

國立交通大學

資訊科學與工程研究所

博士論文

語意式、無所不在、且可自訂流程的智慧型家庭自動化系統

Semantic, Ubiquitous, and User Configurable
Smart Home Automation

研究生：高永威

指導教授：袁賢銘 教授

中華民國一百零一年三月

語意式、無所不在、且可自訂流程的智慧型家庭自動化系統

Semantic, Ubiquitous, and User Configurable Smart Home Automation

研究生：高永威

Student : Yung-Wei Kao

指導教授：袁賢銘

Advisor : Shyan-Ming Yuan



Submitted to Department of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in
Computer Science

March 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年三月

語意式、無所不在、且可自訂流程的智慧型家庭自動化系統

學生：高永威

指導教授：袁賢銘

國立交通大學資訊科學與工程研究所

摘要

智慧型家電與智慧型家庭自動化的觀念已經被提出了很久。然而，當人們提到未來的智慧家庭系統，許多相關研究皆著重於將不同的智慧家電與設備配置在家庭的環境中，並且依照預先定義好的流程來進行自動化的家電控制。使用者可自訂流程的智慧型家庭自動化議題至今還沒有被人們普遍重視。提供使用者可自訂流程的困難之處源自於異質、複雜、且不斷變化的家庭環境。此外，使用者通常會以語意的方式來思考如何控制家中的電器，例如“我想要將所有位於二樓的電燈全部關掉”。如果一個智慧型家庭自動化系統無法接受語意式的命令，及提供語意式的回應，使用者則必須為了一件工作制訂多個流程，並且時常修改與維護它們。本論文提出了一個可接受語意式命令及提供語意式回應，並且允許使用者自行制訂流程的智慧型家庭自動化系統，稱為 USHAS (User-configurable Semantic Home Automation System)。USHAS 採用 Web Service 及 WSBPEL 的技術來執行自動化流程、OWL 與 OWL-S 的技術來定義家庭環境及服務以及一個我們定義的語言：SHPL (Semantic Home Process Language)，來描述語意化的流程。

為了提供方便且親切的使用者介面，USHAS 讓使用者可以透過網頁介面來定義家中的環境以及所有的自動化流程。如此一來，使用者不但可以藉由個人電腦上的網頁瀏覽器存取 USHAS，也可以藉由手持式裝置上的瀏覽器隨時隨地觀察家中的環境現況並且控制家電。除此之外，USHAS 也可扮演家庭資訊中心的角色。USHAS 可對使用者有興趣的網頁及所安裝的網頁應用程式提供在 PC 與手持式裝置上皆可離線瀏覽與執行的功能。

關鍵詞：Smart Home; Semantic Home; Home Automation; Web Service; WSBPEL; OWL; Offline Mobile Web Application; Mobile Content Adaptation

Semantic, Ubiquitous, and User Configurable Smart Home Automation

Student: Yung-Wei Kao

Advisor: Shyan-Ming Yuan

Department of Computer Science and Information Engineering
National Chiao Tung University

Abstract

The ideas of smart home and home automation have been proposed for many years. However, when discussing homes of the future, related studies have usually focused on deploying various smart appliances (or devices) within a home environment and employing those appliances automatically by pre-defined procedures. The issue of user-configurable home automation has largely been neglected recently. The difficulties of supporting user-configurable automation are due to the complexity of various dynamic home environments. Within their home domains, users usually think semantically; for example, “I want to turn off all the lights on the second floor”. If a home automation system cannot accept semantic commands or provide semantic feedback, users may have to create multiple processes for one task and modify them frequently. This thesis proposes a semantic home automation system, USHAS (User-configurable Semantic Home Automation System), which adopts Web Service and WSBPEL for executing automated process; OWL and OWL-S for defining home environments and service ontology; and a self-defined markup language, SHPL (Semantic Home Process Language), for describing semantic processes.

To provide convenient and friendly user interface, USHAS allows users to define their home environments and automation processes via Web pages. In this manner, users can access USHAS and control appliances not only on their PCs but also on their mobile devices. Moreover, USHAS is also able to play the role of home information broker. USHAS can provide the capability of offline browsing and executing the user-selected Web pages and Web applications on PCs and mobile devices.

Keywords : Smart Home; Semantic Home; Home Automation; Web Service; WSBPEL; OWL; Offline Mobile Web Application; Mobile Content Adaptation

誌謝

Acknowledgement

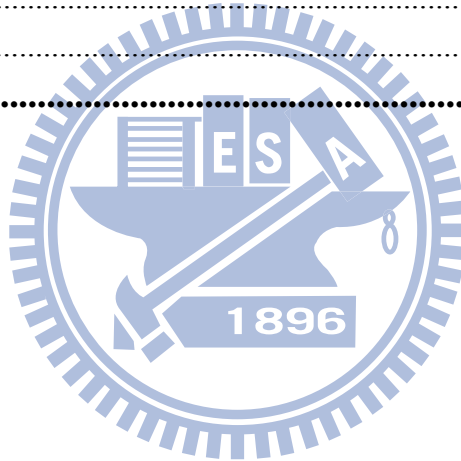
感謝我的指導老師袁賢銘教授在我的研究過程中提供了相當豐富且寶貴的指導，才能讓我有今天的研究成果。另外要感謝我的家人與親戚的關懷，讓我可以安心無慮的進行研究。最後要感謝我最深愛的老婆蕙嬪。蕙嬪的支持與陪伴，是我一路走來最大的動力。因為有蕙嬪，我才能有足夠的勇氣與毅力在六年內完成我的博士學位。謹將此博士論文獻給我最深愛的蕙嬪。



Table of Contents

| | |
|---------------------------------------------------------------------------------|-------------|
| 摘要 | I |
| Abstract | II |
| 誌謝 | III |
| Table of Contents..... | IV |
| List of Figures | VI |
| List of Tables..... | VIII |
| 1. Introduction | 1 |
| 2. Background and Related Work | 7 |
| 2.1 Smart Home and Home Automation | 7 |
| 2.2 SOA and OSGi-Based Smart Home | 7 |
| 2.3 Web Service, WSBPEL, and Web Service Based Home Automation | 9 |
| 2.4 Semantic Web, Context-Aware Home, OWL-S, and Semantic Home Automation | 10 |
| 2.5 User-Configurable Smart Home | 11 |
| 2.6 Mobile Web content adaptation | 13 |
| 3. Design Issues | 15 |
| 3.1 System Assumptions | 16 |
| 3.2 System Overview | 16 |
| 4. USHAS Ontology..... | 20 |
| 4.1 Person class | 21 |
| 4.2 Device clas | 22 |
| 4.3 Event | 24 |
| 5. Semantic Home Process Language (SHPL)..... | 26 |
| 5.1 Variables | 27 |
| 5.2 Time_set | 28 |
| 5.3 Preconditions | 29 |
| 5.4 Flow | 31 |
| 5.5 SHPL example | 32 |
| 6. System Architecture | 34 |
| 6.1 The USHAS Content Adaptation Sub-System | 37 |
| 6.2 The Information Broker Sub-System | 45 |
| 6.3 The Offline Mobile Web Browsing Sub-System | 50 |
| 6.4 The Physical Access Control Sub-System | 52 |
| 6.5 The Face Recognition Sub-System | 53 |

| | |
|-----------------------------------------------------|------------|
| 7. Detailed Design | 59 |
| 7.1 System Model and Detailed Design of USHAS | 59 |
| 7.2 Detailed Design of CAS | 60 |
| 7.3 Detailed Design of IBS | 71 |
| 7.4 Detailed Design of PACS | 77 |
| 7.5 Limitation Analysis | 83 |
| 8. System Prototype and Evaluation..... | 86 |
| 8.1 System Prototype | 86 |
| 8.2 System Scenarios | 97 |
| 8.3 User Satisfaction Evaluation | 98 |
| 8.4 Usability Evaluation | 99 |
| 8.5 Face Recognition Accuracy Evaluation | 104 |
| 8.6 System Comparison | 108 |
| 9. Conclusion and Future Work..... | 111 |
| 9.1 Conclusion | 111 |
| 9.2 Future Work | 112 |
| References | 114 |



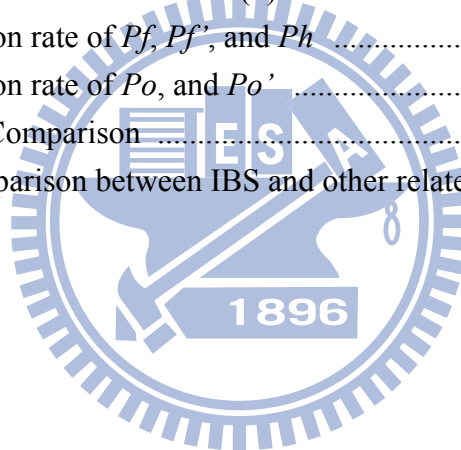
List of Figures

| | |
|--------------------------------------------------------------------------------------|----|
| Figure 3-1. Conceptual stack of USHAS | 17 |
| Figure 4-1. The high level structure of USHAS ontology | 20 |
| Figure 4-2. The low level structure of the Person class | 22 |
| Figure 4-3. The low level structure of the Device class | 23 |
| Figure 4-4. The low level structure of the Event class | 25 |
| Figure 5-1. The high level XSD of SHPL | 26 |
| Figure 5-2. Formal expression of the variables element | 27 |
| Figure 5-3. Formal expression of the time_set element | 28 |
| Figure 5-4. Formal expression of the preconditions element | 29 |
| Figure 5-5. Formal expression of the flow element | 31 |
| Figure 5-6. An example of using SHPL | 34 |
| Figure 6-2. Overview of USHAS Content Adaptation Sub-System | 37 |
| Figure 6-3. Personalize web pages using PC or laptop | 38 |
| Figure 6-4. Browse web pages via mobile devices | 39 |
| Figure 6-5. Page Tailor in Firefox Web browser | 40 |
| Figure 6-6. Select blocks at different granularity | 42 |
| Figure 6-7. Rearrange the selected blocks | 42 |
| Figure 6-8. Internal expression of user preferences about a Web page | 43 |
| Figure 6-9. The communication between these three components | 45 |
| Figure 6-10. High level system architecture of the Information Broker Sub-System ... | 46 |
| Figure 6-11. The detailed architecture of Application Market | 46 |
| Figure 6-12. The detailed architecture of Offline Layer | 47 |
| Figure 6-13. The detailed architecture of Content Adaptation Layer | 49 |
| Figure 6-14. Overview of the Offline Mobile Web Browsing Sub-System | 50 |
| Figure 6-15. The Overview of Physical Access Control Sub-System (PACS) | 52 |
| Figure 6-16. Integration process of face and password | 54 |
| Figure 6-17. Integration process of face and hand gesture | 57 |
| Figure 6-18. Integration process with error | 58 |
| Figure 7-1. System model | 59 |
| Figure 7-2. System components and their interfaces of CAS | 61 |
| Figure 7-3. This is the bookmarklet used to launch Page Tailor | 62 |
| Figure 7-4. The containment hierarchy of elements | 64 |
| Figure 7-5. The selected block (a) and its corresponding sub-tree (b) | 65 |
| Figure 7-6. How to load the user preferences (pseudo-code) | 67 |
| Figure 7-7. The response of query service (pseudo-code) | 68 |

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 7-8. How to update the user preferences (pseudo-code) | 69 |
| Figure 7-9. Produce a corresponding DOM tree | 70 |
| Figure 7-10. A new DOM tree (right) would be created to hold the replicas of selected elements in original DOM tree (left) | 71 |
| Figure 7-11. The sequence diagram of application publication | 72 |
| Figure 7-12. The sequence diagram of application installation | 73 |
| Figure 7-13. The sequence diagram of using the to-do list application | 75 |
| Figure 7-14. The sequence diagram of using the RSS reader application | 76 |
| Figure 7-15. Protocol of user registration phase | 79 |
| Figure 7-16. Protocol of authentication phase | 82 |
| Figure 8-1. The living room of UAHAS prototype | 87 |
| Figure 8-2. The bed room of UAHAS prototype | 88 |
| Figure 8-3. Semantic Environment Editor | 89 |
| Figure 8-4. A semantic process of the USHAS prototype in Semantic Process Designer | 90 |
| Figure 8-5. An example of using SHPL in the prototype | 91 |
| Figure 8-6. A practical example of using the USHAS content adaptation sub-system . | 92 |
| Figure 8-7. The result page of Google Mobile Proxy | 93 |
| Figure 8-8. (a) The mobile version homepage (b) The PC version homepage | 94 |
| Figure 8-9. (a) The to-do list application (b) The to-do list application with one new item added | 95 |
| Figure 8-10. (a) The RSS reader application (b) The online external Web page without content adaptation (c) The offline external Web page with content adaptation | 95 |
| Figure 8-11. The system demonstration of PACS prototype | 96 |
| Figure 8-12. Two ways to specify the precondition of the “Bath Time” scenario | 101 |
| Figure 8-13. The analysis result of usability evaluation (average degree of agreement) | 102 |
| Figure 8-14. The analysis result of usability evaluation (Cronbach α) | 103 |
| Figure 8-15: Usability test (a) Page Tailor in Internet Explorer (b) Page Tailor in Firefox Web browser | 103 |
| Figure 8-16. Recognition rate of P and P_i (1) | 104 |
| Figure 8-17. Recognition rate of P and P_i (2) | 105 |
| Figure 8-18. Recognition rate of P_f , P_f' , and P_h | 106 |
| Figure 8-19. Recognition rate of P_o , and P_o | 107 |

List of Tables

| | |
|-----------------------------------------------------------------------------------------------------|-----|
| Table 6-1. Face and password mapping example | 53 |
| Table 6-2. Nine hand gesture patterns | 55 |
| Table 6-3. Face and hand gestures mapping example | 56 |
| Table 7-1. Examples of origin comparisons | 66 |
| Table 7-2. The database schema of the table AppSync | 74 |
| Table 8-1. Examples of different scenarios and corresponding semantic automation processes | 97 |
| Table 8-2. Analysis of user satisfaction | 98 |
| Table 8-3. Analysis of usability of Semantic Environment Editor | 99 |
| Table 8-4. Analysis of usability of Semantic Process Designer | 100 |
| Table 8-5. Questions included in questionnaire | 101 |
| Table 8-6. Recognition rate of P and P_i (1) | 104 |
| Table 8-7. Recognition rate of P and P_i (2) | 105 |
| Table 8-8. Recognition rate of P_f , P_f' , and P_h | 106 |
| Table 8-9. Recognition rate of P_o , and P_o' | 107 |
| Table 8-10. System Comparison | 108 |
| Table 8-11. The comparison between IBS and other related technologies | 109 |



Chapter 1 Introduction

Previous studies have focused on providing solutions for smart homes, home automation, and ubiquitous homes. Numerous of these studies concentrating not only on controlling devices, but also on combining additional factors before doing so. For example, facial recognition can be performed for opening a door, and hand gesture recognition can be utilized to change the channel of a TV. However, regardless of whether these home appliances are controlled manually by users or indirectly and automatically by computers, most systems provide only pre-defined control of appliances. For example, hand gestures indicating numbers for changing channels on a TV cannot be used as an input for changing the temperature of air conditioner, unless users modify the programs by themselves. Such modification requirement limits the flexibility and scalability of numerous home control and automation systems.

Controlling appliances without pre-defined procedures is difficult, as several requirements must be fulfilled. First, all devices, including home appliances, sensors, and remote controls, should be designed separately with standard interfaces provided. This criterion is reasonable, because users usually purchase devices manufactured by different companies. Second, how automation processes receive information from sensors and control devices should be standardized. Finally, these processes should be designed by users in an easy and understandable manner.

To provide interconnectivity between different parties, various Service-Oriented Architecture (SOA) technologies have been designed. In general, numerous home systems adopted the Open Services Gateway Initiative (OSGi) architecture for smart home implementation. However, the OSGi platform does not provide a complete process management and execution

solution; therefore, pre-defined procedures for executing processes are usually designed for the OSGi. By contrast, Web Service focuses on providing interconnectivity between parties. Moreover, the OASIS Web Services Business Process Execution Language (WSBPEL or BPEL4WS) has been defined for process management and execution of Web Services. Therefore, this thesis adopted the Web Service standard for defining operations of devices and the WSBPEL standard for defining automation processes.

Although WSBPEL provides an effective process definition standard, a gap remains between what users actually want and what they must define in processes. This is because users usually think semantically; for example, “I want to turn off all the lights on the second floor”. In this instance, the WSBPEL process cannot be defined, because the home system does not know about the lights located on the second floor. To solve this problem, this thesis adopted the Web Ontology Language (OWL) to create a knowledge base in a semantic home, and the OWL-S technology to describe the Web Services of the devices.

Although the OWL-S standard provides the functionality of process definition, the difference between the OWL-S process and the WSBPEL process is quite small. Only the input and output variables are mapped to the OWL ontology, and users must specify many binding details. To achieve the goal of semantic home automation, this thesis adopts OWL-S only for semantic service description and discovery, but not for semantic process definition and execution. Because no appropriate candidate is available for defining semantic home processes, this thesis designed a markup language, SHPL (Semantic Home Process Language), to describe semantic processes. In addition, the SHPL execution runtime is developed, which dynamically generates a WSBPEL process for each semantic process based on the current home environment defined in the knowledge base.

This thesis proposes a semantic home automation system, USHAS (User-configurable Semantic Home Automation System), which adopts Web Service and WSDL to execute automation processes; OWL and OWL-S to describe home environments and service ontology; and SHPL to define semantic processes. To prove that the proposed system can satisfy user needs adequately, this thesis designed nine demonstration scenarios: the living room, long vacation, home gym, morning rush, dinner time, good student, sweet dreams, bath time, and party night. Furthermore, this thesis designed and analyzed a questionnaire to determine which scenarios were more appealing to users. Finally, the usability of USHAS was evaluated.

To provide convenient and friendly user interface, USHAS allows users to define their home environments and automation processes via Web pages. In this manner, users can access USHAS and control appliances not only on their PCs but also on their mobile devices. However, numerous environment settings and processes could be defined in USHAS, and only a small part of them are interesting to mobile users; Displaying all the contents of USHAS makes users difficult to read and use.

Not only the USHAS Web pages but also general Web pages have the content adaptation problem when they are displayed on mobile Web browsers. Widespread of mobile devices makes it common to browse Web pages via mobile browsers. However, most Web pages are mainly designed for desktop computers that are equipped with large screens. When browsing on mobile devices, a user might have to scroll up and down, left and right all the time to find the information they want. Because of the limited screen size, this kind of operation is really not user-friendly at all.

Fortunately, some famous websites have another simplified version of Web content specially provided for mobile devices, such as Google Mobile [61]. On the other hand, it is a heavy burden on Web developers to craft and maintain multiple versions of the same website [62-66]. Even with the help of the fascinating toolkits.

If we resize the original web page to fit the width of mobile device, the vertical scroll bar will be too long to view, and the information is crowded. On the other hand, if we provide another version of the original Web page, there may be some important information lost in the mobile version, and the transformation of each page costs a lot for Web page developers. Hence, USHAS is also designed to help users personalize their mobile Web pages for handheld device browsing. Four major objectives are listed and introduced briefly as follows.

Easy-to-use

It does not make sense to launch another program other than the browser to personalize a Web page. When a user surfs on the Internet and finds a Web page that interests him/her, the configuration tool of this system should be able to pop up in the browser window somehow right away. Moreover, all the codes needed to accomplish this job (i.e. personalize web pages) should be downloaded on the fly when accessed, thus allowing a user to work on different computers at different places.

Personalizing Web Pages Visually

Web pages are usually composed of header, footer, sidebar, and content areas [67]. Parts of them are used to maintain a consistent style for the website, and other parts of them are used for navigation. Some renowned Web sites may even contain a lot of advertisements on them.

In many Web pages, only a few of information is really needed to be shown on the mobile phone screen [68][69]. This thesis also aims at allowing users to determine which parts of Web pages should be retained while browsing these pages with their mobile devices.

A friendly user interface should thus be available for users to perform this task [70][71]. For example, with appropriate visual aids (such as highlight), a user can choose blocks in a Web page one by one with different granularity. Through the operation of drag-and-drop, users can determine the relative position of the chosen blocks according to their personal preferences. In short, users can re-construct mobile Web pages simply with visual manipulations, and do not have to write any line of code.

Reducing Wireless Bandwidth Consumption

More than screen size constraints, the limited memory and wireless network bandwidth also make it unsuitable for delivering the entire Web page untailed to mobile devices. Before returning a Web page to mobile devices, some adaptation must be taken to pre-process a Web page according to a user's preferences. So that the volume of data transmission to a mobile phone could be reduced, and thus reduce the consumption of wireless bandwidth as well.

Automatic mobile webpage content extraction

The proposed content adapting algorithm should automatically adapt Web pages to mobile devices. Users always only care about a part of the Web contents. Content adapting applications should provide a function to extract these parts form Web pages.

To solve this problem, USHAS also includes a Web proxy so that only the user-selected parts of Web pages are displayed on the mobile devices. The major contributions of this solution are listed below.

- a. A cross-browser configuration tool is designed.
- b. The Web-based nature of our configuration tool allows a user to configure the settings from different computers, and requires no pre-installation of any software.
- c. Blocks in a Web page can be chosen correctly under the premise that the layout of a Web page does not change frequently.
- d. A Web-based management interface is provided.
- e. An automatic algorithm for mobile web page generation is proposed.

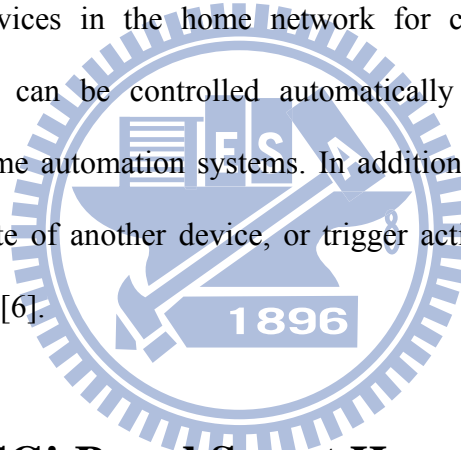
Moreover, USHAS provides the capability of home information broker. Web contents such as Blog articles, News, or RSS seeds shared or subscribed by home members can be pre-tailored and pre-fetched by USHAS according to the user-defined schedules. In this manner, home members can download the adapted and pre-fetched contents so that these contents can be browsed on mobile phones even under offline mode.

The thesis contains seven sections. In Section 2, the background and related work of the proposed system are introduced. Section 3 discusses the design issues of USHAS. The USHAS ontology and SHPL are described in Section 4 and 5 respectively. We present the system architecture and detailed system design of USHAS in Section 6 and 7 respectively. System demonstrations and evaluations of scenarios are presented in Section 8. Finally, we end up with a conclusion and discuss the future work of USHAS in Section 9.

Chapter 2 Background and Related Work

2.1 Smart Home and Home Automation

Smart homes and home automation are popular topics, referring to devices and appliances in the home environment that can be controlled automatically in an intelligent manner. Thus far, numerous studies have proposed system designs and even smart home products [1-4]. In smart home systems, devices and appliances are usually controlled to facilitate the duties of daily life. Home automation systems generally contain an internal network, as well as intelligent rules and devices in the home network for convenient or special purposes. Devices and appliances can be controlled automatically by, or provide environmental information to, these home automation systems. In addition, changing the state of a device may also change the state of another device, or trigger actions in other devices within the smart home environment [6].



2.2 SOA and OSGi-Based Smart Home

Before defining a high level of logic for automation processes, the most crucial challenge is to facilitate the interconnectivity between different devices. Due to highly varied standards for home devices, such as X10 [20], INSTEON [21], UPnP [22], and Jini [23], communication between different devices is difficult to establish using dissimilar interfaces under various standards. For such heterogeneous network integration, the Service-Oriented Architecture (SOA) [14] design principle provides interoperability between various loosely coupled services. Open Services Gateway initiative (OSGi) [15] is one of the technologies that implement the SOA paradigm, and numerous researchers have implemented smart home

systems based on OSGi. Li et al. [24] designed a home network system, providing a Web interface for users to control appliances directly on the OSGi platform. Ishikawa et al. [25] proposed SENCHA, a smart appliance integration middleware framework based on OSGi. They indicated several limitations of OSGi in implementing smart home automation, such as the lack of multiple views of abstraction levels. In other words, the capability of multilevel abstraction of OSGi is not enough. Wu et al. [1] combined the OSGi platform with mobile agents [26] in their design, which involves multiple mobile agents responsible for different tasks, distributed among multiple OSGi platforms. Liao et al. [8] adopted the Message-Oriented Middleware (MOM) [27] paradigm for event handling in their context-aware smart home system. Rui et al. [9] presented a physical structure model and a multi-agent [28] based software architecture based on OSGi; this architecture encapsulated the device sensing as well as control operations into the AmI-Adaptor, and encapsulated the computation logic into the AmI-Box.

The primary problem of OSGi is that only bundles installed on the same OSGi container can inter-communicate. Therefore, technologies such as mobile agents in [1] must be designed to establish communication between different containers. Another problem of OSGi is that no standard process definition is provided. Hence, in the OSGi based systems, high level device control decisions are usually made by multi-agents and pre-defined by programmers. As a result, automation processes are fixed and not allowed to be configured by users. Although the system designed in [9] provides the capability of user configuration to AmI systems, the configuration level is at the AmI-Adaptor and AmI-Box levels, not at the process level; users must select and configure which AmI-Adaptor or AmI-Box to communicate.

2.3 Web Service, WSBPEL, and Web Service Based Home Automation

Web Service [16] is another technology that implements the concept of SOA, consisting of three main standards: Web Services Description Language (WSDL) [29], Simple Object Access Protocol (SOAP) [30], and Universal Description, Discovery and Integration (UDDI) [31]. Similar to OSGi, Web Service provides interoperability between different services; therefore, it is also a candidate for smart home platforms. Uribarren et al. [32] proposed a middleware system based on Web Service for controlling devices with different protocols. Unlike OSGi, a process execution standard, Web Services Business Process Execution Language (WSBPEL) [17], is designed to support process definition for executing Web Services. Because devices are usually provided by different manufacturers, the concept of device functionalities can be mapped to services provided by enterprises; and the concept of process execution using different devices can be mapped to cross-business process execution, including different enterprises. Anke et al. [33] revealed the drawback of OSGi: OSGi bundles are not directly accessible from clients outside of the OSGi container. To solve this problem, Anke et al. designed a system that exposes OSGi bundles using Web Service interfaces, and then executes these bundles using processes defined in WSBPEL. However, using both OSGi and Web Service involves a duplicate design, because both of them follow the SOA paradigm. Systems supporting WSBPEL process definition and execution can adopt device drivers directly implemented by Web Service, instead of by OSGi bundles, to reduce system overhead.

2.4 Semantic Web, Context-Aware Home, OWL-S, and Semantic Home Automation

Although WSBPEL is already a higher layer of both Web Service and OSGi, it is still difficult for users to write a WSBPEL document directly by themselves; many details of static binding, such as portTypes or URLs of services, must still be provided. To enable communication based on semantic ontology between different programs across the boundaries of different organizations, Semantic Web [34] technology is designed on the basis of Resource Description Framework (RDF) [35] and Web Ontology Language (OWL) [18]. In general, Semantic Web technology is usually used for context-aware home automation systems. Wang et al. [36] designed the CONON ontology for context reasoning in pervasive computing environments, including home environments. Moreover, several reasoning engines, such as OWL-QL [37], have been developed for understanding semantic meanings of OWL, and can be highly useful in semantic home systems. For example, if Light L is located in the living room, and the living room is located on the first floor, then the query “all the lights on the first floor” can identify Light L. Furthermore, OWL-S [19] extends the capability of OWL to describe the operations of Web Service and semantic process execution of these operations. Mokhtar et al. [38] developed a QoS-aware dynamic service composition mechanism in ambient intelligence environments, using OWL-S for service description and process definition.

Although OWL-S is based on Semantic Web, it is still not abstract enough for users to express high-level concepts of processes, such as “I want to turn off all the lights on the second floor”. Necessary semantic processes should still be defined explicitly by OWL-S. Given a dynamic home environment, for example, after installing a new light on the second floor, the OWL-S

process should be modified to meet the requirements of using that light. Ha et al. [10] proposed an infrastructure for a ubiquitous home network service, adopting numerous technologies such as Web Service, WSBPEL, and OWL-S. Moreover, they designed a high level semantic process definition. Processes based on this definition are interpreted to WSBPEL processes dynamically according to the current home environment. However, the proposed semantic process definition language is too simplified, and not user-configurable.

Based on Semantic Web, situation-driven approaches [39-41] provide a higher level of semantic process automation abstraction. The concepts of situation, user goals, and broker goals are proposed so that Web Services can be controlled to fulfill brokers' goals, which further fulfill user goals based on particular situations. Users or experts can control process actions by providing various situational parameters. In the smart home domain, the concept of situations can be mapped to the home environment, such as the lighting, current time, and status of appliances. Although users can control the processes by adjusting the situations, the process definitions, or the user goals, should also be pre-defined. In other words, users cannot create their own customized automation processes.

2.5 User-Configurable Smart Home

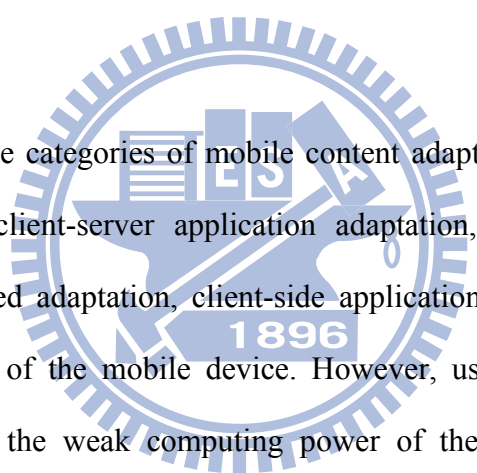
Rodden et al. [42] used the idea of Jigsaw to compose abstract processes for smart home automation. For example, a doorbell piece fitting a webcam piece, which then fits a mobile phone piece, implies that when the doorbell rings, the webcam takes a picture and sends this image to the user's mobile phone. This is a compelling idea because it seems highly simple and user friendly. However, this simple representation causes many ambiguity problems. First, the item definitions are ambiguous; if ten lamps are in a house, there must be a manner to

distinguish them. Identifying each piece merely using the Jigsaw interface is difficult. Determining how users can define and reconfigure their home automation systems in an understandable manner is crucial. In addition, several abstract concepts, such as “no one is home”, are not easily represented; all possible abstract concepts must be defined as pieces before using them. Second, the process definition is ambiguous; the manner in which operations should be executed is not defined. If every item is treated as the same type of piece, three pieces of webcams concatenated one-by-one is possible; however, the meaning of this process is not understandable. Finally, arguments of operations cannot be defined. For example, “turn on the air conditioner if the temperature is higher than 30 oC” cannot be defined because the number 30 cannot be assigned. Moreover, only one precondition piece is allowed to be defined in a process in this system. A trade-off exists between usability and unambiguity; a user interface design that is too simplified is not always understandable to users.

Drey et al. [43] proposed a taxonomy-driven approach to visually prototyping pervasive computing applications, including those of smart homes. This system allows users to create their own semantic automation processes, or rules, based on the sensor-controller-actuator development paradigm. Furthermore, they support the concept of “all” when defining rules using the symbol “*” after entity class names. However, the meaning of “*” differs between the sensor and actuator sides; using “*” on the sensor side implies “any one instance of this category”, but implies “all of the instances of this category” on the actuator side. The ambiguous design of “*” may confuse users when designing processes. Moreover, this system supports the concept of clock time only, and concepts of repeated time, such as every day or every weekend, are not supported.

2.6 Mobile Web content adaptation

Mobile content adaptation is a significant issue when browsing Web sites on mobile devices. In general, Web pages are designed for PC users with large screens. Therefore, when users browse these Web pages on small-screen mobile devices, the display results are usually unsatisfactory. Many mobile content adaptation technologies were designed to solve this problem. The challenges of mobile content adaptation include different device profiles and user preferences. Moreover, since mobile devices do not usually have computation capabilities as powerful as those of PCs, only certain multimedia formats are supported for multimedia content.



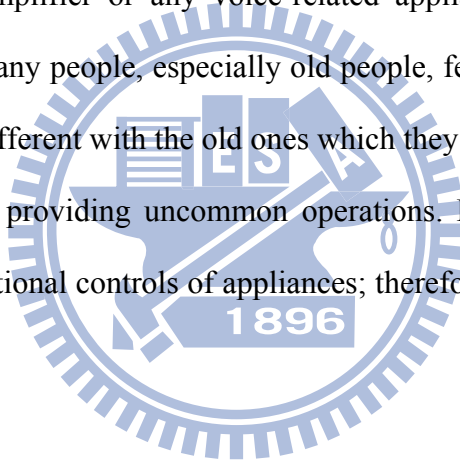
In general, there are three categories of mobile content adaptation architecture: client-based application adaptation, client-server application adaptation, and proxy-based application adaptation. In client-based adaptation, client-side application performs content transcoding according to the profile of the mobile device. However, user preferences must be stored locally on devices, and the weak computing power of the mobile device may decrease efficiency. In server-side adaptation, the server generates or prepares different versions of content and decides what type of content should be delivered to clients according to their profiles. The problem is that server-side programs must be modified to support different types of clients; adapted content for mobile users is not available if the server does not support them. In proxy-based adaptation, content is transcoded on the fly during delivery from server to client; this can solve the problems of client-side and server-side adaptation architectures. However, in certain cases, such as SSL-encrypted communications, Web content can neither be modified nor adapted during transmission between clients and servers.

One client-based application adaptation technology is Opera's Small-Screen Rendering™ in Opera's mobile Web browser. This technology intelligently reformats Web pages to fit the width of screen on mobile devices, thereby eliminating the need for horizontal scrolling. Only the layout of page is changed; all the Web content remains. Another example of client-based application adaptation is the Smart-Fit Rendering technology in ACCESS NetFront. Similar to Opera's Small-Screen Rendering, Smart-Fit Rendering also renders Web pages to fit the narrow screen width of mobile devices.

The main difference between client-server application adaptation and proxy-based application adaptation is whether the content provider or third-party service provider is responsible for conducting the adaptation. Therefore, many adaptation systems can either be deployed server-side or proxy-side. Chen et al. proposed a mobile content adaptation system, which can be either a server-side or a proxy-side architecture. Their system separates a large page into several regions, and creates a minified image as an index page; if any region on the index page is selected, the user is redirected to another page that only displays the content of the selected region.

Chapter 3 Design Issues

From our observation, numerous home appliances are easy to be operated, even by old people, but others are not. The difference between these operations is that several operations are quite common even over different appliances. For example, almost all appliances have power buttons to switch them on or off. It is quite easy to turn an appliance on by pressing the power button even it is a new kind of appliance that you never see it before. Another example is that it is easy to change the volume of TV by pressing the volume up and volume down buttons on the remote control. If you know how to change volume of TV, it is quite easy to learn how to change volume of an amplifier or any voice-related appliance providing volume up and volume down buttons. Many people, especially old people, feel frustrated if the operations of new devices are totally different with the old ones which they already familiar with. Computer is an example of device providing uncommon operations. Except the power on operation, nothing is similar to traditional controls of appliances; therefore many old people are afraid of using computers.



On the basis of our observation, we attempt to identify certain common operations between different appliances, or within the same category of appliance. We believe that common operations not only provide an understandable interface to users, but also provide a high level of abstraction of control. For example, a command “turn off all devices at the second floor” can be created for power saving if all devices support the operation of “turn off”. However, not all devices are the same; several devices, such as multi-purpose washing machines, are valuable because they provide multiple, usually uncommon operations. Another issue is that operations which are not common may become common in the future; therefore, the definition of common operations must be extendable.

Furthermore, this thesis used the idea of common operations to design the definition of automation process. Since no common process definition exists for home automation until now, we attempt to identify the pattern of common requirements in smart home environment. A trade-off also exists between simplicity and capability of process definition; if too many factors are included, the system may frustrate users, but if too little factors are included, some scenarios may not able to be expressed. Finally, the issues of semantic, unambiguous, and user-controllable process definition which are already discussed in Section 2 are also crucial to be addressed.

3.1 System Assumptions

On the basis of the scope of this thesis and the analysis of design issues, several assumptions for USHAS are defined as follows:

- a. All the devices to be controlled have been deployed to users' home environments. For example, lights have been connected to X.10 modules for USHAS to control.
- b. Manufactures or third-party driver providers can provide device drivers which are implemented as Web Services and follow the driver standard of USHAS.
- c. Operations are more understandable if they are common operations.
- d. Automation processes are more understandable if semantic concepts are included.
- e. Automation processes are more understandable if they are defined under a common process pattern.

3.2 System Overview

As discussed in Section 1 and 2, USHAS adopts the Web Service technology for exposing services provided by drivers, WSBPEL technology for execution level process definition, Semantic Web technology for knowledge base construction, and a semantic process definition language, SHPL, for semantic process definition. The conceptual stack of USHAS is shown in Fig. 3-1.

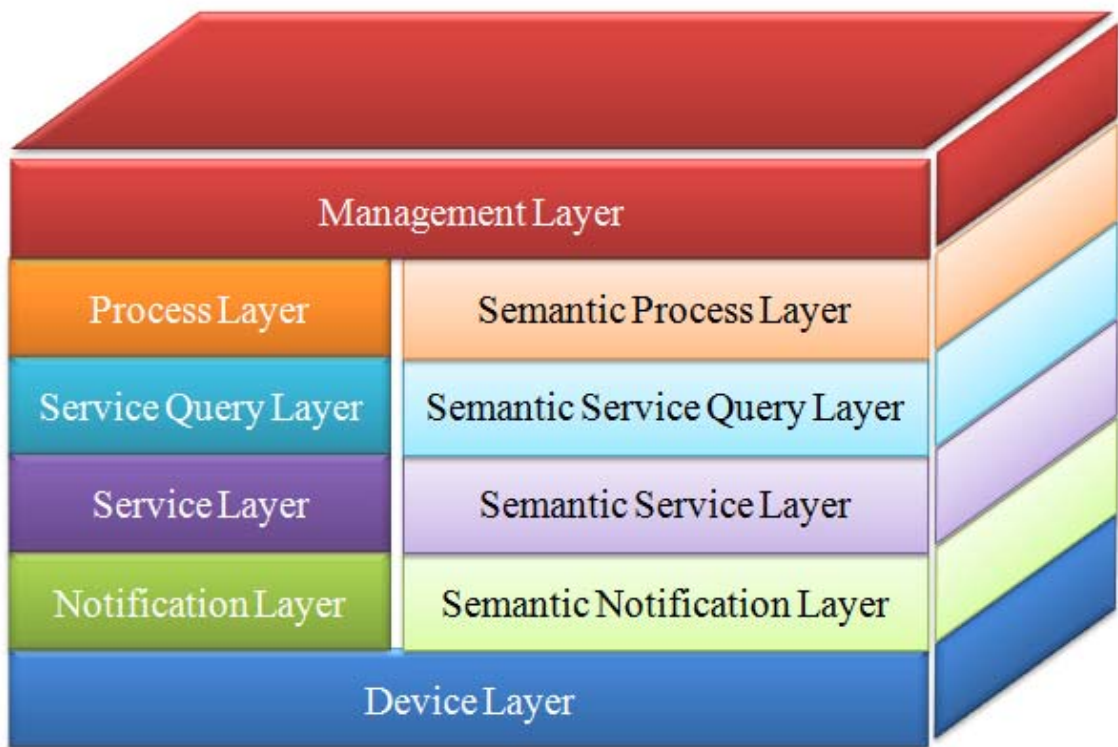


Figure 3-1. Conceptual stack of USHAS

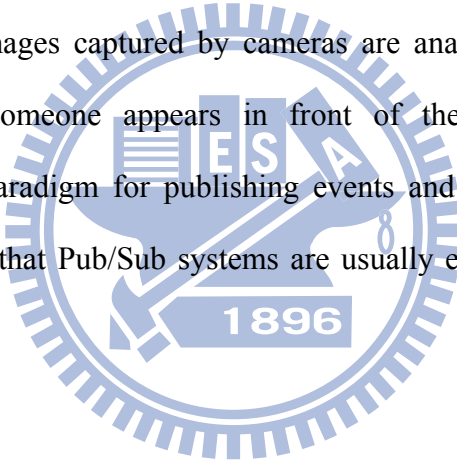
Traditionally, devices can be controlled by two kinds of execution paradigms of smart home: user-controlled execution and event-driven execution. In the proposed system, user-controlled device execution is easily achieved simply by invoking the services provided by device drivers; therefore, this thesis focused on event-driven execution. Layers of conceptual stack are introduced as follows:

a. Device Layer

Two categories of device are designed: sensor and actuator. Sensors collect environment information, and actuators enforce executions based on commands coming from users or automation systems. In USHAS, both sensor and actuator style devices are encapsulated by device drivers following the Web Service standard.

b. Notification Layer

Traditionally, some kinds of agent are designed for receiving sensing data generated by sensors. For example, images captured by cameras are analyzed, and then a face-detected event is generated if someone appears in front of the camera. USHAS adopts the Publish/Subscribe [44] paradigm for publishing events and notifying subscribers based on pre-defined topics, since that Pub/Sub systems are usually employed for event management [45-46].



c. Service Query Layer and Process Layer

Since USHAS follows the Web Service standard for service description, it adopts UDDI for service query. Also, USHAS uses the WSBPEL as the execution level process definition language, and a WSBPEL runtime for process execution.

d. Semantic Notification Layer

Some semantic events such as NoOneAtHomeEvent are defined in this layer. Except explicitly defined events, we found that the concept of semantic event handling is quite

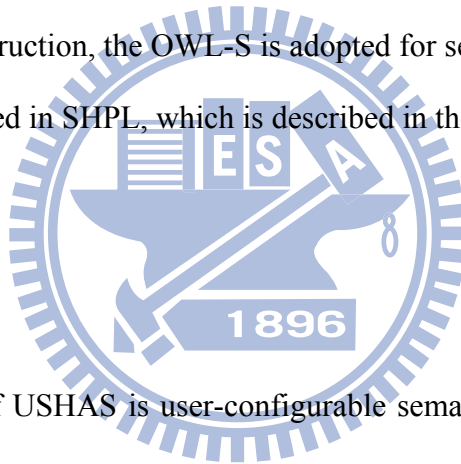
similar to the concept of semantic process execution; semantic events can be mapped to pre-conditions of semantic process, and semantic event handling can be mapped to what must be executed in the body of semantic process. Therefore, the semantic notification layer not only handles explicitly defined events, but also includes pre-condition checking before semantic process execution.

e. Semantic Service Query and Semantic Process Layer

Although OWL-S is not abstract enough for semantic process definition, it is mature for describing and querying services of Web Services. Since USHAS uses OWL as the ontology for knowledge base construction, the OWL-S is adopted for semantic service query. Moreover, semantic process is defined in SHPL, which is described in the next section.

f. Management Layer

One of the main goals of USHAS is user-configurable semantic process design; therefore, a management layer for users to create semantic processes must be provided. Also, since home environment differs from user to user, there must be an interface provided for users to define the ontology of their home environments.



Chapter 4 USHAS Ontology

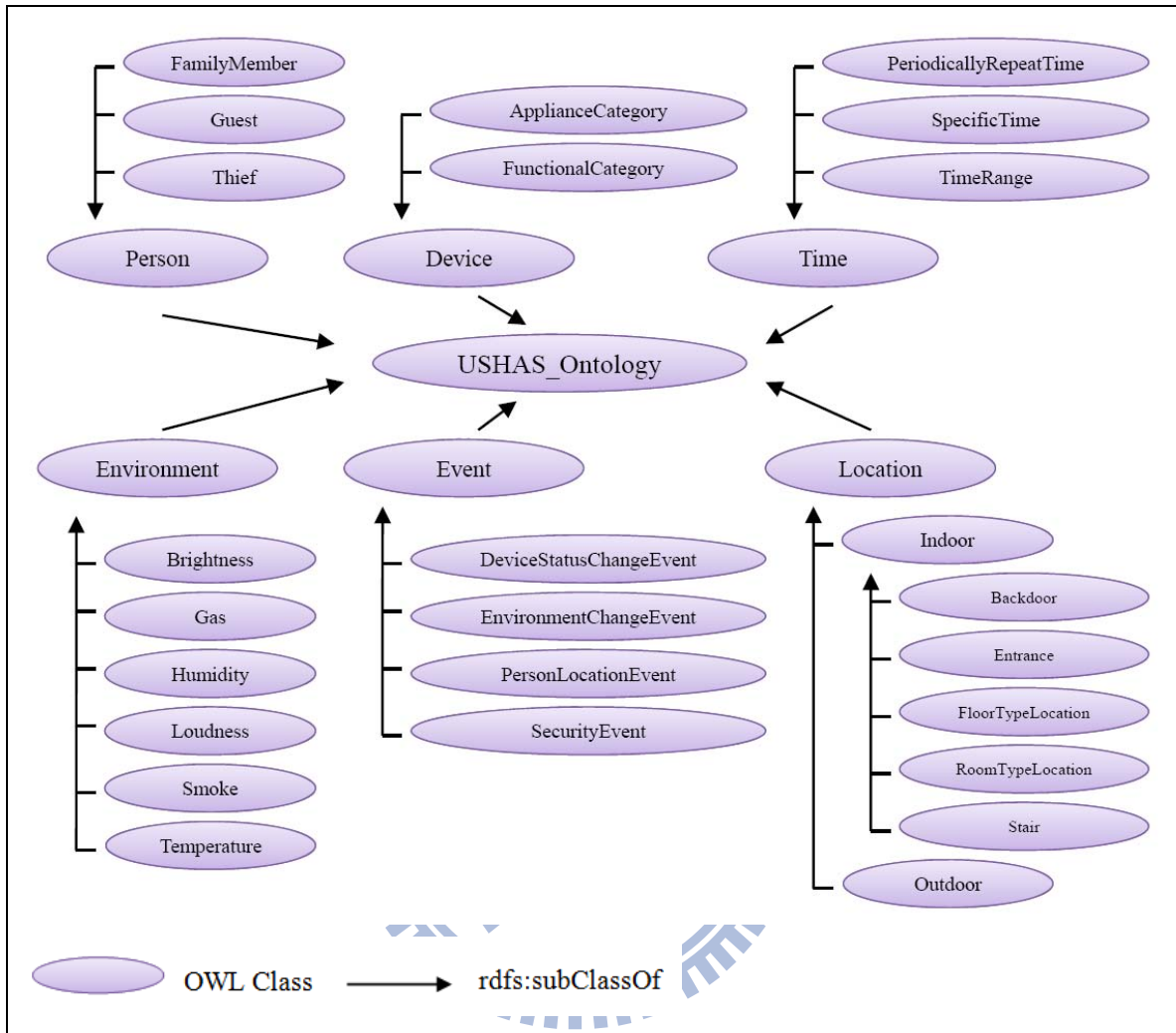


Figure 4-1. The high level structure of USHAS ontology

To describe the status of home, ontology is needed to be defined for home domain; thus, high level information can be maintained, queried, and reasoned. Since a lot of controls and automation processes are usually executed based on the descriptions of user's home, home ontology is usually defined as the skeleton of knowledge base in smart environment systems. The OWL technology is usually used for illustrating home ontology, since it provides a well-defined concept representation model for describing semantic concepts. Wang et al. [36] proposed the CONON ontology, which includes four main classes: CompEntity, Location, Person, and Activity. However, the concept of time is not included. Also, Chen et al. [47]

proposed the SOUPA ontology, which includes nine classes in the SOUPA core: Person, Agent, Policy, BDI, Event, Action, Space, Time, and Geo-M. However, since they only focus on meeting rooms, some environment concepts such as Humidity are not included. On the basis of the requirements of USHAS, the USHAS ontology is defined, which is shown in Fig. 4-1.

Six first-level classes are defined in the USHAS ontology: Person, Device, Time, Environment, Event, and Location. In the Person class, different in-home roles are defined, such as adult member, child member, or even a thief. In the Device class, all the sensor or actuator devices are classified based on the well-known appliance names, such as TV, or their purposes. The Time class includes the definitions of specific time, a range of time, or periodically repeated time (e.g. “Everyday”). Some invisible environmental information, such as brightness or humidity, is classified into the Environment class. Also, some semantic event classes are defined under the Event class. Finally, the Location class maintains the names of spaces, such as living room or first floor in the home domain. Details of some first-level classes are described further as follows.

4.1 Person class

Figure-4-2 shows the low level structure of the Person class. The Person class has two data type properties, FirstName and LastName, to record everyone’s name. In addition, this class has an object type property for maintaining the location of this person. The FamilyMember class has five sub-classes: ElderlyMember, AdultMember, ChildMember, BabyMember, and PetMember. Many applications are developed based on the age of residents. For example, fall detection usually involves elderly members, and adult members are usually used to ensure that particularly dangerous activities, such as cooking, are safely executed.

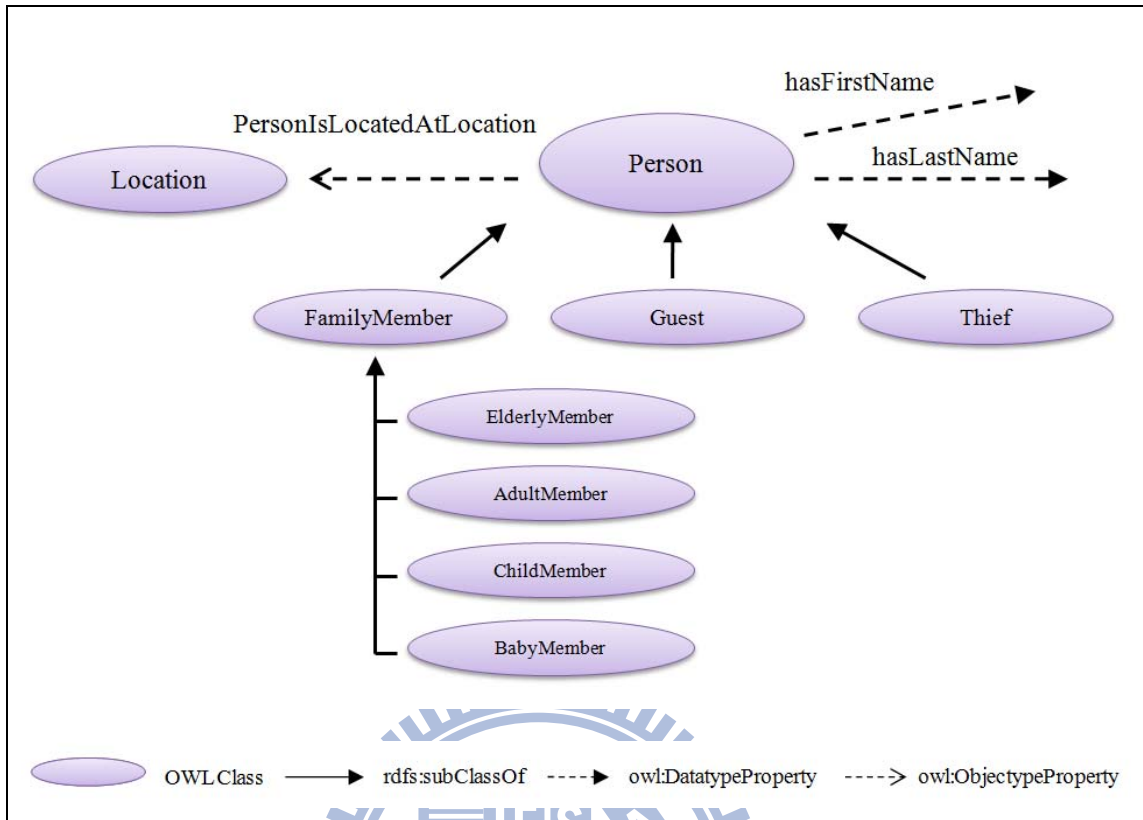


Figure 4-2. The low level structure of the Person class

4.2 Device class

Figure 4-3 shows the low level structure of the Device class. Similar to the Person class, the Device class has an object type property “IsLocatedAt” for indicating the location of this device. The Device class also has a “Generates” property, which maintains an event log for all events generated by this device. Although events are usually generated by sensor devices, generating events is not limited to sensor-style devices; for example, the TV is not a sensor device, but it is allowed to publish a “DeviceStatusChangeEvent” if someone changes the channel. More details of event are described in the Event class section.

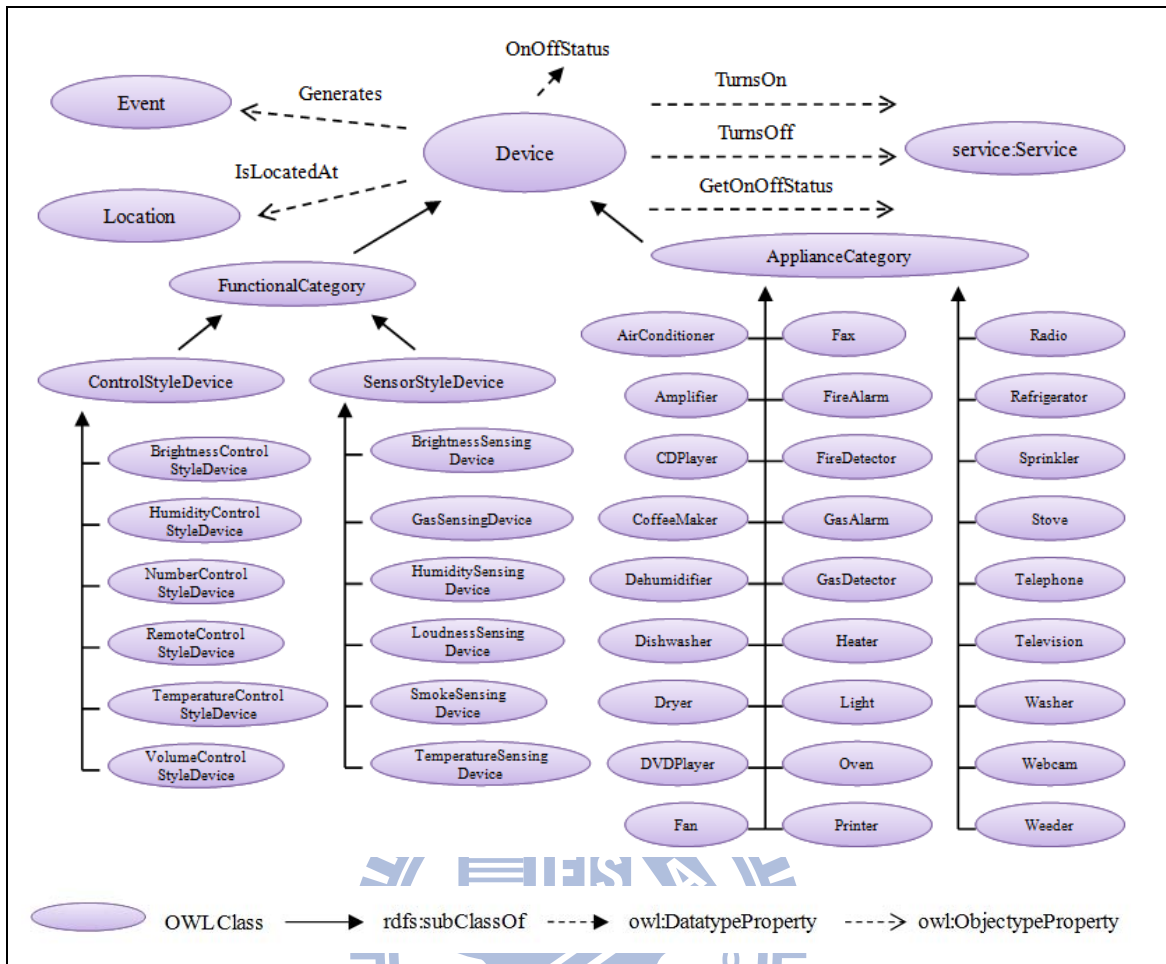


Figure 4-3. The low level structure of the Device class

Three basic service pointers, “TurnsOn”, “TurnsOff”, and “GetOnOffStatus”, are maintained by this class. In OWL-S, each operation of a Web Service is encapsulated as an OWL-S service (service: Service) for invocation. In other words, each Web Service of device must support at least three operations for turning it on, turning it off, and querying its current on-off status. The reason for this requirement is that the on-off status is the most basic and fundamental information of a device. For some simple devices, such as a lamp, provide only the on-off functionality for users to control; for other complex appliances, such as a TV or air conditioner, users also have to turn them on before using them. Controlling the on-off status is the most common and vital functionality shared by all devices at home. With these three

definitions, USHAS can turn on devices, turn off devices, or query the on-off status of devices to maintain the “OnOffStatus” property in a batch process.

Two sub-classes, FunctionalCategory and ApplianceCategory, are defined under the Device class. The FunctionalCategory is further sub-divided into the ControlStyleDevice and SensorStyleDevice. The SensorStyleDevice is designed for sensor devices. Other appliances and devices can be classified by the common product name or the category of their functionalities. By defining the category or product name, similar products can be controlled in a batch process; for example, we can turn on all air conditioners in the home, even the brand names are different. Conversely, by defining the category of functionality, various types of products with shared functionalities can be controlled in a batch process; for example, we can set the audio volume of all appliances at zero after midnight. Although two different manners of classification are available, one device can belong to both of them simultaneously. For example, a TV belongs to the Television class under ApplianceCategory, and also the NumberControlStyleDevice under FunctionalCategory.

4.3 Event

In general, events are generated by agents after analyzing the measurements of sensors. For example, BrightnessSensingDevice generates BrightnessChangeEvent. Moreover, semantic events are also defined under the event class. For instance, the NoAdultAtHome event is generated, based on the presence of each person and the definition of the Person class at home.

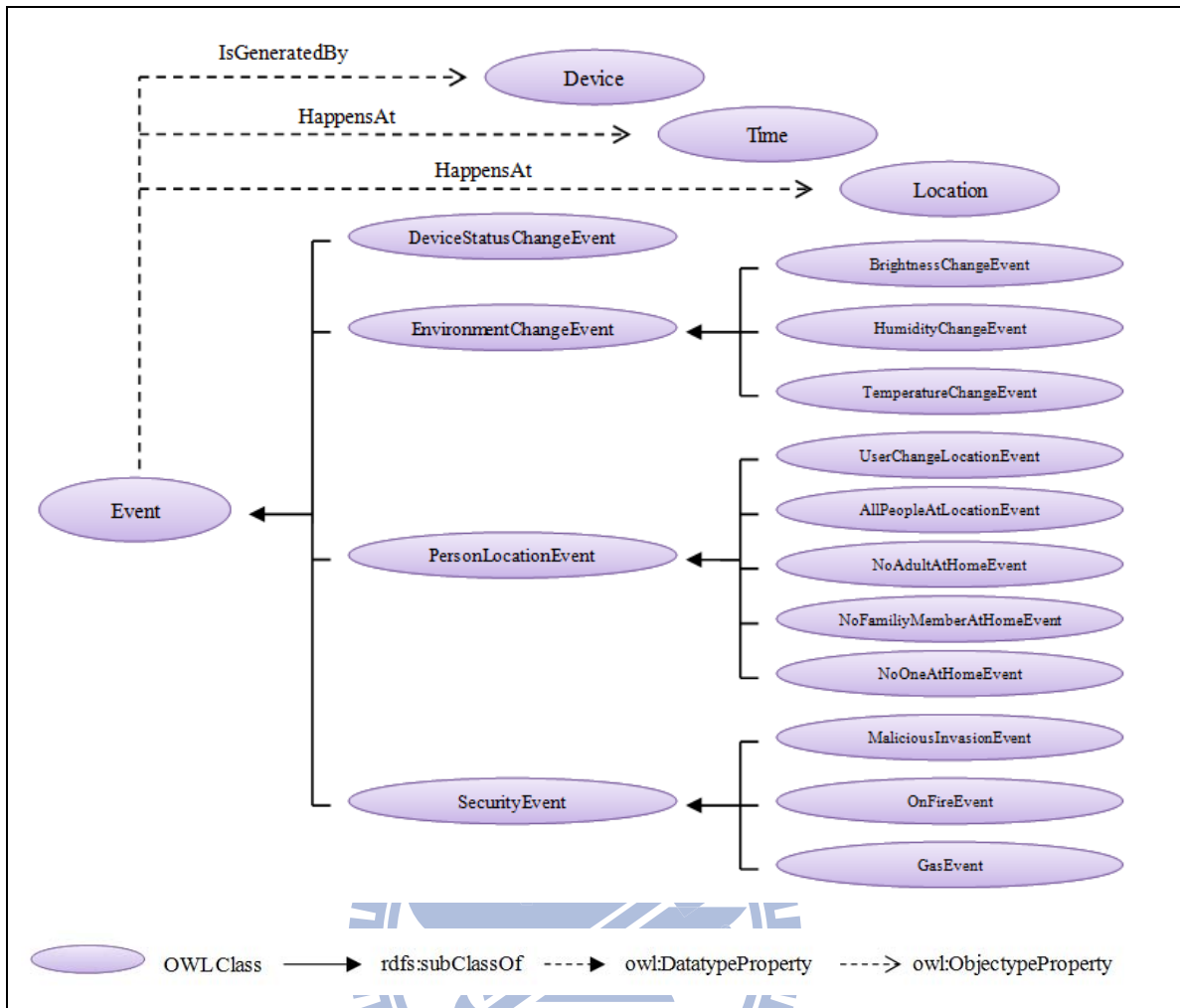


Figure 4-4. The low level structure of the Event class

Chapter 5 Semantic Home Process Language

(SHPL)

Since the semantic process definition of OWL-S is not abstract enough, and no higher level process description based on it exists until now, we decide to define a new process description language, Semantic Home Process Language (SHPL), to support semantic process definition without static service binding.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema elementFormDefault="qualified"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="SProcess">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="variables"/>
        <xsd:element name="time_set"/>
        <xsd:element name="preconditions"/>
        <xsd:element name="flow"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

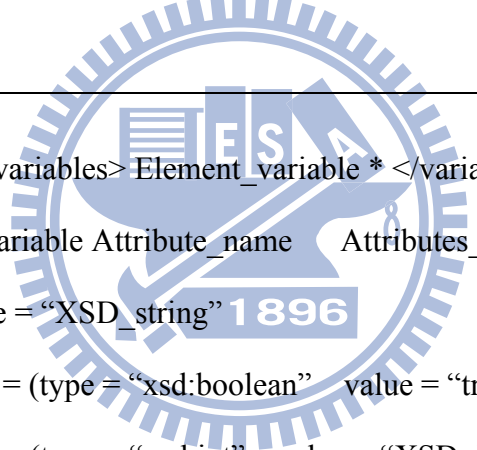
Figure 5-1. The high level XSD of SHPL

From our observation, four vital factors are usually included in home automation processes: pre-conditions, variables, execution time, and flow of invocations. For example, given a command “if the temperature is higher than 30 oC at 6:00 PM, turn on the air conditioner”, the value 30 is a variable; “if the temperature is higher than 30 oC” is a pre-condition, 6:00 PM is an execution time, and “turn on the air conditioner” is an invocation. Although post-condition is usually defined by the generic process description, in home automation

situations, users usually know what the post condition is after executing each invocation; therefore, we can eliminate this definition to decrease the complexity of SHPL. The high level XSD of SHPL is shown in Figure-5-1.

5.1 Variables

The design of variables provides users a manner to express information that can be input by users when defining pre-conditions and invocations. For example, we can define a variable with type integer and value 10, and then let it be the input of the “set_channel” invocation of the TV. The formal expression of the “variables” element is shown in Figure-5-2.



```
Element_variables := <variables> Element_variable * </variables>
Element_variable := <variable Attribute_name Attributes_type_value />
Attribute_name := name = “XSD_string”
Attributes_type_value := (type = “xsd:boolean” value = “true | false”) |
                        (type = “xsd:int” value = “XSD_int”) |
                        (type = “xsd:double” value = “XSD_double”) |
                        (type = “xsd:string” value = “XSD_string”)
```

Figure 5-2. Formal expression of the variables element

Only one “variables” element is in each process, which can contain several “variable” elements. Each variable element has three attributes: name, type, and value. In SHPL, only four basic variable types are supported: boolean, integer, double, and character string. The reason why only these types are supported is because they are the most understandable types

for people to input. In general, home appliances do not require complex input, because inputting information is difficult if it is complex. Many operations, such as switching TV channels, receive numbers as inputs; most operations, such as turning on lamps, require no input from users. Although string-type input is rarely supported by home appliances, it is understandable for users to input string-type with particular devices in the future home environment. For example, users may be able to key-in the name of a movie on a mobile phone, and the DVD player would then start displaying the movie in the living room. Therefore, the type string is also supported for a higher flexibility under the concern of usability.

5.2 Time_set

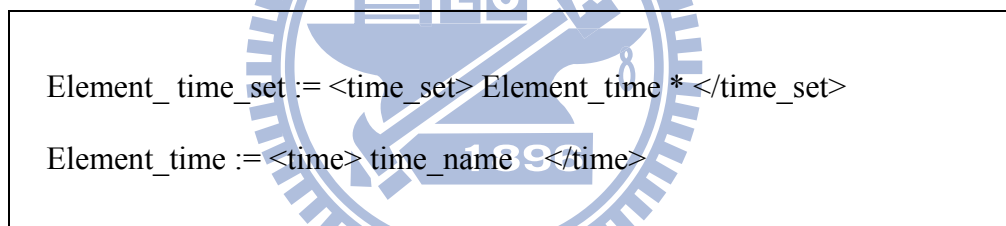


Figure 5-3. Formal expression of the time_set element

The “time_set” element maintains all time instances, which are defined under the Time class of the USHAS ontology, and are execution time of processes. The relationship between time instances is conjunction; in other words, only when all time elements are satisfied will the process be triggered. Although the conjunction relationship limits the capability of expression, it simplifies the logic of time_set; when both conjunction and disjunction relationships are permitted, it is too complex for users to define and comprehend. If users wish to define a process using the disjunction relationship of time instances, they can define multiple

processes using different execution time instances instead. The formal expression of the time_set element is shown in Figure-5-3.

5.3 Preconditions

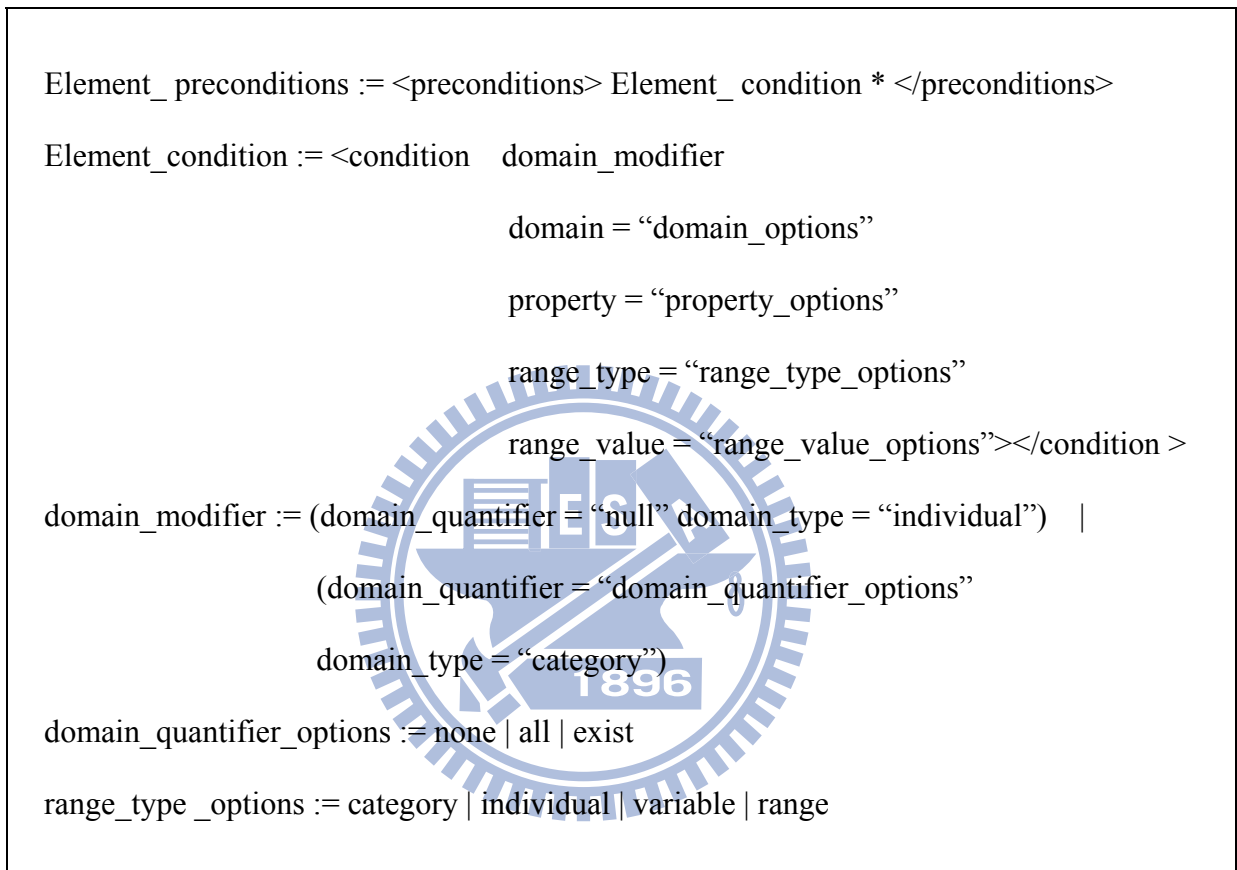


Figure 5-4. Formal expression of the preconditions element

Each “preconditions” element is allowed to contain several “condition” elements. Similar to “time_set”, the relationship between different “condition” elements is also conjunction; thus, only when all conditions are satisfied will the process be triggered. Similar to the reason for using only conjunction in time_set, users may be confused if both the conjunction and disjunction are supported in preconditions. For example, given the process “if the room temperature is lower than 20 degrees and Mary is in the living room or the baby is at home, then, switch on the heating”, the conditions can be explained in two different ways: “if the

room temperature is lower than 20 degrees and Mary is in the living room” or “the baby is at home” and “if the room temperature is lower than 20 degrees” and “Mary is in the living room or the baby is at home”. As a result, users may create unfavorable processes due to the complex logic combination. When only conjunction is allowed, users can still create multiple processes such as “if the room temperature is lower than 20 degrees and Mary is in the living room, then, switch on the heating” and “if the baby is at home, then, switch on the heating” for achieving the same goal. The formal expression of the “preconditions” element is shown in Figure-5-4.

The description of each condition contains three main parts: domain, property, and range. In other words, if particular subjects have properties with values, the condition is satisfied. Moreover, the domain can be further described by the “domain_quantifier” and “domain_type”. The “domain_type” can be a category or an individual. For example, we can specify a person or the Person class as a domain. The “domain_quantifier” is the quantifier of a domain; it is specified only when the “domain_type” is a category. For example, we can specify “all people in the FamilyMember class” or “at least one instance of the OnFireEvent exists” as the subject. The property attribute can be any supported property of a specified domain; if the “domain_type” is a category, only the shared property of this category is allowed to be defined. Moreover, four kinds of range_type: category, individual, variable, and range, are defined. The “individual” and “variable” are the two most basic types; the “individual” type is used for values of object properties, and the “variable” type is used for values of data properties. The range type is used when specifying a range of value. For example, we can specify a range of temperature value in the range_type. Finally, if the category type is specified as range_type, all instances belonging to this category are examined. For example, the pre-condition “all family members are in the bedrooms” is satisfied even when family members are located in different bedrooms.

5.4 Flow

Each flow element is allowed to contain several invoke elements, which enable users to specify which operations of which devices are controlled. In most home automation scenarios, only actuators are controlled; therefore, we mainly focus on invoking devices. If the home automation system providers intend to provide some special agents to be invoked, such as MMS sender, they can implement these agents as special devices following the general device interface.

```
Element_flow := <flow> Element_invoke * </flow>
Element_invoke := <invoke domain_quantifier = "domain_quantifier_options"
                  category = "category_options"
                  device_name = "device_name_options"
                  location_type = "location_type_options"
                  location = "location_options"
                  operation = "operation_options"
                  variable = "variable_options"></invoke >
domain_quantifier_options := all | one
location_type_options := category | individual
```

Figure 5-5. Formal expression of the flow element

Similar to pre-conditions, users can also specify whether all devices are, or only one device of the device category is controlled. We observe that devices are usually specified associated

with their locations with their categories, such as “the lights in the living room”. Therefore, the location and category of controlled device are also included. Finally, the operation supported by the specified device is also included into the invoke element with input variable defined in the variables element.

5.5 SHPL example

```
<?xml version="1.0" encoding="utf-8"?>
<SProcess name="turn off lights on 2nd floor and turn on tv"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="SHPL_Schema.xsd">
  <variables>
    <variable name="input1" type="xsd:boolean" value="true"/>
  </variables>

  <time_set><time>DinnerTime</time></time_set>

  <preconditions>
    <condition domain_quantifier="none" domain_type="category"
      domain="Person" property="IsLocatedAt"
      range_type="category" range_value="SecondFloor"></condition>
    <condition domain_quantifier="exist" domain_type="category"
      domain="Person" property="IsLocatedAt"
      range_type="individual" range_value="MyDiningRoom"></condition>
  </preconditions>

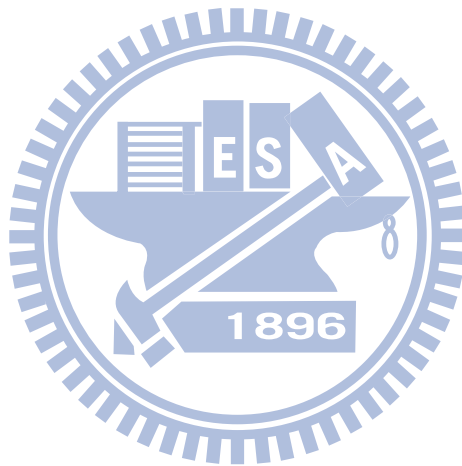
  <flow>
    <invoke domain_quantifier="all" category="Light" device_name="null"
      location_type="category" location="SecondFloor"
      operation="TurnsOff" variable="input1">
    </invoke>

    <invoke domain_quantifier="one" category="Television" device_name="MyTV"
      location_type="category" location="LivingRoom"
      operation="TurnsOn" variable="input1">
    </invoke>
  </flow>
</SProcess>
```

Figure 5-6. An example of using SHPL

Figure-5-6 shows an SHPL example of the scenario “if anyone is in the dining room, and there is no one on the second floor at dinner time, then turns off all lights on the second floor, and turns on the TV in the living room”. The variable “input1” is a simple variable with a true value for confirming the operation execution. The time “DinnerTime” must be defined as a

part of the home environment before creating this process. Two preconditions are specified in this process: the first has the Person category domain and the SecondFloor category range, and the second has the Person domain and the MyDiningRoom individual range. Two operations are executed when these two preconditions are satisfied; the first is the TurnsOff for all Light categories on the SecondFloor, and the second is the TurnsOn for MyTV in the LivingRoom.



Chapter 6. System Architecture

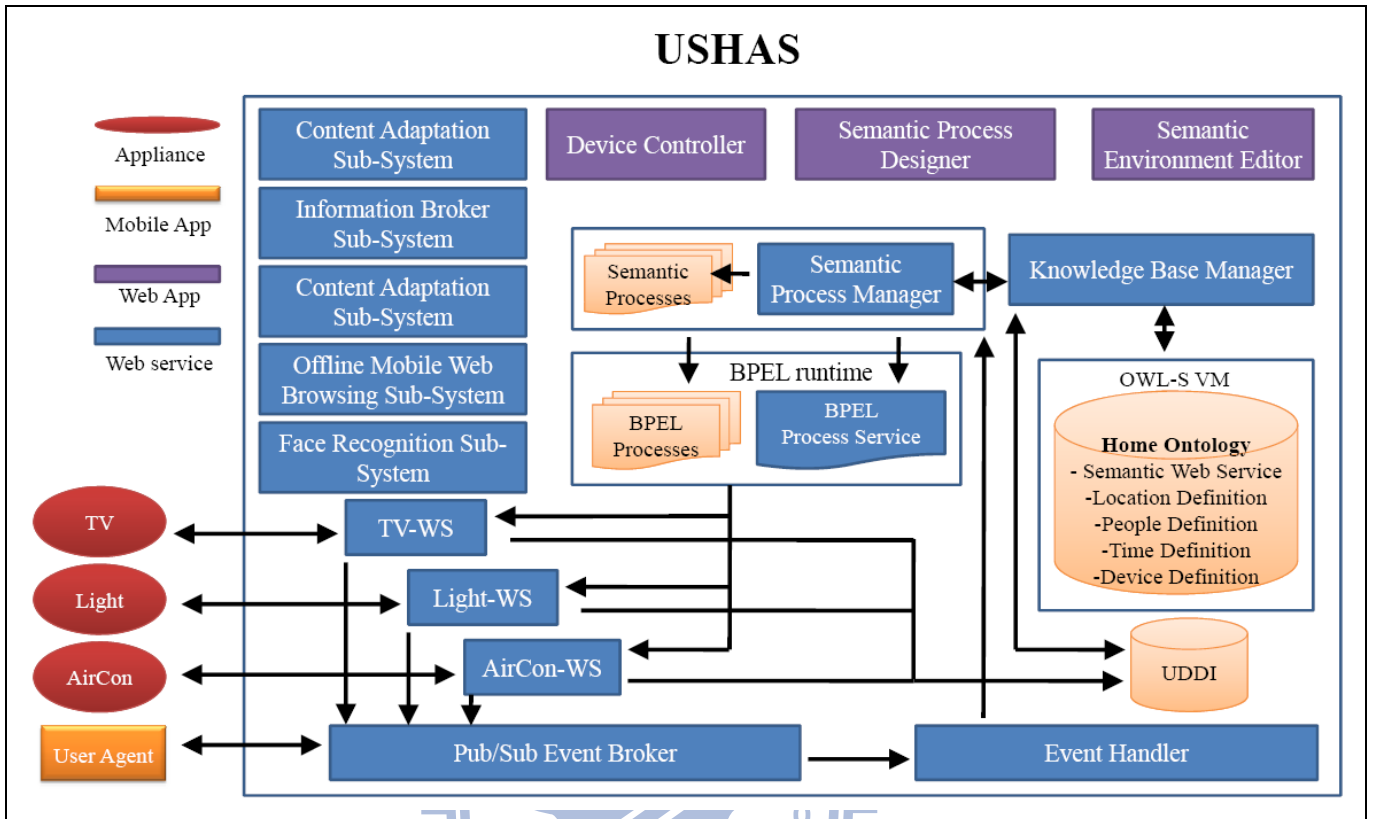


Figure 6-1. System architecture

The system architecture of USHAS is shown in Figure 6-1. For each device to be controlled, a Web Service driver must be implemented based on the protocol and functionalities of this device by device manufacture or third-party developers, such as TV-WS, Light-WS, and so on. Also, detail binding configurations such as house code and key code, must be able to be configured by users via these drivers. For WSBPEL automation, a BPEL runtime is included, which is able to control device drivers based on the BPEL process descriptions. Similarly, for semantic process automation, a semantic process runtime is included, which is able to generate BPEL process dynamically based on current home environment.

To maintain all the home knowledge and status of home environment, a knowledge base following the OWL-S standard is included together with the manager of it. Within the knowledge base, semantic Web Service information, location definition, people definition, time definition, and device definition are maintained. Although semantic Web Service information and device definition are both related to devices, the device definition is a higher level of representation, such as device category, which is user understandable, while the semantic Web Service information is a lower level of representation, such as binding information, which is not user understandable, and is not managed by user directly.

The execution of semantic automation process depends on the decision of the Semantic Process Manager. The Semantic Process Manager constantly checks whether the current time is satisfied with any time instance defined in the semantic processes. Moreover, the Semantic Process Manager checks whether all the pre-conditions are satisfied in each semantic process once the Home Ontology has been updated. Also, for each semantic process, an execution flag is set to indicate whether this process has been executed. Only when all the pre-conditions and time instances are satisfied when the process has not been executed, the process will be executed. The design of this execution flag is important, since users may want to change the result of execution when the pre-conditions and time instances are still satisfied. For example, if no execution flag is set when executing the process “If there is anyone in the living room from 7:00 PM to 9:00 PM, turn on the TV and set the channel to 20”, the TV channel will always be 20 during 7:00 PM to 9:00 PM even users intend to change it. In this manner, this is an annoying process. Therefore, only when the execution flag is set as “not has been executed”, the process can be executed. After the process is executed, this flag will be set as “has been executed”, and, it will be set as “not has been executed” again if any of the time instance or pre-condition is not satisfied. After the Semantic Process Manager decides to execute a semantic process, it translates the flow of this process into a WSBPEL process

based on the current home environment, and then invokes the BPEL process service for WSBPEL process execution.

For event handling, firstly, both kind of explicitly defined basic events and explicitly defined semantic events must be published to the Pub/Sub Event Broker. Developers can create their own services to handle events by subscribing to the Pub/Sub Event Broker, otherwise, all the pre-defined events are sent to the Event Handler which invokes the BPEL runtime directly for basic events, and invokes the semantic process runtime for explicitly defined semantic events. Also, a user agent is designed to send location information and update users' real-time locations.

Finally, three user interfaces are provided via HTTP. The Semantic Environment Editor facilitates users to define information of home environment, such as room definition, family member definition, and time definition. Also, users can deploy bundles of devices drivers via this editor. For defining semantic processes, the Semantic Process Designer is included, which contains variable definition, pre-condition definition, invocation definition, and so on. Options provided in the Semantic Process Designer are basically provided depend on the definition of current home environment, which is based on the input coming from the Semantic Environment Editor and real-time events.

As to user interface, several home automation systems also include the maps of the homes. However, the problem of providing a map is that the structures of users' houses are usually different; users must construct their own maps before using them; it is not an easy task for users. Actually, from our observation, users do not need maps of their house for specifying and understanding the locations within their houses; people usually use semantic properties of

locations for doing so, such as “the living room” or “the bath room on the second floor”. Therefore, we decide not to include the map interface into the proposed system.

6.1 The USHAS Content Adaptation Sub-System

The overview of the USHAS Content Adaptation Sub-System (CAS) is shown in Figure 6-2. The steps of using it are described as follows: (1) User opens a Web page which is going to be customized. (2) User clicks a bookmark to download and execute the Page Tailor. (3-4) Using the Page Tailor to select and store the interesting parts of Web page. (5) Using the mobile device to browse Web page via Web Proxy. (6-9) Filtering out the un-selected parts and sending the selected parts of Web page to mobile device.

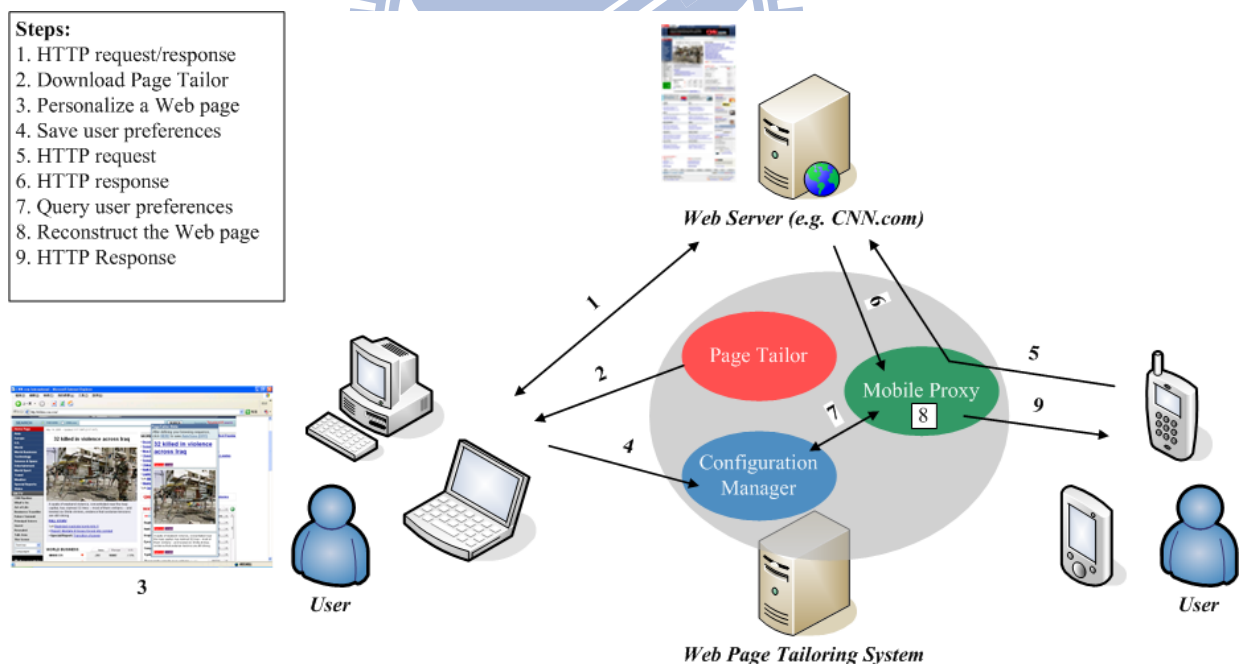


Figure 6-2. Overview of USHAS Content Adaptation Sub-System

The personalizing process in the USHAS content adaptation sub-system comprises two steps. First, user must specify his/her preferences of a web page using a PC or laptop. Second, he/she has to configure the browser on his/her mobile device to go through a specially made proxy, which is responsible for adjusting the content of web pages according to the preferences set in the first step. Two pictures are given below to illustrate separately the relationship between a user, our web page tailoring system, and a remote Web server (such as CNN.com) in each step.

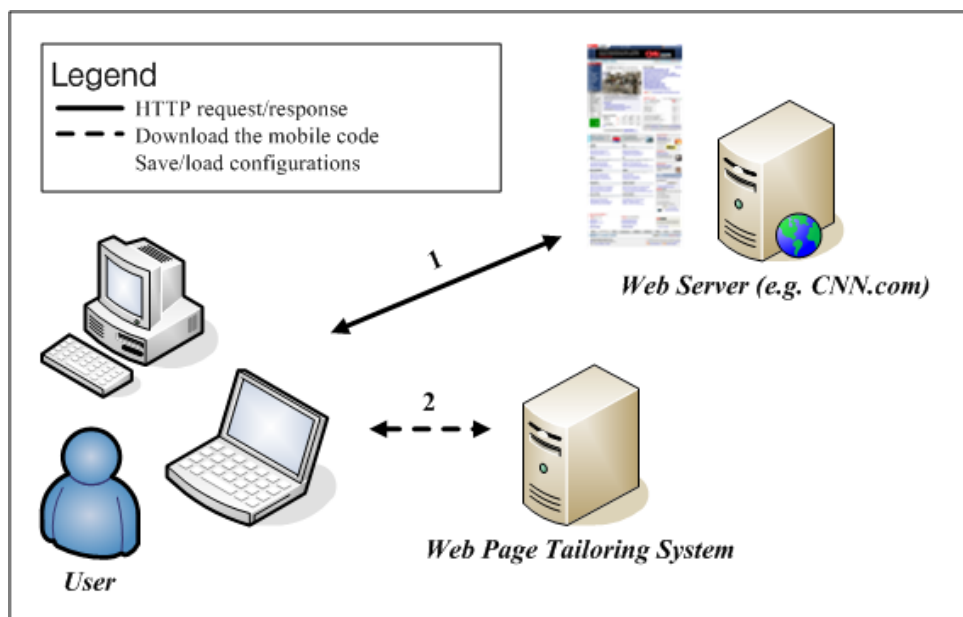


Figure 6-3. Personalize web pages using PC or laptop

Figure 6-3 describes the interaction in the first step. When a user enters a URL in his/her Web browser, a HTTP request is sent to (Line 1) the corresponding Web server specified in the URL. After processing the request by the server, a HTTP response is sent back (Line 1). If the user wants to personalize that page, a program hosted on a tiny Web server included in our system would be downloaded (Line 2) and executed in his/her browser. With the help of that program, the user can specify his/her preferences simply by visual manipulations. After

finishing the job, preferences about this page are sent back and stored in a database for later use (Line 2).

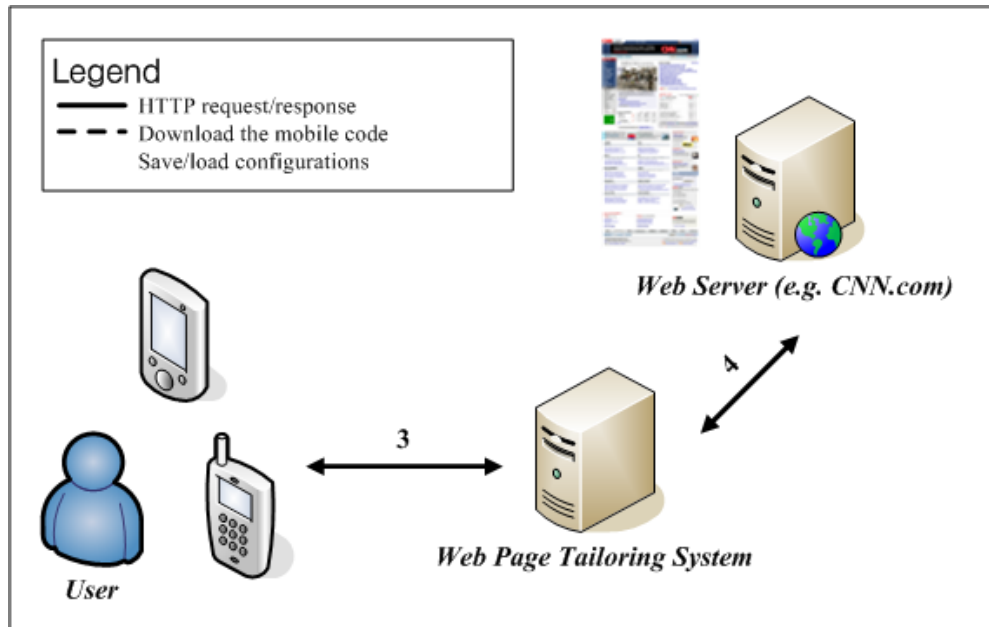


Figure 6-4. Browse web pages via mobile devices

Figure 6-4 pictures the interaction in the second step. Since the user would configure the browser on his/her mobile device to use a proxy included in our system, we would snoop each HTTP request and modify its corresponding response (Line 3 and 4) in between. For example, if the user visits a web page that has been personalized before, some actions would be taken to tailor the web page to meet the user preferences.

In order to achieve the above tasks, three components are designed in our system: Page Tailor, Configuration Manager, and Mobile Proxy. The purpose and functions of each component will be introduced separately.

a. Page Tailor

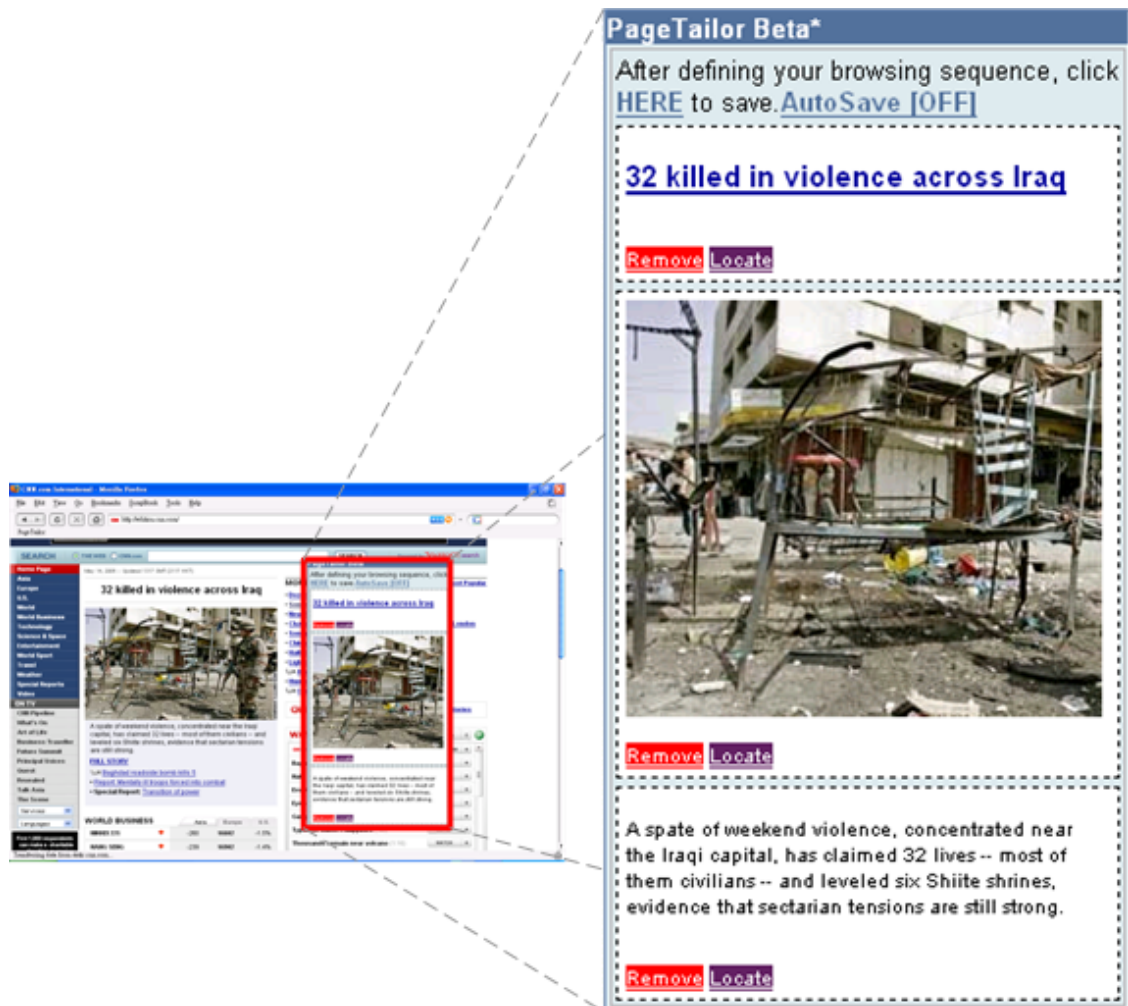


Figure 6-5. Page Tailor in Firefox Web browser

Page Tailor in the form of mobile code can be downloaded and executed in a user's browser when he/she is about to personalize a Web page. It provides some visual manipulations for users to help them specify their preferences about a Web page. The preferences here include: blocks of a Web page that should be retained and their final arrangement. All the preferences about this page would be saved in a remote database that is managed by Configuration Manager. Figure 6-5 is a snapshot when executing Page Tailor in a Firefox Web browser.

When browsing Web pages, a user can click on the installed bookmarklet to download and execute Page Tailor. From the other perspective of users, it seems that the Web page itself provides the personalizing functions.

After the Page Tailor window is launched in the user's browser, some actions are performed in the background automatically. First, Page Tailor will connect with Configuration Manager to retrieve the user preferences about this current visited page. If the user has personalized this page before, Page Tailor would retrieve the old preferences, and then use the data retrieved to reconstruct the past, such as blocks that had been selected and their order.

On the contrary, if there are no preferences about this page, nothing will happen, of course. The purpose of this action is to help users accelerate the setting time; particularly when he/she only wants to perform a slight modification.

In order to help a user specify his/her preferences about a Web page, Page Tailor provides some visual manipulations. Figure 6-6 demonstrates the feature that a user can select a block in a Web page at different granularity. For example, in the top half of this picture, a block containing more information than that in the bottom is selected. A selected block is highlighted in yellow.

Figure 6-7 illustrates another feature – drag and drop. In this picture, three views are shown from left to right. In the beginning, three blocks has already been selected (left). Next, we switch the last two blocks (middle), and then the final result comes out (right). The sequence of blocks in the Page Tailor window would be the same as that in the browsers on mobile devices.

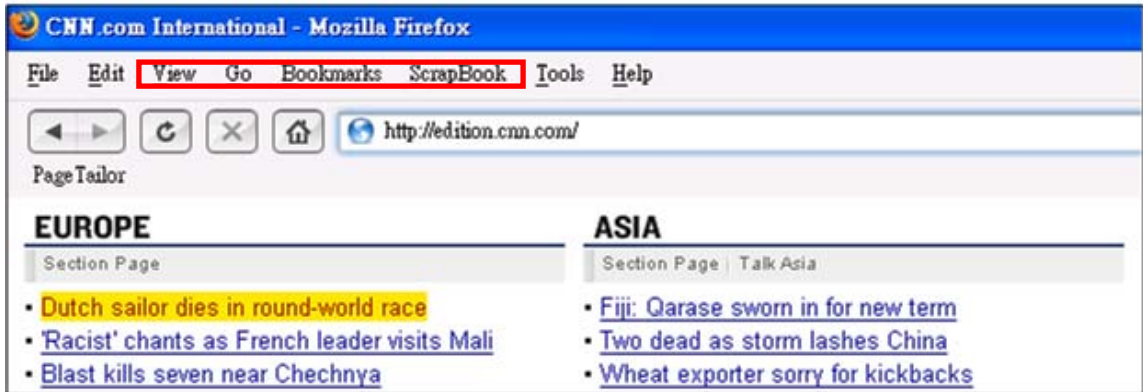
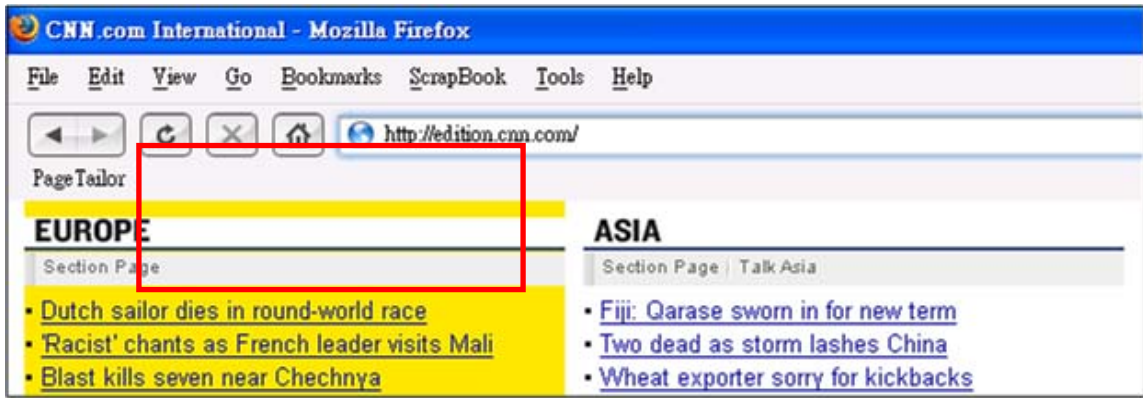


Figure 6-6. Select blocks at different granularity



Figure 6-7. Rearrange the selected blocks

XPath [72] is a language that describes how to locate specific elements in a document. This is the standard we adopted to store the user preference. In our Web Page Tailoring system, the user preferences stored are composed of XPath expressions. In other words, when a user adds a block to the Page Tailor window, Page Tailor would internally generate an XPath expression for that block. By using XPath expressions, we can uniquely identify this block in the future provided that if the layout of this page does not change too frequently.

As for the sequence of selected blocks, the XPath expressions of selected blocks are concatenated together according to their order in the Page Tailor window (separated by commas) to form the user preferences about this page. Figure 6-8 is a practical example.

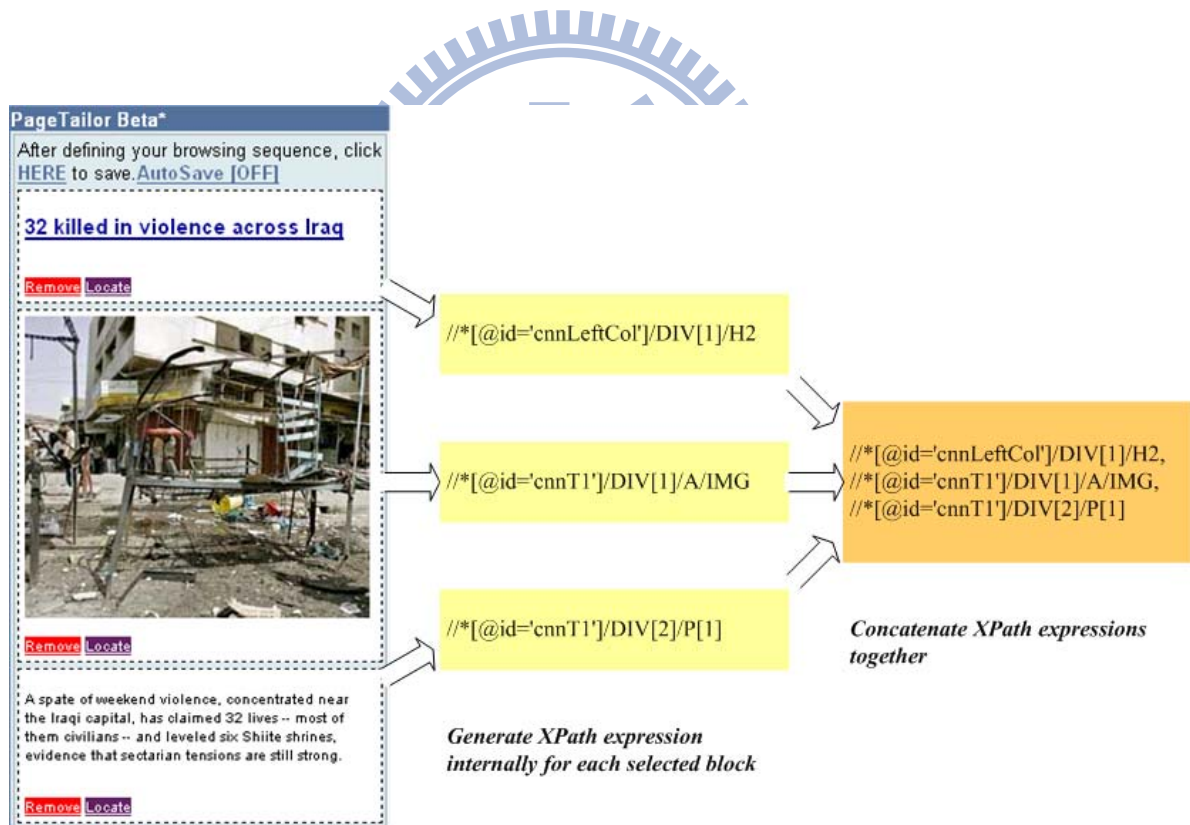


Figure 6-8. Internal expression of user preferences about a Web page

b. Configuration Manager

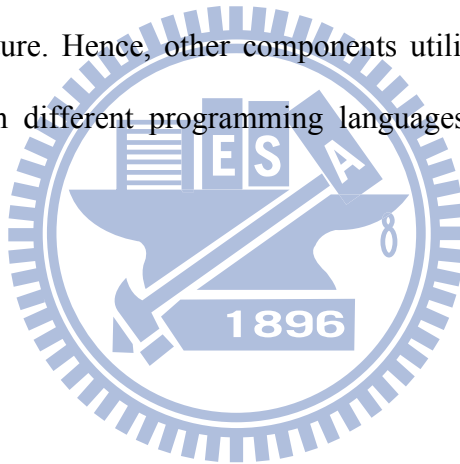
The preferences specified by a user will be stored in a database. Configuration Manager serves as a gatekeeper to control the access to the backend database. It provides a Web-based interface (Figure 2-9) for a user to manage his/her preferences base on the HTTP 1.0 Web protocol. Two Web services are also exported to allow other components in this system to access the preferences programmatically. One is used for querying the database and the other for updating.

The reason why we adopted the Web service approach ultimately is because of its language- and platform-specific nature. Hence, other components utilizing the services in our system could be implemented in different programming languages that are more appropriate for specific tasks.

c. Mobile Proxy

Mobile Proxy is a specially made proxy that is responsible for the final step in completing the personalizing process. It would monitor every HTTP request and makes Web service calls (query service) with the request URL as the parameter to Configuration Manager. The returned user preferences, if any, could then be employed to filter out the unwanted Web page content and rearrange the remaining blocks.

Figure 6-9 summarizes the communication between these three major components. Page Tailor is first downloaded and executed in a user's browser. Then the Web services exported by Configuration Manager are used by it and Mobile Proxy to access the backend database where user preferences are stored.



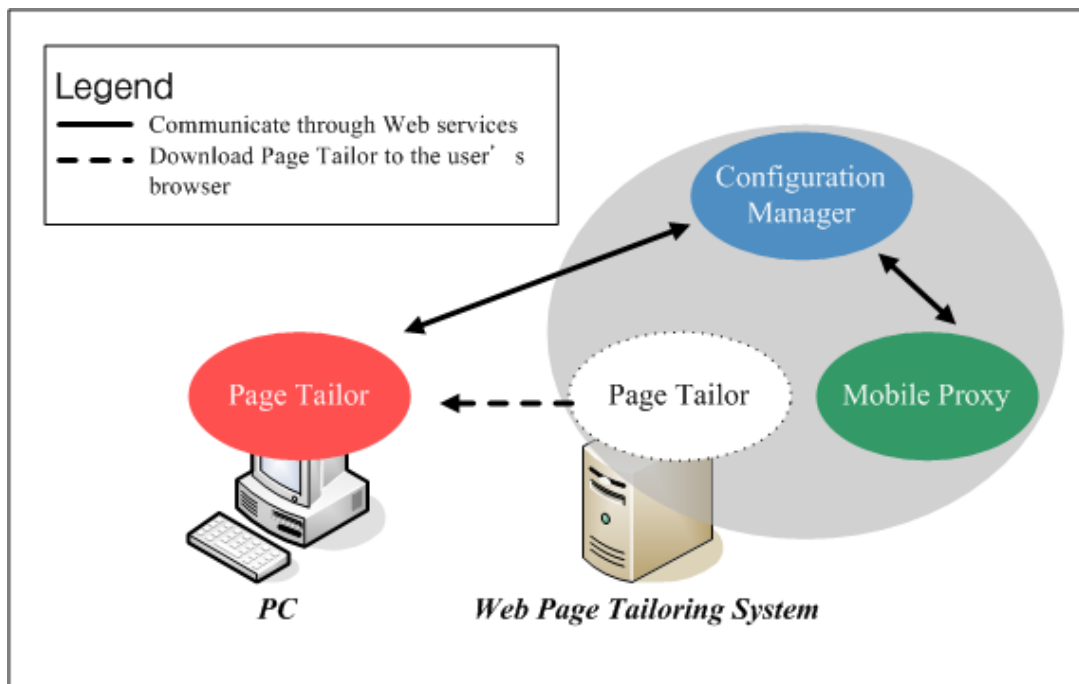


Figure 6-9. The communication between these three components

6.2 The Information Broker Sub-System

The high level system architecture of the Information Broker Sub-System (IBS) is shown in Figure 6-10. There are three main layers included: Application Market Layer, Offline Layer, and Content Adaptation Layer. The application market provides an interface for developers to publish their applications and users to install applications. The Offline Layer is mainly responsible for offline data management such as offline content preparation and offline data synchronization. If some external Web content are included into an application, the developer can use the components of Content Adaptation Layer to provide a simplified content representation. The detailed system architectures of the application market, Offline Layer, and Content Adaptation Layer are shown in Figure 6-11, 6-12, and 6-13 respectively.

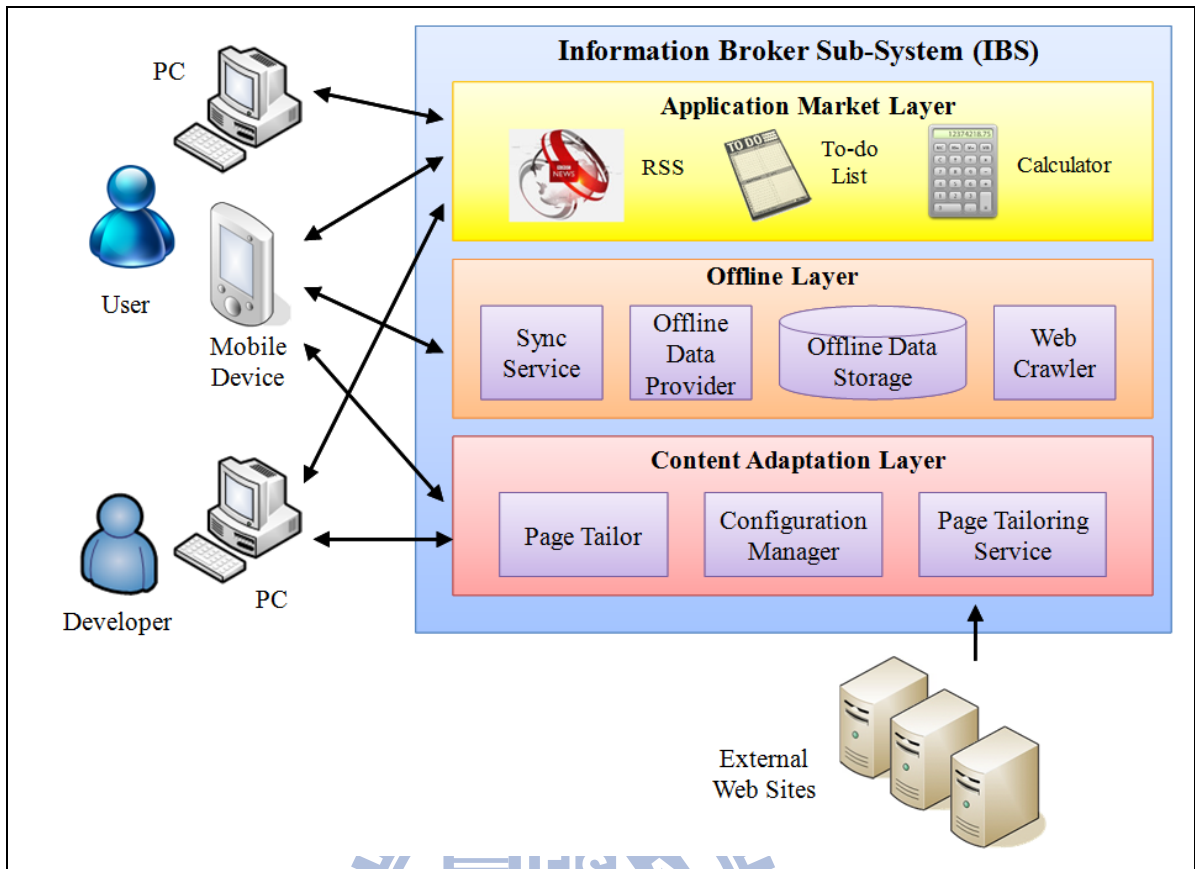


Figure 6-10. High level system architecture of the Information Broker Sub-System

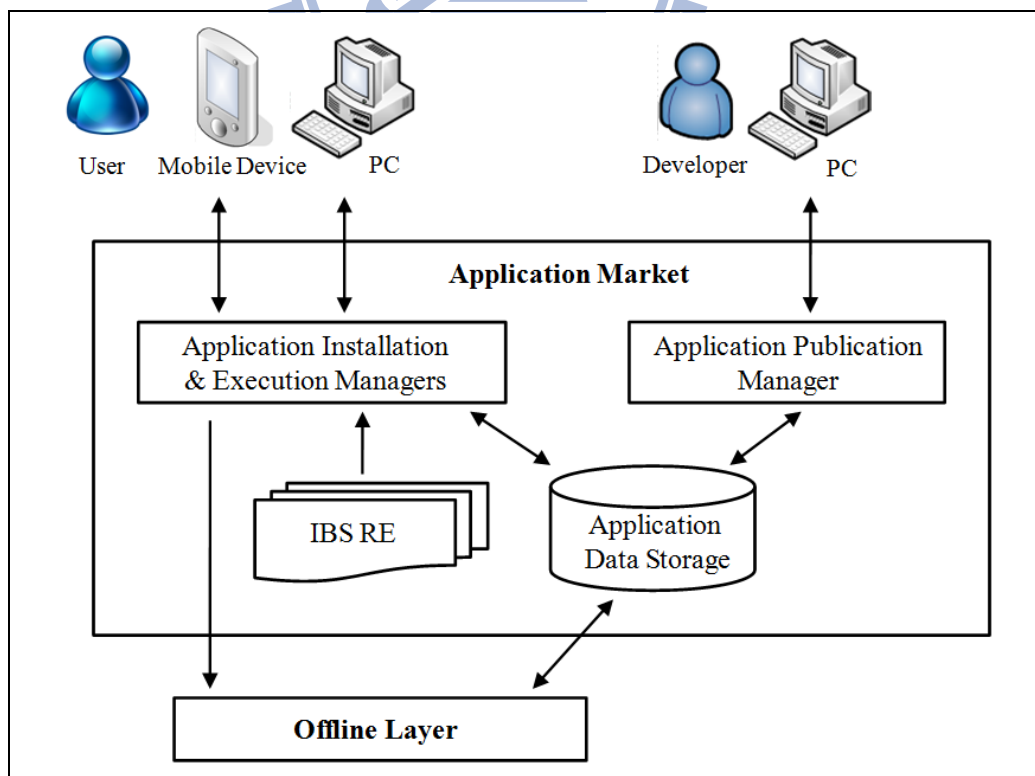


Figure 6-11. The detailed architecture of Application Market

Figure 6-11 shows the detailed architecture of the Application Market. During the application-uploading phase, first, developers can upload their gadgets through the application publication manager. Second, application-related files such as the gadget itself and images are stored in the application data storage. If developers do not wish to maintain an online database by themselves, the application data storage also provides a database for storing application-specific user data. During the application-downloading phase, first, users can go to the application market and select applications via the application installation manager. Second, this manager invokes the Offline Data Provider of the Offline Layer to prepare for offline data during the process of installation. Later, through the application execution manager, users can execute their applications on mobile devices or PCs either online or offline after downloading and executing the IBS Runtime Environment (IBS RE).

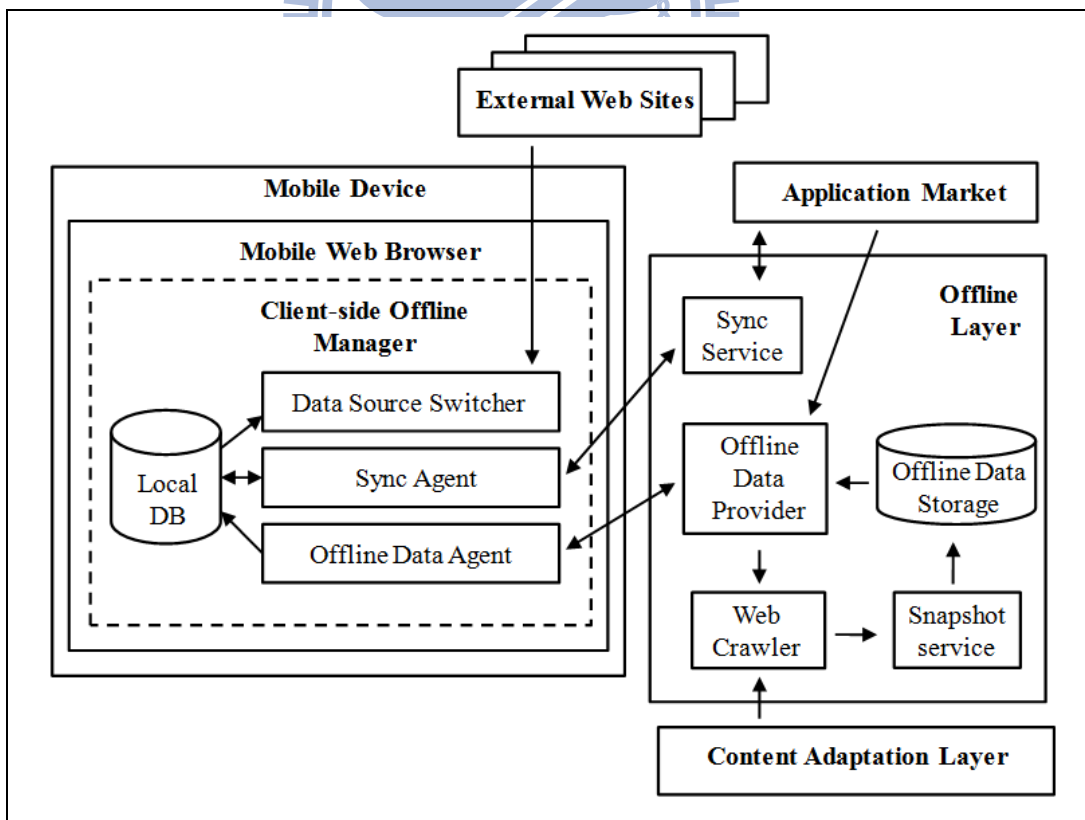
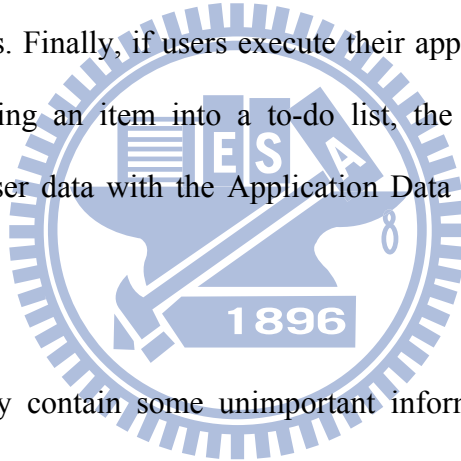


Figure 6-12. The detailed architecture of Offline Layer

In Figure 6-12, before using the Offline Layer, users must first install a Client-side Offline Manager on their mobile browsers. Currently, Google Gears and some JavaScript programs implement this manager. The Data Source Switcher component determines whether it is online or offline, and chooses external Web sites or the local DB as the data source. During the offline data preparation phase, first, the Offline Data Agent component invokes the Offline Data Provider to prepare and download offline Web content. Second, the Offline Data Provider executes a Web Crawler to acquire adapted external Web content if it is included in users' applications, and transforms this content into images by the Snapshot Service. The width of transformed images can be specified by developers so that they can be viewed clearly on mobile devices. Finally, if users execute their applications offline, and change the user data such as inserting an item into a to-do list, the Sync Agent and Sync Service synchronizes the local user data with the Application Data Storage after the mobile device returns to online mode.



Since Web pages usually contain some unimportant information, such as advertisements, browsing them on mobile devices is easier if only important content is selected to be displayed. Moreover, the structure of Web content usually tends to be static; otherwise, certain external services such as external RSS readers cannot extract Web content correctly. Therefore, the DOM tree paths are used to specify which content is important.

Figure 6-13 shows the detailed architecture of the Content Adaptation Subsystem. The Content Adaptation Layer of IBS provides an authoring tool for developers to specify which Web content from which Web pages are included in their applications. For example, developers want to create a RSS news reader application and provide links in an index page for linking to pages under a news Web site, they may want to specify that the title and main

body of article are important, and only display the content of them on users' mobile phones. With the Content Adaptation Layer, developers can do so by the following steps. First, they have to open the external Web page that they want to specify, and download the Authoring Tool onto the current browsers. In our implementation, the Authoring Tool is implemented in JavaScript; therefore, it is easy to download and execute this tool by clicking a bookmark. Second, with the help of the Authoring Tool, developers can specify which blocks, or DOM tree paths, are important for which applications, and save the selection results into the Configuration Database via the Configuration Manager. Also, developers can specify whether all the Web pages under the same domain should follow the same pattern of selection. Finally, if the user or the Offline Layer wants to obtain the content of these Web pages through this application, the Page Tailoring Service (a Web proxy) filters out unspecified content and provides adapted pages that contain only specified content.

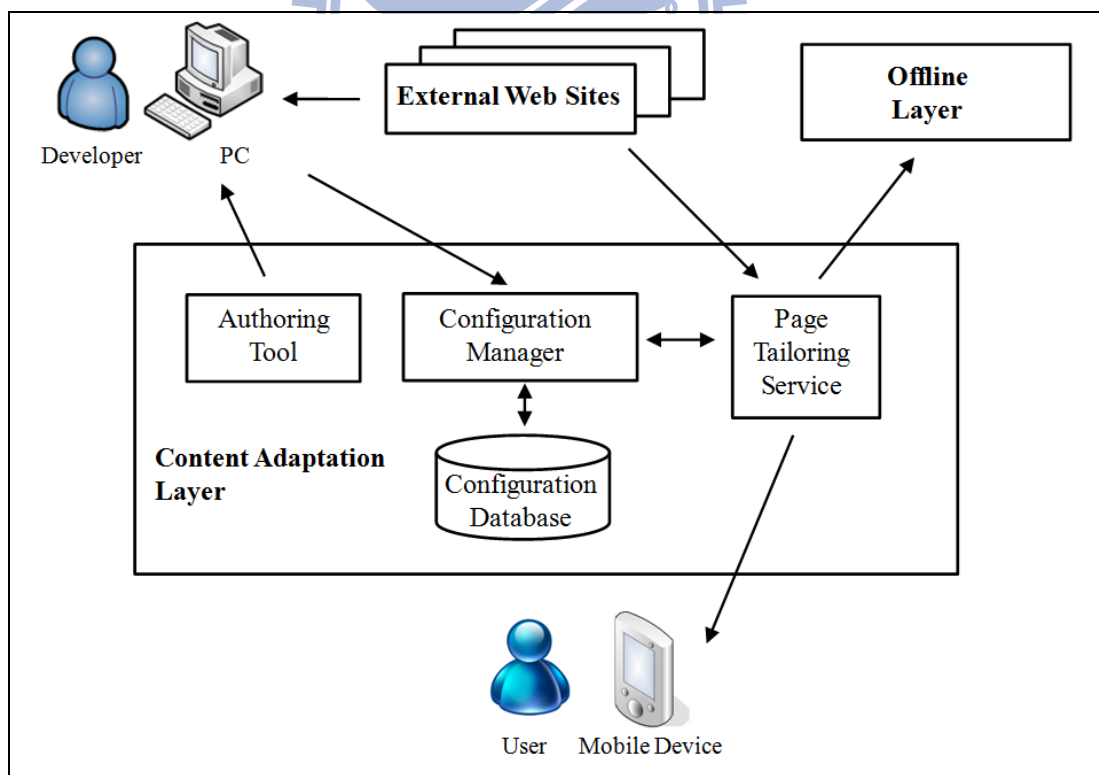


Figure 6-13. The detailed architecture of Content Adaptation Layer

6.3 The Offline Mobile Web Browsing Sub-System

To support offline Web browsing capability of USHAS, the offline mobile Web browsing sub-system is designed. Traditionally, Web pages will not be available without network connection; it is very inconvenient for users. In this research, we propose a framework, which makes Web browsing possible on mobile devices without any available network to the Internet. The overview of our framework is shown in Figure 3-1. There are three main components in our system: Original Web Sites (OWS), Offline Data Pre-Fetcher, and Mobile Device with Local Server. Web pages can be pre-fetched by Offline Data Pre-Fetcher, downloaded to mobile device when network connection still exists, and browsed by user even if there is no network connection anymore.

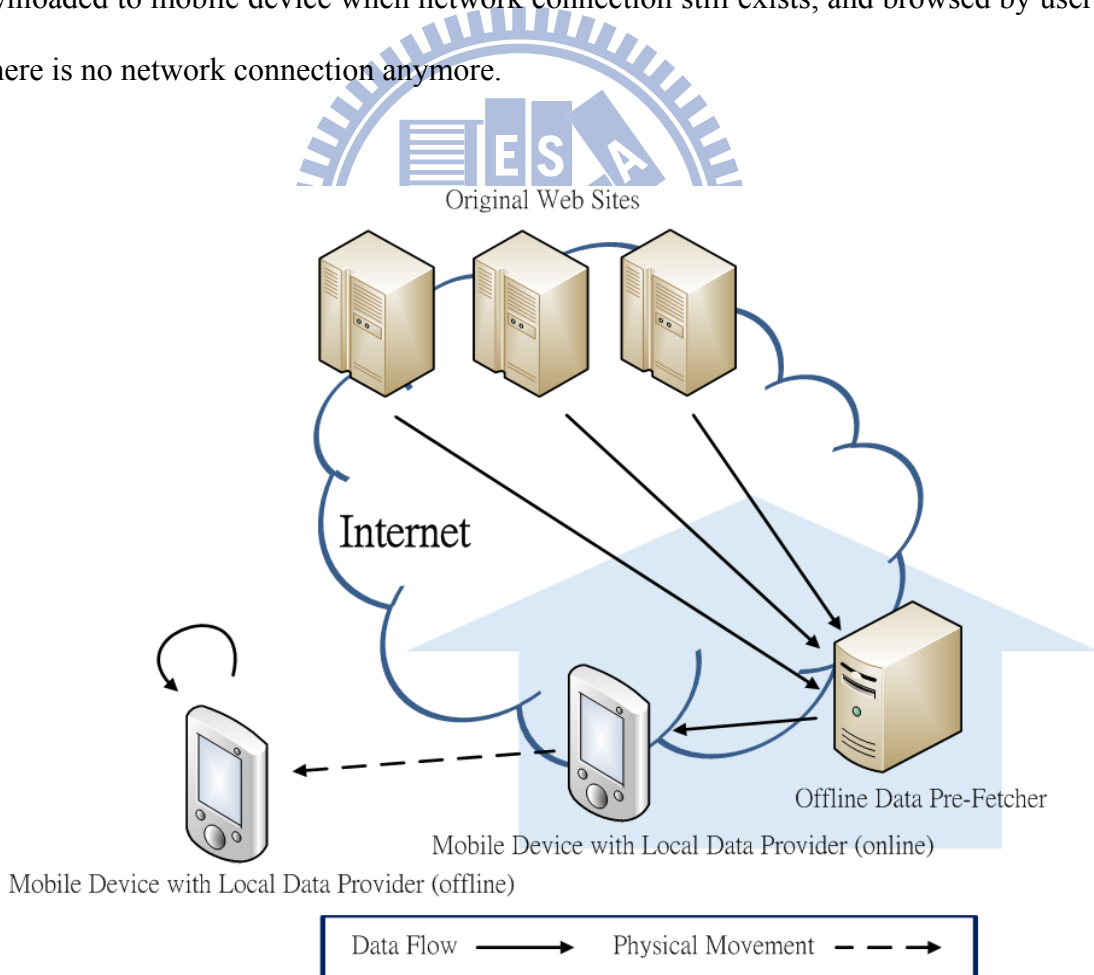


Figure 6-14 Overview of the Offline Mobile Web Browsing Sub-System

a. Original Web Sites

Original Web Sites are content providers providing Web page services on the Internet. Web documents provided by Original Web Sites can be either browsed from mobile device directly or pre-fetched by Offline Data Pre-Fetcher.

b. Offline Data Pre-Fetcher

The Offline Data Pre-Fetcher is responsible for offline data organization. It provides a management interface for users to specify which Web pages they are interested in. Based on the users' interests, the Offline Data Pre-Fetcher pre-fetches Web documents and related data and stores them for cache purpose. It also allows users to define how frequent these Web pages should be updated. After all, those organized Web pages are re-published for mobile device to access.

c. Mobile Device with Local Data Provider

When there is a network connection available, for example, if a user is going to leave his home, he can download the organized Web pages through Offline Data Pre-Fetcher. After that, even if there is no network connection on his way to office, Web pages are still accessible which are provided by the Local Data Provider on his mobile device.

Instead of preparing offline data directly by mobile devices, there is an Offline Data Pre-Fetcher playing the role of Web Proxy. There are several advantages of this design; first, users do not have to visit each original Web site respectively each time; it can be performed in a batch way; second, pre-fetched data can be updated more frequently even if there is no

connection of mobile device; finally, the Offline Data Pre-fetcher can be accessed and managed not only through mobile device, but also via desktop PC.

6.4 The Physical Access Control Sub-System

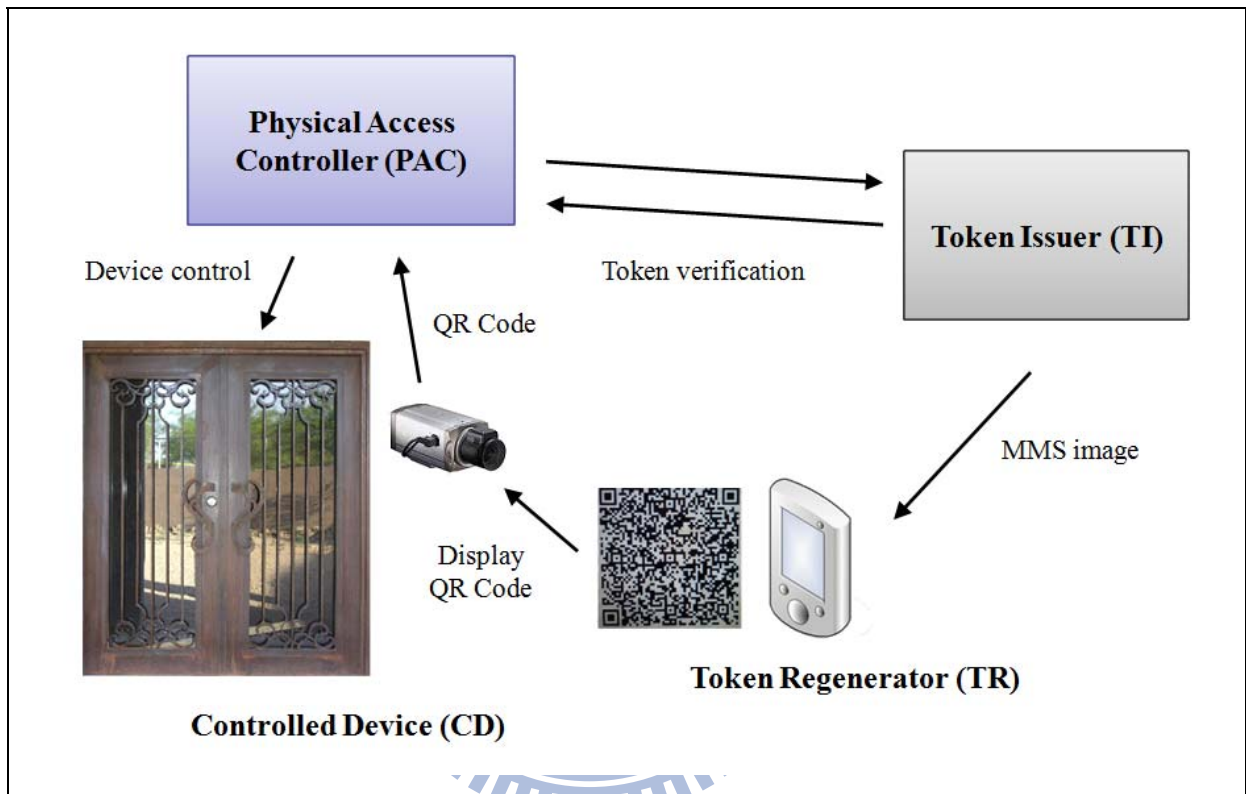


Figure 6-15. The Overview of Physical Access Control Sub-System (PACS).

Figure 6-15 shows the overview of Physical Access Control Sub-System (PACS). In Figure 6-15, when users want to enter the locations controlled by PACS, the TR regenerates the tokens which are sent from *TI* during registration phase at the first time, and stored on mobile devices. After the *TR* regenerates the tokens, encodes them into 2D barcode images, and displays these images on mobile screens, the users can present these images in front of the cameras which are bound to the controlled devices, such as doors and gates. Then, the *PAC* decodes the matrix code images, retrieves the tokens, and sends these tokens to *TI* for token verification. If the verification is successful, the *TI* generates new tokens for users, and sends







these tokens to them via MMS channel for their next visits. Finally, the *PAC* controls the controlled devices to open.

6.5 The Face Recognition Sub-System

To track the current locations of people, the face recognition technology is utilized. Moreover, the hand gesture recognition and passwords inputted by users are adopted to improve the recognition rate of face recognition.

The Face Recognition Sub-System (FRS) of USHAS maintains the mapping between the user faces and passwords, as shown in Table 6-1. It can be noticed that, different person can choose the same password, although it is rare. For example, the first person and the second person choose the same password “4398”; the third person and the fourth person choose the same password “3142”.

Table 6-1. Face and password mapping example

| | 1 | 2 | 3 |
|-----------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Faces |  |  |  |
| passwords | 4398 | 4398 | 3142 |
| | 4 | 5 | 6 |
| Faces |  |  |  |
| passwords | 3142 | 3933 | 5915 |

We integrate the password with the recognition engine to improve recognition rate. The integration point takes place at the final stage of face recognition engine. The integration process is shown in Figure 6-16. First of all, we take input password, and check the available face password mapping. If this password is not available for any candidate, such candidate will be eliminated before the final stage of recognition process. After that, the remaining processes of traditional face recognition are conducted.

For example, in Figure 6-16, the average correctly guesses rate is 1/5 without the elimination. On the other hand, after integration, the average correctly guesses rate is 1/2, which is 2.5 times higher than the previous one. It is the main idea that how the face recognition rate can be improved. The recognition rate can be enhanced if the error recognized face symbol is eliminated.

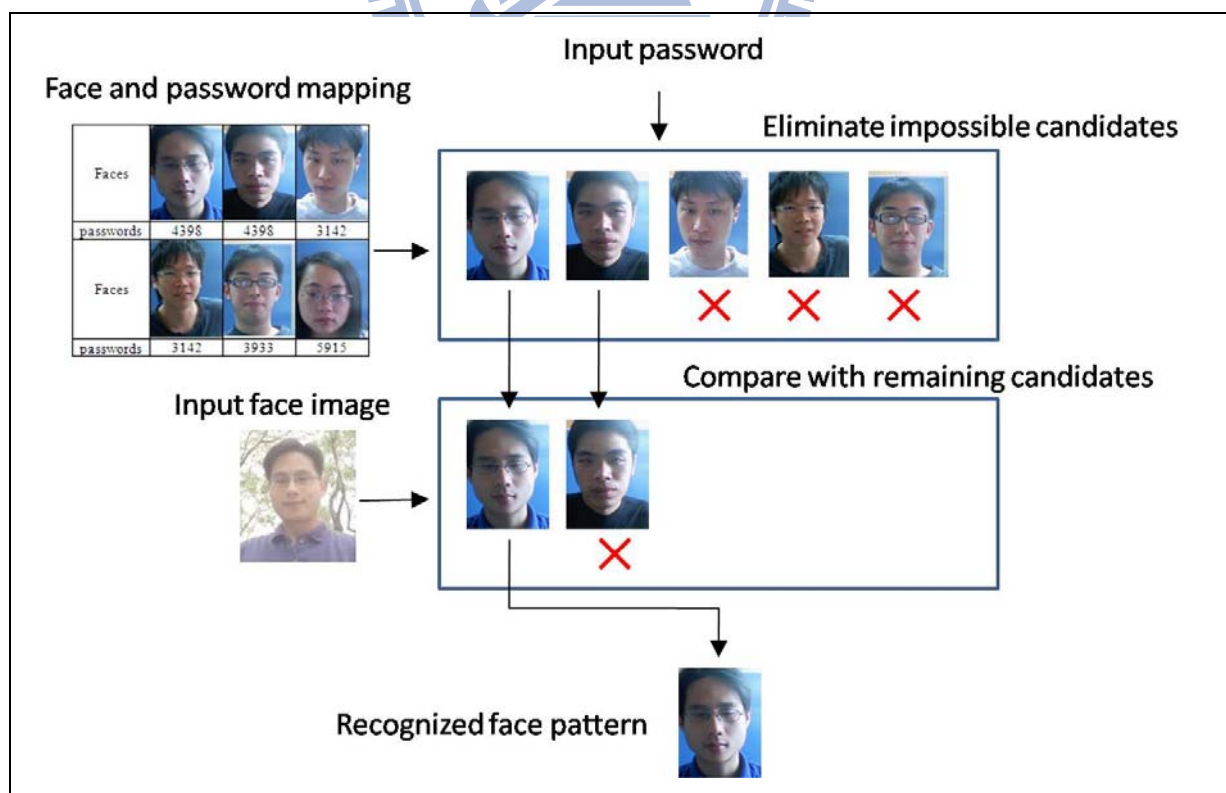
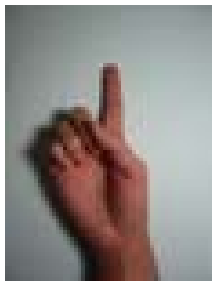





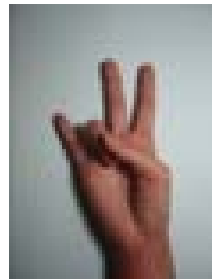
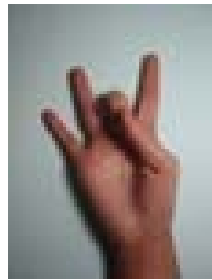
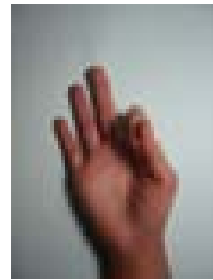


Figure 6-16. Integration process of face and password

The face recognition sub-system also utilizes the recognition result of hand gesture to improve the recognition rate of face recognition. In this system, the input hand gesture and face images are extracted first. After that, these two images are processed by the hand gesture and face recognition engine partially simultaneously. After the hand gesture recognition result is produced, the face recognition engine eliminates the impossible candidates based on the recognized hand gesture dynamically, and figure out which is the recognized person. Finally, we check that is this person permitted to reach the floor indicated or not based on his hand gesture. If he is permitted, then the elevator will bring him to this floor, otherwise, the elevator takes no action.

Table 6-2. Nine hand gesture patterns






























| | | | | |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 1 | 2 | 3 | 4 | 5 |
|  |  |  |  |  |
| 6 | 7 | 8 | 9 | |
|  |  |  |  | |

We assume that the security elevator can reach from the 1st floor to the 9th floor. Hence, we defined nine hand gesture symbols, from one to nine, based on the American Sign Language [10]. The hand gesture symbols are listed in Table 6-2. On the other hand, the permitted floor

number for each person is based on the face and hand gesture mapping. For example, Table 6-3 is the mapping we generate randomly. Here, we assume that the 1st floor is available for everyone.

The integration point takes place at the final stage of face recognition engine. The integration process is shown in figure 4. First of all, we take the recognized hand gesture as input, and check the available hand gesture mapping. If this hand gesture is not available for any candidate, then this candidate will be eliminated before the recognition. After that, the remaining processes of traditional face recognition are conducted.

Table 6-3. Face and hand gestures mapping example

| Face | Available Hand Gestures | | | | | |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
|  |  |  |  |  |  | |
|  |  |  |  |  |  |  |
|  |  |  |  |  | | |
|  |  |  |  |  |  |  |
|  |  |  |  | | | |

Take Figure 6-17 for example, without the elimination, the average correctly guess rate is 1/5. On the other hand, with the elimination method, the average correctly guess rate is 1/2, which is 2.5 times higher than the previous one. This is the main idea that how the face recognition rate can be improved.

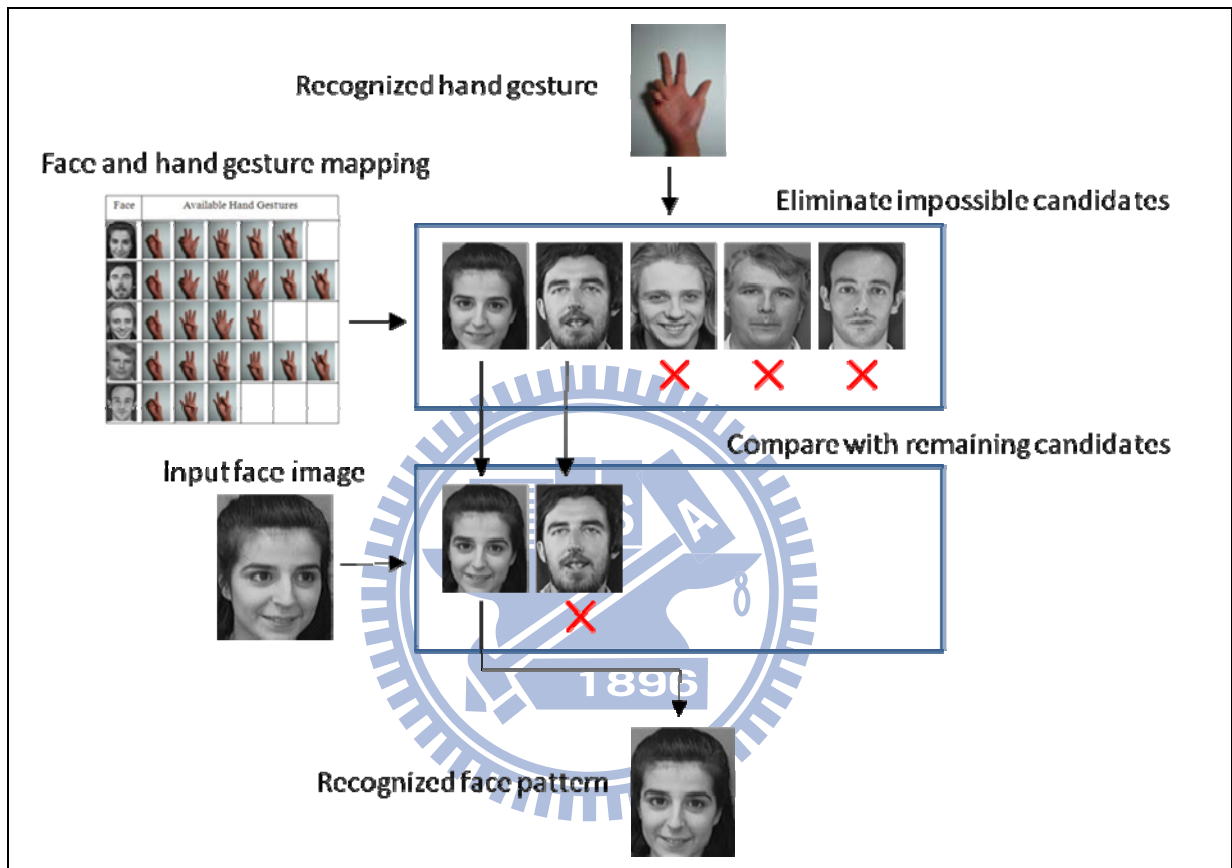


Figure 6-17. Integration process of face and hand gesture

However, there are still risks of this elimination method. For example, in Figure 6-18, if we incorrectly eliminate the expected face candidate, the result of the face recognition must be wrong. This kind of error can be resulted from the wrong recognition result of hand gesture recognition engine, or the hand gesture is in fact unavailable for this person. Hence, the first recognition method chosen to conduct the elimination is very important. We choose hand gesture recognition to be the first recognition method, because it is much easier to be a high recognition rate method.

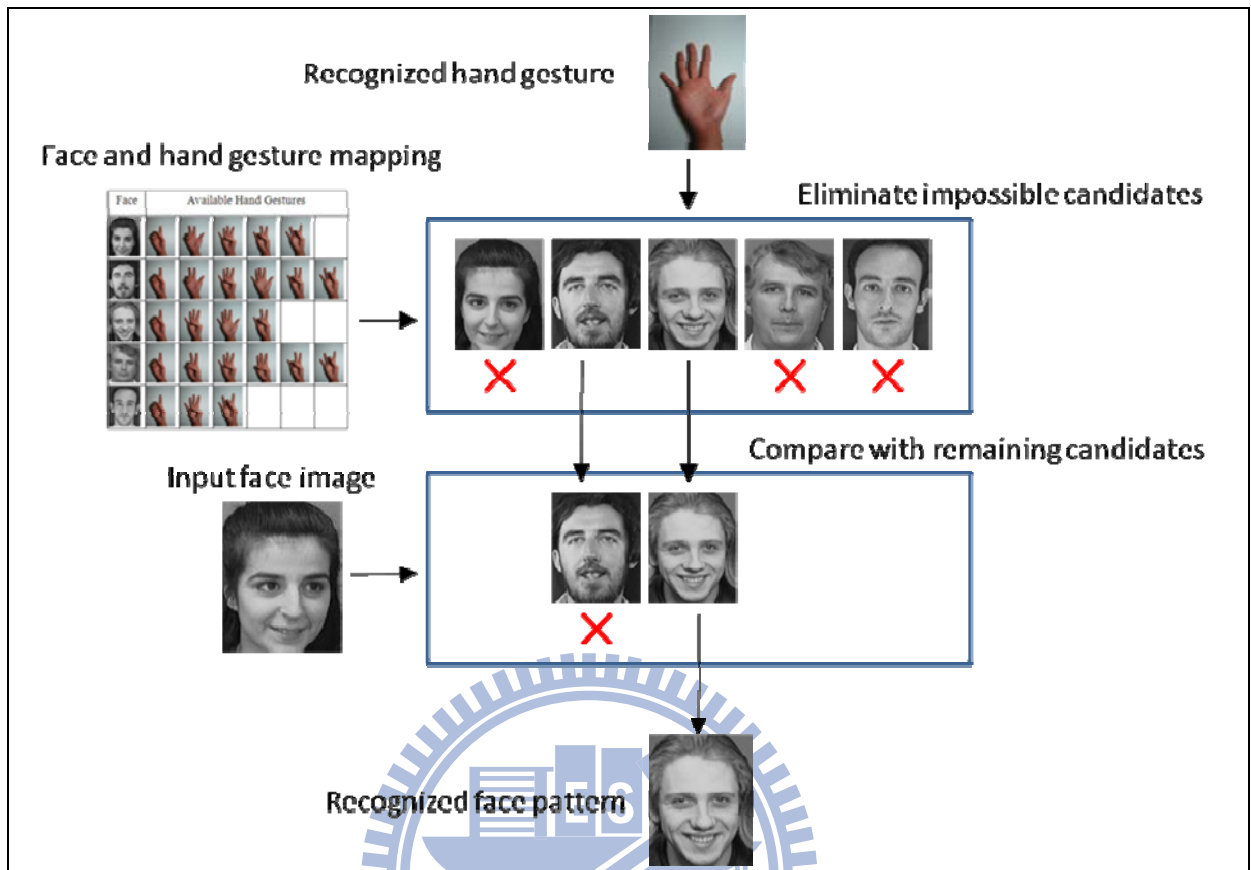


Figure 6-18. Integration process with error

Chapter 7. Detailed Design

7.1 System Model and Detailed Design of USHAS

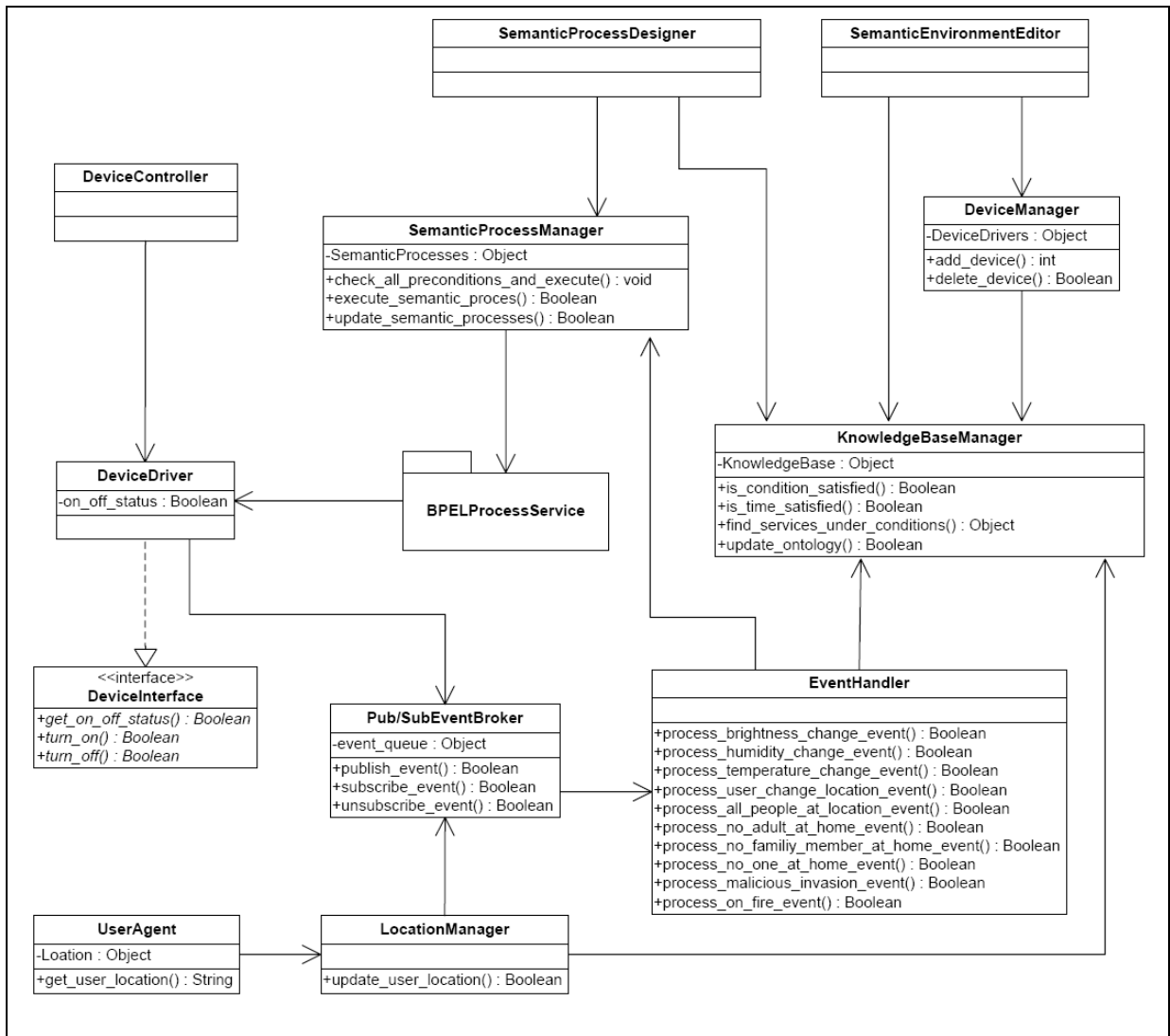


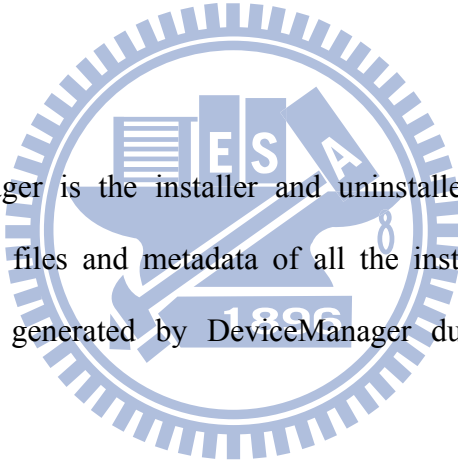
Figure 7-1. System model

The system model is shown in Figure 7-1. All the device drivers must implement the three methods (`get_on_off_status`, `turn_on`, and `turn_off`) defined in the `DeviceInterface` interface. For each category of appliance, an interface is also defined for different functionalities of that category, such as the “`set_channel`” method defined in the `TVInterface` interface. To simplify

this figure, numerous detailed definitions of these categories and functionalities are not included.

A LocationManager class is designed for translating physical location information into semantic information. This class can be implemented differently based on different location detection systems. For example, if an RFID tag is deployed in each room to transmit the ID of the room, and a table is designed to store the mapping between this ID and the semantic name of this room, then the LocationManager can be implemented as the manager of this table. For another example, if someone is recognized by a face recognition system from a camera, and this camera is located at the living room, then this person can be realized to be in the living room too.

Finally, the DeviceManager is the installer and uninstaller of device drivers; this class manages all the physical files and metadata of all the installed drivers. Also, an OWL-S service is automatically generated by DeviceManager during each operation of driver installation.



7.2 Detailed Design of CAS

In Figure 7-2, almost all of the interfaces are *iw*, i.e. HTTP1.0 Web interface. Because our system is designed for Web application, the *iw* interface enable us to develop different system components by different programming languages or platforms. Hence, it provides cross-platform communication for system development.

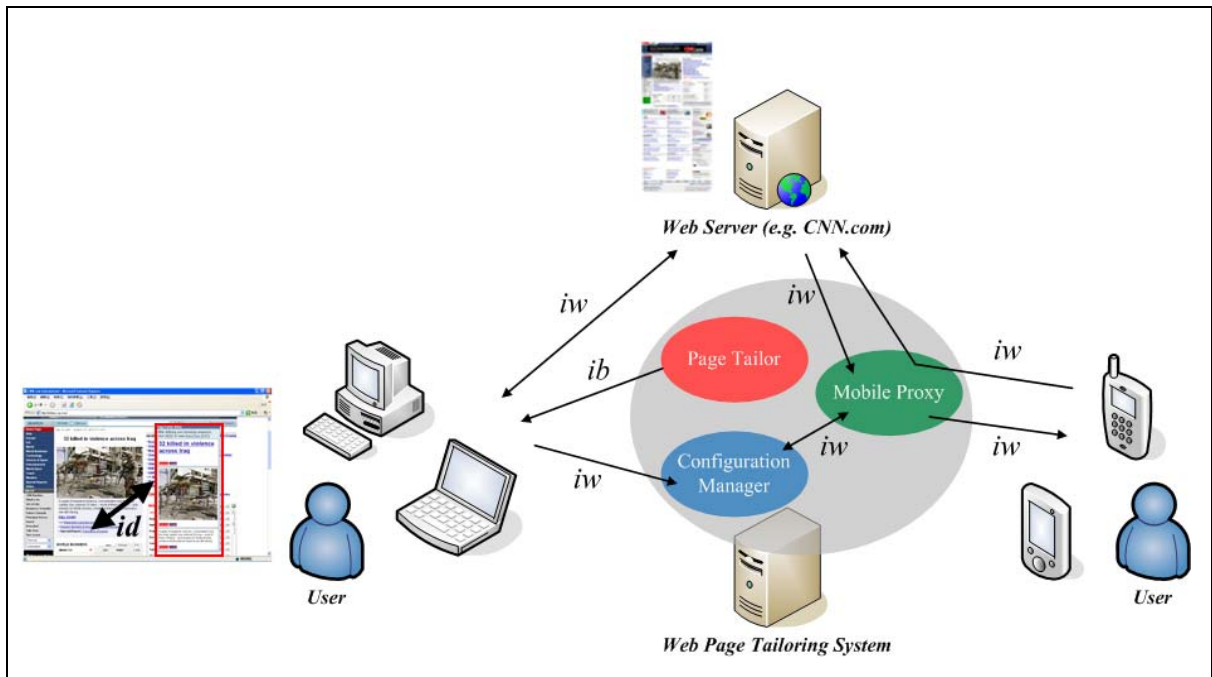


Figure 7-2. System components and their interfaces of CAS.

The Page Tailor is designed to be implemented in JavaScript because of the following considerations.

1. JavaScript was designed to add interactivity to HTML pages. This point conforms to the first objective of this thesis, i.e. easy-to-use, since we can embed the configuration tool in a Web page, so that users can configure their settings directly in the browser.
2. By using JavaScript, we can manipulate a Web page through the DOM interface. DOM provides the interface between the Javascript and original Web pages. In other words, we can change the appearance of a Web page to reflect the user's choice. This one conforms to the second objective, i.e. personalizing Web page visually.
3. JavaScript is one of the most popular scripting languages on the Internet, and works in all major browsers, such as Internet Explorer and Firefox. This one agrees with the third objective, i.e. support for mainstream browsers.

The interface between user and Page Tailor is bookmarklet. That is, the interface between user and Page Tailor in Figure 7-2. The source code of this bookmarklet is shown in Figure 7-3. When a user clicks that, an element with tag name “script” is created (Line 2). After being created, the new element contains no properties. The “src” property (Line 5) is employed to indicate the location of a remote JavaScript file. A created element exists only in the browser’s memory, and not as part of this Web page. In order to let the “script” element take effect, we have to add it to the Web page. For that reason, we find the first occurrence of “body” element (Line 6) in the Web page, and then append the “script” element as a child of it (Line 7).

```
01 javascript:
02 void(s=document.createElement('script'));
03 void(s.type='text/javascript');
04 void(s.id='tk_bookmarklet');
05 void(s.src='http://localhost:3000/javascripts/loader.js');
06 void(b=document.getElementsByTagName('body')[0]);
07 void(b.appendChild(s));
```

Figure 7-3. This is the bookmarklet used to launch Page Tailor

In this manner, we can dynamically load an external JavaScript file on demand to do something with a Web page. It can also work for other types of files such as CSS. JavaScript has been used in many applications. Some open source projects have already incorporated the commonly used functions such as drag-and-drop and visual effects. In Page Tailor, we do not reinvent the wheel but take advantage of the Script.aculo.us JavaScript framework.

Although we can dynamically load these JavaScript files, the exact time spent on loading, however, depends on the browsers. In other words, even if we add the “script” elements to a Web page in some order, which is not equivalent to load the first javascript file and then the second one.

To address the above problem, we must postpone the loading of scriptaculous.js until after prototype.js has been loaded. By selecting a unique identifier from the file (prototype.js or scriptaculous.js) and checking periodically whether the identifier has been defined or not, we can indirectly deduce the current state of that file. In other words, an identifier has been defined, which means that the file containing it must have been loaded, too. For example, we choose the identifier named Prototype in prototype.js and Draggable in scriptaculous.js.

After pagetailor.js is loaded, it would create the containment hierarchy of elements that together compose the Page Tailor window, and the entire hierarchy would be appended to the Web page. Figure 7-4 shows the correspondence between each element and their appearance displayed on the browser. The tag name of each element contained in this hierarchy is labeled on the left-hand side of the figure.

The DOM tree and XPath standards are used to store the user preferences. Page tailor extract the XPath expressions of blocks which users interest in on Web pages, and store these XPath expressions to the Configuration Manager. The degree of support for XPath over HTML varies in different types and brands of browsers, so we must deal with the conversion between a selected block and its corresponding XPath expression by ourselves in Page Tailor. The XPath expressions contained in the user preferences are restricted to internal use only. Then, we will make a statement on how to generate the XPath expressions of a selected block.

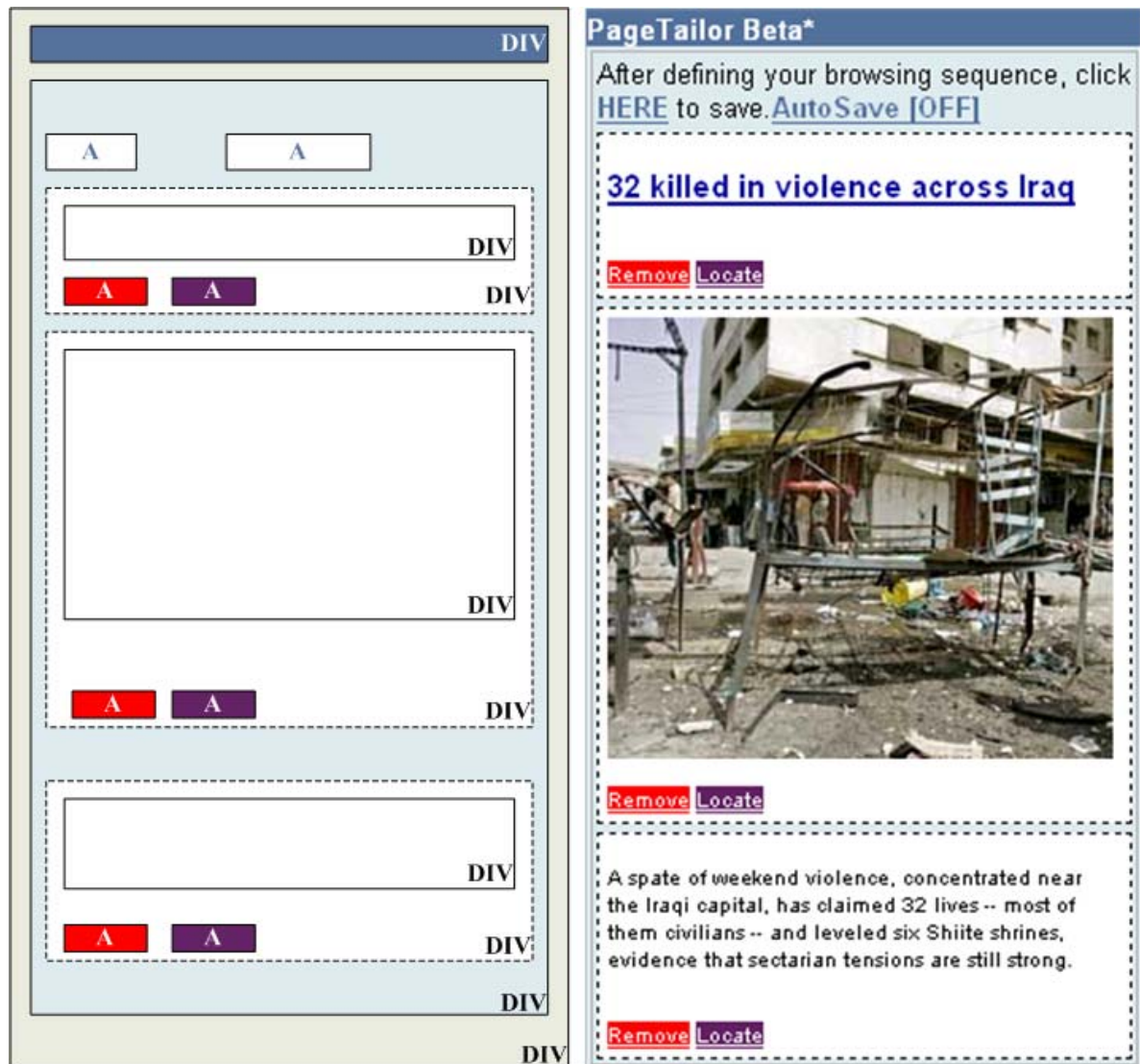


Figure 7-4. The containment hierarchy of elements

As shown in Figure 7-5, every selected block corresponds to a subtree of the document. The way to find its XPath expression can be divided into two steps. First, start from the selected block (or node) and walk recursively up the tree to find all of its ancestors. Then, for each node in the path count the number of its siblings that have the same tag name as that of the node. With these two data, we can finally generate the XPath expressions of the selected block.

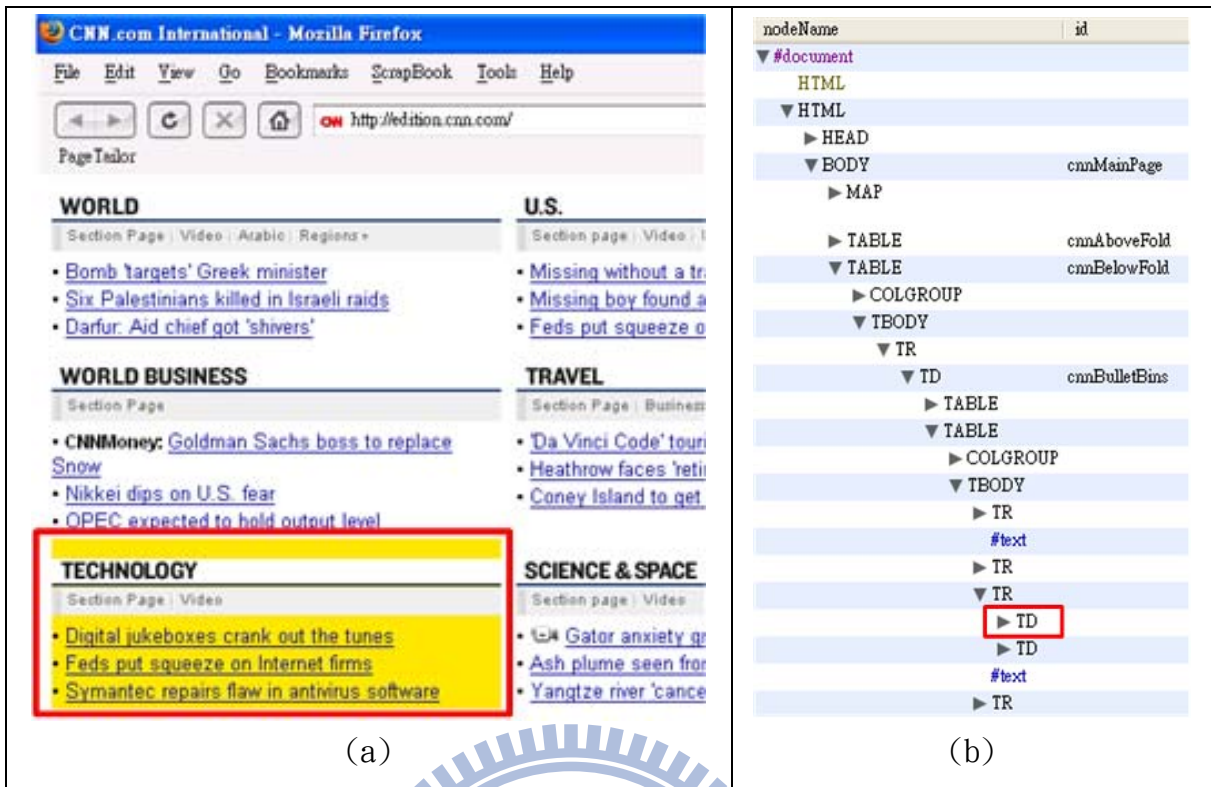


Figure 7-5. The selected block (a) and its corresponding sub-tree (b)

It is the XMLHttpRequest object that can be used by JavaScript to transfer data to and from a Web server using HTTP. It is supported by almost all popular browsers. For security reason, however, browsers often impose some restrictions (also referred to as the “same origin policy”) on this object.

The “same origin policy” dates from Netscape Navigator 2.0. This policy prevents a document from one origin from getting or setting properties of another document from a different origin. In general, two documents are considered to have the same origin if they are identical in three aspects: protocol, host, and optionally port. Table 7-1 lists some examples of origin comparisons with the URL “http://store.company.com/dir2/other.html”.

Table 7-1. Examples of origin comparisons

| URL | Result | Description |
|-------------------------------------------------|---------|---------------------|
| http://store.company.com/dir2/other.html | Success | |
| http://store.company.com/dir/inner/another.html | Success | |
| https://store.company.com/secure.html | Failure | Different protocols |
| http://store.company.com:81/dir/erc.html | Failure | Different ports |
| http://news.company.com/dir/other.html | Failure | Different hosts |

From the comparisons in the above table, it is not difficult to imagine that in Page Tailor, it is impossible to retrieve data from Configuration Manager using the XMLHttpRequest object, because a Web page surely comes from the Web server that is situated in a different origin from where Configuration Manager might live. Hence, we do not use the XMLHttpRequest of AJAX standard in Page Tailor.

Using the method, we can overcome the restriction imposed by the “same origin policy”. In the Page Tailor bookmarklet, we dynamically create a “script” element, assign some attributes to it, and add this element to the Web page. After the remote file has been loaded, the JavaScript code written in that file would be executed accordingly. Hence, the limitation of “same origin policy” of communication between the Page Tailor and Configuration Manager can be overcome by the dynamical javascript loading mechanism in HTTP 1.0 standard.

However, the value assigned to its “src” attribute does not point to a remote file. It points to the query service provided by Configuration Manager. Therefore, adding this element to a Web page is equivalent to using the query service. Moreover, since the type attribute of the “script” element has a value of “text/javascript”, the response of the query service should be in the form of JavaScript, or some runtime errors might occur. As for how to make use of the response of query service, we can see Figure 7-6.

```

01 query(current url, callback);
02
03 function query(current_url, callback_function_name){
04   var req = document.createElement("script");
05   req.type = "text/javascript";
06   req.id   = random number
07   req.src  = "http://.../.../query_service?url=" + current_url
08             + "&callback=" + callback_function_name
09             + "&id=" + req.id;
10
11   Add the element to the web page
12 }
13
14 function callback(service_response, element_id){
15   Process the response(i.e. service_response)
16   Find and remove the element with the specified value (i.e.
17   element_id) of the ID attribute
18 }

```

Figure 7-6. How to load the user preferences (pseudo-code)

The pseudo-code is taken from pagetailor.js. When we need the user preferences of a Web page, the function named query would be invoked (Line 01). What the function does is as described in the last paragraph. There are three parameters in the query string: url, callback, and id. When Configuration Manager receives a request, it would extract the values of each parameter. Among them, the value of url is employed to query the database to find the corresponding user preferences about this Web page. The value of callback contains the name (“callback” in this case) of a function to process the response of query service. And id, the last one, is utilized to identify the “script” element that makes this request, since more than one “script” element could be added to a Web page at the same time.

As mentioned before, the response will be interpreted as JavaScript. What Configuration Manager returns is actually a function invocation like the one in Figure 7-7. The function name, “callback”, is identical to the one involved in the query string. Two parameters are passed to this function. The first one contains the user preferences about the URL specified in the query string, the second one is the id specified in the query string as well, which is left intact.

```
callback(user preferences, id specified in the query string);
```

Figure 7-7. The response of query service (pseudo-code)

When the response is sent back to the client side, the function named “callback” would be invoked with the user preferences filled by Configuration Manager. Since Page Tailor has obtained the user preferences, the reconstruction of the past could be performed in the body of this function. Finally, it comes to the id’s turn. We can use the id to find the “script” element that was added to the Web page to trigger this series of events and remove it. Here the callback invocation is sent back from Configuration Manager to the Page Tailor through the *iw* interface again.

The above descriptions illustrate how to load the user preferences without being restricted by the “same origin policy”. Figure 7-8, on the contrary, illustrates how to update the user preferences. It is not surprising that these two operations are very similar. They differ only in two respects. First, one more parameter is involved in the query string, which is the data required to update. Second, the callback function only serves as a scavenger to remove the “script” element.

```

01 update(current url, user preferences, callback);
02
03 function update(current_url, user_preferences,
callback_function_name){
04   var req = document.createElement("script");
05   req.type = "text/javascript";
06   req.id   = random number
07   req.src  = "http://.../.../update_service?url=" + current_url
08             + "&prefer=" + user_preferences
09             + "&callback=" + callback_function_name
10             + "&id="+req.id;
11
12   Add the element to the web page
13 }
14
15 function callback(service_response, element_id){
16   Find and remove the element with the specified value (i.e.
17   element_id) of the ID attribute
18 }

```

Figure 7-8. How to update the user preferences (pseudo-code)

We adopt Ruby on Rails to implement the Configuration Manager. Controllers are the subprograms in a Rails application that performs tasks. Controller actions are sequences of Ruby code that correspond directly to the tasks this application can be asked to do. When an action is executed, it not only has access to the data from a submitted form but also to the models. In our system, two Web services exported by Configuration Manager are implemented as separate actions. That is, `query_service` and `update_service`. These two interfaces enable the Page Tailor to manipulate the data stored in the backend database.

Mobile Proxy in our system is developed from Muffin. We can modify the content of the Web page before it is sent back to the client according the user preferences specified by the user.

The interfaces in the filter logic class depend on what kind of filtering we intend to perform. There are two interfaces designed: RequestFilter and ReplyFilter. The RequestFilter interface is to filter requests before they are sent to a server. On the contrary, the ReplyFilter is to modify replies after a server responds. These two interfaces follow the HTTP 1.0 standard of *iw* interface.

If a request issued by the client is destined for a URL that has corresponding user preferences stored in our database, Mobile Proxy would start a chain of processing steps. First of all, it would use a HTML parser, NekoHTML to parse that Web page and produce a corresponding DOM tree (Figure 7-9). After that, an XPath engine, Jaxen is employed to process XPath queries over the DOM tree. Some specific elements would be selected at this time, and a new DOM tree would be created alongside to hold the replicas of these elements (Figure 7-10). The sequence of the replicas in the new DOM tree would refer to the sequence of the XPath expressions in the user preferences.



Figure 7-9. Produce a corresponding DOM tree

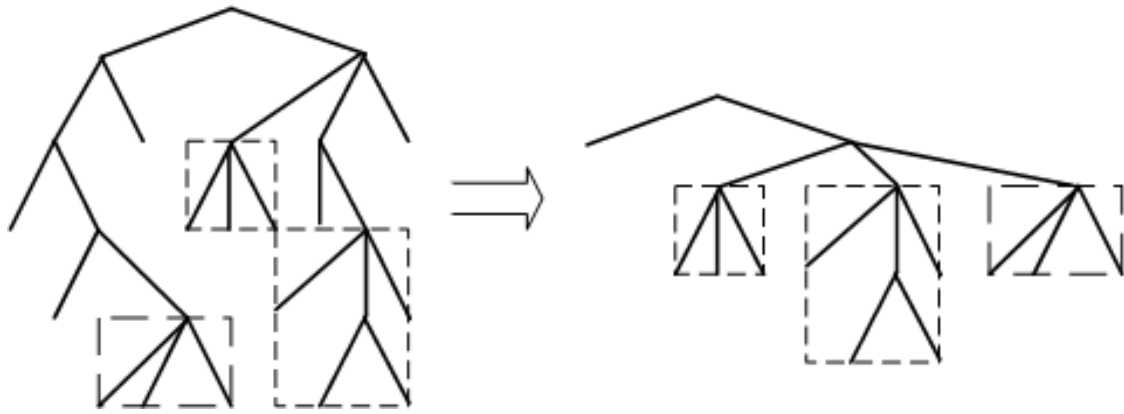


Figure 7-10. A new DOM tree (right) would be created to hold the replicas of selected elements in original DOM tree (left)

The result page (i.e. the new DOM tree) has already been constructed now. However, it is not necessarily self-validating because what we have done already is to find, clone and collect the blocks of content directly. To further polish that, an open source utility, JTidy, would be used. This tool was originally designed to fix mark-up errors and also offers a means to convert existing HTML content into well-formed XML, such as XHTML. After the process of JTidy, a well-formed and validated result page is finally turned up.

7.3 Detailed Design of IBS

Figure 7-11 shows the sequence diagram of how developers can publicize their applications. After developers create their applications, they can upload the gadget XML and related images to our application market. For each uploaded gadget, the application market extracts all the external URLs, creates a list of them, and create local URLs of them for offline browsing; the list of external URLs are referenced by the Offline Layer later when this application is installed by any user. Since most of the functionalities of IBS are provided via JavaScript interfaces, it is very easy to use them within gadgets. Also, developers can use the

Content Adaptation Layer if they want to include some external Web content into their applications. By using the Authoring Tool, developers can login to IBS, specify which application uses the result, decide whether single page or the whole domain is applied, choose the important regions of current Web page, and store the tailoring result to the Content Adaptation Layer.

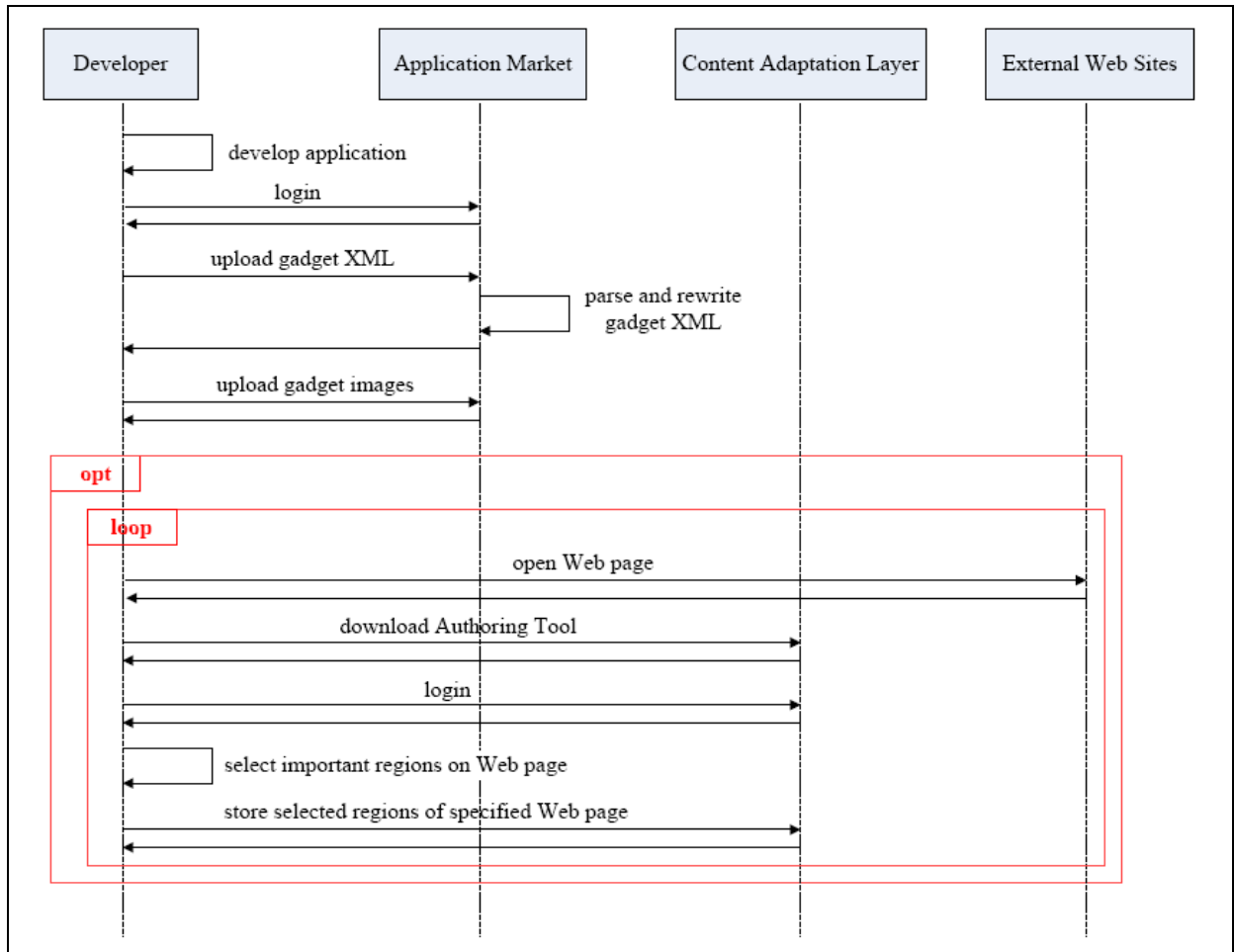


Figure 7-11. The sequence diagram of application publication

Figure 7-12 shows how users can install applications of IBS. For each time a user decides to install an application, the application market asks the Offline Layer to prepare for offline Web content. Two ways are supported for preparing offline data: normal Web page processing and RSS feed processing. When processing normal Web pages, the URLs specified by

applications are the target pages directly; however, in the case of RSS feed processing, the RSS feed is only an index, which contains multiple URLs for multiple targets. Therefore, the Offline Data Provider has to acquire the RSS feed first, parse it, and download all the Web content of all the targets referenced from it with the Web Crawler if a RSS-feed type of URL is specified in application. Since the Web Crawler crawls Web content via the Page Tailoring Service of Content Adaptation Subsystem, the returned Web content is adapted based on the application ID and developer ID. In this way, users can download the adapted Web content for offline browsing directly from the Offline Layer.

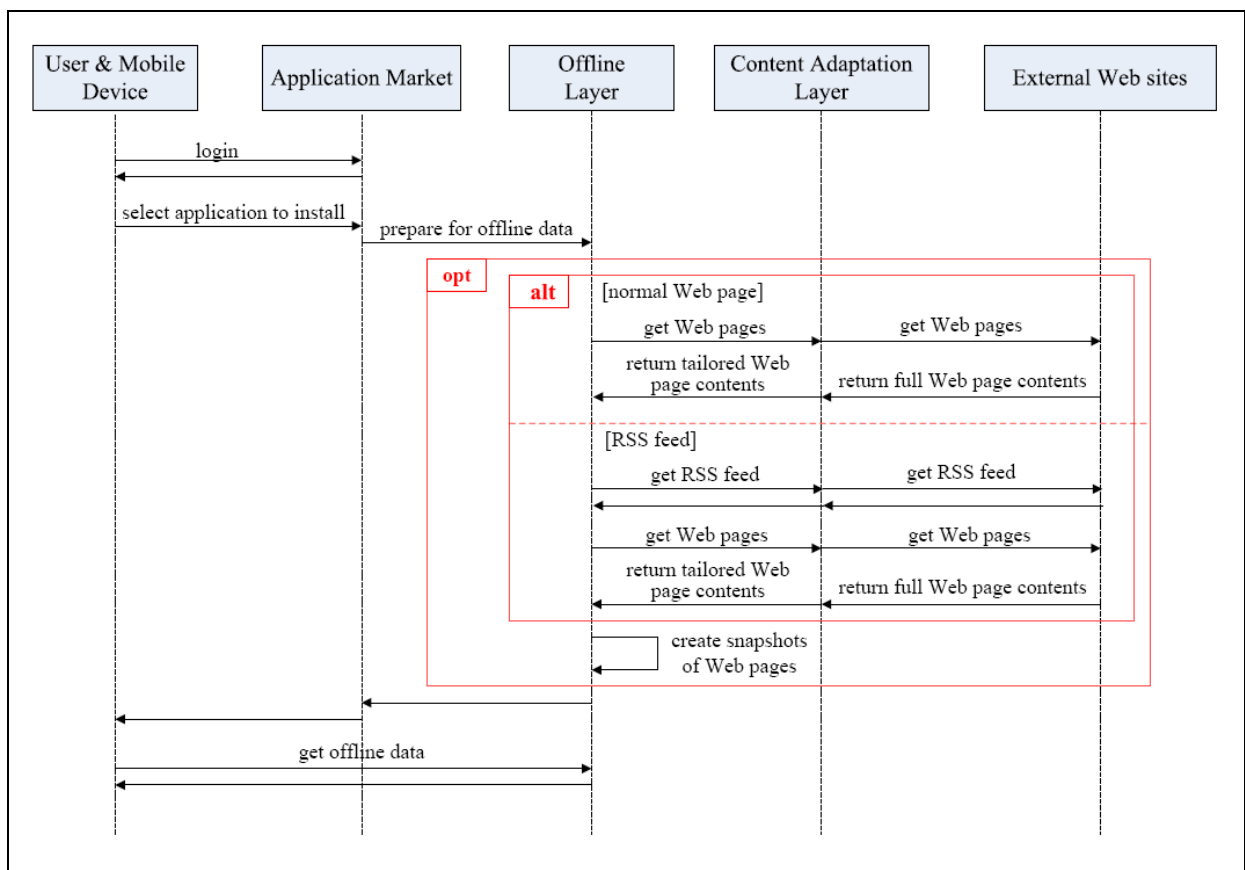


Figure 7-12. The sequence diagram of application installation

One of problems of Google Gears is that it doesn't provide any synchronization mechanism; it leaves this problem to the developers, since different applications usually have different synchronization requirements. However, it implies that developers have to provide an online

server and design their own synchronization mechanisms by themselves. In IBS, we provide a simple and basic synchronization service so that developers can use this service directly without maintaining their own online servers.

Table 7-2. The database schema of the table AppSync

| AppSync | | |
|-------------|----------|---------|
| Field | Type | Key |
| ID | int | primary |
| UserID | int | foreign |
| DeveloperID | int | foreign |
| AppID | int | foreign |
| Key | varchar | |
| Value | varchar | |
| Timestamp | datetime | |

In IBS, all the user data for application synchronization is stored in the AppSync table shown as Table 7-2. Each record of this table maintains the user ID, application's ID, developer's ID, a key-value pair of values, and the latest time when it is updated. Moreover, we provide a table manager which provides some basic operations such as insert, update, and delete, so that developers can use this table easily. The AppSync table is implemented within both the Local DB on the mobile device and Offline Data Storage, and the table manager is implemented within the Offline Data Agent on mobile device and the Offline Data Provider. During the synchronization process, the Sync Service checks that if there are two records have the same UserID, DeveloperID, AppID, and Key values in these two tables then replaces the older record with the newer one.

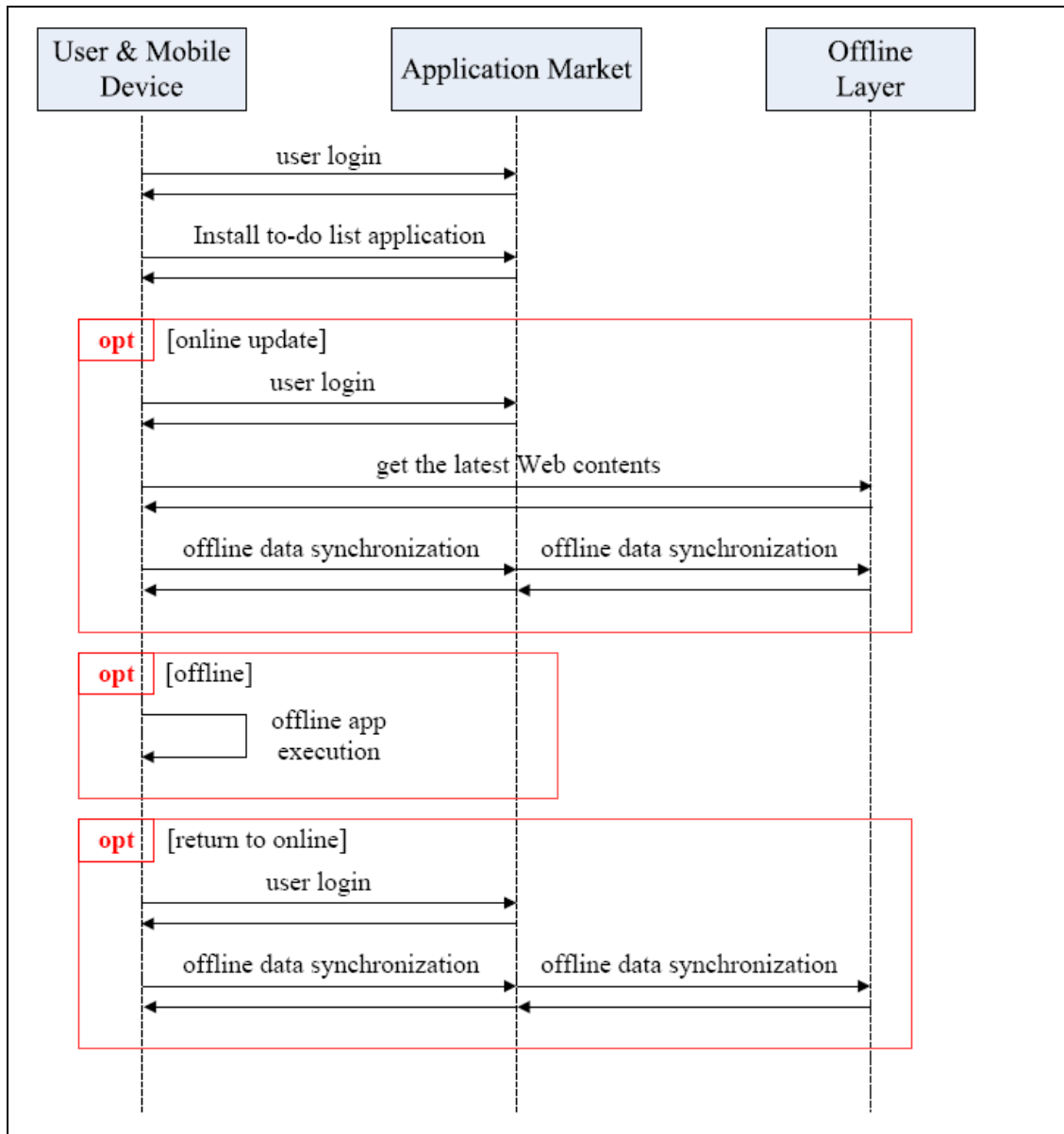


Figure 7-13. The sequence diagram of using the to-do list application

A “to-do list” application is designed for demonstrating the capability of Offline Layer. To-do list is a list which includes several tasks which should be finished. Although it is simple, it is useful for users to have an overview of tasks; therefore, users can easily decide the priorities of tasks and arrange their time properly. A to-do list can be implemented as an application of IBS so that it can be executed offline on mobile devices. The sequence diagram of using it is shown in Figure 7-13. After installing this application, users can execute it in offline mode, and add new items into their to-do lists. The local offline data is synchronized with the online

server when users execute this application again in online mode. In the implementation of the to-do list application, each item from each user is recorded into a record of the AppSync table. For each record, a hash value of item content is chosen as the Key value, and the Value value is the content of item. If any item is deleted, Value is set to null, but Key remains the same; this record is deleted during the synchronization process. If any item is edited, the value of Key is not re-hashed until it has been synchronized.

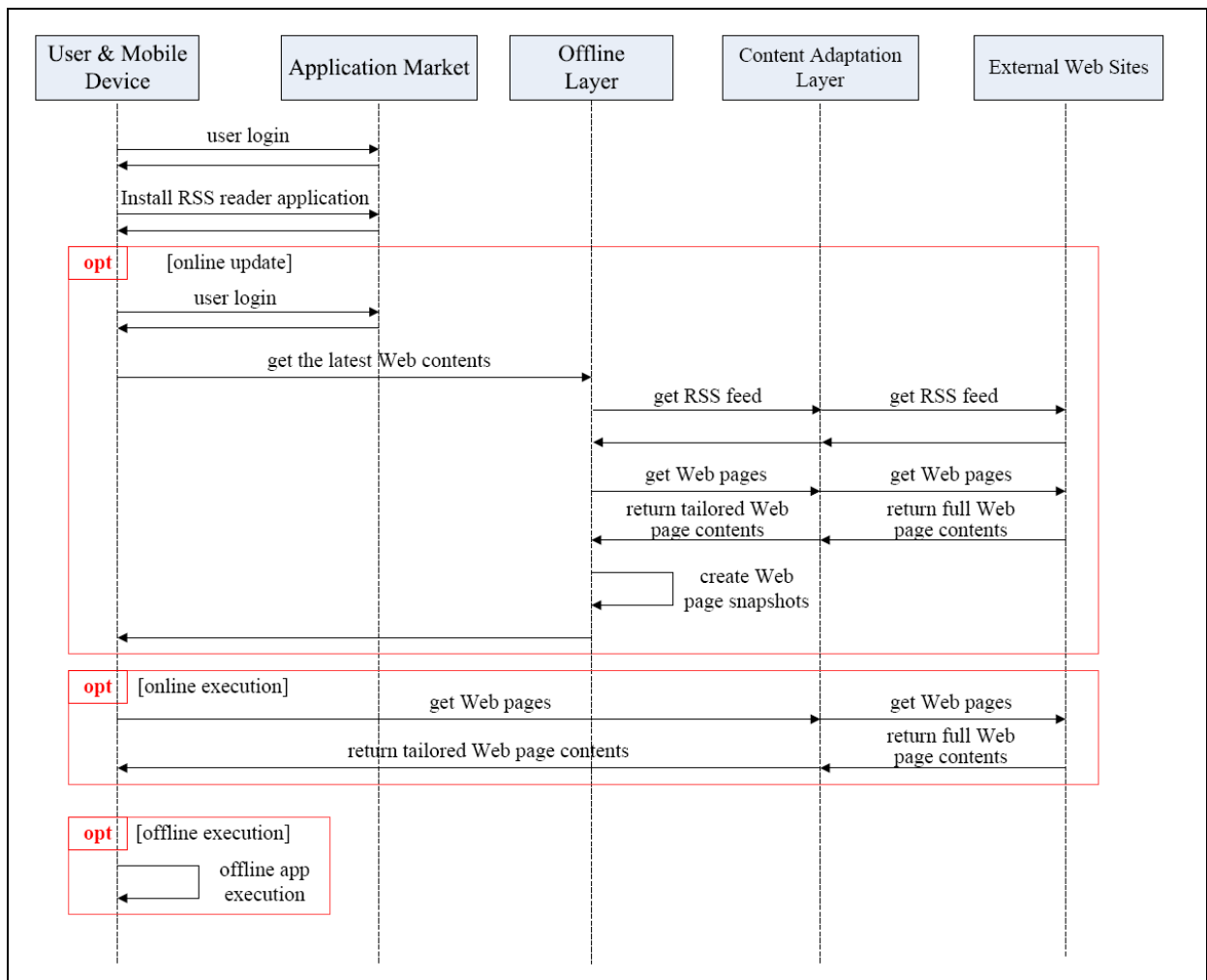


Figure 7-14. The sequence diagram of using the RSS reader application

We design a RSS reader application of BBC News for demonstrating the capability of Content Adaptation Layer. Figure 7-14 shows the sequence diagram of using the RSS reader application. In the gadget of this application, a JavaScript function is invoked for preparing

for Web pages with the type of RSS feed specified. Also, since we use the Authoring Tool to indicate that all the Web pages which are under the specified domain and accessed by this application should be tailored based on a selected pattern, all the returned Web pages are adapted for mobile devices during the online updating process. Moreover, before downloading each Web page, the Offline Layer checks that whether this article exists in the Offline Data Storage; if it exists, then it will not be downloaded again to speed up the updating process. For online execution, if users want to browse external Web pages with adapted content, they can set our Page Tailoring Service of Content Adaptation Layer as Web proxy; in this way, no matter these pages are browsed online or offline, they are adapted for mobile devices.

7.4 Detailed Design of PACS

The design of PACS includes three phases: the initialization phase, user registration phase, and authentication phase. To describe these protocols, numerous notations are defined as follows and employed throughout this thesis.

- a. $\langle EK_\tau, DK_\tau \rangle$: The public/private key pair of TI .
- b. $\langle EK_\pi, DK_\pi \rangle$: The public/private key pair of PAC .
- c. $\langle EK_\mu, DK_\mu \rangle$: The public/private key pair of user (and TR).
- d. PID : The identifier of PAC .
- e. SID : The service identifier indicating the service provided by PAC .
- f. UID : The identifier of user.
- g. π_{long}, π_{one} : The long-term and one-time user passwords.
- h. R_x : Random numbers (can be used as session keys) ($x = 1, 2, 3, \dots$)
- i. $Ea(d, k), Da(d, k)$: Asynchronous encryption and decryption functions for data d and key k

- j. $Es(d,k), Ds(d,k)$: Synchronous encryption and decryption functions for data d and key k
($Es = Ds$)
- k. $Me(d), Md(d)$: Matrix code encoding and decoding functions for data d

The TI is designed as an external authentication server to authenticate $UIDs$. Each SID is associated with a CD and at least one camera by PAC . Also, the PAC maintains the relationships between $UIDs$ and $SIDs$.

According to the requirements of PACS, several assumptions are defined and listed as follows:

- a. The user must have at least one mobile device which is equipped with a digital camera and a large enough screen.
- b. There is no Internet connection available, but the MMS channel is activated on mobile device.
- c. The organization which maintains the TI must already have a PKI, $\langle EK_\tau, DK_\tau \rangle$, to use, and the public key EK_τ is included directly into the TR .
- d. Since the proposed scheme is designed based on RSA, RSA must be trust worthy.
- e. Attackers are able to forge, modify, and replay messages of any kind of communications of PACS.
- f. User's mobile phone may be stolen, but no malware exists on it before it is stolen.
- g. The TI is not compromiseable. Since all the secrets of all users are stored in TI , it is the root of trust of PACS.

During the deployment of PACS several initialization steps must be conducted. First of all, the TI maintains a public/private key pair $\langle EK_\tau, DK_\tau \rangle$ for protecting messages in communications established with mobile phones or PAC . The public key EK_τ is included into

the distributions of TR , so that TR can establish secure channels with TI . Moreover, in case that TI and PAC are operated by different organizations, PAC must maintain its own public/private key pair $\langle EK_\pi, DK_\pi \rangle$ and register the EK_π with its PID to TI . If the TI and PAC are operated by the same organization, $\langle EK_\pi, DK_\pi \rangle$ can be equal to $\langle EK_\tau, DK_\tau \rangle$. In addition, each controlled locations must be associated with a SID , a CD , and at least one camera by PAC .

On the user side, users must download the TR from the Android Market and install it first. At the first time of TR execution, a pair of public/private keys $\langle EK_\mu, DK_\mu \rangle$ of TR is automatically generated for protecting and signing messages in the registration and authentication phases.

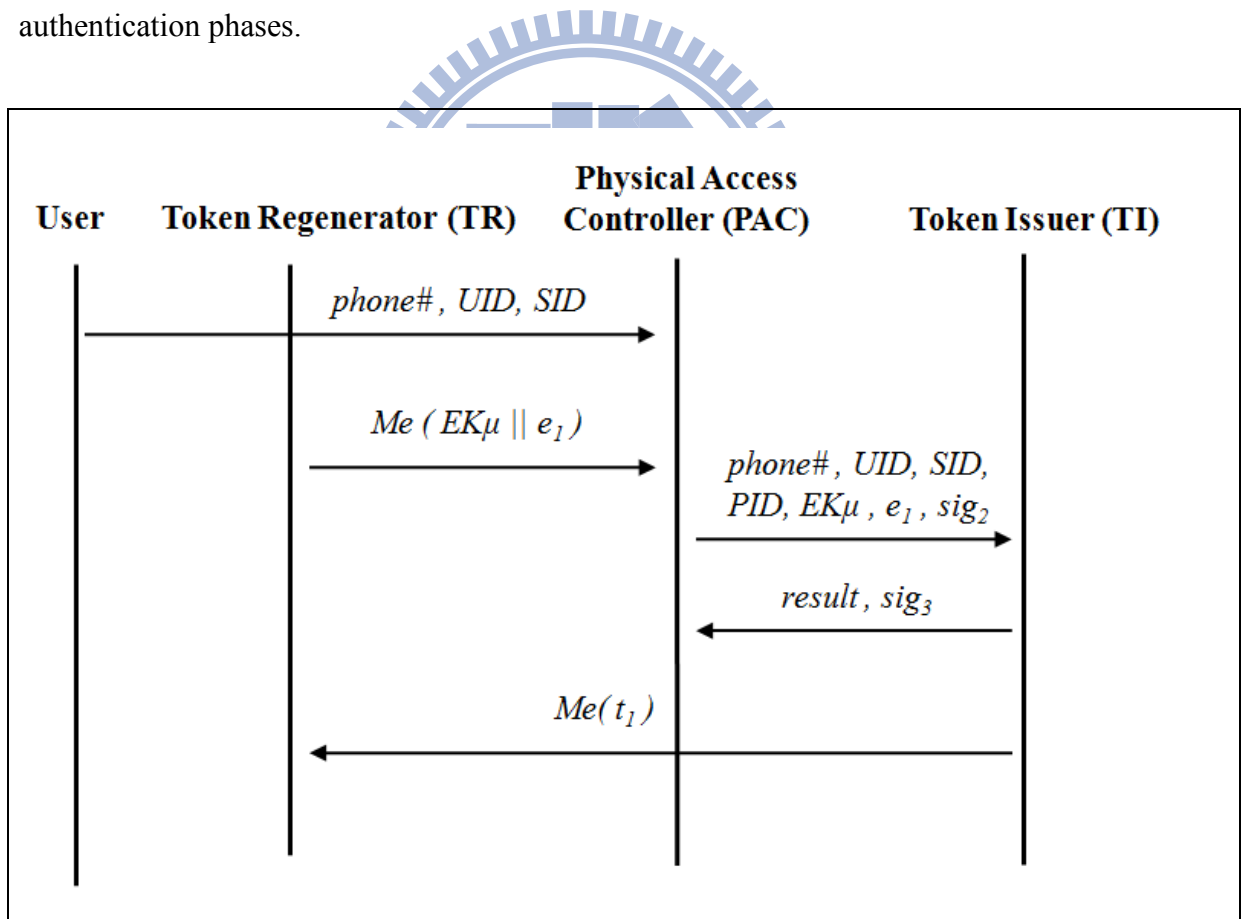


Figure 7-15. Protocol of user registration phase.

Figure 7-15 shows the protocol of user registration phase. During the user registration phase, the user must select a pair of account and password (UID and π_{long}) first. Then, the user uses a PC, goes to the Web page maintained by PAC , and enters the selected UID , SID , as well as user's phone number. After that, the PAC asks the user to execute TR , enters the password π_{long} into TR , and displays the QR code image m_1 in front of the camera of PC. At this time, a Flash program embedded on the Web page repeatedly tries to capture the m_1 until m_1 is successfully captured and decoded by computing $Md(m_1)$.

$$e_1 = Ea(R_1 || IMEI || h(\pi_{long}), EK_\tau)$$

$$m_1 = Me(EK_\mu || e_1)$$

In m_1 , R_1 is a random nonce, which makes the encrypted result diverse each time for preventing replay attack, and h is a hash function, which prevents the π_{long} from being exposed. Once the PAC obtains the decoded m_1 , it asks the PAC manager to confirm the establishment of association between UID and SID ; this step can be replaced by other customized authorization mechanisms such as empowering the users who have finished the payment. If the SID can be associated with UID , it implies that the user with UID is authorized to enter the location associated with the same SID . After the confirmation, the PAC sends the message m_2 to TI ; here, the function hs is another hash function which creates the digest of message to be signed.

$$sig_2 = Da(hs(phone\# || UID || SID || PID || EK_\mu || e_1), DK_\pi)$$

$$m_2 = phone\# || UID || SID || PID || EK_\mu || e_1 || sig_2$$

After receiving m_2 , the TI checks that whether the UID has already been used. If the UID is unused, the TI decrypts e_1 , and stores the UID , $h(\pi_{long})$, phone number, and EK_μ into its database. Finally, the TI sends back the message m_3 to PAC and a QR code image m_4 to TR

via MMS channel according to user's phone number if the registration is successful. Whenever the *TR* receives a MMS message, it checks that whether the message can be decoded and parsed successfully; then, the *TR* stores the token t_1 into its storage according to the *PID* and *SID*.

$$sig_3 = Da(result, DK_\tau)$$

$$m_3 = result || sig_3$$

$$e_2 = Ea(R_2 || f, EK_\mu)$$

$$e_3 = Ea(\pi_{one} || ts || te, EK_\tau)$$

$$sig_4 = Da(hs(UID || SID || PID || e_2 || e_3), DK_\tau)$$

$$t_1 = UID || SID || PID || e_2 || e_3 || sig_4$$

$$m_4 = Me(t_1)$$

In e_2 , f is a flag indicating whether the token has been regenerated. The e_2 is initially encrypted by the user's public key EK_μ so that only the user who possesses the DK_μ can regenerate this token. In the π_{one} is a one-time password randomly generated by *PAC* and associated with *UID*, and e_3 , ts and te represent the starting and ending valid time, respectively. The starting and ending valid time varies according to distinct *PID* and *SID*; if both of their values are negative, then the token t_1 can be used at anytime.

Figure 7-16 shows the protocol of authentication phase. To enter the controlled location, the user must execute the *TR* to regenerate the token t_2 from t_1 and encode the t_2 into a 2D barcode image, then place this image in front of the camera associated with the controlled location. The differences between t_1 and t_2 are that the flag f is updated to f' with the "true" value and the message is encrypted by *TI*'s public key EK_τ .

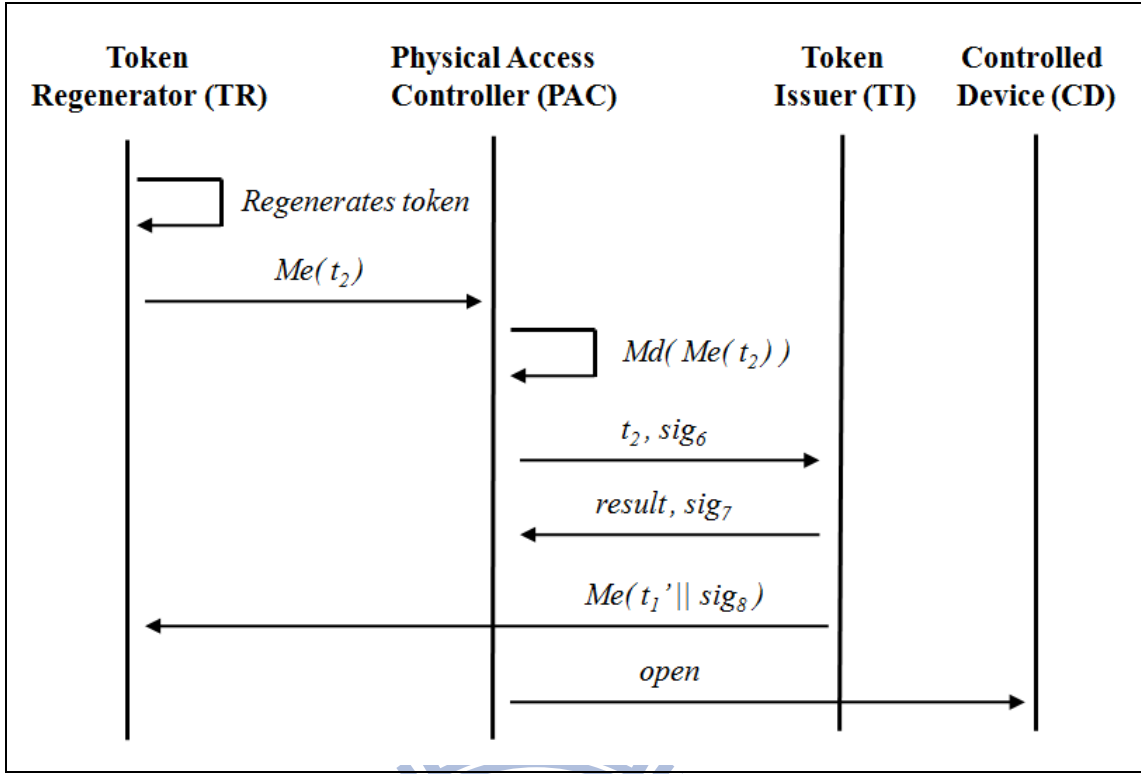


Figure 7-16. Protocol of authentication phase.

$$e_4 = Ea(R_2 || IMEI || f', EK_\tau)$$

$$sig_5 = Da(hs(UID || SID || PID || e_4 || e_3), DK_\mu)$$

$$t_2 = UID || SID || PID || e_4 || e_3 || sig_5$$

$$m_5 = Me(t_2)$$

After receiving the message m_5 , the *PAC* decodes the 2D barcode, extracts the token t_2 , and sends the message m_6 to *TI*.

$$sig_6 = Da(hs(t_2), DK_\pi)$$

$$m_6 = t_2 || sig_6$$

Whenever the *TI* receives the authentication message m_6 , it verifies the e_4 , e_3 , *IMEI*, and π_{one} according to the *UID* and the shared secret R_2 , then returns the authentication result m_7

back to *PAC*. Also, the TI sends a new token t_1' with a new random password π'_{one} to TR for the user's next authentication via MMS channel.

$$sig_7 = Da(result, DK_\tau)$$

$$m_7 = result || sig_7$$

$$e_4 = Ea(R_3 || f, EK_\mu)$$

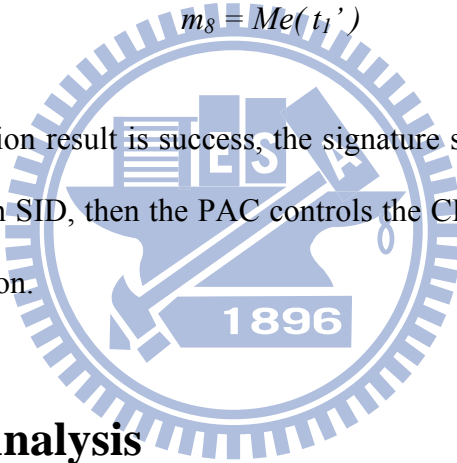
$$e_5 = Ea(\pi'_{one} || ts' || te', EK_\tau)$$

$$sig_8 = Da(hs(UID || SID || PID || e_4 || e_5), DK_\tau)$$

$$t_1' = UID || SID || PID || e_4 || e_5 || sig_8$$

$$m_8 = Me(t_1')$$

Finally, if the authentication result is success, the signature sig7 is verified successfully, and the user is associated with SID, then the PAC controls the CD to open and allows the user to enter the controlled location.



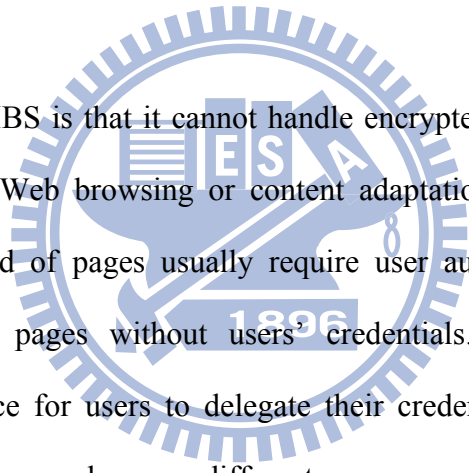
7.5 Limitation Analysis

Because almost all system modules are designed as Web Services, external systems or third-party designed modules must be able to interact with these modules easily. However, currently, we only design the functionalities for USHAS modules, not for other systems. Therefore, it is possible that several functionalities required in other systems are not provided in USHAS. For example, a facial recognition system integrating facial training and recognition operations into the USHAS for family members is difficult to design because no facial database exists in the USHAS.

In addition to the aforementioned difficulties, process conflicts may exist in the current design. For example, if a process “if Light A is on, turn off Light B” is defined with another process “if Light A is on, turn on Light B”, then the status of Light B depends on which process is defined later. In this scenario, because a semantic conflict exists between these two processes, the execution result may not be the same as the needs of the user. Several models, such as Petri Nets and Colored Petri Nets (CPN), can be used as verification tools to solve this problem. However, in the current design of USHAS, we simply assume that users are careful enough to avoid conflict in semantic processes.

Another limitation of USHAS is the security concern. Security is absolutely essential in the home environment. Various levels of security concerns are related to different types of security threats. The lowest security level is device level security: devices being controlled by no one other than the trusted home server or people must be guaranteed. Guaranteeing the security of the entire system is extremely difficult, unless devices are uniformly designed with security functionalities, such as providing a trust engine based on the framework designed by Seigneur et al. The second level is home network security. Network components such as firewalls and proxies must be configured properly so that devices and servers can communicate with each other within a proper and safe environment. For example, Fitzgerald et al. proposed their system architecture to perform network-based access control between semantic Web services by configuring firewall rules. The third security level is home server security: home servers should be well protected to be controlled only by trusted users. Some host-based mechanisms, such as TrustVisor, can be adopted to ensure that only trusted software can be executed under the control of hypervisor. The fourth security level is personal privacy protection. Because webcams may be deployed throughout the home environment for facial recognition and person identification, privacy must be guaranteed for activities such as changing clothes in the bedroom. A possible solution involves installing an application onto a

PC or IP camera that negotiates a secret key with a home server and encrypts each frame before transmission. The final security level is user authentication and access control. People do not usually want their home appliances to be controlled by those whom they do not trust, but they may be willing to share some control with relatives or close friends. For example, a user's friend allowed to change the channel on the TV can do so using a mobile phone. To integrate semantic Web service and security, several recent studies have defined certain semantic security ontologies and performed rule-based access control based on these ontologies. These matters are quite complex and out of the scope of the proposed system; therefore, solutions for solving these security concerns have not been included into USHAS currently.



The major limitation of IBS is that it cannot handle encrypted Web content such as HTTPS pages for either offline Web browsing or content adaptation. In the case of offline Web browsing, since this kind of pages usually require user authentication, the Web Crawler cannot access to these pages without users' credentials. Although we considered of implementing an interface for users to delegate their credentials, it is very difficult to be applied to general Web pages because different pages usually use different authentication mechanism. In the case of content adaptation, since the Page Tailoring Service is designed based on Web proxy, and one of the goals of Web page encryption is to prevent pages from being tempered, it is impossible to create adapted Web pages in the middle of external servers and mobile devices. Therefore, we suggest developers to develop the applications which only reference to public external Web pages. If user data should be encrypted during communications, it is better to handle the data encryption and decryption operations by local JavaScript programs and server-side programs within the user-data level, than within the Web-content level.

Chapter 8. System Prototype and Evaluation

8.1 System Prototype

Figure 8-1 shows the system prototype of USHAS. In the USHAS prototype, almost all the core modules, such as the Knowledge Base Manager, Semantic Process Manager, and Pub/Sub Event Broker, are implemented as Web Services running on a Apache Tomcat Web server of version 5.5.27, so that functionalities of these modules can be shared easily to other modules or even external systems. Among the existing mature BPEL runtime systems, this thesis used the open source ActiveBPEL [55] system. The Semantic Process Designer, Semantic Environment Editor, and Device Controller user interfaces are implemented in JSP; they are also deployed on the Tomcat server.

Constructing a complete smart home environment requires a lot of space and smart devices. Initially, the prototype of USHAS only contains two pseudo rooms with some controllable devices for demonstration. The first floor living room of USHAS prototype, a fan controlled by X10 module, an UPnP media player, an INSTEON light, and an USB webcam are deployed; in the second floor bedroom, two X10 lights and a webcam are included. Moreover, to simplify the user location detection, only one person is defined in this prototype, and the presence of this person is detected by the webcams based on moving detection.



| | |
|----------------------------|------------------------------|
| (a) USHAS Server | (b) D-Link UPnP Media Player |
| (c) INSTEON Module for fan | (d) INSTEON Module for light |
| (e) INSTEON-controlled fan | (f) INSTEON-controlled light |
| (g) Demonstration door | |

Figure 8-1. The living room of UAHAS prototype

Two semantic processes are deployed in the USHAS prototype. First, if there is someone in the living room, and the media player in the living room is turned on, then turn off all the lights and turn on the fan in the living room. Second, if there is no one in the bedroom at noon, then turn off all the lights in the bedroom. As a result, these two processes work perfectly in the USHAS prototype. The definitions of the home environment and the first process are shown in Figure 8-3 and 8-4.



Figure 8-2. The bed room of UAHAS prototype

The Web user interfaces of Semantic Environment Editor and Semantic Process Designer are shown in Figure 8-3 and 8-4. By using the Environment Editor, users can create instances of locations, people, time, and devices according to the actual home environments. Moreover, users can install device drivers, configure the detailed binding information, and control the on/off status of devices. Next, users can create their own processes based on these instances in the Semantic Process Designer. To avoid the ambiguous meanings of pictures, we use English sentences with some replaceable words for defining semantic processes. Each sentence provides all the information that SHPL needs, so that we can easily translate these sentences into SHPL documents. Moreover, we believe that users prefer selecting options than filling blanks; therefore, we list all the legal options for each replaceable word. Whenever each replaceable word is modified, all the other dependent words are checked again; if any illegal result exists during modification, the dependent words are set to null for users to choose again.

In this manner, we can ensure that all the sentences are correct. Finally, an example of using the SHPL in the prototype is shown in Figure 8-5.

Semantic Environment Editor

Location

| Location Name | Class | RoomIsLocatedAtFloor | | |
|----------------|--------------------|----------------------|--------|--------|
| F1 | FloorTypeLocation | | modify | delete |
| F2 | FloorTypeLocation | | modify | delete |
| MyBedRoom | Bedroom | F2 | modify | delete |
| MyLivingRoom | LivingRoom | F1 | modify | delete |
| NotAtHomePlace | Outdoor | | modify | delete |
| | <-- select one --> | <-- select one --> | add | |

Person

| Full Name | Class | hasFirstName | hasLastName | PersonIsLocatedAtLocation | | |
|------------|--------------------|--------------|-------------|---------------------------|--------|--------|
| John Smith | AdultMember | John | Smith | MyLivingRoom | modify | delete |
| | <-- select one --> | | | <-- select one --> | add | |

Time

| Time Name | Category | Start Time | End Time | Repeat Style | | |
|----------------------|--------------------|----------------------------------------------------|----------------------------------------------------|--------------------|--------|--------|
| WorkingDayWakeUpTime | Specific Time | Clock time: 6 : 0 : 0 | | EveryMonToFri | modify | delete |
| MovieTime | Time Range | Date time: August 1 2011 Clock time: 19 : 0 : 0 | Date time: August 1 2011 Clock time: 21 : 0 : 0 | none | modify | delete |
| | <-- select one --> | | | <-- select one --> | add | |

Device

| Device Name | Class | OnOffStatus | DeviceIsLocatedAtLocation | | | | | |
|------------------|-----------|-------------|---------------------------|---------|----------|--------|--------|--------|
| MyDVDPlayer | DVDPlayer | off | MyLivingRoom | turn on | turn off | modify | delete | config |
| LivingRoomFan | Fan | off | MyLivingRoom | turn on | turn off | modify | delete | config |
| LivingRoomLight | Light | off | MyLivingRoom | turn on | turn off | modify | delete | config |
| BedRoomLight2 | Light | off | MyBedRoom | turn on | turn off | modify | delete | config |
| BedRoomLight1 | Light | off | MyBedRoom | turn on | turn off | modify | delete | config |
| LivingRoomWebcam | Webcam | off | MyLivingRoom | turn on | turn off | modify | delete | config |
| BedRoomWebcam | Webcam | off | MyBedRoom | turn on | turn off | modify | delete | config |

add device:

Figure 8-3. Semantic Environment Editor

Process 1

Process name:

Variables:

| Name | Type | Value | |
|-------------------------------------|-------------------------------------------------------|-----------------------------------|-----------------------------------------------------------------------------|
| <input type="text" value="input1"/> | <input type="text" value="true/false"/> | <input type="text" value="true"/> | <input type="button" value="modify"/> <input type="button" value="delete"/> |
| <input type="text"/> | <input type="text" value="<-- select one -->"/> | | <input type="button" value="add"/> |

Preconditions:

| | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| exists one of <input type="text" value="Person"/> , which is a(n) <input type="text" value="category"/> , has(have) property <input type="text" value="IsLocatedAt"/> , with a(n) <input type="text" value="instance"/> of value <input type="text" value="MyLivingRoom"/> , and | <input type="button" value="modify"/> <input type="button" value="delete"/> |
| <input type="text" value="MyDVDPlayer"/> , which is a(n) <input type="text" value="instance"/> , has(have) property <input type="text" value="OnOffStatus"/> , with a(n) <input type="text" value="variable"/> <input type="text" value="input1"/> , and | <input type="button" value="modify"/> <input type="button" value="delete"/> |
| <p style="color: red;">Subject type:</p> <input type="text" value="<-- select one -->"/> <p style="color: magenta;">(Example: In precondition "all of Person, which is a(n) category, has(have) property PersonIsLocatedAtLocation, with a(n) instance of value F1", the subject type is "category")</p> | <input type="button" value="add"/> |

Execution Time:

Time Name

Execution Flow:

| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| for all <input type="text" value="instance"/> of the <input type="text" value="Light"/> , which is(are) located at the <input type="text" value="MyLivingRoom"/> , execute(s) operation <input type="text" value="TurnsOff"/> with variable <input type="text" value="input1"/> | <input type="button" value="modify"/> <input type="button" value="delete"/> |
| for one <input type="text" value="instance"/> of the <input type="text" value="Fan"/> (<input type="text" value="LivingRoomFan"/>) , which is(are) located at the <input type="text" value="category"/> <input type="text" value="LivingRoom"/> , execute(s) operation <input type="text" value="TurnsOn"/> with variable <input type="text" value="input1"/> | <input type="button" value="modify"/> <input type="button" value="delete"/> |
| <p style="color: red;">Subject type:</p> <input type="text" value="<-- select one -->"/> ; Location type: <input type="text" value="<-- select one -->"/> <p style="color: magenta;">(Example: In the invocation "for all of the Light, which is(are) located at the category LivingRoom, execute(s) operation Device_turn_off_Service with variable input1", the subject type is "Light", and the location type is "category")</p> | <input type="button" value="add"/> |

Figure 8-4. A semantic process of the USHAS prototype in Semantic Process Designer

```

<?xml version="1.0" encoding="utf-8"?>
<SProcess name="Home Theater"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="SHPL_Schema.xsd">
  <variables>
    <variable name="input1" type="xsd:boolean" value="true"/>
  </variables>

  <time_set><time>MovieTime</time></time_set>

  <preconditions>
    <condition domain_quantifier="exist" domain_type="category"
      domain="Person" property="IsLocatedAt"
      range_type="individual" range_value="MyLivingRoom"></condition>
    <condition domain_quantifier="null" domain_type="individual"
      domain="MyDVDPlayer" property="OnOffStatus"
      range_type="variable" range_value="input1"></condition>
  </preconditions>

  <flow>
    <invoke domain_quantifier="all" category="Light" device_name="null"
      location_type="category" location="MyLivingRoom"
      operation="TurnsOff" variable="input1">
    </invoke>

    <invoke domain_quantifier="one" category="Fan" device_name="LivingRoomFan"
      location_type="individual" location="LivingRoom"
      operation="TurnsOn" variable="input1">
    </invoke>
  </flow>
</SProcess>

```

Figure 8-5. An example of using SHPL in the prototype

An example of the mobile content adaptation sub-system result is shown in Figure 8-6 and 8-7. As seen in Figure 8-6 (a), only parts of the Web page can be presented on the limited screen at one time. There is plenty of room for scrolling in both horizontal and vertical directions. In Figure 8-6 (b), we personalize this Web page by selecting three blocks of content that are located in the middle of this page. After the personalizing process (shown in Figure 8-6 (c)), the unwanted web page content will be filtered. The result web page of our system is precise and users can read this page with a glance.



Figure 8-6. A practical example of using the USHAS content adaptation sub-system



Figure 8-7. The result page of Google Mobile Proxy

Compare to Google Mobile Proxy, which is shown in Figure 8-7. We can see that the result Web page of the Google Mobile Proxy is still too long in length for scrolling up and down, as

in Figure 8-7 (c). There is still some unimportant information that maybe the users are not interested in, as in Figure 8-7 (d). Compare to our result Web page, as in Figure 8-6 (c), our result page can be more personalized and precise for page presentation.

The mobile version homepage and PC version homepage of IBS are shown in Figure 8-8. In the mobile homepage, the application execution manager displays only icons of applications, so that it matches the traditional user experience of using mobile applications. In the PC version of homepage, the application execution manager provides more details of each application, such as its name and which mobile browser is supported. The reason why some applications are not supported by some browsers is because that the JavaScript program developed by developers may not be able to be executed on some mobile browsers due to the different levels of support of JavaScript.

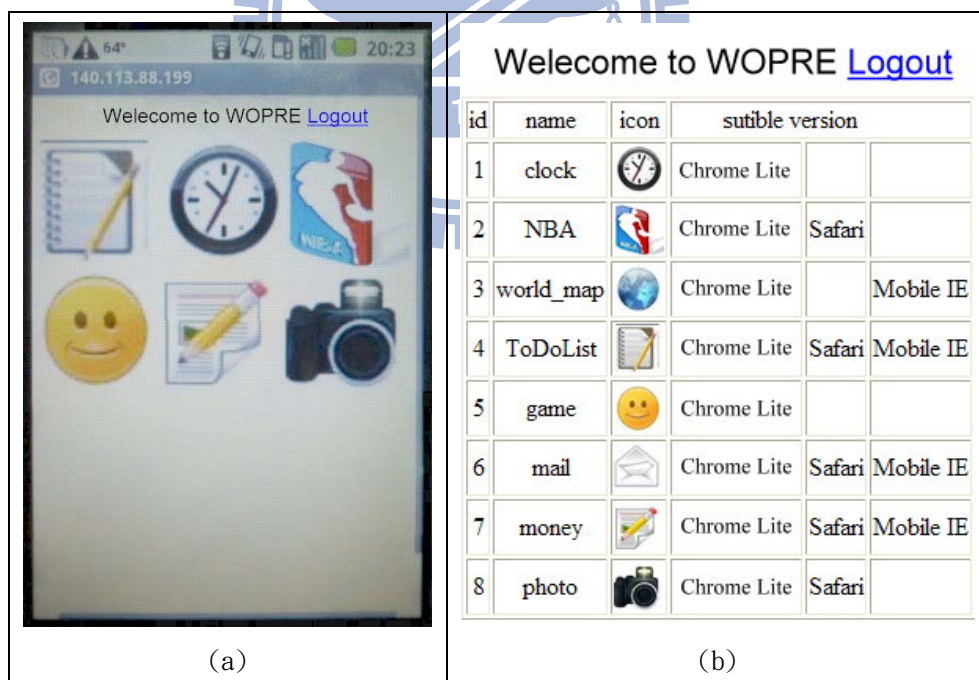


Figure 8-8. (a) The mobile version homepage (b) The PC version homepage

Figure 8-9 shows the to-do list application on mobile device. Even it is under offline mode, user can still do some operations such as add a new item or delete an old item.

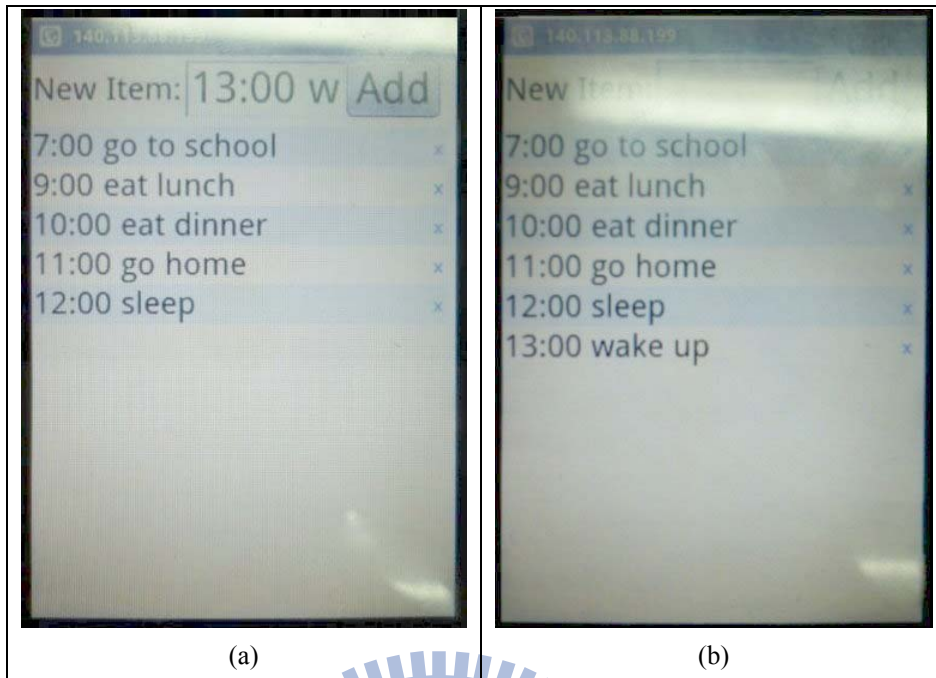


Figure 8-9. (a) The to-do list application (b) The to-do list application with one new item added

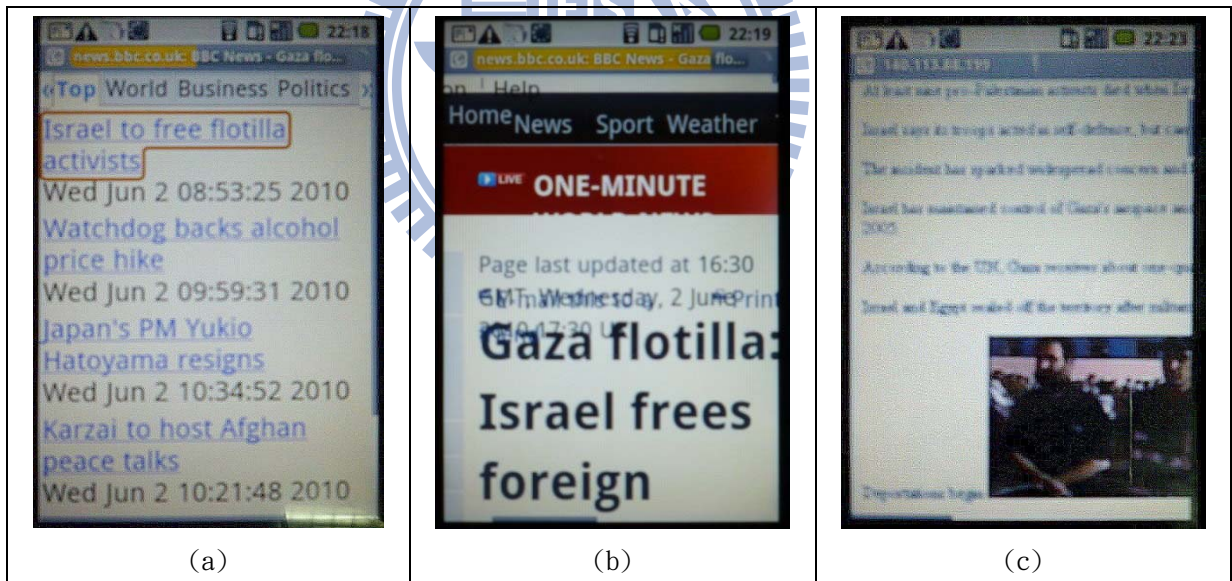


Figure 8-10. (a) The RSS reader application (b) The online external Web page without content adaptation (c) The offline external Web page with content adaptation

Figure 8-10 (a) shows the RSS reader application for BBC News on mobile device. When this application is executed under online mode without our content adaptation mechanism, the result is shown in Figure 8-10 (b); obviously, this Web page is not designed to be browsed on mobile device. With our content adaptation mechanism, the result is shown in Figure 8-10 (c).

Either the mobile device is under online mode or offline mode, adapted Web pages can be provided to mobile users.

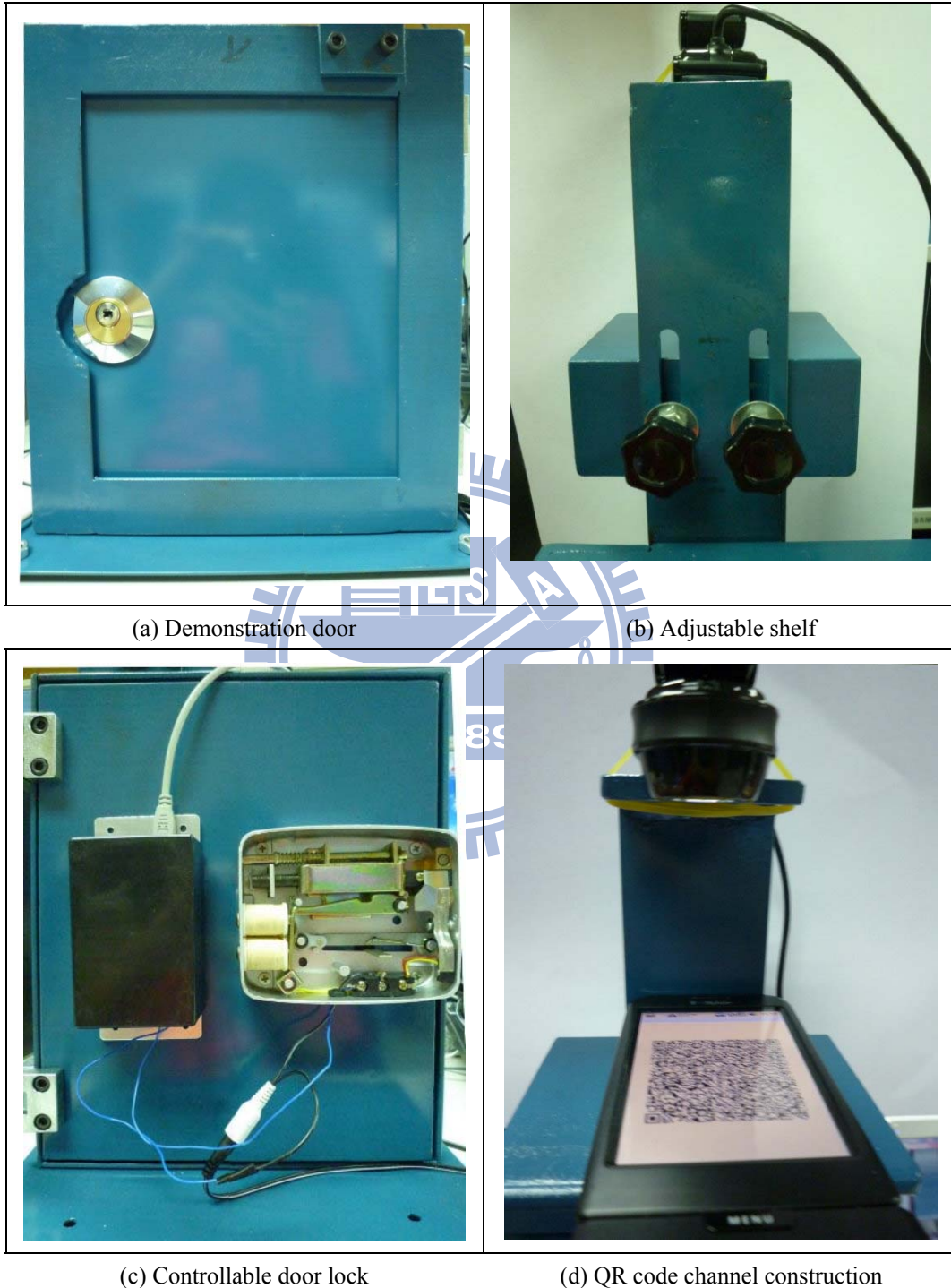


Figure 8-11. The system demonstration of PACS prototype.

Figure 8-11 shows the demonstration door, adjustable shelf, controllable door lock, and QR code channel construction of PACS. If the QR code image is decoded and the token is authenticated successfully, the door is automatically opened.

8.2 System Scenarios

Table 8-1. Examples of different scenarios and corresponding semantic automation processes

| Scenario | Semantic Automation Process |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Long Vacation | During the period of vacation time (from Aug. 1 st to Aug. 10 th), turn on all the lights in the house at 7:00 PM, and turn them off at 11:00 PM every day. |
| Home GYM | If there is anyone in the gym room, then turn on the light and TV in the gym room. |
| Morning Rush | From Monday to Friday every week, at 7:00 AM, turn on all the lights of bedrooms and bath rooms, and the coffee maker starts to make coffee. |
| The Living Room | If the temperature in the living room is higher than 30.0 °C, then turn on air conditioner in the living room and set the temperature of air conditioner as 26.0 °C. |
| Dinner Time | If any family member is in the dining room at dinner time, then turn on the TV in the living room, and set the channel of TV as 20. |
| Good Student | If there is someone in the reading room during 9:00 PM to 11:00 PM every day, set the volume of all the volume style devices at home as 0. |
| Sweet Dreams | If all the family members are located in bed rooms during 0:00 AM to 7:00 AM every day, turn off all the lights in living room, gym room, kitchen, reading room, and bath room. |
| Bath Time | If there is a gas event published by the gas sensor in the bath room, then turn on the gas alarm. |
| Party Night | At 10:00 PM on Sep. 12 th , 2010, turn on the party light and the DVD player in the living room. |

To analyze the user satisfaction and user usability of USHAS, we create some examples of automation processes for different scenarios; these examples also can be defined through the

USHAS Web interface. Since the capability of device control has been shown in the USHAS prototype, we only focus on the expression of SHPL and usability of USHAS with some simulated devices when designing these scenarios and processes. The variety of scenarios guarantees that USHAS and SHPL are able to support a large enough amount of cases which users may need. These scenarios and corresponding processes are listed in Table 8-1.

8.3 User Satisfaction Evaluation

The user satisfaction analysis is shown in table 8-2. We conducted a survey over 136 people in Taiwan who are randomly chosen on the Internet. Each one of them can select more than one scenario which they are interested in. The result shows that almost all the people are satisfied with at least one scenario and the corresponding example; the ratio is 98.53%. Even for the scenario which gets the lowest ratio, Home GYM, 48 people are interested in it. The reason why this scenario gets the lowest count maybe is that people in Taiwan usually do not have the habit of home exercise.

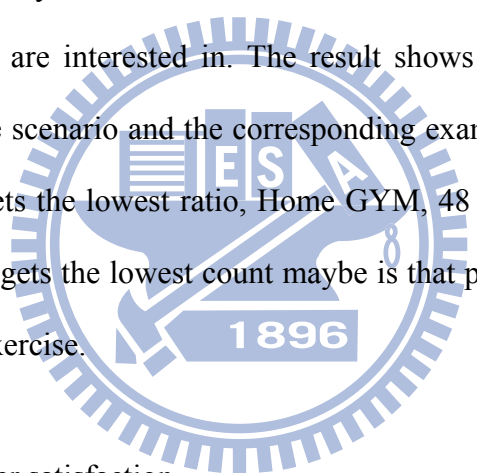


Table 8-2. Analysis of user satisfaction

| Q: Which scenario(s) and the corresponding example(s) you are interested in? | | |
|-------------------------------------------------------------------------------------|--------------|--------------|
| Answer | Count | Ratio |
| Long Vacation | 116 | 85.29% |
| Home GYM | 48 | 35.29% |
| Morning Rush | 96 | 70.59% |
| The Living Room | 90 | 66.18% |
| Dinner Time | 54 | 39.71% |
| Good Student | 74 | 54.41% |
| Sweet Dreams | 120 | 88.24% |
| Bath Time | 87 | 63.97% |
| Party Night | 51 | 37.50% |
| None | 2 | 1.47% |

8.4 Usability Evaluation

To analyze the usability of USHAS, we conduct a preliminary survey based on the system scenarios defined in 8-2. Twenty of 136 people in 8-3 are willing to do the questionnaire of using USHAS. Totally, 7 locations, three people, 8 time definitions, 11 devices, and 10 processes (the “Long Vacation” scenario requires two processes) must be defined by each one of them. The average spent time, average difficulty (1 for very easy and 10 for very hard), and average accuracy rate of using the Semantic Environment Editor have been listed in Table 8-3. The accuracy rate is defined as the number of correctly created instances over the number of all the processes which are required to be defined. The correctness of each instance is judged based on whether it makes the scenario work; for example, using different name of device is not counted as incorrect definition. In general, people can define instances within a little time and feel easy to do so with high accuracy rate.

Table 8-3. Analysis of usability of Semantic Environment Editor

| | Average Spent time (per instance) (min.) | Average difficulty (1-10) | Average accuracy rate |
|----------------------------|-------------------------------------------------|----------------------------------|------------------------------|
| Location Definition | 3.25 (0.46) | 1.7 | 100 % |
| Person Definition | 2.15 (0.717) | 1.7 | 100 % |
| Time Definition | 6.9 (0.8625) | 1.7 | 98 % |
| Device Definition | 6.95 (0.63) | 1.5 | 100 % |
| Average | 4.81 (0.66) | 1.23 | 100 % |

Similar to the Semantic Environment Editor, the Semantic Process Designer is also analyzed according to the average spent time, average difficulty, and average accuracy rate of using it. The accuracy rate of process creation is defined as the number of correctly specified inputs over the number of all the inputs required for each process.

Table 8-4. Analysis of usability of Semantic Process Designer

| | Process Type | Average Spent time (min.) | Average difficulty (1-10) | Average accuracy rate |
|------------------------|-----------------------|----------------------------------|----------------------------------|------------------------------|
| Long Vacation | On-Off control | 3.15 | 2.1 | 92.5% |
| Home GYM | On-Off control | 3.8 | 2.35 | 99% |
| Morning Rush | On-Off control | 3.2 | 2.1 | 97% |
| The Living Room | Control with variable | 7.95 | 8.35 | 76% |
| Dinner Time | Control with variable | 7.45 | 6.35 | 88.25 |
| Good Student | On-Off control | 4.55 | 4.4 | 94% |
| Sweet Dreams | On-Off control | 3.1 | 2.1 | 98.5 |
| Bath Time | Event handling | 8.65 | 8.9 | 56% |
| Party Night | On-Off control | 3.25 | 2.35 | 98.5 |
| Average | | 5.01 | 4.32 | 88.86 |

Moreover, the type of each process is also listed in Table 8-4; here the type indicates the most difficult part of each process. For the nine scenarios, most of them are considered easy and created within a short time with high accuracy rate, while some of them are difficult with longer time and lower accuracy rate. In general, users feel easy to specify processes which control the on-off status of devices; this kind of control is also the most common and intuitive one matching the daily home control experiences. The second easy type of process is the controls with variables. We find that some people do not have the sense of variable, especially for those who do not have science backgrounds. Supporting multiple type of variable is complex to some people, although it provides the flexibility for different kind of operation. Finally, the most difficult kind of process is the event handling type process. Compared with variable, fewer people have the sense of event, and more people failed to create a correct precondition with event included. Actually, we provide two ways to make a correct

precondition for the “Bath Time” scenario; they are shown in Figure 8-12. The precondition of “Bath Time” can be specified from either the device’s or event’s point of view; however, many people still cannot make it correct. Even though some of the processes are not easy to everyone to specify, the result of overall average spent time, difficulty and accuracy rate are satisfied.

Answer 1:

MyGasSensor , which is a(n) instance , has(have) property Generates , with a(n) category of value GasEvent , and

Answer 2:

exists one of GasEvent , which is a(n) category , has(have) property IsGeneratedBy , with a(n) instance of value MyGasSensor , and

Figure 8-12. Two ways to specify the precondition of the “Bath Time” scenario

Table 8-5. Questions included in questionnaire

| Category | Question | Average Degree of Agreement |
|-------------|---------------------------------------------------------------------------------|-----------------------------|
| Appeal | 1. I think the user interface is suitable for being displayed on small screens. | 4.50 |
| Appeal | 2. I am willing to use this system frequently. | 3.92 |
| Content | 3. All the functionalities are well-integrated in the system. | 4.00 |
| Ease of Use | 4. This system is easy to use. | 4.25 |
| Ease of Use | 5. I don't need professional help when I use this system. | 3.83 |
| Ease of Use | 6. I can easily find the buttons that I need every time. | 4.42 |
| Ease of Use | 7. I clearly know what the result of each step is. | 3.83 |
| Performance | 8. The processing speed of this system is fast. | 3.92 |
| Support | 9. This system can do what I thought it could do. | 4.08 |
| Support | 10. This system makes my life convenient. | 4.25 |

The principal factors proposed by Benbunan-Fich [88] were used for designing a questionnaire to evaluate the usability of IBS. The questionnaire contains 10 questions based on factors of appeal, content, ease of use, performance, and support. The questions are listed in Table 8-5 with 6 (0-5) agreement degrees ranging from “strongly disagree” to “strongly agree.” This questionnaire was used to survey 24 users who were randomly chosen on the Internet.

Table 8-5 and Figure 8-13 show that the lowest average degree of agreement appears in Questions 5 and 6. Actually, these two questions are highly rigid; users usually do not require professional help only when they are highly familiar with a system and know what the result of each step is. Although the two questions with the lowest degree of agreement both belong to the Ease of Use category, the category that has the lowest agreement degree is Performance. Since downloading offline data requires time during the application installation and update processes, users usually think that these operations are not fast enough. However, the overall degree of agreement is satisfactory. In addition, Figure 8-14 shows the Cronbach α of the survey; the support category has the smallest α (0.81), and the overall α is 0.95.

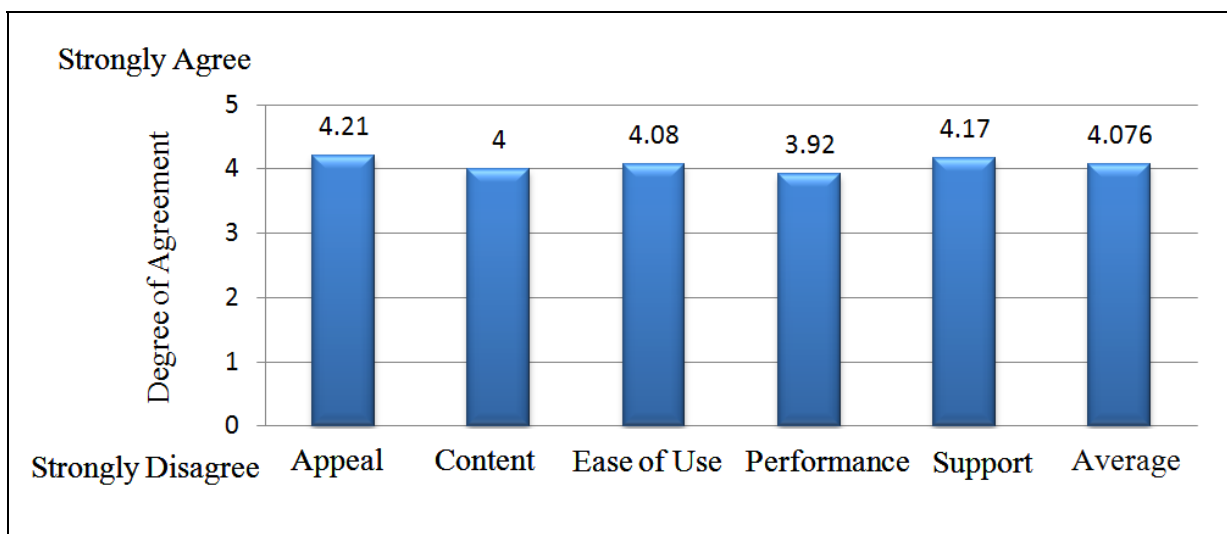


Figure 8-13. The analysis result of usability evaluation (average degree of agreement)

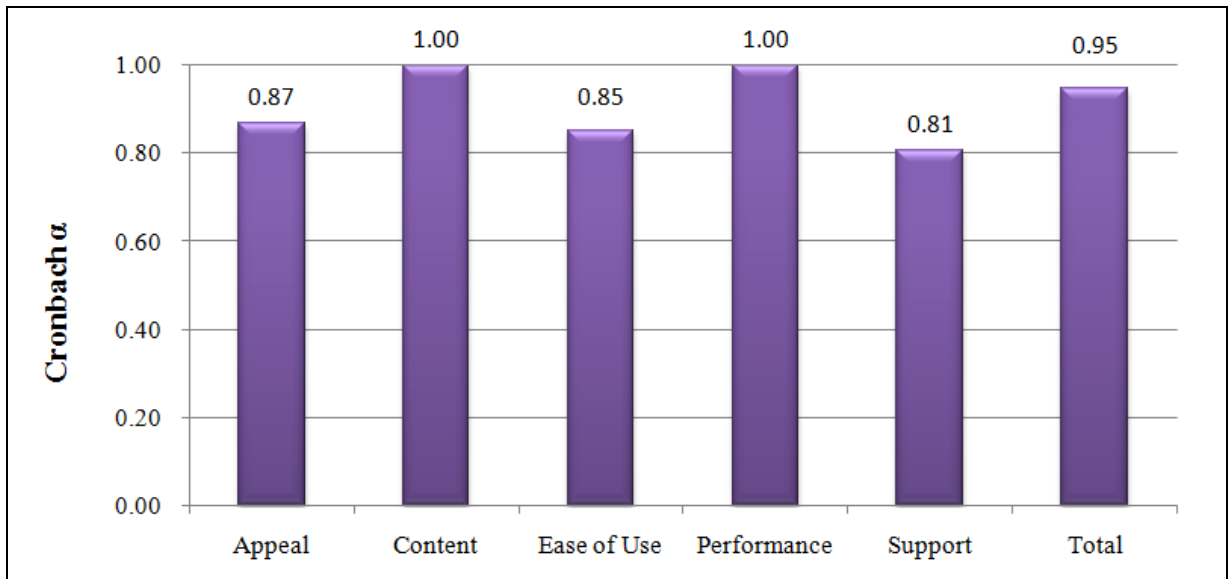


Figure 8-14. The analysis result of usability evaluation (Cronbach α)

To test the usability of Page Tailor on different browsers, we personalize a Web page in Internet Explorer and launch Page Tailor (on the same Web page) in Firefox Web browser to check if the same result can be obtained, and vice versa. Figure 8-15 shows the result of the test.

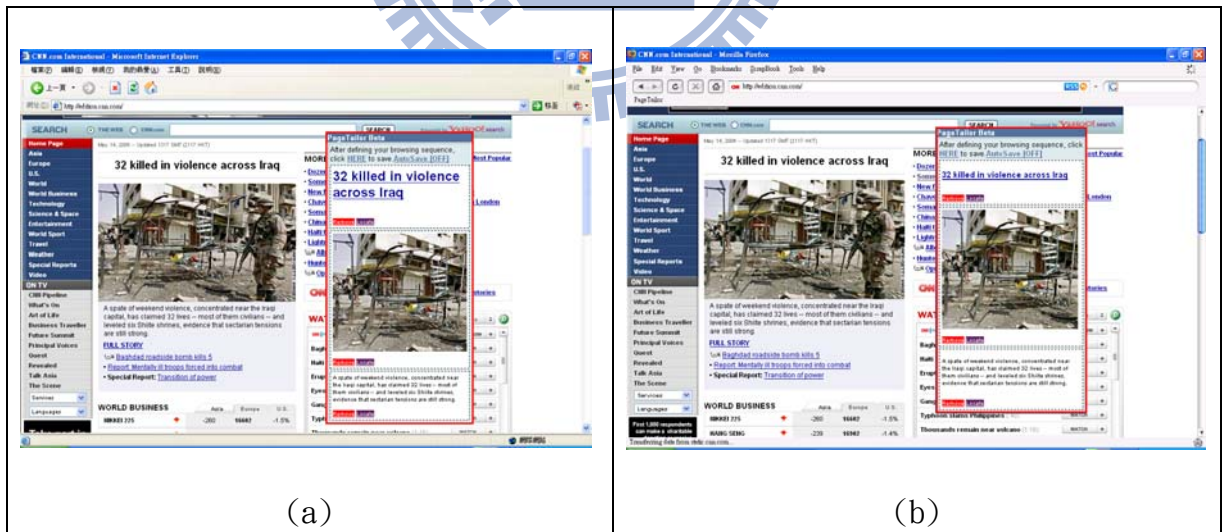


Figure 8-15. Usability test (a) Page Tailor in Internet Explorer (b) Page Tailor in Firefox Web browser

8.5 Face Recognition Accuracy Evaluation

We implement the face recognition sub-system in C++. The face recognition is performed by the basic PCA method. In the thesis, we conduct four experiments for password and hand gesture integrations. In the first experiment, the face database we used is constructed by our self, which contains twenty face patterns and ten images per pattern.

Table 8-6. Recognition rate of P and P_i (1)

| | training data | | | | |
|-------|---------------|-------|-------|------|-------|
| | 1 | 2 | 3 | 4 | 5 |
| P | 0.83 | 0.88 | 0.885 | 0.9 | 0.915 |
| P_i | 0.98 | 0.985 | 0.99 | 0.99 | 0.995 |

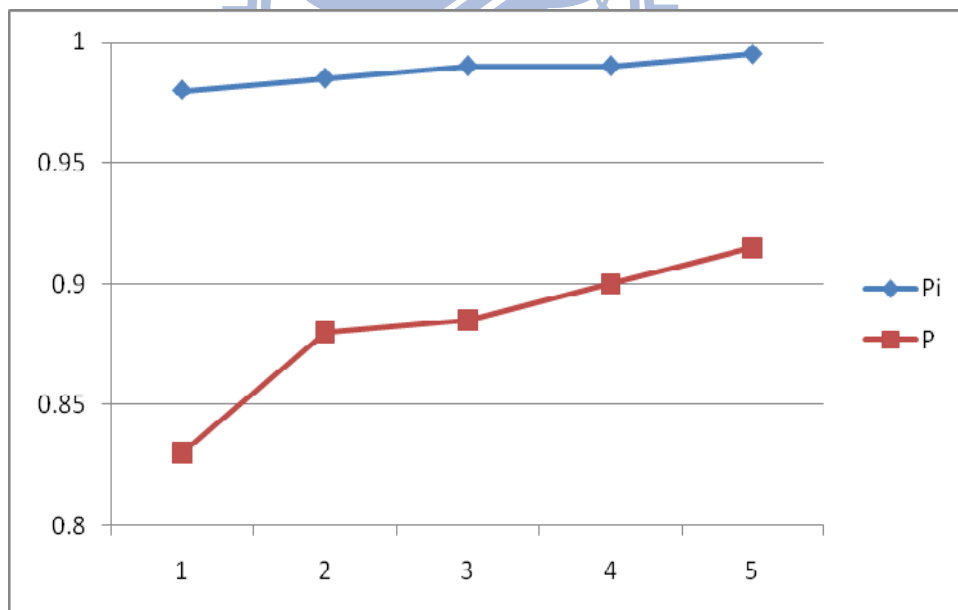


Figure 8-16. Recognition rate of P and P_i (1)

Table 8-6 and figure 8-16 illustrate the recognition rate of original face recognition P (Probability without integration), and integrated face recognition P_i (Probability with

integration). We analyze the recognition rate if there are one to five training images of these two recognition method. In general, P_i is always higher than P .

In the second experiment, the face database we used is a subset of gray FERET database [89], which contains twenty face patterns and ten images per pattern. Table 8-7 and Figure 8-17 show the recognition rate of original face recognition and integrated face recognition. In this experiment, P_i is also always outperforms than P .

Table 8-7. Recognition rate of P and P_i (2)

| | training data | | | | |
|-------|---------------|------|-------|-------|-------|
| | 1 | 2 | 3 | 4 | 5 |
| P | 0.46 | 0.53 | 0.615 | 0.665 | 0.745 |
| P_i | 0.765 | 0.88 | 0.945 | 0.985 | 0.985 |

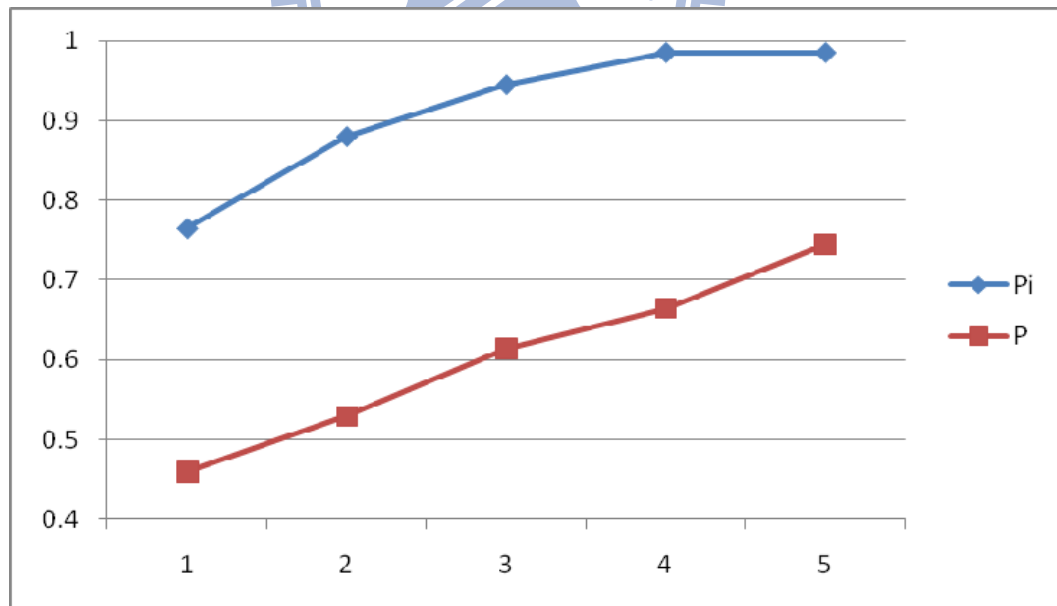


Figure 8-17. Recognition rate of P and P_i (2)

For the hand gesture integration, both the face and hand gesture recognition are performed by the basic PCA method. The face or hand detection is not the topic we focus on, so we use

different database for face and hand gesture. In the experiment, the face database we used is ORL database [90], which contains forty face patterns and ten images per pattern.

Table 8-8. Recognition rate of P_f , $P_{f'}$, and P_h

| | # of training data | | | | |
|----------|--------------------|-------|-------|-------|-------|
| | 1 | 2 | 3 | 4 | 5 |
| P_f | 0.703 | 0.82 | 0.873 | 0.905 | 0.963 |
| $P_{f'}$ | 0.757 | 0.83 | 0.885 | 0.915 | 0.96 |
| P_h | 0.907 | 0.933 | 0.958 | 0.953 | 0.983 |

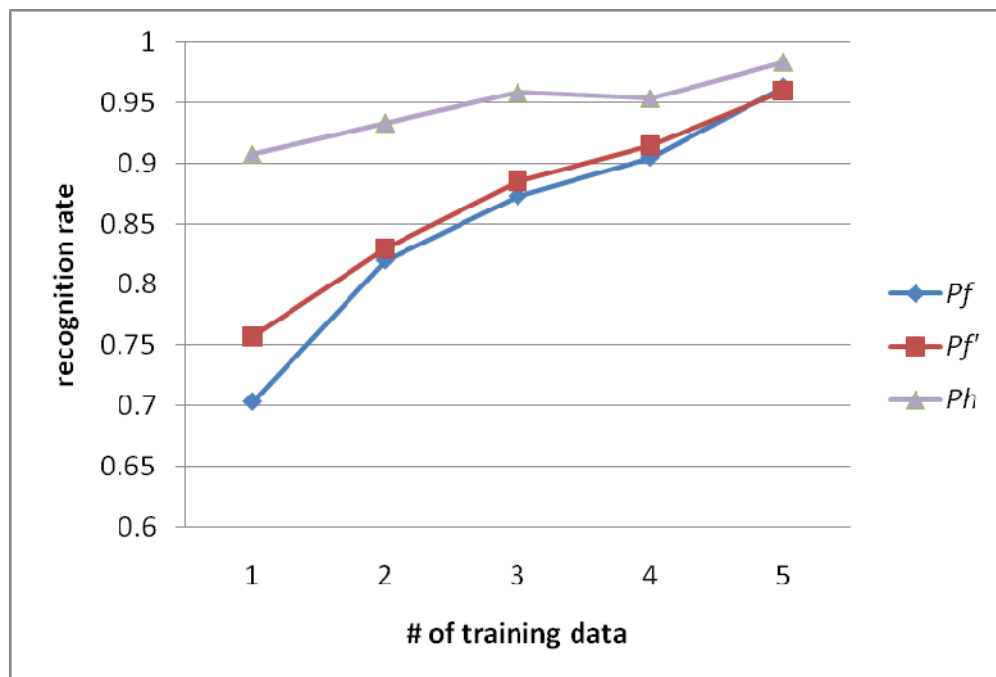


Figure 8-18. Recognition rate of P_f , $P_{f'}$, and P_h

Table 8-8 and figure 8-18 illustrate the recognition rate of original face recognition, integrated face recognition, and hand gesture recognition. We analyze the recognition rate if there are one to five training images of these three recognition method. In general, P_h is always higher than P_f and $P_{f'}$, and $P_{f'}$ almost outperforms than P_f . However, we may notice that if P_f is high enough, the $P_{f'}$ is even a little bit lower than P_f .

Table 8-9 and figure 8-19 illustrate the recognition rate of the original and integrated overall accuracy. We can see that, the same, P_o is almost higher than P_o' . Also, we may notice that P_o' is able to be higher than 94% under five training data.

Table 8-9. Recognition rate of P_o , and P_o'

| | # of training data | | | | |
|--------|--------------------|-------|-------|-------|-------|
| | 1 | 2 | 3 | 4 | 5 |
| P_o | 0.638 | 0.765 | 0.836 | 0.862 | 0.947 |
| P_o' | 0.687 | 0.774 | 0.848 | 0.872 | 0.944 |

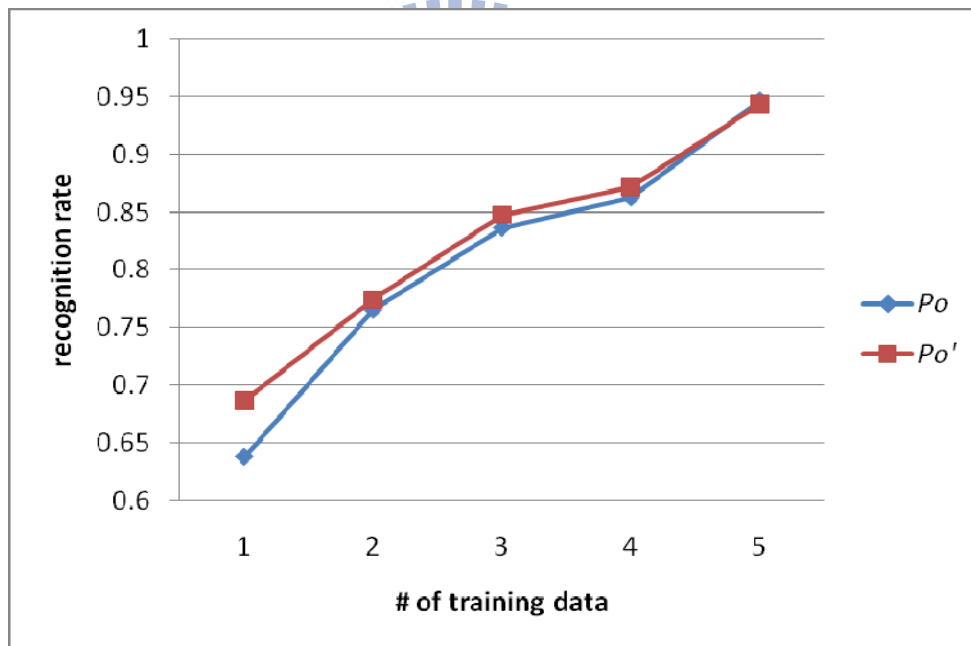


Figure 8-19. Recognition rate of P_o , and P_o'

8.6 System Comparison

Table 8-10. System Comparison

| | Automation Process Execution | Event Handling | Semantic Automation | The “all” quantifier supported | User Config--urable | Repeated Time supported |
|----------------------------------------------|-------------------------------------|-----------------------|----------------------------|---------------------------------------|----------------------------|--------------------------------|
| System designed by Li et al. [24] | No | No | No | No | No | No |
| System designed by Wu et al. [1] | Yes | Yes | Yes | No | No | No |
| System designed by Ha et al. [10] | Yes | No | Yes | No | No | No |
| System designed by Rodden et al. [42] | Yes | Yes | No | No | Yes | No |
| System designed by Drey et al. [43] | Yes | Yes | Yes | Yes (by an ambiguous symbol “*”) | Yes | No |
| USHAS | Yes | Yes | Yes | Yes | Yes | Yes |

The comparison of USHAS and many related systems mentioned in Section 2 are shown in Table 8-10. We compare them in six dimensions: automated process execution, event handling, semantic process automation, automation of semantic process which allow the “all” quantifier, user configuration, and repeated time, and check that these factors are supported or not. Many OSGi based systems, such as [24], only allow user to control appliances directly; automation is not supported. Some researches such as [42] only focus on whether and how automated processes can be created by users; semantic process description is not supported. For semantic automation systems such as [1] and [10], processes are usually fixed; users can not create their own processes. In the system designed by [43], user configurable semantic

automation process is supported with event handling. Although this system also support the “all” quantifier, this is defined by using an ambiguous “*” symbol, which implies “exist one of” on the sensor side but “all of” on the actuator side. As to the concepts of repeated time such as every day, none of the above systems supports them. Finally, only USHAS supports all these features.

Table 8-11. The comparison between IBS and other related technologies

| | Android/ iPhone/iPad App. | Java ME | iGoogle | Flash | Windows Mobile Widget | Opera Widget | Yahoo! Mobile | IBS |
|------------------------------------------------------------|------------------------------------------|--------------------|----------------|--------------|--------------------------------------|-------------------------|--------------------------|------------|
| Cross-Platform (Mobile device) | No | Yes | Yes | Yes | No | No | Yes | Yes |
| Cross-Platform (Mobile Device & PC) | No | No | Yes | Yes | No | Yes | Yes | Yes |
| Easy to develop Application (Gadget-based or not) | No | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Offline App. Execution | Yes | Yes | No | No | Yes | Yes | Yes | Yes |
| Adapted external Web content | No | No | No | No | No | No | No | Yes |
| Offline external Web content available | No | No | No | No | No | No | No | Yes |

The system comparison between IBS and other related technologies is shown in Table 8-11. The Android/iPhone/iPad applications are not cross-platform on mobile device, and Java ME is not cross-platform on PC. Since iGoogle and Flash are technologies based on Web browser, they cannot be executed under offline mode. Regarding the widget-type technologies, Windows Mobile Widget is not cross-platform for both mobile device and PC; Opera widget

can be executed on PC, but not on most of mobile platforms; Yahoo! Mobile includes both the traditional mobile application and Web application technologies, so it is cross-platform when using Web application and offlinable when using mobile application; however, it does not provide content adaptation mechanism for online or offline Web content. Finally, all of these requirements are satisfied in IBS.



Chapter 9. Conclusion and Future Work

9.1 Conclusion

In conclusion, we propose the USHAS system for providing user-configurable semantic home automation. Because different automation processes should be defined to meet the requirements of diverse users and home environments, the USHAS is designed to be user-configurable via Web pages. To solve the problem of interconnectivity between different devices that support distinct communication protocols, this study adopts Web Service technology, which is one of the implementations of the SOA paradigm. Web Service is selected, rather than OSGi, because WS-BPEL has been well defined for automated Web Service execution.

To improve the usability of the proposed system, this study provides the semantic automation process definition and execution, based on the USHAS home ontology. The USHAS ontology contains six first-level classes: Person, Device, Time, Environment, Event, and Location. For semantic process definition, we define the SHPL, which encompasses the definitions of variables, execution time, preconditions, and execution flow. In addition, SHPL supports the capability of expressing the concept of “all” or “none” of something with particular semantic limitations, such as “all the lights in the bed rooms”. For semantic process automation, the semantic process runtime module is designed to create WSBPEL processes automatically, based on the current home environment.

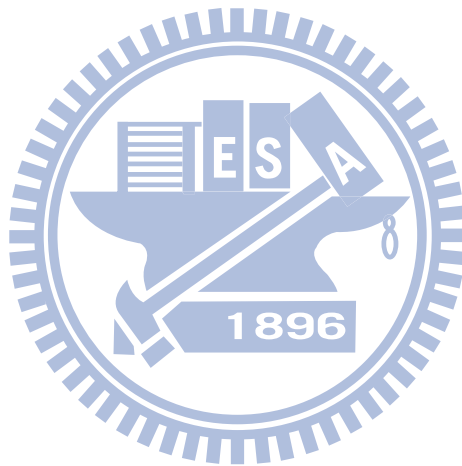
Having implemented the USHAS prototype, several scenarios are designed with corresponding examples. Using these scenarios as examples, user satisfaction and system

usability are evaluated and the results are satisfactory. Finally, USHAS is compared with other related home automation systems. Among these systems, only USHAS supports all features of automation process execution, event handling, semantic automation, the “all” quantifier, user configurable, and repeated time.

9.2 Future Work

As described in Section 7, numerous concerns remain. For future research, we will include an external facial recognition system and attempt to integrate it into USHAS to ensure that the interfaces of USHAS are adequately extensible. To solve the process conflict problem, we will model all semantic processes using Colored Petri Nets, and query the existence of conflict before including each process. If conflict exists after a process is included, then will forgo incorporating this process. Furthermore, only a few smart devices are included in the current USHAS prototype; additional smart appliances, devices, sensors, and scenarios will be included in the future. Moreover, the current user location detection is only simulated in the USHAS prototype; an RFID-based location detection system, such as the prototype designed by Ni et al. [56] will be included. In addition, the user interfaces of both the Semantic Environment Editor and Semantic Process Designer will be improved. In the Semantic Environment Editor, a graphical designer containing a map of the house with drag and drop items will be included. In the Semantic Process Designer, the concept of NLP (Neuro-Linguistic Programming) [57] will be included so that processes can be generated automatically when introducing processes via sentences and their semantic meanings in the context of the current system. Finally, we will address the security concerns and improve the security level of USHAS in the future.

There are several aspects could be improved to the mobile content adaptation sub-system. For example, the functionality of Mobile Proxy can be further extended. At this point, it is only capable of generating XHTML, and the mechanism for detecting the capabilities of mobile devices has not been integrated yet. On the other hand, our system has not been completely evaluated yet, we will accomplish the evaluation in the future.



Reference

- [1] Chao-Lin Wu, Chun-Feng Liao and Li-Chen Fu, "Service-Oriented Smart Home Architecture based on OSGi and Mobile Agent Technology," IEEE Transactions on Systems, Man, and Cybernetics - Part C, Special Issue on Networking, Sensing, and Control, vol.37, no.2 , 2007.
- [2] V. Ricquebourg, D. Menga, D. Durand, et al., The Smart Home Concept : our immediate future, E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference, pp. 23 - 28, Dec. 2006.
- [3] C. Cetina, P. Giner, J. Fons, et al., Using Feature Models for Developing Self-Configuring Smart Homes, Fifth International Conference on Autonomic and Autonomous Systems, pp. 179-88, Apr. 2009.
- [4] Li Jiang, Da-You Liu, Bo Yang, Smart home research, Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, vol.2, pp.659- 663. Aug. 2004.
- [5] J. Ryan, Home Automation, Electronics & Communication Engineering Journal, Vol.1, Issue 4, pp. 185-192, July, 1989.
- [6] Chun-Yuan Chen, Chi-Huang Chiu, Shyan-Ming Yuan, "A MOM-Based Home Automation Platform", Lecture Notes in Computer Science, Vol. 4413, 373-84, Springer Berlin / Heidelberg, 2007.
- [7] Thomas Coopman, Wouter Theetaer, Davy Preuveneers, Yolande Berbers, "A user-oriented and context-aware service orchestration framework for dynamic home automation systems," International Symposium on Ambient Intelligence (ISAmI 2010), Portugal, 16-18 June 2010, Ambient Intelligence and Future Trends. Advances in Intelligent and Soft Computing, volume 72, pages 63-70, Springer, 2010.
- [8] Chun-Feng Liao, Ya-Wen Jong, and Li-Chen Fu, "Toward a Message-Oriented Application Model and its Middleware Support in Ubiquitous Environments," International Journal of Hybrid Information Technology, vol.1, no.3, July 2008.
- [9] Chen Rui , Hou Yi-bin , Huang Zhang-qin , He Jian, "Modeling the ambient intelligence application system: concept, software, data, and network", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, v.39 n.3, p.299-314, May 2009.
- [10] Young-Guk Ha , Joo-Chan Sohn , Young-Jo Cho , Hyunsoo Yoon, "A robotic service framework supporting automated integration of ubiquitous sensors and devices," Information Sciences: an International Journal, v.177 n.3, p.657-679, February, 2007.
- [11] Yung-Wei Kao, Hui-Zhen Gu, and Shyan-Ming Yuan, "Personal based authentication by face recognition," in Proc. of 4th International Conference on Networked Computing and Advanced Information Management (NCM 2008), vol. 2, pp. 581-585, Sep. 2~4, 2008, Gyeongju, South Korea.

- [12] Yung-Wei Kao, Hui-Zhen Gu, and Shyan-Ming Yuan, "Integration of face and hand gesture recognition," in Proc. of 3rd 2008 International Conference on Convergence and hybrid Information Technology (ICCIT08), pp. 330-335, Nov. 11~13, 2008, Busan, Korea.
- [13] William T. Freeman and Craig D. Weissman, "Television control by hand gestures," in Proc. of Int'l Workshop on Automatic Face and Gesture Recognition, pp. 179-183, June 1995.
- [14] Mike P. Papazoglou, Willem-Jan van den Heuvel, "Service-Oriented Architectures: Approaches, Technologies and Research Issues," VLDB J., vol. 16, no. 3, pp. 389-415, 2007.
- [15] OSGi Alliance, OSGi Service Platform Release 3, Amsterdam, The Netherlands: IOS Press, Dec. 2003.
- [16] W3C Recommendation: Web Services Architecture. Available from: <http://www.w3c.org/TR/ws-arch/>
- [17] Diane Jordan, John Evdemon et al., "Web Services Business Process Execution Language Version 2.0," OASIS Standard, 11 April 2007. Available at: <http://docs.oasis-open.org/wsbpel/2.0/>.
- [18] Guus Schreiber and Mike Dean, "OWL web ontology language reference," 2004. Available at: <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>.
- [19] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara, "OWL-S: Semantic markup for web services," W3C Member Submission, November 2004. Available at: <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>.
- [20] X10 industry standard. Available at: <http://www.x10.com/>.
- [21] INSTEON - Wireless Home Control Solutions for Lighting, Security, HVAC, and A/V Systems. Available at: <http://www.insteon.net/>.
- [22] UPnP Forum. Available at: <http://www.upnp.org/>.
- [23] Jini Community. Available at: <http://www.jini.org/>.
- [24] Xie Li and Wenjun Zhang, "The design and implementation of home network system using OSGi compliant middleware," Consumer Electronics, IEEE Transactions on Volume 50, Issue 2, 528 – 534, May 2004
- [25] H. Ishikawa, Y. Ogata, K. Adachi, and T. Nakajima, "Building smart appliance integration middleware on the OSGi framework," in Proc. Seventh IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput., Vienna, Austria, May 2004, pp. 139-146.
- [26] Danny B. Lange and Mitsuru Oshima, "Seven good reasons for mobile agents," Communications of the ACM Vol.42 No.3, pp88-89, March 1999.

- [27] Phong Tran , Paul Greenfield , Ian Gorton, “Behavior and Performance of Message-Oriented Middleware Systems,” in Proceedings of the 2second International Conference on Distributed Computing Systems, p.645-654, July 02-05, 2002.
- [28] Jacques Ferber, “Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence,” Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.
- [29] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, 2001.
- [30] W3C. Soap version 1.2, w3c working draft 17 december 2001. Available at: <http://www.w3.org/TR/2001/WD-soap12-part0-20011217/>, 2001.
- [31] UDDI. The UDDI Technical White Paper. Available at: <http://www.uddi.org/>, 2000.
- [32] Aitor Uribarren, Jorge Parra, J.P.Uribe, Kepa Makibar, Ione Olalde, and Nati Herrasti, “Service oriented pervasive applications based on interoperable middleware,” in Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures, Dublin, Ireland, May 2006.
- [33] Jürgen Anke, Christian Sell, and Jürgen Anke, “Seamless Integration of Distributed OSGi Bundles into Enterprise Processes using BPEL,” in Proc. of Kommunikation in Verteilten Systemen - 15. ITG/GI-Fachtagung vom 26, in Bern, Schweiz - Universität Bern, February 2, March 2007.
- [34] Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., & Horrocks, I. (2000, September/October). The Semantic Web: The Roles of XML and RDF. IEEE Internet Computing 4, 63-74.
- [35] Beckett, D., "RDF/XML Syntax Specification (Revised)", W3C rdf-syntax-grammar. Available at: <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>, February 2004,
- [36] Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung, “Ontology Based Context Modeling and Reasoning using OWL,” in proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, p.18, March 14-17, 2004.
- [37] Richard Fikes, Pat Hayes, and Ian Horrocks, “OWL-QL - a language for deductive query answering on the Semantic Web,” in Journal of Web Semantics, vol. 2, Elsevier, 2004.
- [38] Sonia Ben Mokhtar, Jinshan Liu, Nikolaos Georgantas, Valérie Issarny, “QoS-aware Dynamic Service Composition in Ambient Intelligence Environments,” in Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'05), Long Beach, California, USA, November 2005.
- [39] Stefan Dietze, Alessio Gugliotta, and John Domingue, “Situation-driven processes for semantic web services,” in 3second Annual IEEE International Computer Software and Applications, COMPSAC'08 (2008), pp. 987-992, 2008.

- [40] Minsoo Kim and Minkoo Kim, "Handling Exceptions in Situation-Driven Agent Systems," International Conference on Networked Computing and Advanced Information Management, pp. 870-875, 2009 Fifth International Joint Conference on INC, IMS and IDC, 2009.
- [41] A. Garcia-Crespo, B. Ruiz-Mezcua, J.L. Lopez-Cuadrado, and I. Gonzalez-Carrasco, "Semantic model for knowledge representation in e-business," Knowledge Based Systems 24(2), pp. 282-296, 2011.
- [42] Tom Rodden, Andy Crabtree, Terry Hemmings, Boriana Koleva, Jan Humble, Karl-Petter Akesson, and Pär Hansson, "Configuring the Ubiquitous Home," in Proceedings of the 6th International Conference on Designing Cooperative Systems (Hyères, France, May 11--14, 2004). COOP'04. IOS Press, Amsterdam, 227—241, 2004.
- [43] Zoé Drey, Julien Mercadal, and Charles Consel, "A Taxonomy-Driven Approach to Visually Prototyping Pervasive Computing Applications," in Proceedings of the IFIP TC 2 Working Conference on Domain-Specific Languages, Oxford, UK, July 15-17, 2009.
- [44] Patrick Th., Eugster Pascal, A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec, "The Many Faces of Publish/subscribe," ACM Computing Surveys (CSUR), Volume 35, Issue 2, pp.114 – 131, 2003.
- [45] Lukasz Opyrchal and Atul Prakash, "Secure distribution of events in content-based publish subscribe systems," in Proc. of the 10th conference on USENIX Security Symposium, p.21-21, Washington, D.C. , August 13-17, 2001.
- [46] Mariano Cilia , Christof Bornhövd , Alejandro P. Buchmann, Event Handling for the Universal Enterprise, Information Technology and Management, v.6 n.1, p.123-148, January 2005.
- [47] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi, "SOUPA: Standard ontology for ubiquitous and pervasive applications," in First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), pages 258–267, August 2004.
- [48] Tadao Murata, "Petri Nets: Properties, Analysis and Applications," in Proc. of the IEEE, 77(4), pp.541-580, April 1989.
- [49] Kurt Jensen , "Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use", Vol. 1 : Basic Concepts, in EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1992.
- [50] Jean-Marc Seigneur, Christian Damsgaard Jensen, Stephen Farrell, Elizabeth Gray, and Yong Chen, "Towards Security Auto-Configuration for Smart Appliances," in Proceedings of the Smart Objects Conference 2003, 2003.
- [51] W. M. Fitzgerald, S. N. Foley, "Aligning Semantic Web Applications with Network Access Controls," Computer Standards & Interfaces, 33(1), pp.24-34, 2011. [52] Jonathan M. McCune, Ning Qu, Yanlin Li

- [52] Anupam Datta, Virgil D. Gligor, and Adrian Perrig, "TrustVisor: Efficient TCB Reduction and Attestation," in Proceedings of IEEE Symposium on Security and Privacy (Oakland 2010), 2011
- [53] Ángel García-Crespo, Juan Miguel Gómez-Berbís, Ricardo Colomo-Palacios, Giner Alor-Hernández, "SecurOntology: A semantic web access control framework," *Computer Standards & Interfaces*, 33(1), pp. 42-49, 2011.
- [54] Carlos Blanco, Joaquín Lasheras, Eduardo Fernández-Medina, Rafael Valencia-García, José Ambrosio Toval Álvarez, "Basis for an integrated security ontology according to a systematic review of existing proposals," *Computer Standards & Interfaces*, 33(4), pp.372-388. 2011.
- [55] Yi Qian, Yuming Xu, Zheng Wang, Geguang Pu, Huibiao Zhu, and Chao Cai, "Tool Support for BPEL Verification in ActiveBPEL Engine," in Proceedings of The 18th Australian Software Engineering Conference(ASWEC' 07), pp. 90-100. 2007.
- [56] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *Wireless Networks* 10, pp.701–710, 2004.
- [57] Richard Bandler and John Grinder, "Reframing: Neuro-Linguistic Programming and the Transformation of Meaning," Real People Press, ISBN 0-911226-25-7, 1981.
- [58] Jen-Kai Wu, Chia-Feng Lin, Tzu-Han Kao and Shyan-Ming Yuan "A Visualized Kit for Developing Applications on Multiple Mobile Devices", Conference on Web Information System and Application (WISA) 2006.
- [59] Jen-Kai Wu, Tzu-Han Kao, Chi-Han Kao, Shyan-Ming Yuan, A GUI-based application development toolkit for small and mobile devices, Proc. of R.O.C. Military Academy Basic science conference on April 2004
- [60] Tzu-Han Kao, Sheng-Po Shen, Shyan-Ming Yuan, and Po-Wen Cheng, An XML-based Context-Aware Transformation Framework for Mobile Execution Environments, Lecture Notes in Computer Science (LNCS) of Springer-Verlag (APWeb 2003, Xian, China), Vol. 2642 / 2003, pp. 132 – 143.
- [61] Maryam Kamvar, Shumeet Baluja, "A Large Scale Study of Wireless Search Behavior: Google Mobile Search", Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI), 2006.
- [62] M.H. Bulter, Current Technologies for Device Independence, Hewlett-Packard Company, Technical Publications Department HP Labs Research Library, 1501 Page Mill Road Palo Alto, CA 94304-1126, 2001 March, <http://www.hpl.hp.com/techreports/>, <http://www-uk.hpl.hp.com/people/marbut/currTechDevInd.htm>.
- [63] M. Bulter, F. Giannetti, R. Gimson, T.Wiley, Device Independence and the Web, IEEE Internet Computing, IEEE Computer ociety, 10662 Los Vaqueros Circle P.O. Box 3014 Los Alamitos, CA 90720-1314, 2002 October, <http://www.computer.org/>.

- [64] R. Gimson, S.R. Finkelstein, S. Maes, L. Suryanarayana, Device Independence Principles, W3C Working Draft 18, W3C (MIT, ERCIM, Keio), in the United States, in Europe, and in Japan, 2001 September, <http://www.w3.org/>.
- [65] B.N. Schilit, N. Adams, R. Want, Context-aware computing applications, Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, IEEE Computer Society, 10662 Los Vaqueros Circle P.O. Box 3014 Los Alamitos, CA 90720-1314, 1994, pp. 85 – 90, <http://www.computer.org/>.
- [66] Y. Hwang, E. Seo, and J. Kim, WebAlchemist: A Web Transcoding System for Mobile Web Access in Handheld Devices, Proc. ITCOM, SPIE-The Int'l Soc. for Optical Eng., 2001, pp. 37-46.
- [67] Jinlin Chen , Baoyao Zhou , Jin Shi , Hongjiang Zhang , Qiu Fengwu, “Function-based object model towards website adaptation”, Proceedings of the 10th international conference on World Wide Web, p.587-596, May 01-05, 2001, Hong Kong,
- [68] T. Bickmore, A. Girgensohn, and J.W. Sullivan, Web Page Filtering and Reauthoring for Mobile Users, Computer J., vol. 42, no. 6, 1999, pp. 534-546.
- [69] Yu Chen, Xing Xie, Wei-Ying Ma, Hong-Jiang Zhang, Adapting Web Pages for Small-Screen Devices, IEEE Internet Computing, vol. 09, no. 1, pp. 50-56, Jan/Feb, 2005.
- [70] Tzu-Han Kao, Shyan-Ming Yuan, Designing an XML-based context-aware transformation framework for mobile execution environments using CC/PP and XSLT, Computer Standards & Interfaces, Volume 26, Issue 5, September 2004, Pages 377-399
- [71] Tzu-Han Kao, Yung-Yu Chen, Tsung-Han Tsai, Hung-Jen Chou, Wei-Hsuan Lin, Shyan-Ming Yuan, PUML and PGML: Device-independent UI and Logic Markup Languages on Small and Mobile Appliances Lecture Notes in Computer Science (LNCS) of Springer-Verlag, The 2005 IFIP International Conference on Embedded And Ubiquitous Computing (EUC-05), Nagasaki, Japan, 6-9 December 2005.
- [72] James Clark , Steve DeRose , W3C. XML Path Language (XPath) Version 1.0. <http://www.w3.org/TR/xpath>
- [73] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, VIPS: a vision-based page segmentation algorithm, Microsoft Technical Report, MSR-TR-2003-79, 2003.
- [74] M. Lai and D. Wong. Slicing tree is a complete floorplan representation. In DATE '01: Proceedings of the conference on Design, automation and test in Europe, pages 228–232.
- [75] Tzu-Han Kao, Shyan-Ming Yuan, Automatic adaptation of mobile applications to different user devices using modular mobile agents, in Software: Practice and Experience, 35(14), pp.1349—1391, Nov. 2005
- [76] Google Mobile Content Proxy, <http://www.google.com/gwt/n>
- [77] Jin Jing, Abdelsalam Helal, Ahmed Elmagarmid. Client-server computing in mobile environments. ACM Computing Surveys, 1999, 31(2):117~157

- [78] H. Maruyama, K. Tamura, N. Uramoto, M. Murata, A. Clark, et al., XML and Java Second Edition: Developing Web Applications, Addison-Wesley, 75 Arlington Street, Suite 300 Boston, MA 02116, 2002, <http://www.awprofessional.com/>.
- [79] J. Chen et al., Function-Based Object Model Towards Website Adaptation, Proc. 10th WWW Conf., ACM Press, 2001, pp. 587-596.
- [80] Rong-Hong Jan, Ching-Peng Lin, and Maw-Sheng Chern, "An optimization model for Web content adaptation", Computer Networks, vol. 50, no. 7, 2006, pp. 953-965. (NSC 93-2219-E-009-002 and NSC 93-2752-E-009-005-PAE)
- [81] Yue-Shan Chang, Min-Huang Ho, Wen-Chen Sun, Shyan-Ming Yuan, Supporting Unified Interface for Wrapper Generator in Integrated Information Retrieval, in Computer Standards and Interfaces, 24(4), pp.291-303, Sep. 2002
- [82] Wai Yip Lum, Francis C.M. Lau, "A Context-Aware Decision Engine for Content Adaptation," IEEE Pervasive computing, Vol. 1, No. 3, July-September 2002.
- [83] Yui-Wah Lee, Girish Chandranmenon and Scott C. Miller "GAMMAR: A Content-adaptation Server for Wireless Multimedia Applications", Bell Laboratories.
- [84] Ruey-Kai Sheu, Yue-Shan Chang, Shyan-Ming Yuan, Managing and Sharing Collaborative Files Through WWW, in Future Generation of Computer Systems, 17(8), pp. 1039-1049, May 2001
- [85] Tsun-Yu Hsiao, Nei-Chium Perng, Winston Lo, Yue-Shan Chang, Shyan-Ming Yuan, A new development environment for an event-based distributed system, in Computer Standards & Interfaces, 25(4), pp.345-355, Aug. 2003
- [86] Yue-Shan Chang, Reuy Wu, Kai-Chih Liang, Shyan-Ming Yuan, Magic Yang, CODEX: Content-Oriented Data EXchange Model on CORBA, in Computer Standard & Interface , 25(4), pp. 329-343, Aug. 2003
- [87] Benbunan-Fich, R., "Using protocol analysis to evaluate the usability of a commercial Web site", Information & Management, (39), pp 151-163, 2001.
- [88] FERET database, http://www.itl.nist.gov/iad/humanid/feret/feret_master.html
- [89] ORL database, <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>