Computer Standards & Interfaces 34 (2012) 212–224

Contents lists available at SciVerse ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A Web-based, Offline-able, and Personalized Runtime Environment for executing applications on mobile devices

Yung-Wei Kao [a,*], ChiaFeng Lin [a], Kuei-An Yang [a], Shyan-Ming Yuan [a,b]

[a] *Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan*
[b] *College of Computing & Informatics, Providence University, 200 Chung Chi Rd., Taichung 43301, Taiwan*

ARTICLE INFO

Article history:
Received 18 April 2011
Received in revised form 20 June 2011
Accepted 23 August 2011
Available online 8 September 2011

Keywords:
Mobile device
Offline mobile Web application
Mobile content adaptation

ABSTRACT

An increasing number of people use cell phones daily. Users are not only capable of making phone calls, but can also install applications on their mobile phones. When creating mobile applications, developers usually encounter the cross-platform incompatibility problem (for example, iPhone applications cannot be executed on the Android platform). Moreover, because mobile Web browsers have increasingly supported more and more Web-related standards, Web applications are more possible to be executed on different platforms than mobile applications. However, the problem of Web application is that it cannot be executed in offline mode. This study proposes a Web-based platform for executing applications on mobile devices. This platform provides several services for developers such as offline service, content adaptation service, and synchronization service. With the help of the proposed platform, application developers can develop and publish offline Web applications easily with simplified external Web content and synchronization capability.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Benefits from the advancements of mobile devices, mobile platform, and wireless network technologies [1,2,3,4,5] in recent years, the market of mobile devices are growing dramatically: an increasing number of people are using at least one mobile phone, or even tablet PC, daily. Unlike traditional cell phones, users are not only capable of making phone calls or sending and receiving short messages, but can also install applications from the online markets. Apple iPhone, iPad, and Google Android [6,7,8] are three major examples of this type of mobile device platform. Consumers are not only willing to use applications on mobile devices, but also willing to pay for them. Creating an interesting application that can be executed on different mobile devices within a very short time is a significant issue of application development. However, developers must usually write mobile applications in a specific programming language for a specific mobile platform; that is, iPhone applications in Object C, and Android applications in Java. Since applications written in Object C cannot be executed on the Android platform, developers have to spend twice the time writing the same application in two different languages. Moreover, users may have to buy their next mobile devices with the same platforms as their current ones because other platforms do not support their favorite applications.

Due to the variety of mobile platforms, developing cross-platform mobile applications is a challenge [9]. Certain languages (such as Java ME ([10,11]) cross mobile platforms, but not PC platforms. To provide an application that allows users to access their data (such as email) both on their PC and mobile device, a Web server and Web application are usually necessary. Users can consequently use the Web browser on their PC and a mobile application on their mobile device to access the same data. This would be more straightforward if the Web application could be executed on the mobile Web browser; in this manner, users do not have to install one more non-cross-platform mobile application on their mobile devices.

iGoogle [12] provides a Web application execution platform where users can select which applications they are interested in and execute them on their personalized homepages. Moreover, there is a mobile version of iGoogle for mobile users, so users can execute these selected Web applications on their mobile phones. However, the major problem of executing Web applications on mobile devices is that they cannot be executed without an Internet connection. In certain situations, such as on an airplane, or in a basement where the Internet signal is weak or unavailable, mobile applications can usually be executed, though Web applications cannot. Moreover, if the user's data plan does not include unlimited data transmission, the user may be unwilling to execute Web applications on their mobile phones.

To solve the offline Web-browsing problem, the HTML5 [13,14] standard includes the Web storage and Web SQL database APIs. However, most current Web browsers (especially mobile Web browsers) have not yet adopted HTML5. To achieve the same goal of offline Web-browsing, a browser plug-in, the Google Gears solution is developed. The problem of Google Gears is that not all Web pages can be browsed offline; only the pages that support the server-side Google

* Corresponding author.
E-mail address: ej3muse@gmail.com (Y.-W. Kao).

Gears library can. In order to solve this problem, the Gears-Monkey system was developed. However, there are still several drawbacks of Gears-Monkey requires a high level of script-writing skill, and is not supported on mobile phones, and does not provide batch processing for multiple pages.

In our previous works [15,16], we proposed an authoring tool [15] for mobile content adaptation, and an offline mobile Web-browsing framework [16] for browsing offline Web pages on mobile devices. This paper proposes Web-based, Offline-able, and Personalized Runtime Environment (WOPRE), which includes these previous works with some modifications and extensions, as two subsystems for solving the problems previously mentioned. There are four objectives of WOPRE:

1. With the help of WOPRE, applications should be executable on different platforms including PCs and mobile devices.
2. An online market, similar to the Apple App Store and Google Android Market, should allow developers to publish their applications and users to install desired applications.
3. Full, or at least partial, operations of applications should be executable in offline mode.
4. A content adaptation mechanism for external Web content should be provided.

The research contains seven sections. In Section 2, we introduce some backgrounds and related works of WOPRE. The high level and detailed system design issues are described in Sections 3 and 4 respectively. In Section 5, we discuss the system implementation and demonstration. The system evaluation is presented in Section 6. Finally, we end up with a conclusion and discuss the future works of WOPRE in Section 7.

2. Backgrounds and related works

2.1. Cross-platform technologies

It is difficult for programmers to write a program only once before executing it on different mobile devices [9]. The Java language is designed for cross-platform application execution, and Java ME is a cross-platform language for developing mobile applications [11]. There are already many tools available for developing Java ME applications [17]; certain toolkits which include emulators are available for rapid MIDlet development. However, Java ME only crosses mobile platforms, not PC platforms; to provide an application that allows a user to access data on both PC and mobile devices, developers must usually use Web technologies as intermediaries.

Based on Java ME, Mojax Moblets [18,19] provides an AJAX framework so that developers can write their applications in widgets: a much easier approach to write applications. Similar to Moblets, Yahoo! Go [20] is also a widget-based platform for executing applications on mobile devices. After installing Yahoo! Go, mobile users can easily access Yahoo! Web-based services, such as Yahoo! News, Yahoo! E-mail, and Flickr, on their mobile phones. Developers can also create applications by following the standard of Yahoo! Mobile Widgets [21], and allowing users to execute these applications on various mobile and PC platforms.

Since Web technologies and widgets are good solutions for developing cross-platform applications for mobile users, it is more straightforward to execute Web applications via mobile Web browsers. In this manner, users do not have to install one more platform-dependent application on their phones. Similar to Yahoo! Go, Yahoo! Mobile [22] users can install a platform on their phones for executing widgets; moreover, they can also execute widgets directly on the Yahoo! Mobile Website via Web browsers. Other widget technologies such as Windows Mobile Widget [23] and Opera Widget [24] are similar to Yahoo Go! and Yahoo! Mobile; they have to provide local runtime environments to execute their widgets. Another related system is the Google iGoogle. Developers can create applications by following the standard of Google Gadget [25], and upload them onto the iGoogle platform. Furthermore, since Google Gadget integrates several Google APIs such as Google Maps API [26], Google AJAX Search API [27], Google Calendar API [28], and Google Translate API [29], it is extremely easy to invoke other Google services in Google Gadgets. For mobile users, iGoogle also provides a mobile-version Web site, so users can execute their gadgets on their mobile phones via mobile Web browsers.

2.2. Offline mobile Web browsing

To make offline Web content and applications available on mobile devices, local databases and storage should be provided [30]. HTML5 is a new Web standard designed to supplant the existing HTML 4.01, XHTML 1.0 and DOM Level 2 HTML standards. HTML5 aims to reduce the need for browser plug-in-based Rich Internet Applications (RIAs), such as applications based on Adobe Flash [31], Microsoft Silverlight [32], and Sun JavaFX [33]. Web Storage and Web SQL Database are two new APIs provided in HTML5; they are described as follows:

2.2.1. Web SQL database

The browsers that support HTML5 should include a local SQLite database. With this database, client-side applications can store information such as user data via standard SQL communications.

2.2.2. Web storage

Name–Value-based data storage should be supported for storing Web content such as HTML, JavaScript, and image files. The stored data is non-volatile; even though the browser or the phone is turned off, the data still remains in the storage.

However, HTML5 has most Web browsers and especially mobile browsers, do not yet support HTML5. Google Gears [34], an open-source browser extension that supports the offline execution of web applications, is an alternative solution for implementing the offline mechanism. It can be replaced by HTML5 in the future when most mobile browsers support HTML5. Currently, Google Gears is able to be executed on personal desktops, laptops, and handheld devices. It can store and provide application resources locally, and run asynchronous JavaScript to improve application responsiveness. There are three major components of Google Gears:

A. LocalServer caches and serves application resources (HTML, JavaScript, images, etc.) locally.
B. Database stores record-based data locally in a fully searchable relational database.
C. WorkerPool makes Web applications more responsive by performing resource-intensive operations asynchronously.

However, only Web pages that support the server-side Google Gears library can be browsed offline. Currently, only a few Web sites support this library, and most of them are Google Web-based services. The Gears-Monkey [35] system can solve this problem for offline Web-browsing; its design is based on Google Gears and Greasemonkey [36]. An extension of the Mozilla Firefox Web browser, Greasemonkey, is a user script manager that allows users to customize the manner in which a page is displayed by using JavaScript. Based on Google Gears and Greasemonkey, the server-side Google Gears library can be inserted into any downloaded Web page. Thus, users' favorite Web sites can be captured and browsed offline. However, there are several drawbacks of Gears-Monkey. First, it is only supported on PCs and cannot be executed on mobile devices. Second, it requires users to write scripts by themselves; users may not have enough skill to write scripts. Finally, users must download offline data site by site; no batch processing is provided.

A previous work [16] by these authors proposed an offline mobile Web-browsing framework to solve the problems of Google Gears and Gears-Monkey. In this work, there are several drawbacks. First, users

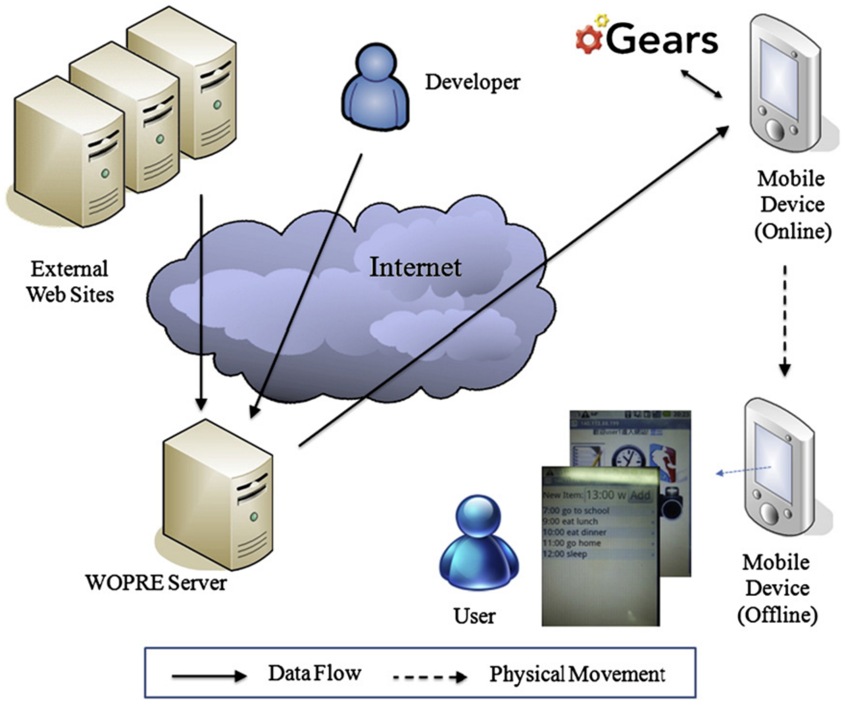


Fig. 1. System overview.

must define their interest lists; otherwise, the framework does not know which Web pages should be downloaded. Second, users must browse offline Web pages via the Offline Web Page Viewer because the offline data is republished under a new domain. Finally, the work only focused on Web page browsing, but not Web application execution; in other words, it did not provide a synchronization mechanism for allowing users to synchronize the data that has been created or modified offline.

Rich Internet Applications (RIAs) [37,38] are Web applications that have many of the characteristics of desktop applications; they are typically delivered by site-specific browsers, browser plug-ins, sandboxes, or virtual machines. Adobe Flash [31], Sun JavaFX [33], and Microsoft Silverlight [32] are currently the three major RIA platforms. Other popular RIA techniques include Adobe AIR [39] and AJAX [40]. RIAs usually improve the richness of data, business logic, communication, and presentation of Web pages. Although RIAs can also be designed for offline Web data management, it is not their main purpose. Conversely, Google Gears focuses on offline data management, and provides a better solution for handling offline issues (e.g., providing a well-established offline data pre-fetch mechanism). Therefore, the authors chose Google Gears to implement the client-side offline data management platform.

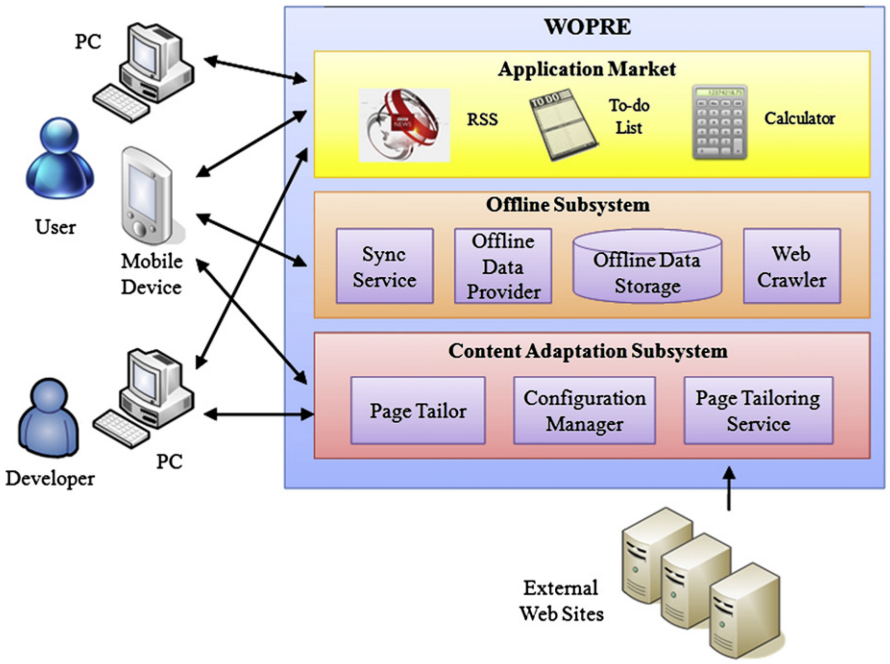


Fig. 2. High level system architecture.

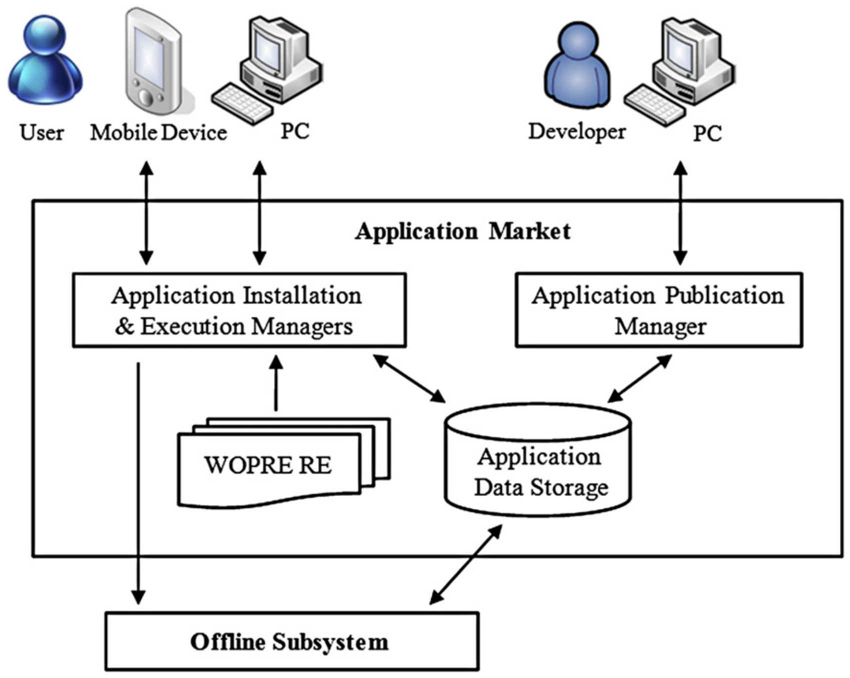


Fig. 3. The detailed architecture of Application Market.

2.3. Mobile Web content adaptation

Mobile content adaptation [41,42,43] 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 [44] 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 [5]. 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

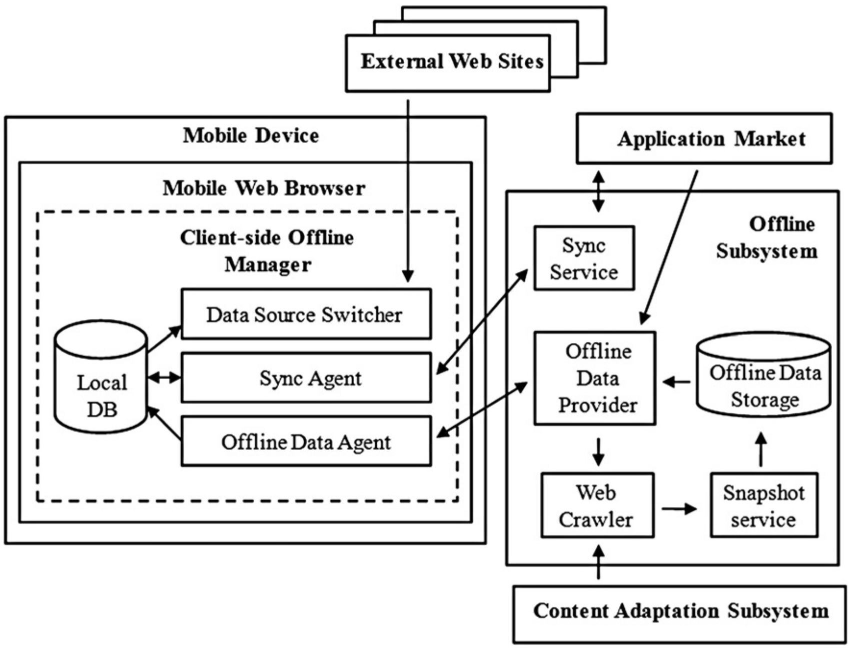


Fig. 4. The detailed architecture of Offline Subsystem.

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™ [45] 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 adaption 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. [46] 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.

In our previous work [15], we proposed a mobile content adaptation system that provides an authoring tool and allows users to select only a part of Web content in a specified Web page for mobile Web-browsing. This system includes a Web proxy for content filtering so that only the specified regions of specified Web pages can be displayed. Although this work also proposed the VIPS algorithm for automatic mobile Web page adaptation for unspecified pages based on the patterns extracted from users' previous selections, users still must select some paths in at least one Web page to create the first pattern.

3. High level system design

3.1. System overview

The system overview is shown in Fig. 1. The WOPRE platform provides some libraries for offline data management and content adaptation so that developers can include these functionalities into their applications. Also, developers can create their applications based on the Google Gadget standard so that the existing iGoogle applications can be executed on our WOPRE platform. Users can select which applications that they are interested in, and install them onto their mobile devices. During the installation process of application, the application gadgets, WOPRE runtime, and external Web content (if they are included within applications), are downloaded to the Google Gears local database. Therefore, when the mobile device is under the offline mode, the selected applications can still be executed upon the local WOPRE runtime.

3.2. System architecture

The high level system architecture is shown in Fig. 2. There are three main Subsystems in WOPRE: Application Market, Offline Subsystem, and Content Adaptation Subsystem. The application market provides an interface for developers to publish their applications and users to install applications. The Offline Subsystem is mainly responsible for offline data management such as offline content preparation and offline data synchronization. If some external Web content is included into an application, the developer can use the Content Adaptation Subsystem to provide a simplified content representation. The detailed system architectures of the application market, Offline Subsystem, and Content Adaptation Subsystem are shown in Figs. 3, 4, and 5 respectively.

Fig. 3 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 Subsystem 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 WOPRE Runtime Environment (WOPRE RE).

In Fig. 4, before using the Offline Subsystem, 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.

There are several differences between the Offline Subsystem and our previous offline framework [16]. First, because all the Web pages which users may browse are under the control of developers, users do not have to define an Interest List in advance. Second, in our previous offline framework, the offline Web content are republished under a new domain; it is very difficult to ask users to browse those pages with different URLs without the help of the Offline Web Page Viewer. However, in the Offline Subsystem of WOPRE, since users always go to the homepage of WOPRE first, which is under the WOPRE domain, the role of Offline Web Page Viewer is replaced by the homepage; therefore, no additional plug-in is required to be installed and used. Finally, our previous offline framework only focused on offline Web browsing, but not offline Web application execution, so there was no synchronization mechanism supported.

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 three paths are used to specify which content is important.

Fig. 5 shows the detailed architecture of the Content Adaptation Subsystem. The Content Adaptation Subsystem of WOPRE 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 Subsystem, developers can do so by the following steps. First, they have to open the external Web

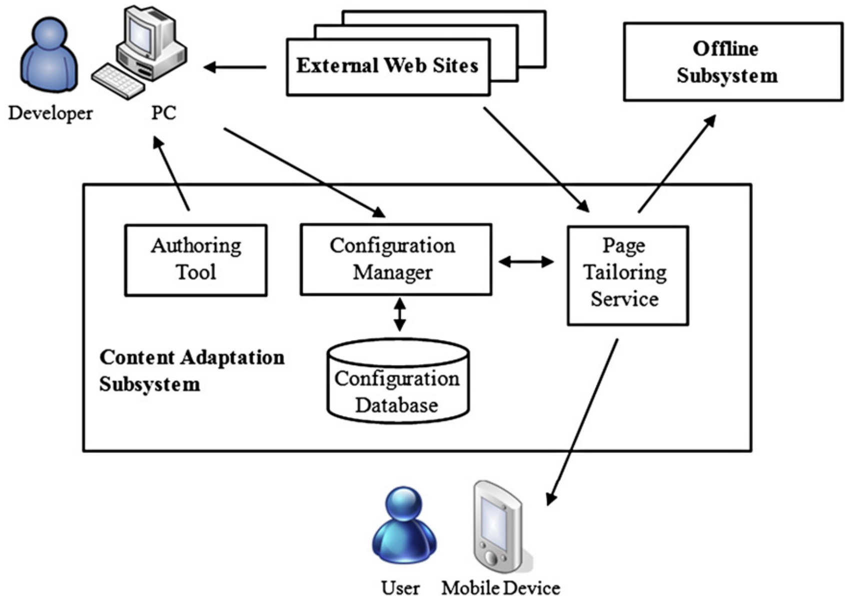


Fig. 5. The detailed architecture of Content Adaptation Subsystem.

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 Subsystem 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.

The major difference between the Content Adaptation Subsystem and the authoring tool in our previous work [15] is that previously, users have to specify which parts of Web pages are important, but now, it's the developers' responsibility in WOPRE. In our previous work, although we proposed the VIPS algorithm for automatic mobile Web page adaptation based on the patterns extracted from users' previous selections, users still have to select some paths in at least one page for creating the first pattern. However, in WOPRE, since users always link to external Web pages via applications, developers can make the decisions of selection for users, so that users don't have to do so by themselves.

3.3. System limitation

The major limitation of WOPRE 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.

4. Detailed system design

In this section, we describe the detailed system design of WOPRE, including the process of application publication and installation. In order to explain how to use the functionalities of Offline Subsystem and Content Adaptation Subsystem, we develop two applications for demonstration; therefore, we also describe the detailed design of these two applications as well as how they interact with our Subsystems.

4.1. Application publication process

Fig. 6 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 Subsystem later when this application is installed by any user. Since most of the functionalities of WOPRE are provided via JavaScript interfaces, it is very easy to use them within gadgets. Also, developers can use the Content Adaptation Subsystem if they want to include some external Web content into their applications. By using the Authoring Tool, developers can login to WOPRE, 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 Subsystem.

4.2. Application installation process

Fig. 7 shows how users can install applications in WOPRE. For each time a user decides to install an application, the application market asks the Offline Subsystem 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

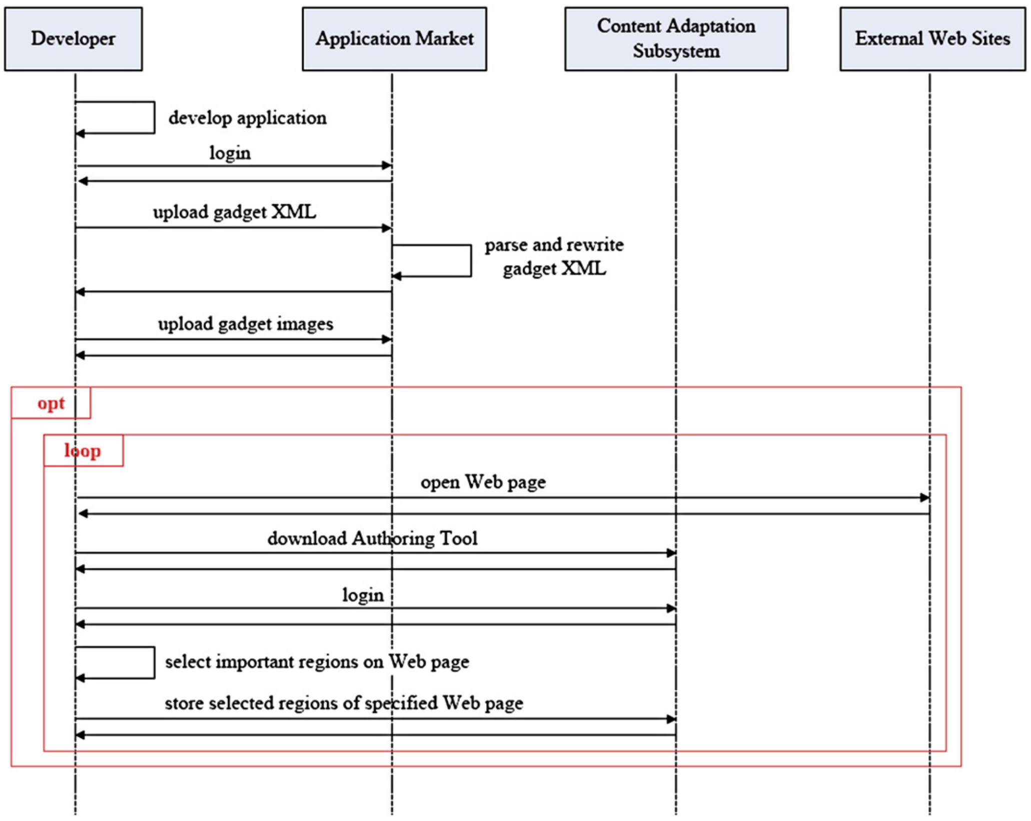


Fig. 6. The sequence diagram of application publication.

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 craws Web content via the Page Tailoring Service of Content Adaptation Subsystem, the returned Web

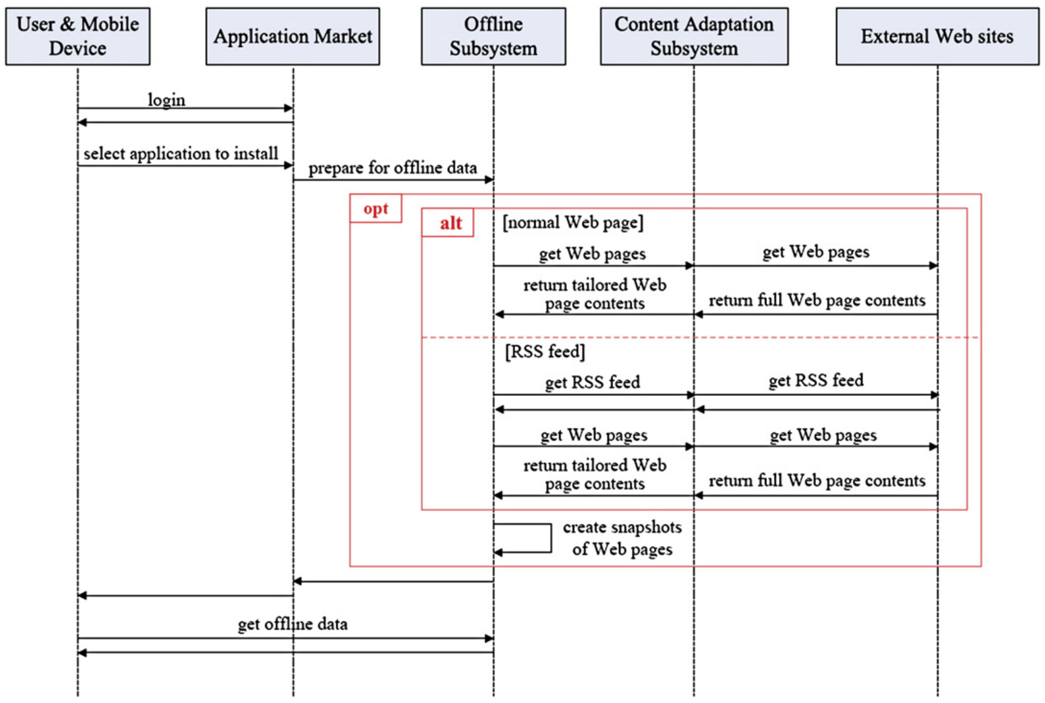


Fig. 7. The sequence diagram of application installation.

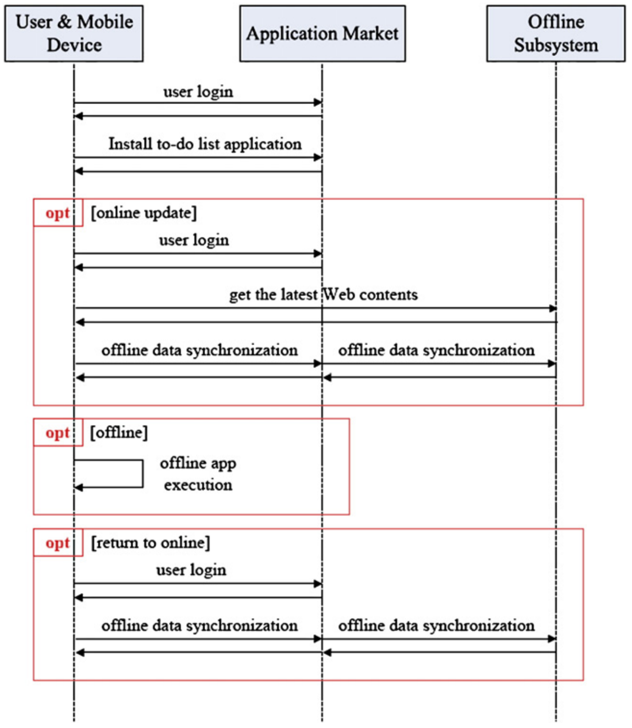

User & Mobile Device
Application Market
Offline Subsystem
user login
Install to-do list application
opt [online update]
user login
get the latest Web contents
offline data synchronization
offline data synchronization
opt [offline]
offline app execution
opt [return to online]
user login
offline data synchronization
offline data synchronization

Fig. 8. The sequence diagram of using the to-do list application.

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 Subsystem.

4.3. Synchronization service

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 WOPRE, we provide a simple and basic synchronization service so that developers can use this service directly without maintaining their own online servers.

In WOPRE, all the user data for application synchronization is stored in the AppSync table. 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.

4.4. The to-do list application

A "to-do list" application is designed for demonstrating the capability of Offline Subsystem. 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 WOPRE so that it can be executed offline

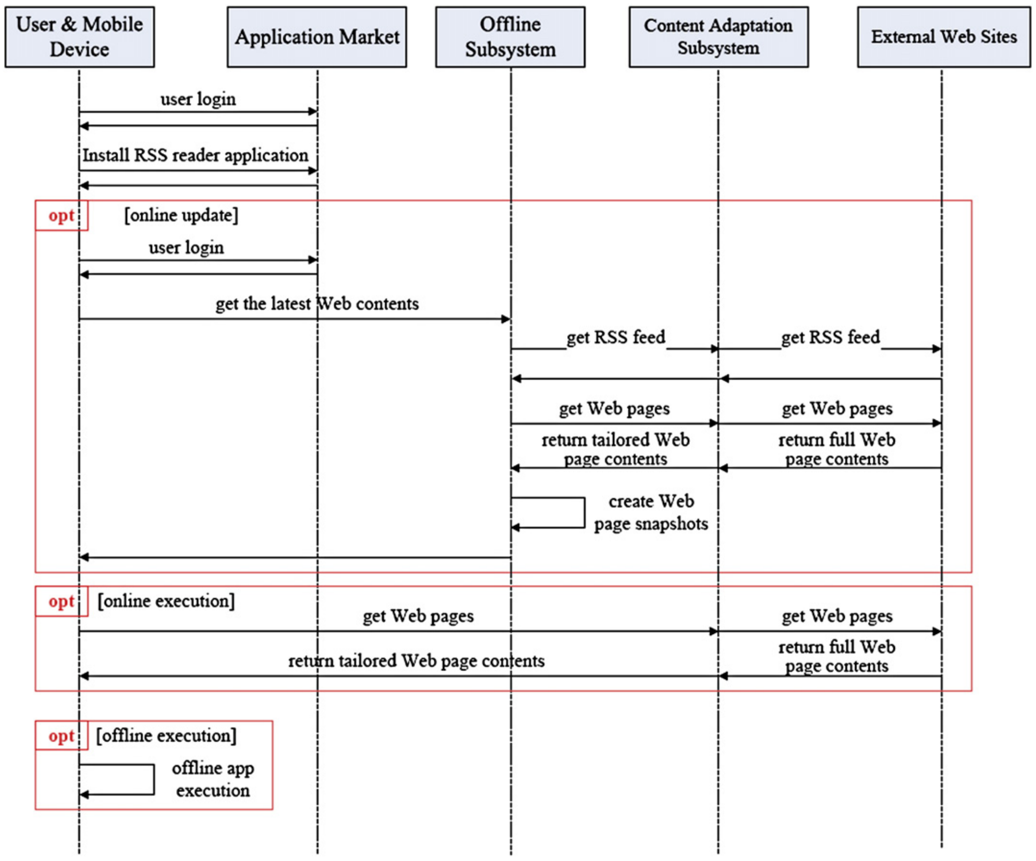


Fig. 9. The sequence diagram of using the RSS reader application.

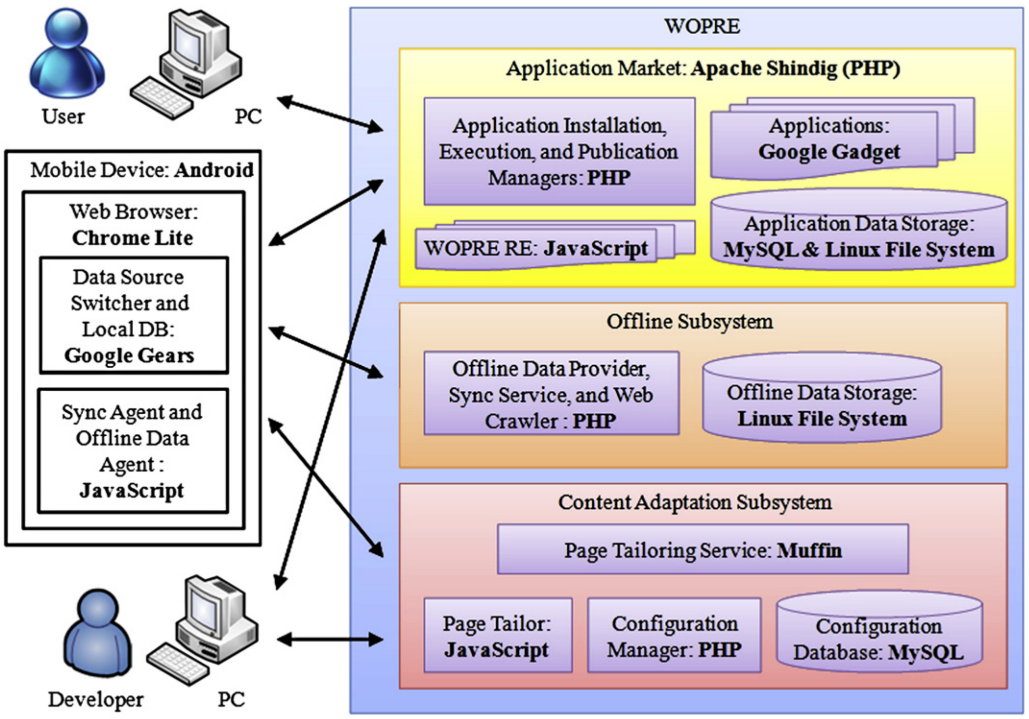

User
PC
Mobile Device: Android
Web Browser: Chrome Lite
Data Source Switcher and Local DB: Google Gears
Sync Agent and Offline Data Agent : JavaScript
Developer
PC
WOPRE
Application Market: Apache Shindig (PHP)
Application Installation, Execution, and Publication Managers: PHP
Applications: Google Gadget
WOPRE RE: JavaScript
Application Data Storage: MySQL & Linux File System
Offline Subsystem
Offline Data Provider, Sync Service, and Web Crawler : PHP
Offline Data Storage: Linux File System
Content Adaptation Subsystem
Page Tailoring Service: Muffin
Page Tailor: JavaScript
Configuration Manager: PHP
Configuration Database: MySQL

Fig. 10. The system implementation of WOPRE.

on mobile devices. The sequence diagram of using it is shown in Fig. 8. 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.

4.5. The RSS reader application

We design a RSS reader application of BBC News for demonstrating the capability of Content Adaptation Subsystem. 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 Subsystem 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 Subsystem as Web proxy; in this way, no matter these pages are browsed online or offline, they are adapted for mobile devices (Fig. 9).

5. System implementation and demonstration

5.1. System implementation

Fig. 10 shows the system implementation technologies of WOPRE. On the server side, the Apache Shindig [47] platform is used to implement the application market. Apache Shindig is an open-source third-party project designed for executing Google Gadget applications. Because the PHP version of Shindig is used, most of the server-side components of WOPRE are implemented in PHP. Due to AJAX security issues, JavaScript codes cannot access Web content under other domains. Therefore, for each gadget uploaded to WOPRE, the application installation manager examines all URLs referenced in the gadget, and rewrites all external URLs to local ones for offline browsing. However, it is better to provide the original URLs for users under online mode so that real-time information is available. Hence, whether to use the original or rewritten URL depends on whether the user is online. Table 2 shows two examples of URLs before and after rewriting. If any external link is included within the "<a>" element, it is automatically rewritten into a short JavaScript program that provides a different URL based on current online status. Conversely, if an external link is referenced within an RSS XML document that is going to be processed offline, the two URLs are concatenated by the "||" symbols, and the offline WOPRE runtime analyzes it and uses a different URL based on current online status.

The Web Crawler of Offline Data Provider is implemented by using SiteShoter [48]. The most challenging problem of Web crawling is deciding how many levels should be crawled. For example, if Page A has a link to page B, and page B has a link of page C, and page A is crawled with level 1, then only the content in Page A is pre-fetched and

Table 1
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	

Table 2
Examples of URL rewriting.

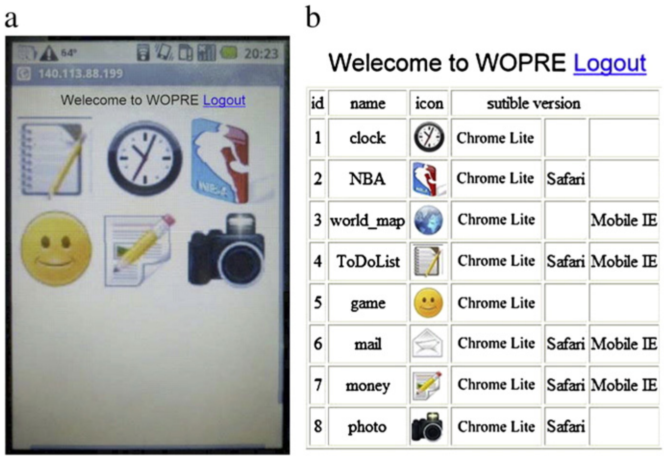
Before rewriting in HTML	<a href="http://sports.espn.go.com/nba/news/story?id=6622805&campaign= rss&source=ESPNHeadlines" level="1">click me</a>
After rewriting in HTML	<script language="JavaScript"> var status=online_or_not(); if (status==1){document.write("<a href=\"http://sports.espn.go.com/nba/news/story?id6622805&campaign=rss&source=ESPNHeadlines\" level=\"1\">click me</a>"); }else{document.write("<a href="http://www.wopre.com/offline/sports.espn.go.com /nba/news/story-9274698167.jpg\">click me</a>");} </script>
Before rewriting in RSS	<link level="1">http://sports.espn.go.com/nba/news/story?id=6622805&campaign=rss&source=ESPNHeadlines</link>
After rewriting in RSS	<link>http://sports.espn.go.com/nba/news/story?id=6622805&campaign=rss &source=ESPNHeadlines || http://www.wopre.com/offline/sports.espn.go.com/nba/news/story-9274698167.jpg </link>

available offline; if the level is 2, then the content of both A and B, but not C, is available offline. Images should be processed at different levels because images are usually regarded to be on the current page, but they are actually treated as external links in HTML documents. Since different depths of levels are required in different applications, it is difficult to automatically decide them by WOPRE. Therefore, we allow developers themselves to specify the level of each external link in the "level" attribute. Examples of using this attribute are also shown in Table 1. Thus, WOPRE can know how many levels should be pre-fetched. At the last level, Level 1, since external links do not work offline, WOPRE simply takes a screenshot of the entire Web page with the width of mobile screen specified so that it can be processed faster. Finally, for all links where the "link" attributes are not specified, they are treated as Level 1.

In the Content Adaptation Subsystem, Muffin [49], a Java-implemented Web proxy, is selected as the Page Tailoring Service. On the mobile phone-side, Google Gears plays the role of Data Source Switcher and Local DB.

5.2. System demonstration

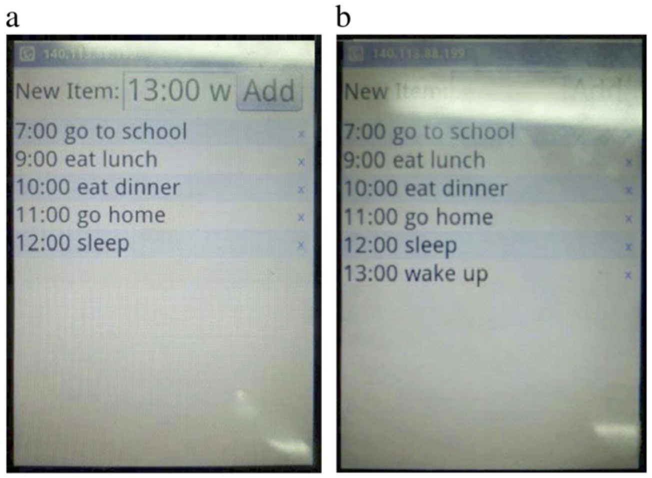
The mobile version homepage and PC version homepage are shown in Fig. 11. In the mobile homepage, the application execution manager displays only icons of applications, so that it matches the traditional user experience of using mobile applications. In the PC version of homepage, the application execution 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



Welecome to WOPRE Logout

id	name	icon	sutible version		
1	clock		Chrome Lite		
2	NBA		Chrome Lite	Safari	
3	world_map		Chrome Lite		Mobile IE
4	ToDoList		Chrome Lite	Safari	Mobile IE
5	game		Chrome Lite		
6	mail		Chrome Lite	Safari	Mobile IE
7	money		Chrome Lite	Safari	Mobile IE
8	photo		Chrome Lite	Safari	

Fig. 11. (a) The mobile version homepage (b) The PC version homepage.



New Item: 13:00 w Add
7:00 go to school
9:00 eat lunch
10:00 eat dinner
11:00 go home
12:00 sleep

New Item Add
7:00 go to school
9:00 eat lunch
10:00 eat dinner
11:00 go home
12:00 sleep
13:00 wake up

Fig. 12. (a) The to-do list application (b) The to-do list application with one new item added.

developed by developers may not be able to be executed on some mobile browsers due to the different levels of support of JavaScript.

Fig. 12 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.

Fig. 13 (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 Fig. 13 (b); obviously, this Web page is not designed to be browsed on mobile device. With our content adaptation mechanism, the result is shown in Fig. 13 (c). Either the mobile device is under online mode or offline mode, adapted Web pages can be provided to mobile users.

6. System evaluation

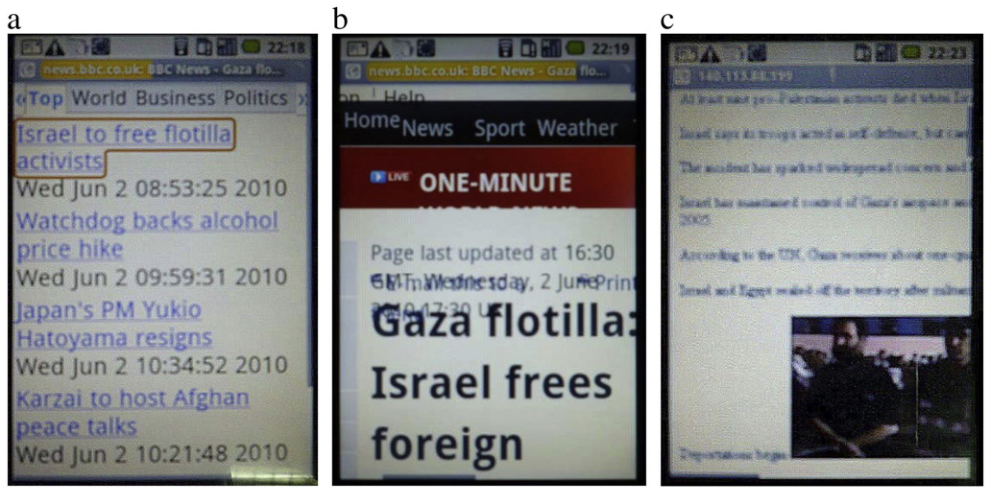
6.1. Performance evaluation

Table 3 lists the performance evaluation result of WOPRE. In order to do so, we measure the consumed time of each application in each operation and calculate the average time after 20 times of executions. The operation which takes the longest time is online application installation for the RSS reader. The reason is that our RSS reader contains five external links in each category, and there are five categories in this application. In other words, all the offline data of these 25 Web pages should be prepared during the installation stage. Although the update process of the RSS reader also downloads external Web content, the pages which already exist in the Offline Data Storage are not downloaded again; therefore, it requires less time for application update than installation. In addition, if real-time external Web pages are downloaded from external Web sites under online mode on mobile device, it takes more time for data transmission than the offline case.

6.2. Usability evaluation

The principal factors proposed by Benbunan-Fich [50] were used for designing a questionnaire to evaluate the usability of WOPRE. The questionnaire contains 10 questions based on factors of appeal, content, ease of use, performance, and support. The questions are listed in Table 4 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 4 and Fig. 14 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



Top World Business Politics
Israel to free flotilla activists
Wed Jun 2 08:53:25 2010
Watchdog backs alcohol price hike
Wed Jun 2 09:59:31 2010
Japan's PM Yukio Hatoyama resigns
Wed Jun 2 10:34:52 2010
Karzai to host Afghan peace talks
Wed Jun 2 10:21:48 2010

Home News Sport Weather
ONE-MINUTE
Page last updated at 16:30
Gaza flotilla: Israel frees foreign

Fig. 13. (a) The RSS reader application (b) The online external Web page without content adaptation (c) The offline external Web page with content adaptation.

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, Fig. 15 shows the Cronbach α of the survey; the support category has the smallest α (0.81), and the overall α is 0.95.

6.3. System comparison

The system comparison between WOPRE and other related technologies is shown in Table 5. 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 WOPRE.

Table 3
Performance evaluation of WOPRE.

Operation	Average consumed time (s)		
	The to-do list App	The RSS reader App	Average of to-do list and RSS reader Apps
Application publication	0.37	0.39	0.38
Application installation	3.27	36.42	19.85
Application update	0.21	27.36	13.79
Application execution on mobile device (online)	0.13 (add a new item)	0.23 (browse an external Web page)	Meaningless
Application execution on mobile device (offline)	0.14 (add a new item)	0.18 (browse an external Web page)	Meaningless
Average of operations	0.82	12.92	

7. Conclusion and future works

7.1. Conclusion

Due to the platform-dependent problem of mobile applications, application developers must spend twice the time writing the same application twice in different languages for different mobile platforms. Even if some technologies such as Java ME are designed as cross-platform languages, they only cross mobile platforms, but not PC platforms. Web applications are cross-platforms because mobile devices usually provide mobile Web browsers. However, Web applications cannot be executed in offline mode. Based on Yahoo! Mobile Widget, Yahoo! Mobile includes both the designs of mobile applications and Web applications, as Web applications are cross-platform, and mobile applications are Offline-able. However, if an application includes external Web pages, which are usually designed for PCs, it is difficult to view them on mobile devices. The Web-based and Offline-able WOPRE system is proposed to solve these problems; it contains an online market for application publication as well as installation and two Subsystems for

Table 4
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

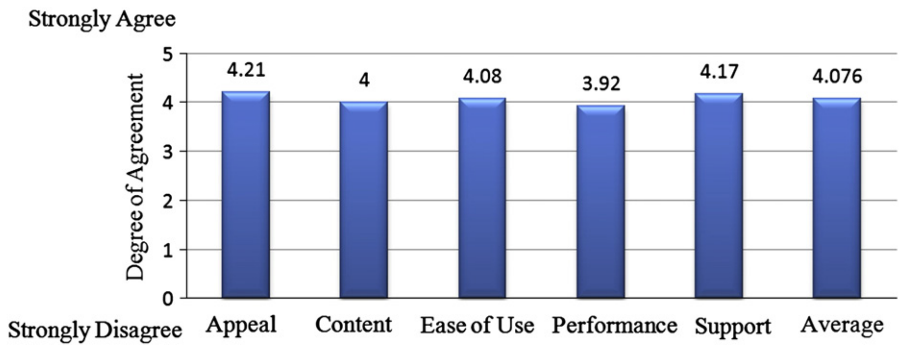


Fig. 14. The analysis result of usability evaluation (average degree of agreement).

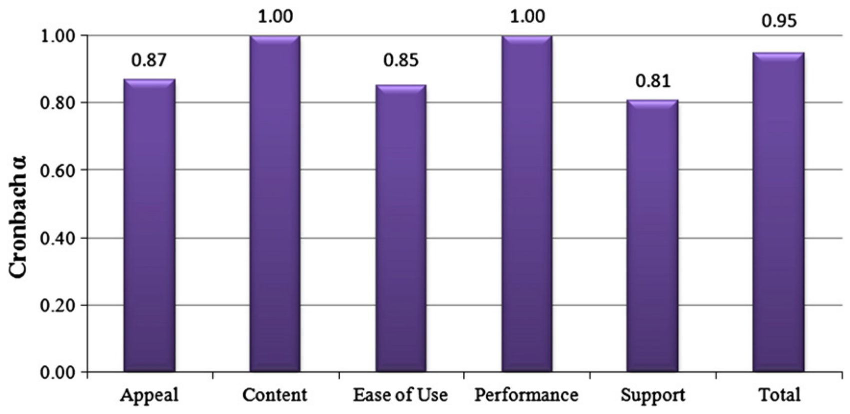


Fig. 15. The analysis result of usability evaluation (Cronbach α).

application developers to design an Offline-able application with adapted external Web pages easily. Based on these two Subsystems, the “to-do list” and “RSS reader” applications are designed to demonstrate their capabilities. Finally, a survey shows that the WOPRE is user-friendly.

7.2. Future works

Currently, the iGoogle Gadget standard is used in WOPRE; application developers are asked to follow this standard for application development. In the future, other widget standards such as Yahoo! Mobile Widget or Windows Mobile Widget will be included, so that more existing widget applications can be executed on WOPRE. In addition, Flash applications will be supported on WOPRE. Since most WOPRE libraries are designed by Web technologies, it is extremely easy to support the functionalities of either Offline or Content Adaptation Subsystems in Flash. The gadget standard must be modified so that developers can include Flash within their gadgets. Moreover, the current work does not address social networks; in the future, the Facebook API [51], will be supported so that existing Facebook applications can be executed on WOPRE, and WOPRE users can interact with each other asynchronously within Web or Flash applications even in offline mode. Furthermore, if HTML5 is supported by most mobile Web browsers in the future, we will replace Google Gears with the HTML5 standard, so that users do not have to install another program. Finally, since the design of WOPRE is purely based on Web technologies, it is highly suitable to be deployed on Web OS platforms such as Chrome OS [52]. In this manner, WOPRE can make the Web OS market more competitive with other markets based on the Apple App Store and Google Android Market. Therefore, we will deploy WOPRE on Web OS systems and devices, and analyze whether Web OS users are satisfied with the proposed system.

Acknowledgments

This work was mainly supported by the National Science Council grant NSC97-2218-E-009-042: Google Android based smart device system and its applications.

Table 5
The comparison between WOPRE and other related technologies.

	Android/iPhone/iPad App.	Java ME	iGoogle	Flash	Windows mobile widget	Opera widget	Yahoo! mobile	WOPRE
Cross-Platform (Mobile device)	No	Yes	Yes	Yes	No	No	Yes	Yes
Cross-Platform (Mobile Device and 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

References

[1] Wang Yi, Lin Jialiu, Annavaram Murali, Quinn A. Jacobson, Hong Jason, Krishnamachari Bhaskar, Sadeh Norman, "A framework of energy efficient mobile sensing for automatic user state recognition", Proc. of the 7th international conference on Mobile systems, applications, and services, June 22–25, 2009, Kraków, Poland.
[2] Ch. Borst, T. Wimb¨ock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. Robuffo Giordano, R. Konietschke, W. Sepp, S. Fuchs, Ch. Rink, "A. Albu-Sch¨affer, and G. Hirzinger. Rollin' Justin — Mobile Platform with Variable Base", Proc. of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009.
[3] Kao Yung-Wei, Peng Pin-Yin, Hsieh Sheau-Ling, Yuan Shyan-Ming, "A Client Framework for Massively Multiplayer Online Games on Mobile Devices", Proc. of International Conference on Convergence Information Technology (ICCIT2007), Nov. 21–23, 2007, pp. 48–53.
[4] Sedat Atmaca, Celal Cekena, Ismail Erturk, A new QoS-aware TDMA/FDD MAC protocol with multi-beam directional antennas, Computer Standards & Interfaces 31 (4) (2009) 816–829.
[5] Edward David Moreno, José Ivo Fernandes de Oliveira, Architectural impact of the SVG-based graphical components in web applications, Computer Standards & Interfaces 31 (6) (2009) 1150–1157.
[6] Sharon P. Hall, Eric Anderson, "Operating Systems For Mobile Computing", Journal of Computing Sciences in Colleges (December, 2009).
[7] R. Godwin-Jones, Emerging technologies mobile-computing trends: lighter, faster, smarter, Language, Learning and Technology 12 (3) (2008) 3–9.
[8] Yung Fu Chang, C.S. Chen, Hao Zhou, Smart phone for mobile commerce, Computer Standards & Interfaces 31 (4) (2009) 740–747.
[9] Tomasz Knyziak, Wieslaw Winiecki, The new prospects of distributed measurement systems using Java 2 Micro Edition mobile phone, Computer Standards and Interfaces 28 (2) (2005) 183–193.
[10] T. Butter, M. Aleksy, P. Bostan, M. Schader, Context-aware user interface framework for mobile applications, Proc. of ICDCS Workshops, 2007.
[11] Lu. Eric Jui-Lin, Yung-Yuan Cheng, Design and implementation of a mobile database for Java phones, Computer Standards & Interfaces 26 (5) (2004) 401–410.
[12] O. Casquero, J. Portillo, O.R. Ramon, J. Romo, M. Benito, iGoogle and Gadgets as a platform for integrating institutional and external services, Workshop on Mash-Up Personal Learning Environments (MUPPLE'08), 2008, pp. 37–41.
[13] I. Hickson (Ed.), HTML 5. Technical report, Web Hypertext Application Technology Working Group HTML 5, 2007http://www.whatwg.org/specs/web-apps/current-work, Working Draft, Available.
[14] T. Melamed, B. Clayton, A comparative evaluation of HTML5 as a pervasive media platform, Proc. of mobile computing, applications, and services: First International ICST Conference (MobiCASE 2009), October 26–29 2009, p. 307, San Diego, CA, USA.
[15] Yung-Wei Kao, Tzu-Han Kao, Chi-Yang Tsai, Shyan-Ming Yuan, A personal webpage tailoring toolkit for mobile devices, Computer Standards & Interfaces 31 (2) (February 2009) 437–453.
[16] Yung-Wei Kao, Tung-Hing Chow, Yuan Shyan-Ming, Offline web browsing for mobile devices, Journal of Web Engineering 10 (1) (2011).
[17] Lingfen Chen, Derek Woods, K. Curran, J. Doherty, Mobile development environments for electronic finance, International Journal of Electronic Finance 4 (2) (2010) 99–119.
[18] Mojax Moblets, [online]Available:, https://code.google.com/p/moblets/.
[19] C. Tong, Analysis of Some Popular Mobile Social Network System, Helsinki University of Technology, 2008.
[20] Yahoo! Go, [online]Available:, http://en.wikipedia.org/wiki/Yahoo!_Go.
[21] Yahoo! Mobile Widgets [online]Available:, http://mobile.yahoo.com/developers.
[22] Yahoo! Mobile, [online]Available:, http://mobile.yahoo.com/.
[23] Windows Mobile Widgets, [online]Available:, http://msdn.microsoft.com/en-us/library/dd721906.aspx.
[24] Opera Widget, [online]Available:, http://widgets.opera.com/.
[25] Google Gadget, [online]Available:, http://www.google.com/webmasters/gadgets/.
[26] Google Maps API, [online]Available:, http://code.google.com/apis/maps/index.html.
[27] Fitzgerald Michael, Google Ajax Search API, O'Reilly, 2007 978-0-596-52953-6.
[28] Google Calendar API, [online]Available:, http://code.google.com/apis/calendar/.
[29] Google Translate API, [online]Available:, http://code.google.com/apis/language/translate/overview.html.
[30] 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 26 (5) (2004) 377–399.
[31] J. Allaire, Macromedia flash MX— A Next-Generation Rich Client, Macromedia White Paper, , 2002.
[32] L. Moroney, Introducing Microsoft Silverlight 2.0, Microsoft Press, 2008.
[33] T. Noda, S. Helwig, Rich Internet Applications, Technical Comparison and Case Studies of AJAX, Flash, and Java based RIA, http://www.uwebc.org/opinionpapers2005.
[34] Google Gears, [online]Available:, http://gears.google.com/.
[35] Gears-Monkey, [online]Available:, http://code.google.com/p/gears-monkey/.
[36] Greasmonkey, [online]Available:, https://addons.mozilla.org/zh-TW/firefox/addon/748.
[37] J. Duhl, Rich Internet Applications, White Paper, IDC, November, 2003.
[38] Daniel Peintner, Harald Kosch, Jörg Heuer, Efficient XML Interchange for rich internet applications, Proc. of 2009 IEEE International Conference on Multimedia & Expo (ICME 2009), 2009.
[39] Mike Chambers, Daniel Dura, Georgita Dragos, Kevin Hoyt, Adobe AIR for JavaScript Developers Pocket Guide, O'Reilly Media, April 2008 ISBN: 978-0-596-51837-0.
[40] J.J. Garrett, Ajax: a New Approach to Web Applications, http://www.pablolfc.com.ar/leer/Ajax.pdfFeb 2005.
[41] W.Y. Lum, F.C.M. Lau, A context-aware decision engine for content adaptation, IEEE Pervasive Computing 1 (3) (2002) 41–49.
[42] A. Carreras, J. Delgado, E. Rodriguez, V. Barbosa, M.T. Andrade, H. Kodikara Arachchi, S. Dogan, A.M. Kondoz, A platform for context-aware and digital rights management-enabled content adaptation, IEEE Multimedia 17 (2) (April-June 2010) 74–89.
[43] Marcos Forte, Wanderley Lopes de Souza, Antonio Francisco do Prado, Using ontologies and Web services for content adaptation in ubiquitous computing, Journal of Systems and Software 81 (3) (March 2008) 368–381.
[44] Jin Jing, Abdelsalam Helal, Ahmed Elmagarmid, Client–server computing in mobile environments, ACM Computing Surveys 31 (2) (1999) 117–157.
[45] Opera Small-Screen Rendering™, [online]Available:, http://dev.opera.com/articles/view/making-small-devices-look-great/.
[46] Yu. Chen, Xing Xie, Wei-Ying Ma, Hong-Jiang Zhang, Adapting web pages for small-screen devices, IEEE Internet Computing 9 (1) (2005) 50–56.
[47] Apache Shindig, [online]Available:, http://shindig.apache.org/.
[48] SiteShoter, [online]Available:, http://www.nirsoft.net/utils/web_site_screenshot.html.
[49] Muffin, [online]Available:, http://muffin.doit.org/.
[50] R. Benbunan-Fich, Using protocol analysis to evaluate the usability of a commercial Web site, Information Management 39 (2001) 151–163.
[51] Facebook API, [online]Available:, http://developers.facebook.com/.
[52] Alex Wright, Ready for a Web OS? Communications of the ACM 52 (12) (2009) 16–17.

Yung-Wei Kao was born on March 12, 1982 in Taipei, Taiwan, Republic of China. He received his MBA degree in Department of Information Management of National Central University in 2006. He interests in Web Technologies, Ubiquitous Computing, Distributed Objects, and Network Security.

Chia-Feng Lin received his MS degree in computer science from National Chiao Tung University, Taiwan ROC in 2006. He is a PhD student in Computer Science from National Chiao Tung University. His current research interests include software architectures for large-scale distributed systems, Cloud computing related issues, SOA and Web2.0.

Kuei-An Yang was born on March 12, 1982 in Taipei, Taiwan, Republic of China. He received his MS degree from Institute of Computer Science & Engineering, in Department of Computer Science of National Chiao Tung University in 2010. He interests in Web Technologies, Nand Flash Memory management, and Network Security.

Shyan-Ming Yuan was born on July 11, 1959 in Mauli, Taiwan, Republic of China. He received his BSEE degree from National Taiwan University in 1981, his MS degree in Computer Science from University of Maryland, Baltimore County in 1985, and his PhD degree in Computer Science from the University of Maryland College Park in 1989. Dr. Yuan joined the Electronics Research and Service Organization, Industrial Technology Research Institute as a Research Member in October 1989. Since September 1990, he has been an Associate Professor at the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. He became a Professor in June 1995. His current research interests include Distributed Objects, Internet Technologies, and Software System Integration. Dr. Yuan is a member of ACM and IEEE.