Transforming Browser-based Rich Clients to Mobile Phone Applications: a Demonstration on Android

Daimeng Wang, Qi Zhao, Gang Huang
Key Laboratory of High Confidence Software Technologies, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China;
Corresponding to hg@pku.edu.cn

ABSTRACT

Rich clients running in web browser become a popular style of web applications. Such browser-based rich clients are usually designed for personal computers and seldom work well on mobile phones due to the phones' incapacity of displaying the relatively large user interface. On the other hand, mobile phones provide many features bringing better user experiences than PC but today's browser-based rich clients are unaware of and unable to use these phone features. In this paper, we propose an automated approach to transforming browser-based rich clients to mobile phone applications and demonstrate the approach on Android by 1) displaying the PC-style user interface on the smaller phone screen and 2) enabling the data embedded in the rich client to be exchangeable with the native Android applications.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *computer-aided software engineering (CASE), user interfaces.*

General Terms

Design.

Keywords

Rich Client, Mobile Phone, Android Apps, User Interface.

1. INTRODUCTION

Web clients, especially browser-based rich clients that have many characteristics of desktop applications, are being adopted widely in the World Wide Web. Running in web browsers with the rich user experience-enabled Web technologies such as HTML 5 and Javascript, they are granted with advanced capacities. Included in these capacities are accessing data and functionalities published by Web Servers through SOAP, RESTful Web services and RSS/Atom feeds, executing business logic based on response data, providing rich user interface (UI), and even supporting service composition [3][4][5].

At the same time, an increasing number of mobile phones are running rich clients in their mobile versions of web browsers. Compared to personal computers (PC), mobile phones have as many weaknesses as strengths. A major weakness of mobile phones is their relatively smaller screen. A laptop PC with a 13 or 15 inch screen has a resolution of 1280*800 or 1440*900, while a mobile phone usually has only a screen of only 2.8 to 4.1 inch with a resolution of 320*240 or 480*320. As a consequence, rich clients originally designed for PC users perform poorly on mobile phones, that mobile phone users are often frustrated with constant zooming and scrolling. The strength of mobile phones is the integration of various features, including basic phone functions and desktop applications. These features make mobile phones capable of much more various tasks and bring better user experiences compared to PC. Due to these differences, it is extraordinary difficult for rich clients to utilize strengths and avoid weaknesses for both platforms at the same time. In addition, despite their characteristics of desktop applications, rich clients are accessed in browsers unlike other mobile phone applications, bringing inconformity to user experience.

In practice, one of the solutions to these problems is to develop a counterpart of the original rich client for mobile phone users, with specially designed browser capable of utilizing some features provided by mobile phones. These versions are often limited to a narrow range of screen sizes. If a new kind of device (with a screen size different from that of PCs and current mobile phones) becomes popular, a new homologous version must be designed and implemented. Besides, they are still accessed via browsers and the inconformity problem is not solved.

A better solution is to develop stand-alone mobile phone applications. This approach provides better user experiences, but is limited to not only the screen size, but also the operating system the applications are running on. Therefore, developing such applications for a variety of operating systems is required, causing enormous amount of coding in more complex programming language such as Java, Object-C, etc. Moreover, updating these applications is a frustrating process which requires downloading and re-installation. In some cases, there exist asynchronous data between stand-alone applications and browser based rich clients. Finally, phone features can be utilized in these two solutions are only the most basic ones. More advanced features provided by various phone applications are still unaware of.

In this paper, we propose an automated approach to transforming browser-based rich clients to phone applications, called rich-client-based applications, with demonstration on Google Android, the most popular smartphone platform. The main contribution of this paper includes:

- UI adaptability: The whole UI of the rich client is split into several small parts before it is displayed on the screen.

- Data exchangeability: A mechanism is established enabling the data embedded in the rich client to be exchangeable with the native functions and applications of mobile phones.

The rest of the paper is organized as follows. Section 2 briefs the architecture of our transformation approach. Section 3 describes our design in detail. Section 4 lists several related work. Section 5 includes conclusion and future work.

2. APPROACH OVERVIEW

Figure 1 illustrates the architecture of a rich-client-based application and its relations with its server, the operating system and other desktop applications. A rich-client-based application contains two major parts: a browser engine and an HTML user interface.

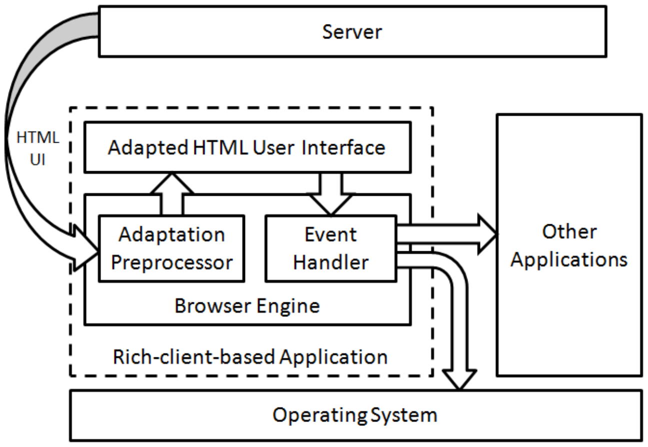


Figure 1. Architecture of a transformed rich-client-based application.

The browser engine is the core of a rich-client-based application. It is similar to those of other common browsers, containing basic browser functions to parse, render and display the rich client's user interface which is written in HTML, CSS and Javascript. Additionally, the browser engine needs to be enhanced to assume two other responsibilities: adaptation preprocessing and event handling. Furthermore, depending on the method of UI adapting and event handling, it may be required that the browser engine contains other special functions discussed in Section 3.

Adaptation preprocessing is the preparation stage of UI adaptation. The UI of rich clients is usually initialized with some basic frameworks and generated dynamically on the fly, which requires a runtime dynamic UI adaptation approach. However, a fully runtime adaptation will give users glances on the process of UI's loading, generating and adapting. To avoid such poor user experience, after the browser engine fetches an HTML document representing the UI from the server, some preprocessing must be accomplished before the UI is displayed to the user.

The event handling function of the browser engine provides an interface for interaction with the operating system (for basic mobile phone functions), desktop applications and other rich-client-based applications. When the user chooses the data to be exchanged with other phone features, a consequent event is published and all mobile phone functions and applications subscribing this event will be notified.

The browser engine is operating-system-sensitive, which means for each kind of operating system a homologous version of the browser engine needs to be developed. On the same operating system, fortunately, the same browser engine can be reused on different rich –client-based applications.

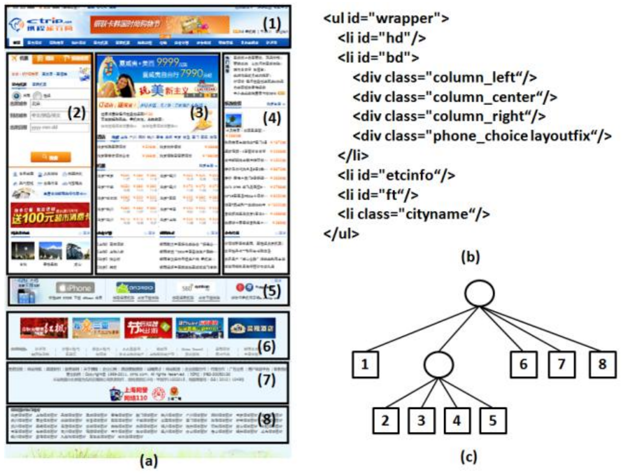
The adapted HTML user interface is generated from the original UI of the browser-based rich client. Its web page structure is analyzed and reconstructed to fit the screen based on the runtime DOM-tree. The whole UI is split into sections reasonably smaller to be displayed on the screen. At the same time, a navigation interface is also generated to allow users to switch between different sections.

3. DESIGN AND IMPLEMENTATION

3.1 User Interface Adaptation

Supported by CSS and Javascript, the user interface of a browser-based rich client can always be viewed as a tree-like structure called Document Object Model (DOM) tree, with each HTML tag as a DOM node.

Figure 2 shows the home page of Ctrip as an example. Figure 2.a demonstrates the page disposition of Ctrip's UI. Figure 2.b and Figure 2.c shows the actual structure of the page in form of HTML text and DOM tree. We will be using this case as the example for UI adaptation section.



```
<ul id="wrapper">
  <li id="hd"/>
  <li id="bd">
    <div class="column_left"/>
    <div class="column_center"/>
    <div class="column_right"/>
    <div class="phone_choice layoutfix"/>
  </li>
  <li id="etcinfo"/>
  <li id="ft"/>
  <li class="cityname"/>
</ul>
```

Figure 2. Example: www.ctrip.com.

3.1.1 Automated Adaptation

The automated approach to adapting user interface of rich clients is discussed thoroughly in work [1]. It provides a mechanism for dynamic UI partition and automatic navigation. It can run on any rich client and on any browser, and provides a setting interface to determine how a page is to be partitioned. In our example, with the correct settings, it can automatically divide the UI of ctrip.com exactly in accordance with its disposition shown in Figure 2.

However, the effectiveness of the settings differs from case to case. It is hardly possible to find one single setting for every rich client we are trying to adapt. Another drawback of automated approach is that the differences between sections of the UI are completely ignored. Taking Ctrip for example, in Figure 2, sections (5), (6) and (7) in are mostly sponsors and ads, while section (8) contains only a long list of hot links. These sections are less important and could be removed in order to simplify the UI on mobile phones and improve user experience. On the contrary, section (1) is the header and contains navigation links throughout the site. It is likely that users want it shown at the top of the screen all the time.

The UI of a rich client is highly dynamic, involving both frequent changes to the inner HTML text of DOM nodes and insertion or deletion of entire DOM nodes and their descendants, causing the partition pattern needing runtime modification. Therefore a mechanism is needed for monitoring such DOM tree changes. The approach introduced in work [1] uses an extraordinary costly method of monitoring the inner HTML code of each section, due to the ordinary browser's lack of supporting of such monitoring mechanism. However, our approach uses a purposefully enhanced browser engine, which makes it possible that we monitor DOM node insertion, deletion and modifications through events, thus significantly reducing the cost of dynamic partition.

3.1.2 Manual Configuration

The automated approach introduced in work [1] has some flaws, which are difficult to fix due to the approach's intention for universality and compatibility. However, these intentions are not included in our study of transforming rich clients to phone applications, where some case-by-case configuration is entirely acceptable. There could be a unique set of configuration data for each rich-client-based application to ensure its best performance and user experience.

There are two levels of manual configuration. The first one is the above mentioned settings in the automated UI adaptation approach. These settings could define how large a partitioned section can be, whether the CSS style of each section should be changed, etc. This level of configuration intends to improve the effectiveness of the automated UI adaptation to its best. It is difficult, though, for this level of configuration to help determine whether a section is important or not. When the UI updates, this level of configuration could remain unchanged, and may need modification only when the disposition of the UI is totally reconstructed.

The second level of configuration directly designates the root nodes of the DOM trees included in each section. Developers need to analyze the HTML and CSS code of the original UI and determine how the UI should be split and how important each split section is. The automated approach is helpful to generating a default pattern. This level of configuration aims to provide best user experience through the analyzing and designing of developers themselves. There are two drawbacks of this level of configuration. The first one is that it requires the chosen designated DOM nodes to have unique identification properties. In our example of Ctrip in Figure 2.b, the "id" property of the "<li>" tags and the "class" property of the "<div>" tags would be suffice for developers to mark each sections. Another drawback is that a minor updating consisting of changes on these identification properties will require an update of the configuration data as well, thus increasing the maintenance cost.

In practice, the two level of configuration can be used simultaneously. The second level of configuration is of high priority to ensure the best user experience. And when it becomes unavailable, the automated partition method is executed with the first level of manual configuration as its settings. Figure 3 shows the UI adaptation for Ctrip on Android emulator, using both automated approach and manual configuration.

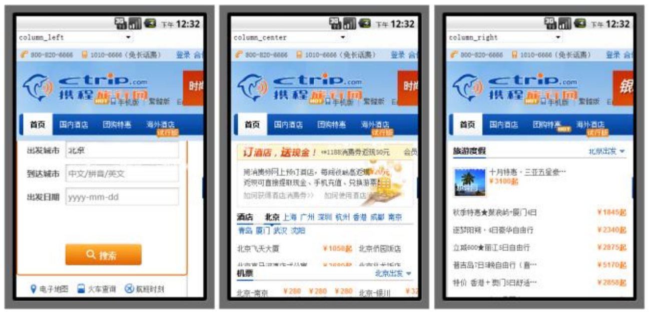


Figure 3. UI adaptation for Ctrip on Android emulator.

3.2 Data Exchange

A browser-based rich client contains a large amount of data, many of which can be inputs for other applications. Enabling the data embedded in the rich client to be exchangeable with the native functions and applications is considerably beneficial to user experience.

To properly establish the connection between different data types and different mobile phone functions and applications, events are introduced to serve as communication agents. When a user selects some text, all events will be listed, sorted by their relations with automatically detected data type. The user may need to choose which event he or she wished to publish. Then the mobile features subscribing this event will take the previously selected text as input.

Figure 4 shows an example of relationships between data types, events and applications. Data types are automatically detected through either regular expression (e.g. phone number) or searching (e.g. contact name). The detectable data types and the detection method are embedded in browser engine and cannot be changed unless the browser engine updates. Each type of data is related to one or more events and each event can have one or more subscribers. Events and its subscribers are stored in a file and can be manually added by developers and users. The browser engine fetches the event and subscriber list from this file and listens to the events published though user's interaction with the UI. In other words, other phone features do not need to modify themselves to subscribe these events. Instead, developers or users need to designate such relationships themselves.

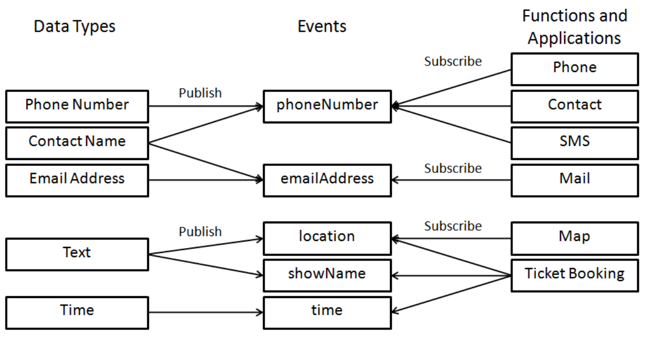


Figure 4. An example of relationships between data types, events and applications.

There are three kinds of event subscribers: basic mobile phone functions, rich-client-based applications and stand-alone desktop applications. For most mobile operating systems, basic mobile phone functions are permanent and APIs of accessing these functions are already provided. Thus, event subscribing of these features can be directly integrated in the browser engine. As for rich-client-based applications, it is easy for their developers to write some configure data designating these applications' subscribing events during their developments. For other desktop applications, however, it is not practical, although possible, for developers to bind each of them to one or more events. If the need arise, the users could create such bindings themselves. Additionally, it is hardly possible that these applications directly accept selected data. Instead, the data should be copied to clipboard for the likely subsequent pasting action.

Luckily, many mobile phones' operating systems provide their own mechanisms for local applications to communicate with each other through run-time messages. Android, for example, uses intent messaging as a facility for binding between its components. The Android system is capable of finding appropriate applications to respond to an intent message. Such mechanisms are very useful and could be the foundation of our data exchange between rich-client-based applications and other mobile phone features, without needing for developers' or users' further settings. Our approach mentioned above, on the other hand, could still be a valuable supplement for further customization.

4. RELATED WORK

For user interface adaptation, Work [6] and [7] introduced an approach for traditional dynamic web pages. Methods introduced in these works are single-shot server-side adaptation based on HTML text analysis. In practice, UC browser and Baidu Transcoding both provide web page adaptability. UC browser could adapt fetched web pages before displaying them, disabling Javascript and some CSS style. Baidu Transcoding adapts web pages and filters Javascript and CSS codes at server side, thus significantly reduces phone traffic. Since Javascript and some CSS are disabled in both practices, it is impossible for them to preserve all features provided by rich clients.

The data exchange part is similar to Mashups, where data and function from different services are combined to create new services. Currently most Mashups are service driven, which means that the combination of services are already determined and data flows through the designated path. Our approach, however, is data-driven. The combination of applications can be determined by the users themselves according to the data they are interested in.

The idea of developing desktop applications for mobile phone using HTML, CSS and Javascript is presented in development tools and middleware such as Phonegap and Rexsee. The applications developed with these tools are hybrid, meaning that they are neither truly native nor purely web based. The difference between the two is that Phonegap stores its user interface and logic code locally, aiming for client-side applications, while Rexsee is an enterprise mobility suite designed for server-side application development.

5. CONCLUSION AND FUTURE WORK

Our paper presents an automated approach to transforming browser-based rich clients to phone applications, using Android platform for demonstration. We combine the automated approach to user interface adaptation with manual configuration to reconstruct the UI of the original rich client to fit the screen of mobile phone. Additionally, we add event handling mechanism to enable the data embedded in the rich client to be exchangeable with the native functions and applications phones. Out method provides a way of developing mobile phone desktop applications based on already existed rich clients with a minimum quantity of coding and maintenance cost.

Future work includes testing and improving performance for both UI adaptation part and data exchange. One possible improvement in future work is adopting better UI adaptation method to deal with rich clients with more complex and less clearly structured UI.

6. REFERENCES

[1] Gang Huang, Daimeng Wang. Adapting User Interface of Service-Oriented Rich Client to Mobile Phones. The 6th IEEE International Symposium on Service-Oriented System Engineering (SOSE), 2011.

[2] Gang Huang, Qi Zhao, Jiyu Huang, Xuanzhe Liu, Teng Teng, Yong Zhang, Honggang Yuan. Towards Service Composition Middleware Embedded in Web Browser. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009, 93-100.

[3] J. Duhl, White paper: Rich Internet Applications. Technical report, IDC, November 2003.

[4] M. Driver, R. Valdes, and G. Phifer, Rich Internet Applications Are the Next Evolution of the Web. Technical report, Gartner, May 2005.

[5] Qi Zhao, Gang Huang, Jiyu Huang, Xuanzhe Liu, Hong Mei, Ying Li, Ying Chen. A Web-Based Mashup Environment for On-the-Fly Service Composition. SOSE 2008, pp. 32-37.

[6] Y. Chen, W.Y. Ma, and H.J. Zhang, Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices, Proc. 12th Int'l World Wide Web Conf., May 2003.

[7] Z. Hua, X. Xie, H. Liu, H. Lu, W. Ma. Design and Performance Studies of an Adaptive Scheme for Serving Dynamic Web Content in a Mobile Computing Environment. IEEE Transantions on mobile computing, vol. 5, no. 12, December 2006.