Visual Wrapping and Functional Linkage of Existing Web Applications

Kimihito Ito
Meme Media Laboratory, Hokkaido University
N-13 W-8, Sapporo, 060-8628, JAPAN
Email: itok@meme.hokudai.ac.jp

Yuzuru Tanaka
Meme Media Laboratory, Hokkaido University
N-13 W-8, Sapporo, 060-8628, JAPAN
Email: tanaka@meme.hokudai.ac.jp

***Abstract*- HTML-based interface technologies enable end-users to easily use various remote Web applications. However, it is difficult for end-users to compose new integrated tools of existing Web applications. In this paper, the authors propose a framework where end-users can wrap remote Web applications into visual components called pads, and functionally combine them together through drag&drop-paste operations. We use, as the basis, a meme media architecture IntelligentPad that was developed by our research group. In the IntelligentPad architecture, each visual component called a pad has slots as data I/O ports. By pasting a pad onto another pad users can integrate their functionalities. Users can visually create wrapper pads for Web applications that he wants to use by defining HTML nodes within the Web application to work as slots. Examples of such a node include input-forms and text strings on the Web page. Users can directly manipulate wrapped Web applications on their desktop screen to define application linkages among them. Since no programming expertise is required to wrap Web applications or to functionally combine them together, end-users can build new integrated tools of wrapped Web applications. Authors also discuss applications of this framework to mobile computing.**

I. INTRODUCTION

During the last couple of years, the main portion of information resources in World Wide Web has been shifted from handmade HTML pages to server-generated HTML pages, such as those using CGI (Common Gateway Interface), ASPs (Active Server Pages), JSP (Java Server Pages) and PHP. A Web application is an application program that has HTML-based front-end for users to utilize some services provided by a remote HTTP server. Many companies and researchers provide Web applications, such as search engines, financial services, scientific analysis tools, and various other kinds of database services. Basically, every Internet user can access these Web applications.

End-users access Web applications through Web browsers. Thus, to combine two Web applications, users need to open these two Web pages, to input some data on the first page, to copy a part of the result, and to paste this copy into the input-form on the second page. Users need to repeat this process, if they want to apply the same processing to other inputs. In UNIX, users can compose command sequences using pipes and redirections. UNIX users can perform a job that is not implemented as a single command. On the other hand, Internet users cannot combine existing Web applications into a single application.

Revision and approval cycle of Web applications are becoming shorter and shorter, because their providers need to increase the quality and competitiveness of their application. Sometimes they change URLs, and sometimes they revise the format of front-end HTML pages.

Many researchers have worked on developing robust wrapping method of Web applications [12][18]. Some wrapping method has robustness for format changes. From the view point of computational learning theory, it is impossible to deal with every kinds of formatting change. We do not focus on such robustness in this paper.

We assume target users for our framework to be novice in computer programming but specialists in a domain such as finance or bioinformatics. In their intellectual activities, they repeat the process of 'think', 'try', and 'see'. When applied to problem-solving contexts, researchers call such a repetitive process a plan-do-see loop. Many feel that computers can be applied to support this process, speeding it up and driving progress around this cycle. This requires seamless support of the three phases: think, try, and see.

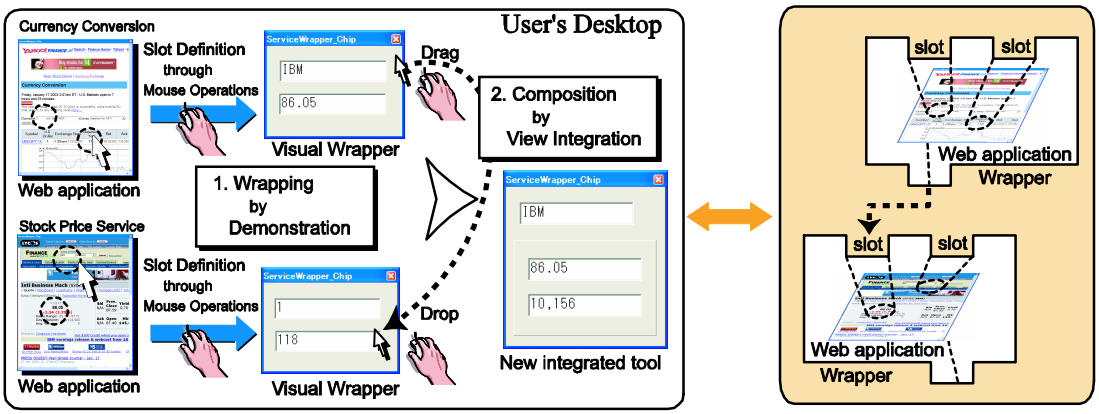
From this point of view, we propose a new framework for the visual wrapping and linking of Web applications to dynamically and visually compose an integrated function. Users can rapidly wrap any Web application with a wrapper component only through direct manipulation. Based on the IntelligentPad [21] architecture, users can combine these wrapped visual components called pads, by pasting pads on another pad, to compose a composite pad. This composition defines application linkages among Web applications to integrate them into a single composite function. It supports the users' seamless repetition of the three phases: think, try, and see.

In our previous poster presentation [10], we have presented only an outline of our approach. In this paper, we describe our method more precisely and discus the application to mobile computing.

II. RELATED WORKS

Web Service technologies such as SOAP(Simple Object Access Protocol)[4] enables us to interoperate services published over the Web. However, they assume that the API (Application Program Interface) library to access such a service is *a priori* provided by its server side. You need to write a program to interoperate more than one Web service. Our technologies, on the other hand, provide only the client-side direct manipulation operations for users to re-edit intellectual resources embedded in Web pages, to define a new combination of them together with their interoperation.

Open Hypermedia Systems [15] allow links and annotations to be added to documents outside the author's control and are designed to be integrated with any number of applications to provide hypertext functionality to everything from spreadsheets to graphics editors. Open Hypermedia Systems use link service [5] to resolve the source anchor of a link to all the possible destinations by querying a link database to identify relevant links. Hyperlinks in Open Hypermedia Systems can be extended to application linkages among Web applications.



User-defined views and their combination Web application linkage

Fig. 1. An outline of our approach to define functional linkage among Web applications
(A video demonstration is available at http://ca.meme.hokudai.ac.jp/people/itok/CHIP/movies).

There are a lots of previous research studies on visual wrapping of Web pages. However, there are a few research studies that allow end-users to wrap and to define functional linkage in the same environment. Bauer and Dengler[3] have introduced a PBD(Programming by Demonstrations) method in which even naive users can configure their own Web based information services satisfying their individual information needs. They have implemented the method into InfoBeans [2]. By accessing a InfoBox with an ordinary Web browser, users can wrap Web applications. By connecting channels among InfoBeans on the InfoBox, users can also integrate them functionally together. However, it seems difficult for users to reuse a part of composite Web applications defined by other users. W4F[19], which is semi-automatic wrapper generator, provides a GUI support tool to define an extraction. The system creates a wrapper class written in Java from user's demonstration. To use this wrapper class, users need to write program codes. DEbyE[7] provides more powerful GUI support tool for the wrapping of Web applications. DEbyE stores the extracted text portions in XML repository. Users have to use another XML tool to combine extracted data from Web applications. LExIKON [8] learns an underlying relation among objects within a Web page from a user-specified ordered set of text strings. There is no GUI support tool for the join of two extracted relations. WebView [6] allows us to define customized views of Web contents for mobile computers. However it seems difficult for end-users to create a new view that integrates different Web applications.

III. Web Application Linkage

A. Requirements

The end-users' creation of functional linkage among Web applications requires the following capabilities:

1. Easy specification of input and output portions of Web applications to reuse embedded functions in them.
2. Easy definition of functional linkage between Web applications to compose a new integrated application.
3. Easy decomposition and re-composition of composite functions in the user-defined integrated applications.

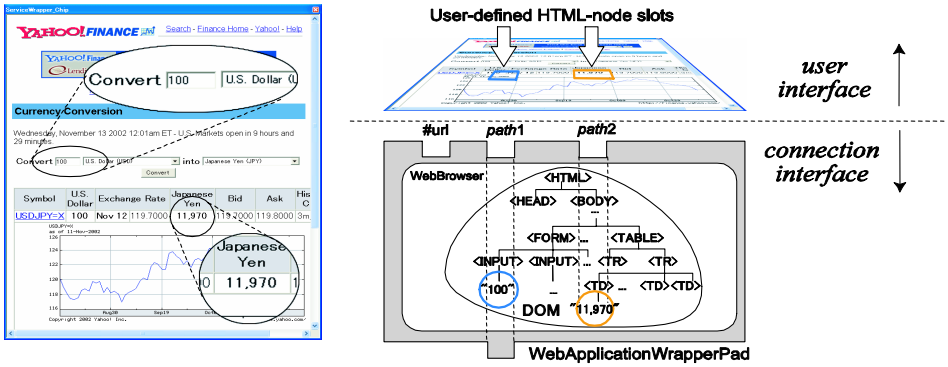
B. Our Approach

We will propose the use of IntelligentPad [20][21][22] technologies to achieve these three capabilities. IntelligentPad architecture allows users to combine media objects (called pads), such as multimedia documents and application programs, through their view integration. Each pad has slots as data I/O ports. Through drag-and-drop and paste operations, users can connect one pad to a slot of another pad. This operation simultaneously creates both a composite view and a functional linkage through a slot connection.

We summarize our approach for Web application linkages as follows:

1. Create visual wrapper components of Web applications by specifying some portions on the Web pages to work as slots using a mouse.
2. Visually combine wrapped Web applications together to compose a new integrated application that performs a composite function.
3. Decompose an integrated application visually to re-compose another composite application visually.

Fig. 1 shows an outline of our approach to define functional linkage among Web applications. The user specifies an input form of the company's name and the text of the retrieved stock quote to work as slots in current stock quote service[14]. The user specifies an input form of the U.S dollar amount and the text of the converted Japanese yen amount to work as slots in currency conversion page[25]. The user connects the slot of the current stock quote in US dollar to the US dollar slot of the currency conversion service through a drag-and-drop and paste operations. The two services are combined to compose a single application. Users can also decompose the composition through drag-and-drop and paste operations. Users require no programming expertise to wrap, to combine Web applications and also to decompose composite ones.



A Web application The wrapped Web application

Fig. 2. A Web application and its user-defined wrapper.

IV. Web Application Wrapping

Web application has HTML-based front-end for the user. This front-end is defined in HTML format. Using the API of a legacy Web browser such as Netscape Navigator or Internet Explorer, we can access HTML-elements within Web Applications. Such legacy browsers parse an HTML document to create a DOM (Document Object Model) [24] of given URL for the rendering of the document view on the display. In this process, the browser converts ill-formed HTML into well-formed HTML in appropriate ways. A DOM forms a tree structure, in which each node corresponds to an HTML element. The DOM defines tag-specific interfaces for many HTML elements.

To create a wrapped visual component of a Web application, users may just open the Web page they want to wrap using a WebApplicationWrapperPad, which is a kind of Web browser. The user can specify an HTML-node or a text string to work as a *slot* on this pad.

Fig. 2 shows an abstract architecture of a WebApplicationWrapperPad. Its rendering and parsing functions are implemented by wrapping Internet Explorer's API [17]. A WebApplication-WrapperPad has the #url slot and user-defined HTML-node slots named with HTML-paths as its connection interface. The view of this pad shows the HTML-page at the URL address specified by the value of the #url slot. The value of an HTML-node slot named with an HTML-path *p* is the text value of the HTML-node at *p*.

A. HTML-Paths

Fig. 3 shows an HTML document with its DOM tree representation of a Web application. To identify a user-specified HTML element, we use an HTML-path.

An *HTML-path* is a concatenation of node identifiers along a path from the root to the specified element. Each element identifier consists of a tag name and an index *i*, where this node is the *i*th sibling that has the same tag name. We define the grammar of HTML-paths as follows:

HTML-path ::= *tagname*[*i*]

| *HTML-path*/*tagname*[*i*]

The HTML-path expression is the specialization of XPath expression [23]. For example, the circled portion in the document in Fig. 3 corresponds to the circled node whose HTML-path is

HTML[1]/BODY[1]/FORM[1]/INPUT[1]

This is the HTML-path of an input element of this Web application. To access the detail of an HTML element, we add the following two extensions to the path-expression.

- *HTML-path*/attr[*attribute-name*]
- *HTML-path*/text[*regular-expression*]

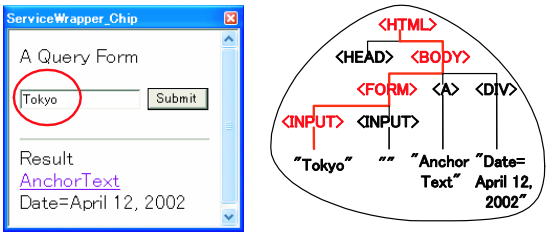
The function attr[*attribute-name*] selects an attribute value named with the specified name. We can set and get the value of the attribute. For example, consider the DOM tree in Fig. 3.

The value pointed by

"HTML[1]/BODY[1]/A[1]/attr[href]"

is the string "http://ca.meme.hokudai.ac.jp".

The function text[*regular-expression*] captures substring of inner text. This regular expression must contains just one pair of parentheses "(" and ")"'. This function allows us to extract and to edit the text within



Document View **Document Object Model**

```
<HTML>
 <HEAD><TITLE>Sample HTML</TITLE></HEAD>
 <BODY>
   A Query Form<BR>
   <FORM method="GET" action=./sample.cgi>
    <INPUT value="Tokyo" name="query">
    <INPUT type="submit">
   </FORM>
   <HR>Result<BR>
   <A href="http://ca.meme.hokudai.ac.jp">
    AnchorText
   </A>
   <DIV>Date=April 12, 2002</DIV>
 </BODY>
</HTML>
```

Fig. 3. An HTML document and its DOM tree.

an HTML element flexibly.

Let e be an HTML element obtained by HTML-path p. And let r_{left}, $r_{capture}$ and r_{right} be regular expressions. Then the text pointed by the following expression: p/text[r_{left} ($r_{capture}$)r_{right}] is a substring s of the inner text of the element e, where s matches $r_{capture}$, s is followed by a string that matches r_{left} and string that matches r_{right} is preceded by s. For example, the text pointed by

"HTML[1]/BODY[1]/DIV[1]/text[Date=(.*),2002]"

is the text "April 12".

B. User Interface for Wrapping

To hide HTML-path from users, the system calculates HTML-path expressions from users' operations. Users can directly specify any HTML-node as a slot using his mouse.

There are two ways for users to specify an HTML-node to work as a slot:

- specify an HTML element to work as a slot,
- specify a text portion to work as a slot.

If a user clicks with the right button on a region that he wants to use as a slot, a popup menu is shown.

Selecting "HTML as Slot" in the popup menu, the user can use the HTML-element at this location as a slot. Let p be the HTML-path of the element.

1. If the element is a textinput element or textarea element, WebApplicationWrapperPad installs a slot named p/attr[value].
2. If the element is an anchor element, WebApplicationWrapperPad installs a slot named p/attr[href].
3. WebApplicationWrapperPad installs a slot named p, otherwise.

In Fig. 4, the user selects a text input element to use this as a text input slot. WebApplicationWrapperPad installed the slot named

HTML[1]/BODY[1]/FORM[1]/INPUT[1]/attr[value].

Selecting "Text as Slot" in the popup menu, the user can use the selected string as a slot. Let p be the HTML-path of the element e that contains the selected text portion. The WebApplication-WrapperPad installs a slot named with p/text[l (.*)r], where l and r are respectably a preceding substring and a following substring of the selected portion in e. A dialog box then appears to ask the user to confirm the two text strings l and r.

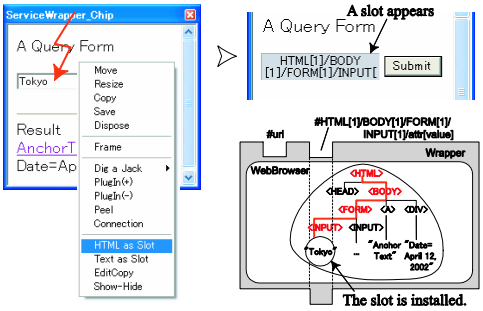


Fig. 4. User interface for slot definition.

C. Wrapping of Single-page Web Applications

A Web application A is called a *single-page Web application*, if every result page returned by A contains an input-form for the next query for A. In other words, we can define slots for both input and output on a single page returned by a single-page Web application. Examples of single-page Web applications include Google Search Page, Lycos' Stocks&News, and Yahoo's Currency Exchange.

Using a WebApplicationWrapperPad, we may wrap a single-page Web application into a pad. Many Web applications return result pages that have exactly the same syntactic and semantic structure for different input queries. If this new page has the same structure to previous page, we can apply new query using the same HTML-path.

A value change of a slot in a WebApplication-WrapperPad causes its view update and possibly the value changes of the other slots.

1. If the value of the #url slot is changed, then the WebApplicationWrapperPad downloads the page at the URL address to updates the DOM representation.
2. If the user clicks an anchor or submits a form, then the wrapper pad downloads the target page to updates the DOM representation.
3. If the value of an HTML-node slot is changed, then the WebApplicationWrapperPad changes the value of each node specified by the corresponding HTML-path.
   - If the HTML-node slot is an input element contained in a form, the wrapper pad submits the form to the appropriate Web server to updates the DOM representation.

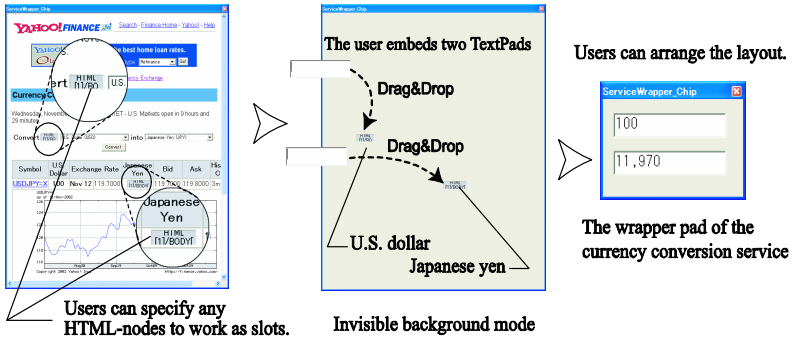


Fig. 5. Visual definition of a wrapped Web application through direct manipulation.

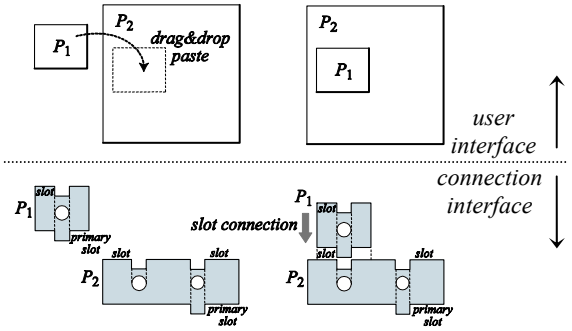


Fig. 6. User interface and slot-connection interface. If a user pastes P_1 on a slot of P_2, the primary slot of P_1 is connected to this slot of P_2.

4. Whenever the WebApplicationWrapperPad updates the DOM representation, it renders a new view. The values of all the HTML-node slots will be changed to the values of the corresponding nodes in the new DOM tree.

In Fig. 5, a user creates a wrapper pad that wraps a currency conversion Web application. Firstly, the user may open the target page. Secondly, using his mouse, he may specify regions that he wants to work as slots. Then the slots will appear as sunken shaded regions on this page. The user may keep the background Web page either visible or invisible. Thirdly, the user may embed other pads, such as TextPads, into these slots. Finally, the user may arbitrarily relocate and resize the embedded pads to design their layout.

V. Functional Linkage Among Wrapped Web Applications

In this section, we present how to visually create functional linkage among Web Applications and local legacy tools. We apply the IntelligentPad architecture to this problem.

A. IntelligentPadArchitecture

IntelligentPad [21] represents each component as a pad, a sheet of paper on the screen. A pad can be pasted on another pad to define both a physical containment relationship and a functional linkage between them. When a pad P_1 is pasted on another pad P_2, the pad P_1 becomes a child of P_2, and P_2 becomes the parent of P_1. No pad may have more than one parent pad. Pads can be pasted together to define various multimedia documents and application tools. Unless otherwise specified, composite pads are always decomposable and re-editable.

Each pad has both a standard user interface and a standard connection interface. The user interface of a pad has a card like view on the screen and a standard set of operations like `move', `resize', `copy', `paste', and `peel'. Users can easily replicate any pad, paste a pad onto another, and peel a pad off a composite pad. Pads are decomposable persistent objects. You can easily decompose any composite pad by simply peeling off the primitive or composite pad from its parent pad. As its connection interface, each pad provides a list of slots that work as connection jacks of an AV-system component, and a single connection to a slot of its parent pad (Fig.6).

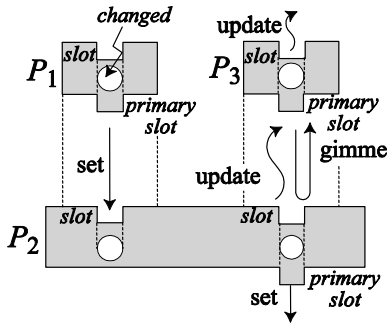


Fig. 7. Three standard messages, 'set', 'gimme' and 'update' between pads.

To set up data connection between pads, IntelligentPad uses three standard messages, set, gimme and update. We show an outline of these three messages in Table 1.

TABLE I
A Summary of Three Standard Messages.

Message	Summary
Set slotname value	a child sets the specified value to its parent's slot
Gimme slotname	a child requests its parent to return the value of its specified slot
update	a parent notifies its children that some slot value has been changed

Each pad is embedded in one parent at most with its connection to one of the parent pad slots. Connected pads form a tree structure. We do not restrict the maximum depth of the tree.

Each pad has one primary slot. When the value of the primary slot of a child is changed, the child sends a set message with the new slot value to its parents. Using this value, the parent changes its own slot values. Then, the parent notifies all of its children of its state change by sending an update message. Each child that has received an update message sends a gimme message to the parent pad, changes its own slot values using the return value of this gimme message, and then sends an update message to each of its children. Using this mechanism, state changes are propagated from one pad to all the pads connected to it through slots(Fig. 7).

B. Functional Linkage among Wrapped Web Applications through View Integration

A pad that wraps a Web application provides slots for some of the original's input forms and output text

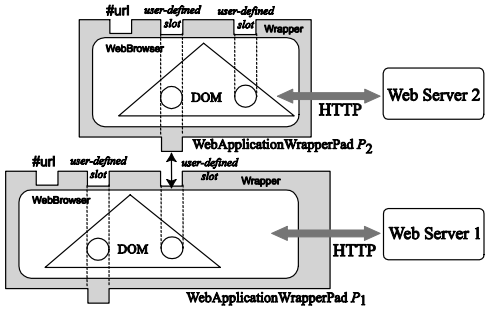


Fig. 8. Functional linkage among different Web applications.

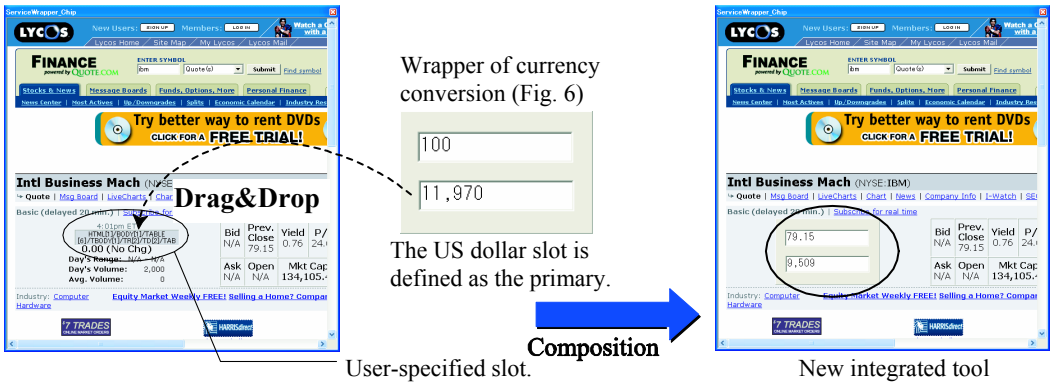


Fig. 9. Dynamic linkage definition between visually wrapped Web applications.

strings. Since wrapped Web applications are pads, you may combine wrapped Web applications together through drag&drop and paste operations. Through data linkage between slots specified by a user, these wrapped Web applications cooperate with each other (Fig. 8).

Fig. 9 shows a WebApplicationWrapperPad showing a Lycos' Web page for a real-time stock quote browsing service. We have wrapped this page, defining a slot for the current stock quote. Then we paste the wrapped currency conversion Web application with its dollar amount slot specified as its primary slot on this wrapped Lycos stock quote page. We connect the conversion pad to the newly defined current stock quote slot. The right hand side in this figure shows a composite pad combining these two wrapped Web applications. For the input of different company names, we use the input form of the original Web page. Since this Web application uses the same page layout for different companies, the same path expression correctly identifies the current stock quote information part for each different company. The current stock quote in US dollar is converted to Japanese yen by the wrapped currency conversion Web application.

VI. Application to Mobile Computing

Our framework is suitable not only for desktop computing but also for mobile computing.

In mobile computing, the surroundings around a user changes dynamically. Different situations require different information service. Even if the user could connect his/her mobile device to the Internet, the user may not find the necessary service. There may not be such a service on the Web. Using our framework, the user may compose a new tool for his/her present problem by wrapping existing Web applications and linking them together on his/her mobile device as long as he or she find a set of appropriate services that can collaboratively solve the problem.

Suppose that a user has a mobile device with wireless LAN unit. And suppose the city that he or she lives in has developed wireless LAN infrastructure which provides not only internet connection service but also Web-based localized information services based on the user's access point [11]. From the localized information service, the user could get the list of restaurants which locate close to him or her. The user knows a large-scale online restaurant guide that provides fair evaluation of restaurant. The user uses this guide so often that he or

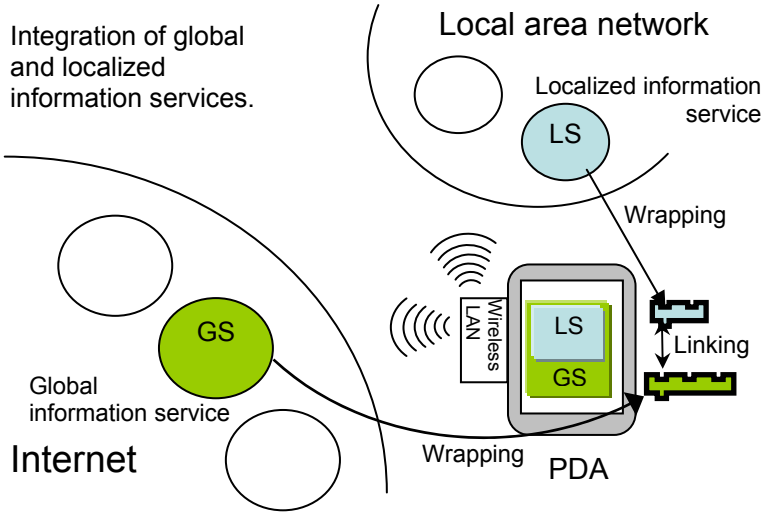


Fig.10. An application of our framework to mobile computing.

she has wrapped this guide into a pad by defining a slot for a restaurant name. The only things the user needs to do are to wrap the Web-based localized restaurant-information service, and to connect its restaurant name to the slot in the wrapped restaurant guide. The result is a new tool that accesses both the localized information service and the global information service (Fig. 10). No programming expertise is required to wrap the localized information service or to functionally combine these two wrapped services together.

VII. CONCLUSION

We have proposed a new framework for visually and dynamically defining functional linkages among Web applications to compose a single application tool. This framework is based on the IntelligentPad architecture. Users can visually and dynamically wrap Web applications into visual components, and visually combine them together to define functional linkages among them. Users can also visually define functional linkages among wrapped Web applications and such local tools in pad forms as chart drawing tools and spreadsheet tools to compose a single integrated tool. We also have discussed an application of the framework to mobile computing.

REFERENCES

[1] V. Anupam, J. Freire, B. Kumar, and D. F. Lieuwen: Automating web navigation with the webvcr. *WWW9 /Computer Networks* 33(1-6) pages 503–517, 2000.

[2] M. Bauer and D. Dengler: InfoBeans - Configuration of Personalized Information Services, In *Proc. of IUI99*, pages 153-156.

[3] M. Bauer, D. Dengler, and G. Paul: Instructible Agents for Web Mining, In *Proc. of IUI2000*, pages 21-28.

[4] M. Birbeck and etc.: *Professional XML*. Wrox Press Ltd., 2000.

[5] L. A. Carr, D. D. Roure, W. Hall, and G. Hill: Implementing an Open Link Service for the World Wide Web. *World Wide Web* 1(2), pages 61–71, 1998.

[6] J. Freire, B. Kumar, and D. F. Lieuwen: Webviews: accessing personalized web content and services. In *Proc. of WWW2001*, pages 576–586, 2001.

[7] P. B. Golgher, A. H. F. Laender, A. S. da Silva, and B. A. Ribeiro-Neto: An Example-Based Environment for Wrapper Generation. In *Proc. of ER Workshops 2000*, pages 152–164, 2000.

[8] G. Grieser, K. P. Jantke, S. Lange, and B. Thomas: A Unifying Approach to HTML Wrapper Representation and Learning. In *Proc. of Discovery Science 2000*, pages 50–64, 2000.

[9] K. Ito: CHIP(Collaborating Host-Independent Pads). http://ca.meme.hokudai.ac.jp/people/itok/CHIP.

[10] K. Ito and Y. Tanaka: Visual wrapping and composition of web applications for their interoperations, *Poster Tracks of WWW2002,* #64, 2002.

[11] R. Jana, T. Johnson, S. Muthukrishnan, A. Vitaletti: Location based services in a wireless WAN using cellular digital packet data (CDPD) In *Proc. of Data Engineering for Wireless and Mobile Access*, 2001

[12] T. Kistler, H. Marais. WebL - A Programming Language for the Web. *WWW7 / Computer* Networks 30(1-7) pages 259-270 1998.

[13] N. Kushmerick: Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence* 118(1-2) pages 15-68 2000.

[14] Lycos,Inc,Stocks&news. http://finance.lycos.com

[15] K. C. Malcolm, S. E. Poltrock, and D. Schuler: Industrial strength hypermedia: Requirements for a large engineering enterprise. In *Proc. of Hypertext1991*, pages 13–24, 1991.

[16] T. Nelson: Transcopyright, Project Xanadu, http://xanadu.com/tco/

[17] Microsoft: MSHTML Reference. MSDN Library.

[18] T.A. Phelps and R. Wilensky: Robust intra-document locations. *WWW9 / Computer Networks* 33(1-6) pages 105-118, 2000.

[19] A. Sahuguet and F. Azavant. Building intelligent Web applications using lightweight wrappers. *Data & Knowledge Engineering* 36(3):283–316, 2001.

[20] Y. Tanaka: From augmentation media to meme media: IntelligentPad and the world-wide repository of pads. *Information Modelling and Knowledge Bases* 6, pages 91–107, 1995.

[21] Y. Tanaka and T. Imataki: Intelligentpad: A Hypermedia System Allowing Functional Compositions of Active Media Objects through Direct Manipulations. In *Proc. of IFIP'89*, pages 541–546, 1989.

[22] Y. Tanaka, A. Nagasaki, M. Akaishi, and T. Noguchi: A Synthetic Media Architecture for an Object-Oriented Open Platform. In *Proc.IFIP Congress* 3, pages 104–110, 1992.

[23] The World Wide Web Consortium: XML path language(XPath) version 1.0, November 1999. http://www.w3.org/TR/xpath.

[24] The World Wide Web Consortium: Document object model (DOM) level 2 HTML specification version 1.0, http://www.w3.org/TR/2001/

[25] Yahoo Corporation. Currency conversion. http://quote.yahoo.com/m3?u.