Retrieval, Description and Security: Towards the Large-Scale UI Component-Based Reuse and Integration

Hao Han*, Peng Gao† and Keizo Oyama*‡
*National Institute of Informatics, Japan
{han.oyama}@nii.ac.jp
†Tokyo Institute of Technology, Japan
gao@tt.cs.titech.ac.jp
‡The Graduate University for Advanced Studies (SOKENDAI), Japan

Abstract—Under the trend of the information/functionality integration, the application integration at the presentation and logic layers becomes a popular issue. Without the open Web service APIs, the integration of traditional Web applications is based on the reuse of UI components usually, which represent the interactive functionalities of applications partially. In this paper, we present some common problems of the current UI component-based reuse and integration, and propose our solutions: a security-enhanced "component retrieval and integration description" method. Our purpose is to construct a reliable large-scale reuse and integration system for Web applications.

Index Terms—UI Component, Reuse, Integration, Mashup, Web Application, Retrieval, Security

I. INTRODUCTION

More and more information/knowledge is available on the Web with the development of the Internet, but they are not always in the forms that support end-users' needs. Mashup implies easy and fast integration of information in order to enable users to view diverse Web contents in an integrated manner. However, there is not an uniform interface used to access the data, computations (application logic) and user interfaces provided by different kinds of Web contents. Although there are widely popular and successful Web services such as Google Maps API [12], unfortunately, most existing Web sites do not provide Web services. Web applications are still the main methods for the information distribution. Compared with Web services and Web feeds, it is difficult to integrate Web applications with other Web contents because the open APIs are not provided. For example, the famous global news site CNN [3] provides the online news search function at site side for general users. However, this news search function can not be integrated into other systems because CNN does not open search function as a Web service. Similarly, BBC Country Profiles [2] does not provide the Web service APIs and it is difficult for the developers to integrate it with other Web services.

Beyond the limitation of open APIs, the integration of Web application contents and functionalities at the presentation and logic layers based on the reuse of UI components becomes a popular issue. Here, the Web application functionality indicates a part of Web application/page, and works as a mechanism to generate the contents responding to end-user request dynamically. Usually, these functionalities are realized by server-side logic processing or client-side scripting, and the end-users use the UI component (e.g. text input field, drop-down list in a form) to complete the request/response message exchange. Some approaches and systems are proposed to realize the integration of general Web applications by reusing the UI component or emulating the functionality (and contents extraction). However, all these approaches have to face some common problems such as component retrieval in a large-scale Web resource. Moreover, with the development of RIA technologies, there are more and more Web contents dynamically generated by client-side scripts or plug-ins. For interacting components (data and code) of multiple-parties together and executing within the Web user's browser, whose original role was just to render and display static contents, it is clear that at a minimum we need certain security mechanism to be enforced. In this paper, we present the main problems of current component reuse in detail, and propose our solutions: component retrieval and integration description. We also state the current security issues of client-side information/functionality integration and reuse. Then we present our solutions by leveraging the existing soon-standardized methods and an original access control mechanism (here, the risk of privacy leakage issue of server-side composition [37] is out of our scope). Our purpose is to construct a reliable large-scale reuse and integration system for Web applications. Our research emphasis is laid on the reuse and integration of the existing Web applications by the general Web users at client side instead of the component or portlet-based development by the application developers.

The organization of the rest of this paper is as follows. In Section 2, we present the current problems and research about the reuse and integration of UI component. In Section 3 we explain our solution in detail. We discuss and evaluate the new opportunities brought to the Web application integration in Section 4. Finally, the conclusion and the future work are given in Section 5.

II. CURRENT PROBLEMS AND RESEARCH

Most Web mashup technologies are based on the combination of Web services or Web feeds. Yahoo Pipes [34] and Microsoft Popfly [24] are the composition tools to aggregate, manipulate, and mashup Web services or Web feeds from different Web sites with a graphical user interface. Mixup [35] is a development and runtime environment for UI integration [7]. It can quickly build complex user interfaces for easy integration by using the Web service APIs available.
Mashup Feeds [28] and WMSL [27] support integrated Web services as continuous queries. They create the new services by connecting the Web services using join, select and map operations. Like these methods, Google Mashup Editor [13], IBM QEDWiki (IBM Mashup Center) [19] and some other service-based methods [23], [32], [36] are also limited to the combination of existing Web services, Web feeds or generated Web components.

In this paper, we focus on the general Web applications. For the reuse and integration of parts from Web applications without APIs, the partial Web page clipping method is widely used. The users clip a selected part of Web page, and paste it into a personal Web page. Internet Scrapbook [22] is a tool that allows users to interactively extract components of multiple Web pages by clipping and assembles them into a single personal Web page. However, the extracted information is a part of static HTML document and the users cannot change the layout of the extracted parts. C3W [11] provides an interface for automating data flows. With C3W, the users can clip elements from Web pages to wrap an application and connect wrapped applications using spreadsheet-like formulas, and clone the interface elements so that several sets of parameters and results may be handled in parallel. However, it does not appear to be easy to realize the interaction between different Web applications (like Safari Web Clip Widgets [4]) and needs a C3W-only Web browser. Extracting Web pages by end-user programming [15] is more suitable to generate mashup applications at client side. Marmite [33], implemented as a Firefox plug-in using JavaScript and XUL, uses a basic screen-scraping operator to extract the content from Web pages and integrate it with other data sources. The operator uses a simple XPath pattern matcher and the data is processed in a manner similar to Unix pipes. Intel MashMaker [9] is a tool for editing, querying, manipulating and visualizing continuously updated semi-structured data. It allows users to create their own mashups based on data and queries produced by other users and by remote sites. However, these methods need the professional Web programming ability [6], or have other limitations (e.g. need the components produced by portlets [21] or WSRP [31]), and have to face the following common problems as Mashroom [29] and Dapper [8].

- "How to find the most suitable functionality from the large-scale Web resources?". Currently, the users need to find the functionalities from Web sites by search engines (e.g. Google, Yahoo) or use the predefined UI component library, which is also constructed manually and limited to small scale. As shown in Figure 1, by search engine, the users have to check the search results by the following steps.

1) The users send the query request (keywords) to search engine and get the response page, which contains a list of URLs linked to search result pages.
2) The users check each result page to find out whether the page contains the suitable Web functionality/UI component or not (not suitable or no component).
3) The users compare the suitable components and select one as the most suitable component (for further reuse and integration).

The traditional Web functionality retrieval method is inefficient (even ineffective) because the current general Web search engines employ the contents-oriented search mechanism mainly, and the information of functionality is beyond the analysis range. Moreover, the search result ranking method of these non-functionality-oriented search engines can not satisfy our functionality retrieval, and leads to the time-costing manual check and comparison.

- "How to share and reuse the developed parts of a mashup application with other users/developers?". Currently, the users can select and integrate the desired parts (functionality or contents) from the different Web applications to develop the mashup applications. However, it is still difficult to share the parts of a developed mashup application if they do not know the detailed information about the parts such as the interface or parameters. For example, as shown in Figure 2, User A integrates a and b from the Web application A and B respectively and generate ab, then the c of Web application C is integrated and the abc is generated finally. However, if User B wants to reuse the ab, or User C wants to integrate the abc with others such as the d of Web application D, they have to know the detailed internal structures and interface of abc. It is difficult for general users to share the developed parts for the further reuse easily because there is not an uniform interface or structure for different Web resources and mashup development.

- "How to realize the flexible integration of various Web applications?". Currently, the most used mode of the integration of Web applications is "emulates the request submitting functionality and gets the response page, then extracts the partial information from the response page for the further reuse and integration". It is the simplest mode and may omit the following possible situations.

- Between the request submitting page and the page containing the target partial information, there are more than one page transitions.
- For a request, there are more than one possible responses. For example, the string comparison is
Fig. 2. Users need to share and reuse the developed parts often used in the emulation of optionlist-based page transition, and the number of the corresponding options is not always one.

- Errors or Exceptions are thrown during the implementation of mashup applications.

"How to secure the interoperation of components from different origins?". Currently, as a main execution platform, the Web browser only provides a single principal trust model based on the Same-Origin Policy (SOP) [20] as a base security mechanism to regulate cross-origin interaction among Web contents. The characteristic of the SOP states that the contents of one origin can only access the ones from its own origin, nevertheless, such access is forbidden for the resources of other origins. This causes that either the component-component separation with controlled interaction or the component-server communication is impossible. However, the nature of the information/functionality integration and reuse involves multiple-parties' contents in interoperating with each other, to provide extra-value functionalities that can give users more various experience. Consequently, a contradiction goes that there is either no trust or full trust when incorporating components of different origins. Therefore, in a lot of cases the composers are forced to abandon the security for getting more functionality, which incurs high security risks because the confidentiality and integrity of the component from one site can get complete control over the ones from other sites.

Base on the analysis of four questions above, We assert that the current approaches can not realize the various and complicated integration systematically. In our past research, we also have given the solutions by end-user programming method [15] or non-programming description system DEEP [14] and WIKE [16]. However, these developed systems also cannot solve the described problems well.

III. OUR SOLUTION

In order to realize the integration of the large-scale Web resources based on the component reuse, it is necessary to construct an UI component retrieval system and propose a description approach to describe the integration process. Here, we propose our framework, which is being developed to solve the current problems.

The component-oriented search mechanism is employed in UI component retrieval system. As shown in Figure 3, the Web functionality search engine designed for component retrieval works as follows.

1) A crawler collects the Web pages containing Web functionalities from the large-scale Web resources.

2) The following related information is extracted from the Web page.
   - page URL
   - meta data
   - component type (tag name)
   - component name and ID (name and ID attribute)
   - request submitting method name and type (e.g. GET and POST)
   - information inside the component (e.g. option values of optionlist)
   - information near the component (e.g. hidden values in the same FORM)
   - structure (e.g. a single text input field, a list of links)

3) The extracted information is analyzed and classified to generate the recognition pattern of each Web functionality, then the index information is created for the components retrieval.

Here, the component type is the type of request-input element, such as InputBox (text input field), OptionList (dropdown option list in selectbox), and LinkList (anchor list), in the page where the request of end user is submitted.

The users search for the components by giving the component type and other search keywords. After the desired components are found, they are described in an annotation method for the integration. Here, we define an uniform format shown in Figure 4 for the following basic entities, which describe the reuse and integration based on the page transition (Figure 8 gives the examples in Section IV).

- Flow: a flow represents the data flow or work flow of an integration. It works like the main function of an executive program and receives the parameters from users as the request.
- Page: a page is a basic information container in a Web application. It contains the desired partial information such as the component or contents, which is extracted by the users’ designation.
- Component: a component is used to emulate the functionality and get the response (page) according to the request.
- Collection: a collection contains polymorphic algorithms that operate on collections such as sort, reverse and swap.
- Checker: a checker is used to check whether the extracted data is valid.
- Convertor: a convertor is used to convert the data into another format.
- Pattern: a pattern is used for comparing strings, searching strings, extracting substrings and other string-based operations as a selective option.
The format contains five main parts: name, comment, input, output and exception. These items reflect the end-user operations (e.g. users find the text input field, input the keywords, submit the request, and search for the target contents in the response page), and give the description of target Web contents type/property. Name is simple description of entity and the comment contains the keywords, which reflects the specification of the entity. Source of input is the URL, link or object output from other entities. Path is used to locate the target contents. The value of path is the XPath expression of a target part in HTML document of Web page. Path is used to find the target parts from the Web page. In the tree structure of HTML document, each path represents a root node of subtree and each subtree represents a part of Web page. Property is text, image, link, object or component. Text is the character string in Web pages such as an article. Image is one instance of the graph. Link is a reference in a hypertext document to another document or other resource. Object is one instance of the video or other multimedia file. Component is the type of request-input element. Type is single, list and table. Single means a part without similar parts such as the title of an article. List means a list of parts with similar path values such as result list in a search result page. Table means a group of parts arranged in 2-dimensional rows and columns such as the result records in a Google Image Search result page. Matcher performs match operations on a string by interpreting a pattern entity. Output gives the names of output elements and the exception defines the error message thrown when the exception happens.

Compared with the Web services, it is not easy to access and integrate Web applications because they are designed for browsing by users, not for the parsing by computer program. The Web pages of Web applications, usually in HTML or XHTML formats, are used to display the data in a formatted way, not to share the structured data across different information systems. Without the interface like SOAP (Simple Object Access Protocol) or REST (Representational State Transfer), we have to use the extraction and emulation technologies to interact with Web applications. The extraction is used to reach the target contents and extract them from Web pages. The emulation is used to realize the process of sending request and receiving response. For the request submitting, there are POST and GET method, and some Web sites use the encrypted codes or randomly generated codes. In order to get the response Web pages from all kinds of Web applications automatically, we use HtmlUnit [18] to emulate the submitting operation instead of URL templating mechanism. The emulation is based on the event trigger. For example, in the case of InputBox (enter the query keywords into a form-input field by keyboard and click the submit button by mouse to send the query), the text input field is found according to the path and the query keywords are inputted. Then the click event of the submit button is triggered to send the request and get the response Web page.

The trend of RIA technologies accelerates the development of rich client solution. For example, many Web sites use the DHTML (Dynamic HTML) or DOM scripting technology to
create the interactive Web pages. It contains HTML, client-side scripts (such as JavaScript), DOM (Document Object Model) and etc. The scripts change variables (including the elements outside the target parts such as the hidden values) programmatically in an HTML document, and affect the look and function of static contents. In this situation, to integrate and reuse component reliably, we propose four specific security requirements and solutions for our framework, part of them are absorbed from the categories summarized by [26].

- Component Separation: The reusable components should be separated with each other. The DOM tree and script cannot be unauthorized modified by other components. In case two or more components from the same origin contain scripts, the separation mechanism also should be enforced.
- Functionality Interaction: Each functionality should have the ability to interact with other functionalities and the integrator. While the restriction is that the entity which contains critical information should not be unauthorized read and rewriting by others.
- Cross-origin Communication: Functionality needs to be able to communicate with not only its own origin but also can interact with other origins under control.
- Usage Control: The author of reusable component should be able to decide whether the specific functionality can be accessed by other component or not.

Concerning about practical and extensible, it is meaningful to leverage the existing standardized technologies. The solutions are illustrated with the scenario in Figure 5: functionality N1 from Web site N is integrated into the integrator page A1 from Web site A. Then the integrator A1 contains two parts: functionality A1 and functionality N1 from different origins respectively. Each functionality possibly contains both static (HTML) and active (Script) contents.

The interaction or communication is restricted with SOP.

- The iframe tag separates one inclusion of site N (N1.html) from site A with no interaction under the SOP restriction. The postMessage [17] is a popular browser API extension which provides principal interaction between frames/iframe. It provides the postMessage() operation on a DOM window and iframe object. To combine the postMessage with iframes offers restricted interaction and separation is an efficient method.
- The script tag containing another inclusion of site N (N1.js) is for getting interacting with other parts such as the functionality A1 and N1.html. The tag offers no separation at all. Therefore it would bring confidentiality or integrity break risks because the script part has the ability to read and write other component’s state. Caja [25] is a safe subset of JavaScript and most widely used for script isolation within the same execution. The functionalities’ scripts can automatically be translated to Caja modules, which are isolated from each other to restrict the unsafe JavaScript, such as eval or with.
- The XMLHttpRequest object of JavaScript is not allowed to make cross-origin request. In case the functionality N1 needs to communicate with integrator A1’s remote Web server, Cross-Origin Resource Sharing (CORS) [5] is an efficient and widely supported extension of the HTTP to provide cross-origin request. It allows the remote Web server to indicate whether the functionality of other origin have the right to access to its resources or not.
- As shown in Figure 6, our framework holds an interface to insert a new item called “access” into the format of entity. The access tag uses the “public” and the “private” attribute to declare whether the entity can be accessed by the normal public flow or only used privately. Moreover it can specify the applications/pages of which origin can access into. Further, this interface can be a part for protecting copyright of some special component.

```html
<access>
  <allow></allow>
</access>
```

Fig. 6. Access Item

IV. DISCUSSION

The construction of the functionality/component search engine solves the first problem described in Section II, and the uniform format of the basic entities gives a solution to the second and third problems as following example.

As shown in Figure 7, we describe the function of country information search by the entities. We search for the selectable option at top page of BBC Country Profiles [2], and submit the request to get the target country page. We extract the map of country and the URL of timeline page from country page. We search for the key events using the given month and extract them from the timeline page. Figure 8 gives the definition of Page (Start) and Component (optionlist) as the examples. We use Firebug [10] to get the XPath of target part by GUI and avoid the manual analysis of the HTML document. The entities are combined with each other and the whole flow is generated finally. The matcher, pattern and checker are used to keep the correct process, and the exception message is thrown if the errors happen. Moreover, The convertor and collection create the event searching function by extracting the corresponding...
information from a simple static Web page. As a part of the 
Flow, the Flow 1 or Flow 2 can be reused by other users, who 
do not need to know the detailed internal structures of these 
sub flows because of the same format.

We realize a component-based reuse and integration of 
Web contents with component retrieval and entity description. 
Similarly, there are WSDL and WADL [30] used to describe 
a series of Web services in detail such as abstract types, abstract 
interfaces and concrete binding. We could not believe that all 
the tasks could be completed fully automatically by handing 
WSDL or WADL files to WSDL2Java or WADL2Java [1]. 
Compared with the WSDL or WADL, our entities need a 
shorter and simpler description format, and are applicable to 
the description of general Web applications. It is easier to read, 
write, reuse and update any part of mashup Web application 
than end-user programming methods. Our annotation approach 
makes it possible for users with no or little programming 
experience to implement the integration of Web contents from 
various Web applications. Any contents from any kind of Web 
applications are available.

Except the component and the object generated by client-
side scripts, the extracted partial information from Web application 
are in XML format. We need a template processor to 
transform XML data into HTML or XHTML documents. The 
users can select our predefined XSLT files as the stylesheet 
transformations, or modify them to get the customized visual 
effects. Also, the full original XSLT files created by users are 
valid in our system.

The aim of the example stated in Figure 7 which contains 
also all static contents from one origin is to address the 
workflow of our system. Yet we still consider the security issues 
cannot be ignored since the active contents are widely 
used within the attractive and complex RIA application and 
Web pages.

V. CONCLUSION

In this paper, we gave the current problems of component-
based large-scale Web application reuse and integration, and 
presented an UI component retrieval system which realizes 
the quick and effective component searching. We also pro-
posed a description method to define the basic entities in 
application reuse and integration. Additionally, we analysis 
the contradictions between traditional security mechanism: 
SOP and the nature of mashup: interoperation. And then 
we present our security requirement and solutions. By our 
retrieval, description security-enhanced system, the typical 
Web users can construct the mashup Web applications easily 
and quickly.

As future work, we plan to modify our approach as follows.
• We will combine and develop the various tech-
niques/tools, and propose a friendly GUI for users to 
generate the entity description file more easily and fully 
automatically.
• We will explore more flexible ways of integration, and 
construct an open community for sharing various mashup 
components in future. The technologies of semantic Web 
and RDF will be used to centralize the various Web applications, Web services and other Web contents.
• We will develop a fine-grained access mechanism which 
could trace the information flow in the functionalities.
• We will propose a stabler extraction method to solve the 
problems of frequently changed contents formats.

VI. ACKNOWLEDGEMENT

This work was supported by a Grant-in-Aid for Scientific Research A (No.22240007) from the Japan Society for the 
Promotion of Science (JSPS).

REFERENCES
