An Internetware Based Approach to Building Web Page Integration Applications for Mobile Devices

Tianwei Sun, Feng Xu, and Jian Lü

1 (Key Laboratory for Novel Software Technology, Institute of Computer Software, Nanjing University, Nanjing 210093, China)
2 (Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract Mobile devices are more and more popular in recent years. As a result, there're huge requests of mobile applications, especially those integrated with multiple information. However, on one hand, most of the mobile applications at present just contain some certain kinds of information and they cannot adapt to the rapid change of users' requirements, either. On the other hand, to build these applications, it's usually time consuming and there are not enough resource components with programmable interfaces. In this paper, we propose an approach based on Internerware to building web page integration applications for mobile device. We introduce a framework that provides abundant internet-programmable interfaces, a flexible integration mechanism to meet the users' rapid changing requirements and a reliable mechanism that guarantees the quality of the referred resources effectively. With this framework, we can rapidly build an application that integrates all the information according to users' requirement.

Key words: Internetware; mobile devices; web page integration; programmable blocks


1 Introduction

With the burgeoning of mobile technologies, mobile devices are more and more popular these years. More and more users enjoy the convenience brought by web applications on mobile devices, such as stock information, weather forecast, the live of sports report, etc. Kaikkonen[6] already predicted the prosperity of mobile systems and applications several years ago. There is a huge number of web resources distributed in the Internet. Among them, websites are important and relatively stable sources that could provide a variety of interesting information. But how to exploit these resources efficiently and effectively is still a challenging problem. On one hand, the fact that users are deluged by a huge amount of information means that we require an effective method to discern the information that users are interest in. Different people might concern about different sorts of information. Redundant data transfer...
can also be a considerable power burden for mobile devices. On the other hand, mobile devices are different from computers. Compared with computers, they have much smaller screens and weaker computing capability. So information displayed on mobile devices should comply with their features.

To current users, a large number of them have multiple independent requirements, e.g. a user might be interested in local weather forecast, stock information of Google and discount information of flight tickets. In most cases, users have to search the applications by themselves and visit these resources separately; however, if we have a tool to integrate these resources and display the information that users need in an optimized way, mobile devices would save much power from transferring unnecessary information and provide a cleaner, simpler and more convenient view to users. To develop such a tool, we have already had a huge number of independent software components which are also autonomous, like web services, web sites, etc. In our work we naturally consider the challenges in front of us from a view of Internetware. Internetware, as an advanced software paradigm, was proposed in academia several years ago\(^8\). Here We refer the definition of Internetware contributed by Jian Lü et al.

Internetware is a software collaboration consisted by a group of autonomous software components which are distributed in an open environment, provided by third-parties and connected by a variety of coordinators with different features\(^9\). The core idea of internetware is resource integration and sharing in the open, dynamic and autonomous network environments. And this is also the main reason why we select the view of Internetware to observe and resolve the problems in mobile web-applications.

A traditional Internetware application usually integrates components with programmable interfaces. These components usually use some typical approaches and methods such as OMG’s CORBA/CCM series, Microsoft’s COM/DCOM, Oracle’s J2EE/EJB, etc. In recent years, despite of the rapid development of these technologies, the number of these kinds of components is still far from users’ demands. To mobile web-applications, websites are still the most important resource providers. The major challenges that we are facing to include: first, there are a huge number of resources but we do not have many programmable interfaces; second, we need a flexible resource-integration mechanism that can adapt to the rapid changes of users’ requirements and the changes from websites; third, we need a reliable mechanism that can identify malicious websites or those websites that cannot work normally; fourth, we need a mechanism to optimize the efficiency for the system. To cope with these challenges, we propose an Internetware based framework that provides abundant internet-programmable interfaces, a flexible integration mechanism to meet the requirements and a security mechanism that guarantees the quality of the referred resources effectively. For the first challenge, we have a block framework to wrap all the resources in programmable units; for the second challenge, the block framework provides a process-oriented design approach for agile development; for the third challenge, we have a selection module based on trust to filter the websites and we also have a web crawler module to collect the certification and monitoring records; for the fourth challenge, we have the web crawler to determine mobile websites to reduce the transcoding workload. In this paper we mainly focus on these challenges.

The rest of this paper is organized as follows. Section 2 covers the related work.
We demonstrate the architecture of the internetware system in Section 3. In Section 4 we give the introduction of our proto-system and an example. Finally, we conclude the paper in Section 5.

2 Related Work

There’s a lot work about integrating web pages. Han et al.\cite{4} proposed a system named “Homepage Live”, which allows users to collect their favorite Web content from existing Web pages and organize them in a single page. Homepage Live wraps these content in multiple semantically coherent blocks and automatically recognizes these blocks and allows users to use drag-and-drop actions to select the ones of interest. The main challenging problem for Homepage Live is how to trace a content block because many web pages are updating everyday. They adopt a novel method where tree edit distance is utilized to trace the block. Internet scrapbook\cite{12} is a system which enables users to collect content blocks on Web pages and trace it in updated versions. It exploits the plan html tags and contents over the Web pages to trace the target block. Knoblock\cite{7} proposed the Ariadne approach to web-based information integration. The approach interprets a request to web sources and returns a structured reply.

All these work focus on the algorithms during the integration procedures. Different from them, our main work is to design a framework which can flexibly integrate web pages and we also introduce some features these work don’t have like trust management.

As our intention is to offer mobile users web-based applications, transcoding is another issue we should consider. Related work about transcoding web page for mobile devices can be classified into two classes in general: manual and automatic reauthoring.

In manual reauthoring, according to original web pages, a serial of mobile optimized web pages are designed for a better look on mobile devices. In automatic reauthoring, web pages are transcoded to mobile friendly pages automatically by some algorithm. Hwang designed a transcoding framework in allusion to automatic reauthoring in Ref.\cite{5}. Some structure-aware transcoding heuristics are applied to preserve the layout of the original Web pages as much as possible. Anderson\cite{1} lays the theoretical foundations for web site personalization, and proposes the implementation for web site personalizer Proteus.

We believe that because of the heterogeneity of the current web pages, automatic reauthoring is not a practical approach to transcode web sites. In our system, we still use manual reauthoring. But we don’t produce a new web page for the original page. We get the webpage from remote server and transcode it with some rules and output the optimized page to users. In this way, we can guarantee that the content is always consistent with the original site. In addition, if the original site’s content has changed, we only need to change the rules without the need of revising website, so that there will be less workload.

3 System Architecture

3.1 Framework
In this section we present the framework of our approach. As Internetware has several important features including software components integration, adaptation to the environment, security management, we designed several modules to realize them.

1. **Resource selection module**: This module is primarily responsible to analysis the user’s request. It converts the structured request into web page URLs. There’s two kinds of requests. One contains web page information, it specifies what web pages the system will return. The other includes service types and user’s customized requirements which we normally use the term *quality of service* (QoS). As this kind of request does not specify the service providers, the module needs to synthesize many factors like QoS and trust measurement to dynamically choose service providers and their services. In our framework, we take web sites as service provider, and web pages as services. Usually a user request is a mixture of both.

2. **Integration and transcoding module**: This module takes web page URLs and integrate the web pages. It also transcodes web pages to be mobile optimized so that they can meet the needs of mobile users. For every web page this module handles, we will define a sort of rules to transcode the page and wrap the content to make it programmable. This module is the most important part of the framework.

3. **Web crawler module**: We use some auxiliary module to improve the efficiency of the framework. The Web Crawler module is one this kind of modules. It will record all related information of a web site including categories, response time, web page size, etc. In particular, we use an algorithm to determine whether a site is a mobile website or has a mobile mirror. We do this work because if a web page is mobile optimized, we can integrate the page directly instead of transcoding it to improve efficiency. Web Crawler Module continuously analyzes and evaluates thousands of web sites from Internet, so that our framework can get the updated information of these sites.

4. **Security management module**: In our system, security management focuses on the reliability and security of the web pages which will be integrated. This module will work during the procedure of Website selection module and Mobile Crawler Module.
We briefly describe our approach. In essence, our system is a middleware that connects web resources and mobile devices. When the system receives a Client’s http request, it sends this request to Resource Selection Module and converts the request into web page URLs and service requirements. Resource Selection Module selects the web page URLs that meet these requirements from database. During the procedure, Security Management Module always checks the reliability and security of the websites. Then the URLs will be sent to Integration and Transcoding Module, the module downloads all the web pages, integrates them to one web page and makes the page mobile optimalized, and then return the http response to the client. The process is described in Fig.2.

![Diagram of system framework](image)

**Figure 2. Framework of the system**

In the following sections, we will give the detailed description of design of Resource Selection Module, Web Crawler Module and Web Crawler Module. We will discuss Security Management Module in our future work.

### 3.2 Resource selection module

There’s two kinds of users. Some people know what kind of web sites to visit. The others don’t care about service providers and just want to get information. So we designed this module to cope with this issue. For the latter users, the request will contain users’ needs of service and constraints of Qos. According to the request, this module will select a collection of web pages that meet the requirements. As there will be more than one web page that provide the same kind of service, the module should evaluate these web pages (service providers) and choose the best one.
We take web sites’ features and trust into consideration. At present, there are few third parties provide certification and monitoring records for web sites. We use Web Crawler Module to gather Qos information of the web sites which we will introduce later. We also find there’s some third-party ranking websites like Alexa.com that provide relatively impartial ranking information for almost all web sites on the Internet. They also provide some other information of the web sites like traffic status, search analytics, reviews, etc. Usually a web site’s ranking represents people’s favorite and trust on it, so we consider ranking as an important parameter in the evaluation approach.

At present, our system uses the weighted average method to calculate the scores of each of the web sites according to their Qos attributes. As there’re several ranking values from different ranking site, first we need to calculate an average value of ranking. As the range and direction of the Qos attributes’ values are different, we think it’s unreasonable to calculate the evaluated value by simple weighted average of the Qos attributes. The attribute values will be normalized before the weighted calculation. When the calculation is finished, the web site with the highest score is our choice.

Suppose the number of the web pages needed to be evaluated is \( m \); the number of the Qos attributes is \( n \); the number of the third-party ranking web sites is \( t \). Let \( c\text{Value}[j] \) be the user’s constraint of the \( j^{th} \) Qos attribute, it has a default value if user doesn’t give one; \( w[j] \) be the weight of the \( j^{th} \) Qos attribute; \( w\text{Rank} \) be the weight of the rank; \( \max(q\text{Value}[j]) \) and \( \min(q\text{Value}[j]) \) be the maximum and minimum constraint value of the \( j^{th} \) Qos attribute; \( g\text{Value}[j] \) be the average value of the \( j^{th} \) Qos attribute; \( r[i][k] \) be the rank of the \( i^{th} \) web site that the \( k^{th} \) third-party ranking web sites gives; \( w[t][k] \) be the trust weight of the \( k^{th} \) third-party ranking web site.

We have constraints,

\[
\sum_{j=1}^{n} w[j] + w\text{Rank} = 1
\]

\[
\sum_{k=1}^{t} w[t][k] = 1
\]

First we normalize the qos attribute value.

\[
\left| \frac{c\text{Value}[j] - g\text{Value}[j]}{\max(q\text{Value}[j]) - \min(q\text{Value}[j])} \right|
\]

Then we give the normalize the rank value.

\[
\left| \frac{\sum_{k=1}^{t} w[t][k] \times r[k] - \max(r[k])}{\max(r[k]) - \min(r[k])} \right|
\]

Now we have the score of the \( j^{th} \) web site. If the value direction is negative which
means the value is smaller the better, \( w[j] \) should be negative.

\[
\sum_{j=1}^{n} w[j] \cdot \frac{cValue[j] - gValue[j]}{\text{max}(gValue[j]) - \text{min}(gValue[j])} + wRank \cdot \frac{\sum_{k=1}^{t} w(t) \cdot r(k) - \text{max}(r(k))}{\text{max}(r(k)) - \text{min}(r(k))}
\]

With the score of each web site, we can use fast sorting algorithm to select the best service.

There have been a lot of research in the area of services selecting algorithm. Wang\cite{13} has introduced an Internetware-software-architecture-oriented trust-driven mechanism for selecting service (ISAOT). ISAOT supports dynamic services binding and selecting in coordination aspect and makes decisions on the basis of trust in services selecting. It addresses the relevant principles, models, approaches which are required when the Internetware systems select services according to the change of application requirement. It also demonstrates a clear trusted Internetware development model. To cope with the high dynamics demand of open system, Pan\cite{11} proposes a reputation-based recommender discovery approach. They divide the Web of trust into personalized trust networks, and locate the recommenders through trust opinion iteration among users. This approach can reduce the cost in information collection and also gain high efficiency. Zong\cite{14} presents an efficient, scalable and effective trust computation model based on artificial neural networks. They propose a broker-assisting information collection strategy based on clustering method. We can apply these approach to our system in the future.

3.3 Integration and transcoding module

Integration and Transcoding Module is the core module of the system. It takes in the web page URLs pretreated by Resource Selection Module and sends the URLs to the remote servers. When all the response contents from the remote servers return to the module, it will integrate all these contents and make the contents mobile optimized if necessary. After this, this module will send the integrated page to the users’ devices.

This process can be seen as a collection of steps. It can be divided into four stages: preparing request; gathering web page sources; integrating web pages; sending response. Each stage can also be divided into more detailed steps. For instance, in the stage of integrating web pages, we might need to change the style of one page and remove irrelevant contents of another one, which means this stage takes two steps and these steps could be synchronous or asynchronous.

So we aim to find an approach to combine these steps in an efficient way. Mannion\cite{10} has introduced an approach for synthesizing behavioral models of eBlock\cite{3,2} networks into code and connectivity for physical eBlocks. Mannion has shown eBlocks can be a fast way to build applications and reduce a heavy workload of code programming. We are inspired by eBlocks and we also designed our framework based on the idea of Programmable Blocks.
We define a block to be a logical programmable block having a pre-defined combinational or sequential behavior. It has an input or output interface or both. In most cases, input data can be a web page URL, or web page contents, and output data is usually a web page content that is processed by the block. Blocks can connect with each other in sequence to complete a certain operation, they can also connect in concurrency if these blocks are independent.

As a block is a basic element and more than one block is needed in the process of integration and transcoding, we define a blockset to be a set of blocks which collaborate to accomplish a logic function. A blockset can be contained in another blockset to provide more complicated function. To web page URLs, we make a mapping between a specific blockset that contains all the integration and transcoding work and the URLs which conform to a specific regular expression.

In our framework, one process will pass through four stages as we mentioned above. So we defined four kind of root blocks correspond to the four stages: RequestBlock, SourceBlock, ContentBlock, ResponseBlock. These blocks can affect the request website separately.

Here we give the detailed description of the root blocks.

**RequestBlock**: It prepares the request that will be sent to the remote server. A request contains two parts: request header and request content. We can use the blocks to change header information like User-Agent, Accept-Encoding, Cookie, etc. We can also pre-treat the content. For example, we designed a block named *FilterUploadBlock* to filter the harmful content uploaded by the user. As RequestBlocks are executed on one request, one request block may affect the other blocks. RequestBlocks within a BlockSet are executed in sequential order.

**SourceBlock**: It obtains web page data according to the request passed by RequestBlocks and returns the data to the system. Usually this kind of blocks obtain data from Internet. As our intention is to integrate web pages, there will be several
SorceBlocks in a blockset. SourceBlocks within a BlockSet are executed simultaneously as they will not affect each other.

**ContentBlock**: It operates on the data that the SourceBlocks have retrieved, whether it be HTML, CSS, Javascript etc. This kind of blocks are for integration, transcoding and also optimizing web pages. With these blocks we can extract some part of a web page or affect the formatting of another. ContentBlock within a blockset are executed in sequential order.

Content block is the core of the blocks, so we’ll introduce it in more detail. The block can be divided into several categories.

1. File block: Operates on css, javascript and other files linked in the web page. For example, we can delete stylesheet link to reduce data transfer.
2. Device block: Many mobile devices’ browsers have their own style tags or they don’t support some features of the traditional browsers. So these blocks optimize these features.
3. Formatting block: This kind of blocks are composed of a serial of formatting operations. It formats the body of the html content such as adding/deleting/moving tags, modifying the attribute of a tag etc.
4. Integration block: This kind of blocks are used to extract the content we need from a web page, wrap the content to make it more flexible to be integrated, and insert it into the integration page.
5. Passthrough block: It rewrites the link attribute of tags so that they pass through the system and are fetched directly from the remote host.
6. Optimization block: This block optimizes the site. For instance, one block handles the image from the remote server and zooms out the image, another block 'LateloadHTMLBlock' delays to load the page.
7. Features block: it adds some features that’s useful to clients like bookmarks, history, etc.

**ResponseBlock**: it collects all the data from SourceBlocks and ContentBlocks and puts it together into a response. There can never be more than one ResponseBlock in a BlockSet because we just return exactly one response to the user.

With our integration and transcoding framework, we can follow the four stage process to design and plan and we have blocks to accomplish every step operation. It’s a fast approach to integrate all the web pages that the user requests.

### 3.4 Web crawler module and algorithm

We use some auxiliary module to supplement our framework and improve the efficiency of the system during the transformation process. The most important is the Web Crawler Module.

As we mentioned in Section 3.1, there are few third parties provide certification and monitoring records for web sites now. So we need to gather these Qos information of the web sites ourselves. Web Crawler Module is responsible for this work.

First We get a list of web sites from a web information website like Alexa.com; then we start our crawl engine to evaluate each one on the list and record whatever we want like response time, page size, etc. This engine will continuously work until finish crawling all the web sites. As the list of web sites will be updated regularly, we also need to run the crawl engine respectively.
Another important feature of Web Crawler Module is that it will determine whether a site is a mobile site or whether it has a mobile optimized mirror. As we know, there's considerable web sites that have their own mobile optimized mirror like (Google.com), (Yahoo.com), etc. As these mobile sites can be shown well on mobile devices, we don’t need to transcode them. We can just download them and integrate the page content we need immediately. So it can save the workload of customizing blocksets. In addition, it can reduce the waiting time that the system transcodes a webpage. This is important because our intention is building the application in an efficient and effective way.

At present, there's no good algorithm to determine mobile sites. W3C has defined a couple of disciplines that one mobile site should implements and offers an algorithm to determine mobile sites. But we have found most mobile friendly sites are not in full accordance with these disciplines. A lot of mobile friendly sites get a low score with the algorithm which means the algorithm judges these sites as not mobile friendly by mistake.

We have studied the disciplines that W3C defines, and analyses some mobile friendly sites. Here we introduce our algorithm to improve the W3C algorithm.

Algorithm of determine mobile websites.

1: Download site.com with an mobile user agent (we call it ‘mua’ site), goto 2
2: Check if there's `<META HTTP-EQUIV="REFRESH">` tag in ‘mua’ site.
   If true, goto 3, else goto 4
3: Extract the URL in the tag, goto 1,
4: If site.com gets redirected, goto 5, else goto 6
5: If the redirected site is like ‘m.site.com’, ‘site.com/m’, etc, goto 13, else goto 6
6: If mobile site has a mobile doctype, goto 13, else goto 7
7: Download site.com with a desktop user agent (we call it ‘dua’ site), goto 8
8: Is ‘mua’ site different from ‘dua’ site by more than 10%? If true, goto 13, else goto 9
9: Is ‘mua’ site different from ‘dua’ site by less than 1% or they are the same?
   If true, goto 15, else goto 10
10: Search Anchor tags, goto 11
11: If ‘m.site.com’, ‘site.com/m’ on the ‘mua’ site but not on ‘dua’ site, goto 13,
    else goto 12
12: Check all website like ‘m.site.com’ etc. If there’s at least one URL gets a ‘200’
    response, goto 13, else goto 15.
13: This may be a mobile friendly site
14: This is a mobile friendly site
15: This is not a mobile friendly site

We got top 300 websites from Alexa and applied our algorithm on them. The result shows that this algorithm can recognize 79% of websites as mobile friendly sites or ordinary sites. And we have to check manually on the left 21% uncertain websites.
4 Case Study

We designed an proto-system using program language ruby and open source framework rack. The system is used to create personalized applications according to different users’ requirement. Its main part is an online visual editor. With click and drag actions, we can add blocks to blocksets, implement them to the specific URL and make customized configuration by setting parameters of blocks.

We have a primary implementation of our framework in Fig.5.

![Figure 5. Panel of the online visual editor](image)

Every block needs a detailed configuration. We use a json object to store it. Here’s an example of a content block’s configuration.

```
[
  "Content::Passthrough::Form",[],
]
Here we give an example of booking airline ticket. To make travel more convenient, we integrate weather information with booking service.

Ctrip(www.ctrip.com) is a Chinese travel website that offers booking service. Users can search for detail airline information by departure city, arrival city and departure time and book tickets.

As there are many weather services on Internet, we do not specify a particular one to integrate. Service selection algorithm will automatically select an appropriate service.

We have Web Crawler to collect weather forecast website. There are three weather websites in the database: www.weather.com.cn, www.nmc.gov.cn, and www.cma.gov.cn. According to the service selection algorithm, We choose these two Qos properties: response time, availability. In addition, we choose authoritative third-party ranking websites www.chinarank.com and alexa.com to get website ranking information.

Table 1 shows the Qos value of the weather services.

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Service Domain</th>
<th>Available</th>
<th>Response Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather China</td>
<td><a href="http://www.weather.com.cn">www.weather.com.cn</a></td>
<td>20/20</td>
<td>1821</td>
</tr>
<tr>
<td>National Meteorological Center of CMA</td>
<td><a href="http://www.nmc.gov.cn">www.nmc.gov.cn</a></td>
<td>19/20</td>
<td>1737</td>
</tr>
<tr>
<td>China Meteorological Administration</td>
<td><a href="http://www.cma.gov.cn">www.cma.gov.cn</a></td>
<td>20/20</td>
<td>1952</td>
</tr>
</tbody>
</table>

Table 2 shows the rank of the services. We can find that www.weather.com.cn has a much lower rank than the others.

<table>
<thead>
<tr>
<th>Service Domain</th>
<th>Chinarank Rank</th>
<th>Alexa Rank(Traffic Rank in CN)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.weather.com.cn">www.weather.com.cn</a></td>
<td>111</td>
<td>115</td>
</tr>
<tr>
<td><a href="http://www.nmc.gov.cn">www.nmc.gov.cn</a></td>
<td>2203</td>
<td>5054</td>
</tr>
<tr>
<td><a href="http://www.cma.gov.cn">www.cma.gov.cn</a></td>
<td>1001</td>
<td>4780</td>
</tr>
</tbody>
</table>
In Section 3.2 we introduced the algorithm. We set the weight of availability to 0.3, the weight of response time to 0.2 and the rank to 0.5. In this case, the value range of response time is [0...2000], the value of user constraint is 2000; the value of availability is 0.9. So each service has a value. As Weather China has the lowest value, the system will integrate www.weather.com.cn with Ctrip.

Figure 6 gives the blockset design for the application. We use several content blocks to take out the weather information content from the page and make it optimized. Then we use integration block to wrap the content in an iFrame to make it programmable. We remove the irrelevant content (stroke content in Fig.7) from Ctrip search result and optimize the flight content. At last we integrate two iFrames in a web page and return it to the client.

![Blockset for ctrip](image1)

![Remove the irrelevant content of ctrip](image2)
It just takes several agile steps to build a customized application with our system. In addition, we make it possible to make programmable interfaces. Our system can be also flexible to adapt to the environment. For example, if the 'div' tag's id changes, what we should do is just modify the xpath parameter of the corresponding formatting block. In this case, we also improve efficiency by identifying the mobile site and
integrating it’s page.

Figure 8 shows a screenshot from the iPhone simulator. The upper part is a 3-day weather forecast and the lower is the flight information.

5 Remaining Issues

As there’s an explosive growth of mobile devices in recent years, it is critical to develop techniques to offer mobile optimized resources to mobile users. In this paper, we designed an Internetware-based system which aims to integrate webpage resources to fit for mobile devices. Our main work is concentrated on the framework that introduces programmable blocks to reduce the workload. We regard this system as an preliminary implementation of Internetware. In our future work, we’ll improve the programmable blocks framework and we will also replenish trust management to make our system more reliable, robust, and dynamic. And from the practical perspective, we hope to make the system easier to use so that everyone can use it to create their own mobile applications and integrate whatever they are interested in.

Acknowledgements

Supported by 973 of China (2009CB320702), NSFC (61073030, 60736015) and JSNSF (BK2008017).

References

