A Method for Integration of Web Applications Based on Information Extraction

Hao Han and Takehiro Tokuda
Department of Computer Science, Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan
{han, tokuda}@tt.cs.titech.ac.jp

Abstract

Integration of Web services from different Web sites has brought new creativity and functionality to Web applications. These integration technologies, called mashup or mixup, have made a shift in Web service development and created a new generation of widely popular and successful Web services such as Google Maps API and YouTube Data API. However, the integration is limited to the Web sites that provide the open Web service APIs, and currently, most existing Web sites do not provide Web services. In this paper, we present a method to integrate the general Web applications. For this purpose, we propose a Web information extraction method to generate the virtual Web service functions from Web applications at client side. Our implementation shows that the general Web applications can be also integrated easily.

Keywords: *Web application integration, information extraction, Web service, mashup, end-user programming*

1 Introduction

With the development of the Internet, the Web becomes the richest source of information. Although there is a tremendous amount of information available, they are not always in the forms that support end-users' needs. There is a growing trend of enabling users to view diverse sources of data in an integrated manner, called mashup or mixup. These integration technologies have made a shift in Web service development and created a new generation of widely popular and successful Web services such as Google Maps API and YouTube Data API.

However, this integration is based on the combination of Web services and limited to the Web sites that provide the open Web service APIs. Unfortunately, most existing Web sites do not provide Web services. Web applications are still the main methods for the information distribution. For example, CNN lets users search for the online news by entering the keywords at Web page. Once the users submit the search, CNN would present the news search results. However, this news search function can not be integrated into other systems because CNN does not open this search function as a Web service. Similarly, Wikipedia does not provide the official Web service APIs and it is difficult for the developers to integrate it with other Web services.

In this paper, we propose a *Web information extraction* method to realize the Web application integration. We select the target Web applications, search for the desired information, and extract the partial information to realize the virtual Web service functions. All the processes of Web information searching and extraction are run at client-side by end-user programming like a real Web service. We developed a Java-based class package for Web information searching and extraction, and the users can integrate Web applications with our class package easily. Our implementation shows that we do not need to write too much program.

The organization of the rest of this paper is as follows. In Section 2 we give the motivation of our research and an overview of the related work. In Section 3 we explain our Web information extraction approach in detail. We construct a Web application integration system and give an evaluation of our approach in Section 4. Finally, we conclude our approach and give the future work in Section 5.

2 Motivation and Related Work

With the development of Web 2.0, there are a rapidly growing number of mashup applications. According to ProgrammableWeb [12], a mashup community Web site, the total number of listed mashup applications is more than 3000, and on average there are more than 3 new systems generated everyday in May 2008. Although many users would like to build mashup applications, the existing Web services are not adequate for the users' needs, and many famous and popular Web sites do not provide Web services.

Many systems have been developed to integrate the Web applications. The most widely used method is partial Web page clipping. The users clip a selected part of Web page, and paste it into a personal Web page. I-know [9] is a sim-

978-0-7695-3261-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICWE.2008.29

ple Web application to generate a customized personal Web page. It extracts the partial text information between the defined start keyword and end keyword from a Web page, and creates a Web page by listing the extracted text information. However, the extracted information is limited to text. Internet Scrapbook [10] is a tool that allows users to interactively extract components of multiple Web pages by clipping and assemble them into a single personal Web page. However, the extracted information is a part of HTML document and the users can not change the layout of the extracted parts. C3W [4] 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 interfaceelements so that several sets of parameters and results may behandled in parallel. However, it does not appear to be easy to realize the interaction between different Web applications.

Extracting typed data from multiple Web pages is more suitable to generate mashup applications. Marmite [14], 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. MashMaker [3] 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, they do not appear to support the integration of dynamically generated Web pages like the result pages from form-based query.

Some approaches are proposed to construct Web services based on the Web applications to realize the integration. Pollock [11] can create a virtual Web service from form-based query interface of Web sites. It generates wrappers using XWrap, and WSDL file using Web site-related information, then publishes the details of the virtual Web service into UDDI, but this system needs the users to parse the HTML documents of the form-based Web pages. H2W [13] also provides a Web services generation method based on information extraction from existing Web applications. These approaches take a great deal of time and skills to create such services in a proxy server run between the target Web applications and users, and it is extremely unlikely that the constructed Web services can support all the needs of all of its end-users.

To address these problems, we propose a Web information extraction approach to integrate the Web applications. Compared with the developed work, our approach has the following features:

- As shown in Fig. 1, all the processes of searching, extraction and integration are run at client-side by end-user. The users can realize the different personal Web services to support all the needs by themselves, and do not depend on the proxy server.
- We focus on extracting typed data from Web pages and the extracted result is structured data.
- We extract all kinds of information including text, link, image and object from different layout such as list and table.
- We support the information extraction in the static Web pages and the dynamically generated Web pages such as the result pages from form-based query.
- The users can realize the continuous information searching and extraction over multiple Web pages by end-user programming.

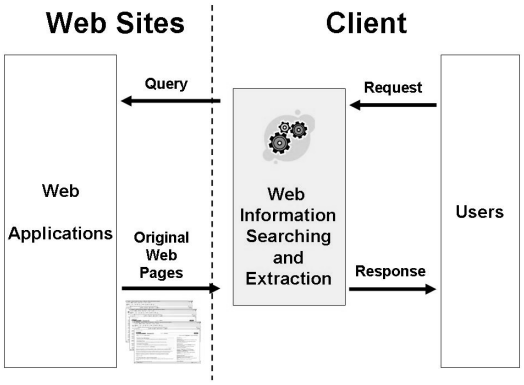

Figure 1. Client-Side-Approach

We explain our approach, construct a Web application integration system, and give an evaluation in the following sections.

3 Web Information Extraction

Usually, a real Web service responds to the requests of users by returning the data in the server-side database. The Web service developers design the query commands by an interactive and programming language such as SQL to retrieve data in the tables of database. For our Web application integration, the Web applications work as the server-side database and the target Web pages work as the tables. The end-users use our Web information extraction method to search for the desired information and extract it.

Compared with Web services, the Web applications are not suitable for integration 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 used to share the structured data across different information systems. In order to realize the Web information extraction, we simulate the browsing of users by end-user programming to reach the desired information, and use string matching and tree structures of Web pages to extract it.

3.1 Data Type

There are many kinds of information in Web applications such as text, picture and link. During the information extraction, the users need to specify the type of target data. Data type represents "*What kind of information is needed?*". For example, in a Web page, usually each link contains an anchor text associated with a URL. Without the specified data type, we can not get the right information because we do not know which one is needed between text and URL.

In a Web page, a visible item represents a piece of information that can not be divided into smaller parts, and is the node value or attribute of a single node in an HTML document. We give our data type definition of visible items as shown in Fig. 2. There are two kinds of data types. The first kind contains text, image, object and link. Text is the character string in Web pages such as an article. Image is one instance of the picture embedded in tag <img>. Object is one instance of the video or other multimedia file embedded in tag <object>. Link is a reference to another document or other resource embedded in tag <a>. Usually, the first kind of data types are the final results of extraction, and used in Web information interaction and integration. The second kind contains select-option and form-input. Many Web applications use them for Web pages transition. Select-option is an option in the drop-down list and each option represents a link. Form-input is an input field in a form used to accept the users' queries. The users fill in a text form-input, submit the form, and the application presents corresponding results.

We use these six data types in information searching and extraction in the following sections.

3.2 Information Searching

Not all the contents of Web applications are necessary and useful for the users. We search for the target information in Web pages to realize the query functions. Like the SQL, we select the information from the target Web pages where the information satisfies some conditions. We define the searching function as follows:

$$\mathrm{Search(P, K, T_1, R, T_2)}$$

where, P is the target Web page, K is the searching keyword that refers to the desired information, T_1 is the data

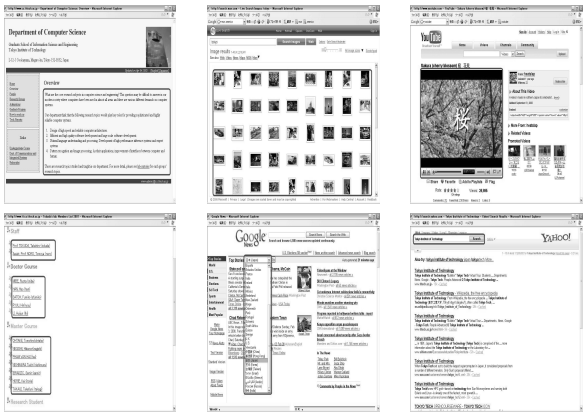

Figure 2. Data types: text, image, object, link, select-option and form-input

type of searching keyword, R is the range of searching, T_2 is the data type of desired information, and the returned value is the node list of search results. The searching range represents the number of visible items between the searching keyword and the desired information, and has five types as shown in Table 1. The searching keyword is a very important element during our searching and works as an index of our desired information. We need to find a suitable searching keyword because a suitable searching keyword can give a more precise search result like a well-written where-clause of SQL.

We give the examples in our implementation.

3.3 Information Extraction

After we find the target node, we need to extract the information from the node in text format excluding the tags of HTML document according to the defined data type. There are three kinds of data structures in the Web pages as shown in Fig. 3: single, list and table. Single means a node without similar sibling nodes such as the title of an article. List means a list of nodes with similar paths such as result list in a search result page. Table means a group of nodes arranged in 2-dimensional rows and columns such as the result records in a Google Image Search result page.

Figure 3. Data structures: single, list, table

Table 1. The Types of Searching Range

Value	Range
R=MAX_VALUE	the first T_2 node following after K in the whole page
$R > 0$	the first T_2 node following after K within R
$R = 0$	the T_2 node containing K
$R < 0$	the first T_2 node previous to K within $\|R\|$
R=MIN_VALUE	the first T_2 node previous to K in the whole page

Each HTML document of Web application can be parsed as a tree structure, and our extraction method is based on the analysis of the tree structure. We define the extraction function as follows:

$$\mathrm{Extract(N, T, S)}$$

where, N is the target node, T is the data type, S is the data structure, and the returned value is the extraction result list.

We give the detailed processes of extraction in the following sections.

3.3.1 Single

Among the information extraction from three kinds of data structures, the extraction from a single target node is the most easy and basic. As described in Table 2, we extract the information according to the defined data type.

For example, the extracted information of a photo is the value of attribute *src* of node <img>, and the extracted information of a link is the value of attribute *href* of node <a>.

We do not extract the information from the select-option and form-input. They are used in 3.4 for request submitting.

3.3.2 List

The node list extraction is based on the tree structure of HTML document. In the tree structure, each node has its own path. We use the following steps to extract the corresponding node list of node N if the data structure is list.

1. We get the path of node N.
2. We get the parent node P of N.
3. We get the subtree S whose root node is P.
4. We get the node list L of which each has the same path as N without considering the orders of child nodes of P under S.
5. If we find two or more than two nodes in L, or P is <body>, then L is the final node list. Otherwise, we set the parent node of P as the root node of S, then go to Step 4.

Each node of the extracted node list L represents a part of list as shown in Fig. 4.

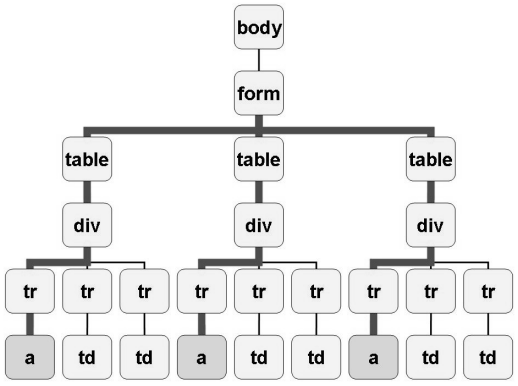

Figure 4. Node List Extraction

We extract the information from each node of node list according to the data type by the method described in section 3.3.1.

3.3.3 Table

The table is often used in Web pages. We extract the information from a table structure by using the node list extraction method twice because the table is 2-dimensional and can be viewed as a list of list.

3.4 Request Submitting

Usually, for the users, there are two basic types of methods to send their requests and get the response Web pages. The first type is to click an option in drop-down list in a Web page by mouse to view a new Web page. The second type is to enter the query keywords into a form-input field by keyboard and click the submit button by mouse to send the query. For the request submitting, there are POST method 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 sites, we use HtmlUnit [8] to simulate the submitting operation.

We define the request-submit function as follows:

Table 2. Information Extraction from Single Node

Data Type	Information
text	node value of corresponding node
image	attribute value of corresponding <img> node
object	attribute value of corresponding <object> node
link	embedded link value of corresponding <a> node

$$\mathrm{Submit(F, K)}$$

where, F is the select-option or form-input, K is the selected item name or query keywords, and the returned value is the result page. The users give the selected item name or query keywords, and get the result page as response.

Our request-submit function is applicable to the general Web sites without the manual analysis.

We use the request-submit function in our implementation.

4 Implementation and Evaluation

In this section, we implement our approach to combine Web services, Web feeds and Web applications from more than one Web sites into a single integrated system. Compared with Web services and Web feeds, we extract the information from Web applications to realize the functions as those of the Web services. In order to interact with different Web sites, we use the programming language such as Java and client-side script language such as JavaScript to call the Web services, parse the Web feeds and extract the information from Web applications. Finally, these information are integrated into a single system and interacted with each other.

Our system, called *World in Web*, has the following functions.

1. We can view the following country information after we select a country from country list as shown in Fig. 5.

   A: The position information in map

   B: The country name, population, capital city and area, and the information and photo of the country leader

   C: The main cities

   D: The latest news

   E: The key events of the given year.

2. We can view the following city information after we click a city from the listed main cities as shown in Fig. 6.

   F: The position information in map

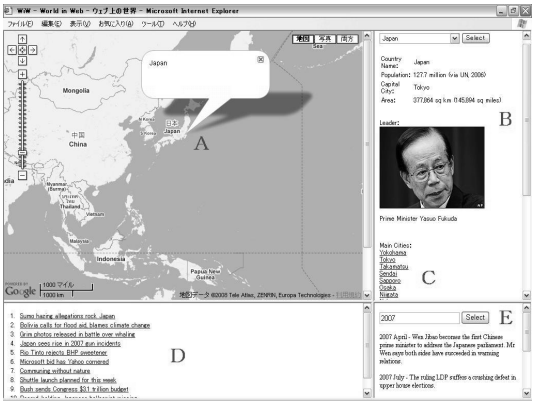

Figure 5. World in Web - Country Information

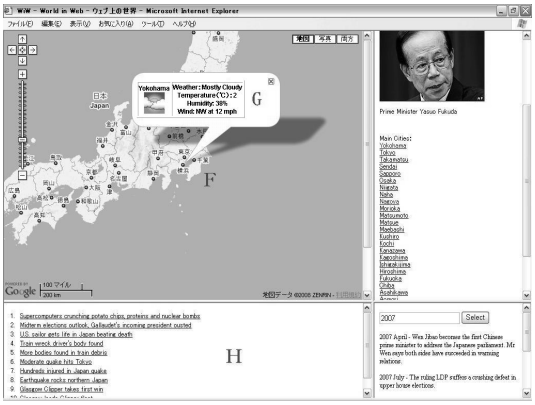

Figure 6. World in Web - City Information

   G: The weather information

   H: The latest news

We use the following steps to construct our system.

1. We select the target Web services, Web feeds and Web applications listed in Table 3.

2. As shown in Fig. 7, we realize the function of CNN news search engine. We search for the form-input from the top page of CNN, and submit the request to get the search result page. We extract the result records from the result page using the searching keyword.

Table 3. The Target Web Services, Web Feeds and Web Applications

Type	Source	Description
Web service(SOAP)	Google Maps API [6]	diagrammatic representation of an area
Web service(REST)	Google Weather API [7]	state of the atmosphere such as heat and rain
Web feed(Country List)	Google Data [5]	country name and ISO code
Web feed(City List)	Google Data	main city names and positions
Web application(Static URL)	BBC Country Profiles [1]	history, politics and economic background of countries
Web application(Dynamic URL)	CNN News Search [2]	CNN Web news article search engine

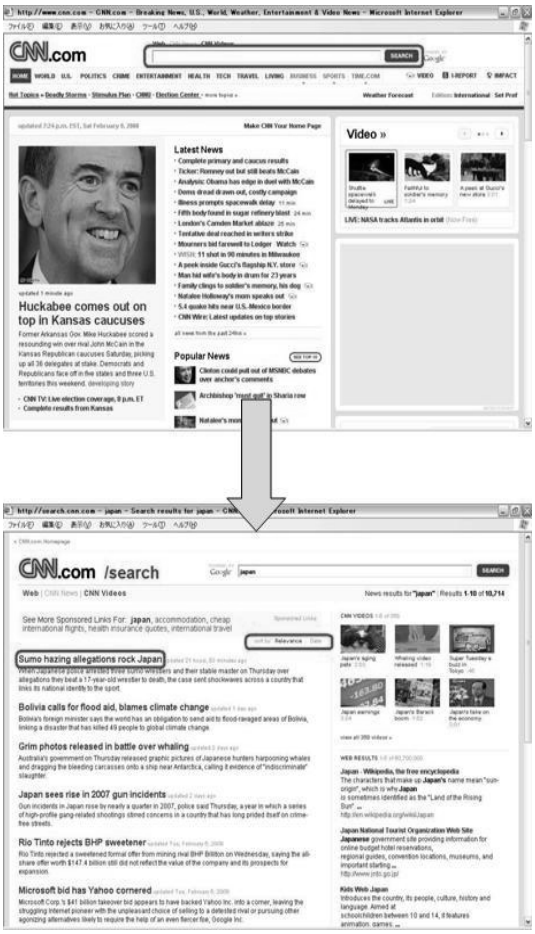

Figure 7. The Process of CNN News Search

3. As shown in Fig. 8, we realize the function of BBC country information search. We search for the select-option at top page of BBC Country Profiles, and submit the request to get the target country page. We extract the information of country and leader, and the URL of timeline page from country page. We search for the key events using the given year and extract them from the timeline page.

4. We program to realize the interaction of CNN news search and BBC country information search with the Web services and Web feeds.

Our approach is based on the string matching and tree structures of Web pages. We realize the searching and extraction functions like the retrieval function of SQL, and complete the integration of Web applications by our defined functions.

We use data type in our information extraction and our extraction result is structured data excluding the HTML tags. The extracted information can be used to interact with other Web sources freely by users' needs. Moreover, We use the data structure of desired information in Web pages and simplify the extraction process because we do not need to search for the desired parts one by one if they are grouped into a list or table.

For the dynamically generated Web pages, the URLs are changed by the request data, and can be parsed to find the generation rules if the Web sites use GET method. Some approaches parse the example Web page URLs manually and get the generation rules, and get the target Web pages using the generation rules. However, besides the GET method, there are POST method and other methods. It is difficult, or even impossible, to generate the target Web page URLs. Instead, we get the target page by simulating the submitting process. It is applicable to all kinds of request-submit and does not need time-consuming manual analysis.

We give a comparison of the actual program code sizes in Table 4. Using the defined *Search*, *Extract* and *Submit* functions, our approach realizes the Web applications integration without writing too much program.

5 Conclusion and Future Work

In this paper, we have proposed a Web information extraction approach to realize the integration of the general Web applications. We developed a Java-based class package for Web applications integration and the users can construct a mashup applications with our class package easily. All the processes of Web information extraction are run at client-side by end-user programming, and the users can realize the personal virtual Web service functions by their own needs without writing too much program.

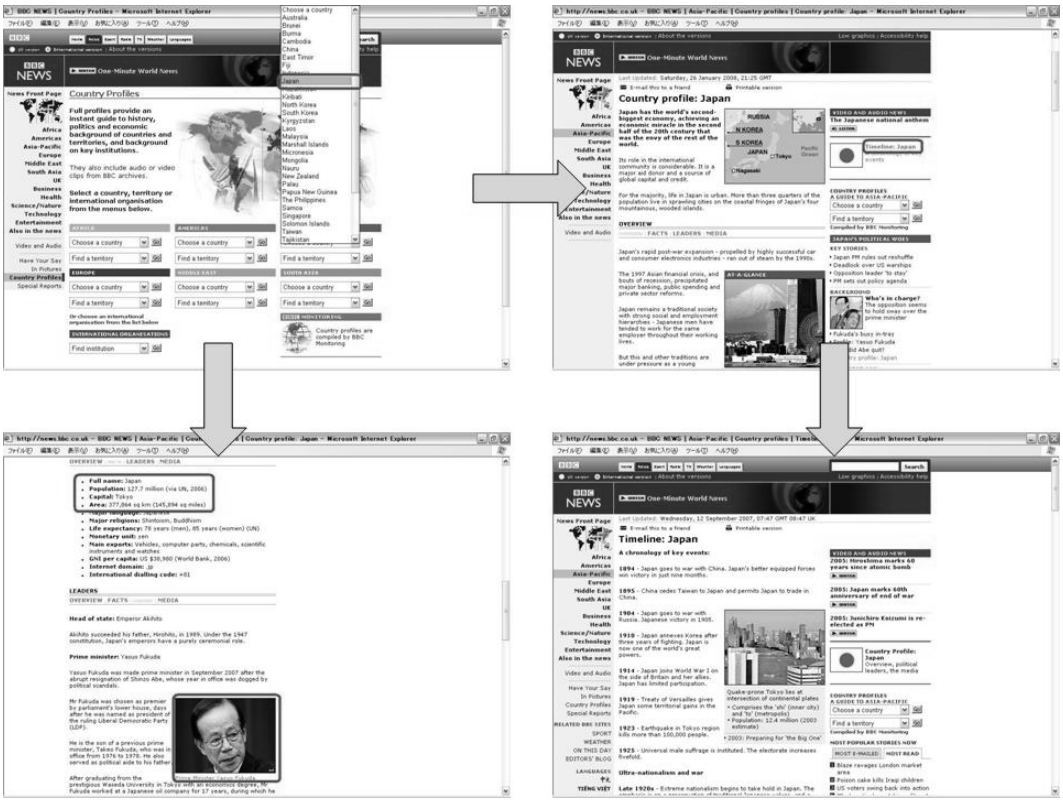

Figure 8. The Process of BBC Country Information Search

Table 4. The Comparison of Program Code Sizes

Country Information	Key Events	News Search	Maps	Weather	Country List	City List
31 lines	9 lines	15 lines	16 lines	37 lines	15 lines	23 lines

Although we realize the Web application integration, our approach still depends on some manual work such as the searching keywords decision. As future work, we will modify our approach to propose an automatic searching keywords decision function. Moreover, we will give the GUI for easier configuration and realization of Web application integration with less programming. Additionally, besides the current developed Java-based class package, we will develop a JavaScript-based package in future.

References

[1] Country Profiles. http://news.bbc.co.uk/2/hi/country_profiles/.

[2] CNN. http://www.cnn.com.

[3] R. Ennals and M. Garofalakis. MashMaker: Mashups for the masses. In *The Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, 2007.

[4] J. Fujima, A. Lunzer, K. Hornbaek, and Y. Tanaka. C3W: clipping, connecting and cloning for the Web. In *The Proceedings of the 13th international World Wide Web conference*, 2004.

[5] Google Data APIs. http://code.google.com/apis/gdata/.

[6] Google Maps API. http://code.google.com/apis/maps/.

[7] Google Weather API. http://www.google.com/ig/.

[8] HtmlUnit. http://htmlunit.sourceforge.net/.

[9] I-know. http://i-know.jp/.

[10] Y. Koseki and A. Sugiura. Internet scrapbook: Automating Web browsing tasks by demonstration. In *ACM Symposium on User Interface Software and Technology*, pages 9–18, 1998.

[11] Y.-H. Lu, Y. Hong, J. Varia, and D. Lee. Pollock: Automatic generation of virtual Web services from Web sites. In *the Proceedings of the 2005 ACM symposium on Applied computing*, 2005.

[12] ProgrammableWeb. http://www.programmableweb.com/.

[13] M. Tatsubori and K. Takashi. Decomposition and abstraction of Web applications for Web service extraction and composition. In *The Proceedings of the 2006 International Conference on Web Services*, 2006.

[14] J. Wong and J. I. Hong. Making mashups with marmite: Towards end-user programming for the Web. In *The Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007.