Hierarchical frame and frameset components for visual programming of web applications

Quan Liang Chen and Takao Shimomura

***Abstract*—**By using the HTML frame facility, Web applications can display a lot of information on one Web page at a time and show it to users. This paper presents a methodology that makes it possible to visually design and program Web applications that use frame facilities. The BioPro system that implements this methodology newly provides *Frameset page design windows* with which we can design Web pages that represent a frameset consisting of several frames and other framesets. We design the contents of ordinary Web pages using *Web page design windows*. Each frame of a *Frameset page design window* has a link to the Web page that is displayed in the frame. The system generates Web application program code from these *Frameset page design windows* and the *Web page design windows*. *Frameset page design windows* provide several editing facilities such as adding, deleting, changing, and nesting of framesets to make it easier to develop Web applications that use frame facilities.

***Keywords*—**Automatic generation, html frames, html framesets, web application, web components.

I. Introduction

Web applications that need to show a lot of information on one Web page at a time often use some inline frames, exchange displayed elements using their tabs, or enable users to see a hidden part of the page using the scroll bars of a Web browser. On the other hand, if Web applications use a HTML frame facility, they can display a lot of information on one Web page at a time and show it to users [11]. The users can easily grasp the outline of the provided information, and at the same time, they can see the contents of the frame of interest in detail by extending the frame to the whole page if they need to. Therefore, some Web applications that need to display a lot of information at a time such as computer-assisted instruction systems, Web-based chat systems, and the Help windows of various kinds of Web applications often use the HTML frame facility [4], [20].

Recently, CSS (Cascading Style Sheets) is popular and used to design Web pages [21]. Without using frames, we can easily create Web pages that contain multiple columns. CSS is suitable for designing Web pages based on a grid system. In the grid system, the whole area of a Web page is divided into grids, and a space that consists of several continuous grids is used as a column. Because the width of each column is fixed in CSS, if the lengths of the columns are different, it will be difficult to see the contents of all the columns at the same time. For example, MIT OpenCourseWare project [9] provides free courseware including curriculums of over 1400 subjects from faculties and graduate schools, where a Web page consists of a side bar column and a main column. The side bar column displays several menu items each of which is linked to a part of the contents of the main column including course objectives, measurable outcomes, assessment methods, examination schedules, meetings, recitations, course materials, assignments, and other resources. If we scroll down the main column, the menu items in the side bar column cannot bee seen.

For the development of Web applications, various kinds of Web application frameworks [2], [3], [7], [10], [12] and integrated development environments [19] have been proposed and utilized. Among them, visual programming for Web applications has been attracting programmers' attention as a tool that makes it easier to design and debug Web applications [6], [13], [16], [17]. Most existing development environments assist programmers to design the <body> parts of HTML documents for Web applications, which mainly receive form data from Web browsers, process database tables using those data, and display the results in Web pages. However, in the existing development environments, it is difficult to visually represent the design of Web pages that use frame facilities because <frame> elements are not written in the <body> parts of HTML documents.

We have developed the BioPro (Brain-Image Oriented Programming) system that enables programmers to visually design the contents of Web pages, database tables, program tables, actions, and Web page transfers [16]. This paper presents a methodology that makes it possible to visually design and program Web applications that use frame facilities. The enhanced BioPro system that implements this methodology newly provides *Frameset page design windows* with which we can design Web pages that represent a frameset consisting of several frames and other framesets [15]. We design the contents of ordinary Web pages using *Web page design windows*. Each frame of a *Frameset page design window* has a link to the Web page that is displayed in the frame. The system generates Web application program code from these *Frameset page design windows* and the *Web page design windows*. *Frameset page design windows* provide several editing facilities such as adding, deleting, changing,

Manuscript received January 19, 2007; Revised received July 24, 2007.
Quan Liang Chen is a doctoral course student at the University of Tokushima, Japan.

Takao Shimomura is with the University of Tokushima, Tokushima 770-8506 Japan (phone: +81-88-656-7503; fax: +81-88-656-7503; e-mail: simomura@is.tokushima-u.ac.jp).

and nesting of framesets to make it easier to develop Web applications that use frame facilities.

II. Requirements for frame design

A. Requirements

Web pages can contain framesets, and framesets contain frames. Moreover, the Web page that is displayed in a frame can also contain framesets. Because this relationship ranges among multiple pages, it is difficult to grasp with the conventional development environments. To make it easier to develop Web applications that use frame facilities, we take into account the following requirements:

(1) We can nest frames in a frameset any number of times, and we can create a frame pointing to another Web page that contains framesets.

(2) We can easily understand the relationship between the frames and the Web pages that are displayed inside of the former.

(3) We can easily change the structure of framesets, easily add and delete frames, and easily change the Web pages that are displayed inside frames by using the mouse dragging.

B. Visual programming for Web applications

The BioPro system is a visual programming environment for Web applications based on the model-view-controller architecture [8] as show in Fig. 1. The design of Web pages (View) and DB tables (Model) and the definition of actions (Control) are separated from each other. We first describe a procedure of development for Web applications using the BioPro system. Let's consider a simple Web application fruitShop as an example. In this Web application, we first enter our name in the entryShop page and then click the "Enter" button. The next fruitShop page will show the images and prices of several kinds of fruits. When we click the "Buy" button to select one fruit, the checkOut page shows the name and price of the selected fruit.

To design this Web application, we first design the contents of these Web pages by choosing appropriate Web components (forms, text fields, buttons, etc.) from menus and pasting them in each of the *Web page design windows*. We next create a *DB table design window* fruit to create a real DB table fruit, which stores several kinds of fruits to be displayed in the fruitShop page. We drag the name, image, and price fields of *DB table design window* fruit and drop them in the fruitShop *Web page design window* so that the fruitShop page will display these fruits. We also drag the name text field of the entryShop *Web page design window* and drop it in the checkOut *Web page design window* so that the checkOut page will display the name that is entered in the entryShop page. We finally choose the target Web page to which control transfers when we click each of submit buttons in these *Web page design windows* to complete the development of this Web application.

The terms this paper introduces are as follows and written in italic:

Web page design window

a BioPro window (Web page for short) to design the contents of a Web page

Frameset page design window

a BioPro window (Frameset page for short) to design a frameset that include frames

Web source window

a BioPro window to write actions executed when the corresponding Web page is accessed

DB table design window

a BioPro window (DB table for short) to create, edit, store a real DB table

Program table design window

a BioPro window (Program table for short) to design a table, which is stored in a session during the execution of a Web application

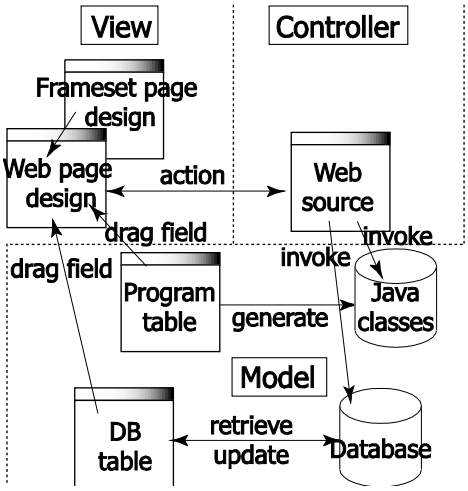


Fig. 1 Model-view-controller architecture for Web applications

III. Frameset Features/Functions

A. Introduction of Frameset page design windows

Requirement 1: We introduce *Frameset page design windows* as root Web pages that represent framesets, and link each frame included in a *Frameset page design window* to an ordinary *Web page design window*. For example, in Fig. 2(b), *Frameset page design window* Y has two frames, and its upper frame is linked to *Web page design window* C, and its lower frame is linked to *Web page design window* D. We make it possible to link each frame included in a *Frameset page design window* to not only an ordinary *Web page design window* but also another *Frameset page design window*. This enables frames to be nested in a frameset any number of times.

Requirement 2: Fig. 3(a) shows two *Frameset page design windows* and four *Web page design windows* created by a programmer using the BioPro system, as described in Fig. 2. Each frame of the *Frameset page design window* is linked with a pink line to the *Web page design window* that is displayed in

it. Fig. 3(b) shows an example of more complicated framesets.

Requirement 3: As shown in Fig. 2(a), the *Frameset page design window* X has two frames, where its upper frame is divided into two other frames, left and right frames, each of which is linked to *Web page design windows* A and B, respectively. Its lower frame is linked to another *Frameset page design window* Y. When we execute the Web application that contains these pages, the *Frameset page design window* X is displayed as a root Web page that includes four Web pages A, B, C and D as shown in Fig. 2(c). In a *Frameset page design window*, we can easily divide, delete, exchange frames by clicking or drag-and-dropping the mouse.

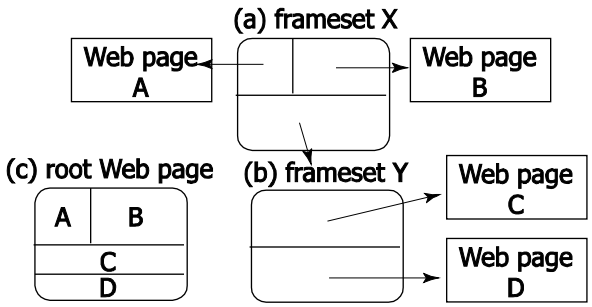
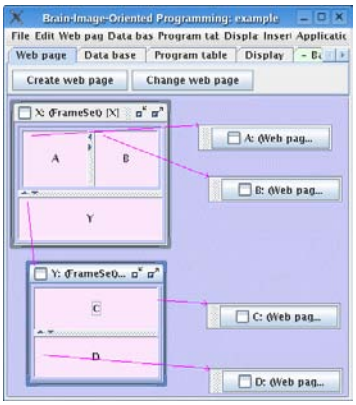


Fig. 2 Page design for framesets

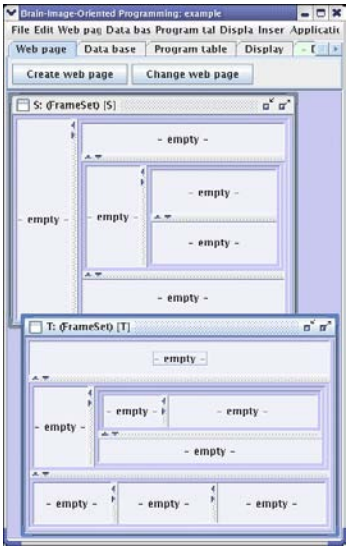
B. Visual programming for Web applications using framesets

As an example of a Web application that uses frames, let's consider a simple Web-based chat system. As shown in Fig. 4, this Web page consists of two frames. The upper frame has a text field to enter a name, and buttons to enter and exit a chat room. Below them, it has an area to display the contents of chatting and its status. The lower frame has a text field to enter a chat message, and a button to send the message. To update the contents of chatting that vary any minute, the upper frame keeps being refreshed at a certain period of time. Because this Web page is divided into these two frames, the lower frame is not affected by refreshing the upper frame even when a user is entering a message.

Fig. 5 shows an example of visual programming of this Web-based chat system. The window in the top left corner of Fig. 5(a) is a *Frameset page*, which corresponds to the *Frameset page* Y in Fig. 2(b). The upper frame is linked to chatFrame *Web page* to display chatFrame Web page. The lower frame is linked to utterFrame *Web page* to display utterFrame Web page. Fig. 5(b) and Fig. 5(c) show the *Web pages* of chatFrame and chatFrame Web pages, respectively. In the *Web source window* of chatFrame *Web page*, we can refer to variables user, enter, exit, and textarea as predefined variables, which are generated from the visual design of the Web pages by the system. For example, variable user has the value of the name text field. We do not need to care about the inconsistency between the parameter names of a sender and a receiver. Using these predefined variables, we write necessary actions for the processes of entering and exiting the room. In the *Web source window* of utterFrame *Web page*, we write the process for adding the submitted message by referring to the contents of the message as a predefined variable.



(a) framesets X and Y



(b) framesets S and T

Fig. 3 *Frameset page design windows* created by the BioPro system

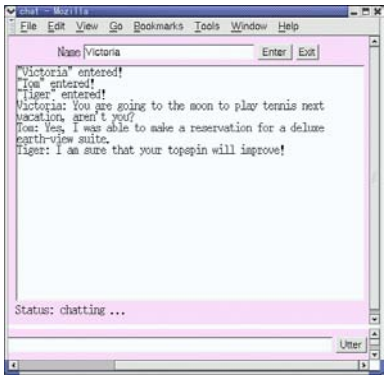


Fig. 4 Example of Web-based chat system that uses frames

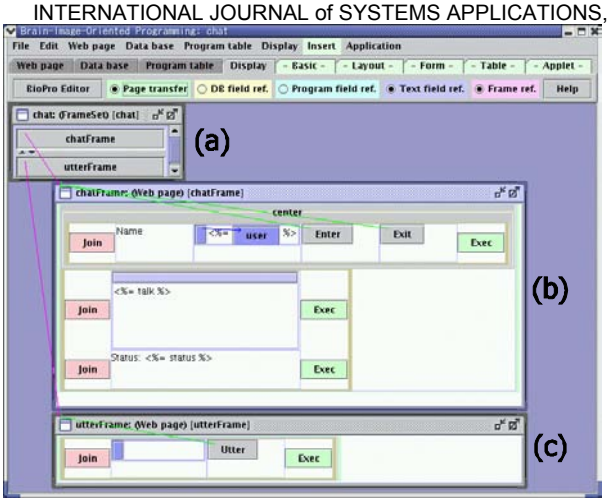


Fig. 5 Visual programming using *frameset pages*

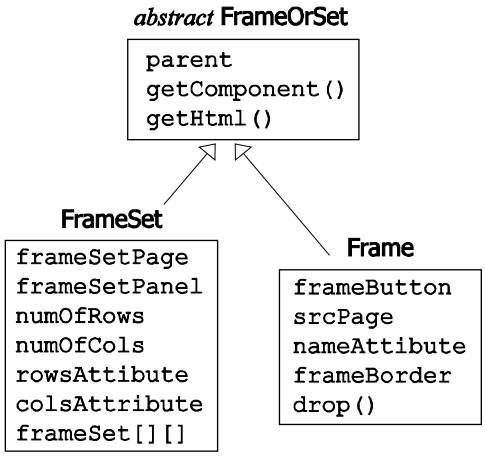


Fig. 6 FrameSet and Frame that extend abstract FrameOrSet

IV. Implementation

A.Implementation of frameset hierarchy

To develop *Frameset page design windows*, we define FrameSet class and Frame class. As shown in Fig. 6, both of FrameSet class and Frame class extend abstract class FrameOrSet. In FrameOrSet class, variable parent refers to its parent frameset or frame. GetComponent() method returns JComponent that displays its frameset or frame. A frameset is displayed as JPanel that contains a JSplitPane, and a frame is displayed as a button in a *Frameset page design window*. GetHtml() method returns the HTML code that represents its frameset or frame. In FrameSet class, frameset[][] array points to its child frameset or frame. In Frame class, variable srcPage points to a *Web page* or a *Frameset page*. Drop() method enables a user to drag and drop a frame to another frame to exchange them.

Fig. 7 illustrates the frameset hierarchy of the two *Frameset pages* shown in Fig. 3(a), where Frame and FrameSet instances also have a pointer that refers to their parent frame or frameset, because they extend the abstract FrameOrSet class shown in Fig. 6.

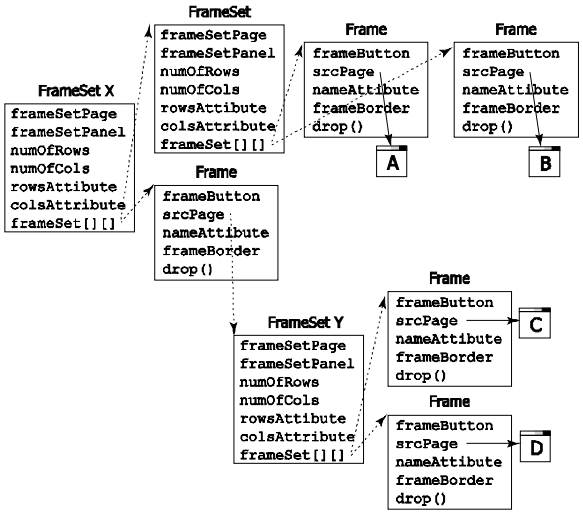


Fig. 7 Frameset hierarchy

B.Actions defined in Web source windows

In the *Web source window* that corresponds to chatFrame *Web page*, we can here define actions to be executed when control transfers to this page. The "Predefined vars:" column of the *Web source window* shows some variables that contain submitted data and these are automatically generated by the system. Fig. 8 shows the *Web source window* that corresponds to utterFrame *Web page*. Although a typical page transition (a default transition) is specified as an arc from one *Web page* to another *Web page*, the other transitions (as when a failure occurs) can be specified in this *Web source window* as an action. The actions defined here will be synthesized with designed Web components to generate the program code of the Web application (See Fig. 10).

C.Automatic generation of variables for submitted data

The BioPro system provides the page-transition framework for Web applications, where submitted data can be automatically received and analyzed, and then the variables that contain received data are automatically generated. In a *Web source window*, programs can use these generated variables as predefined variables. In the BioPro system, Web pages are composed of a variety of Web components such as text fields, textareas, field references for *Program tables* and *DB tables*, submit buttons, and pivot tables. When a request is submitted from a source page A to a target page B, some data are submitted from Web components included in source page A. Target page B automatically receives these submitted data, analyzes the contents of the data, transforms them into appropriate values, and generates some variables to store those values. For example, a text-field Web component whose name is "user" submits the text that is input in this field, and the target page receives this submitted text, generates a variable whose name is "user", and stores the text in this variable.

Fig. 9 illustrates how predefined variables are generated for chatFrame and utterFrame *Web source windows*. When we click on "enter" or "exit" button in chatFrame Web page, a request is submitted to chatFrame Web page itself. Therefore, variables enter, exit, txtArea, user are generated as predefined variables in chatFrame *Web source window*. When we click on "utter" button in utterFrame Web page, a request is submitted to utterFrame Web page itself. Therefore, variable utter for the text field whose name attribute is "utter" is generated as a predefined variable in utterFrame *Web source window* as shown in "Predefined vars:" column of Fig. 8. Although utterFrame Web page contains "utter" button, no variable for that button is generated because this button's name attribute has not been specified in this Web application.

From a *Web page*, code for displaying the corresponding Web page will be automatically generated. If it refers to some fields of a *DB table*, code for retrieving records from the corresponding DB table will be automatically generated. Submitted data can be automatically received and analyzed, and then the variables that contain received data will be automatically generated. The other logic such as updating a DB table needs to be written manually as an action in a *Web source window*.

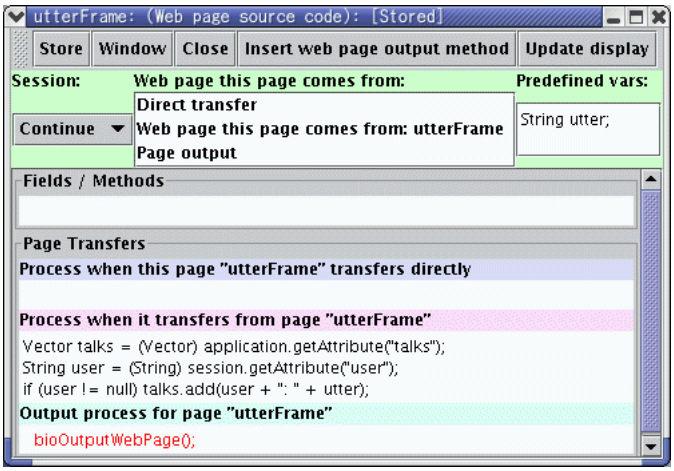


Fig. 8 *Web source window* for utterFrame page

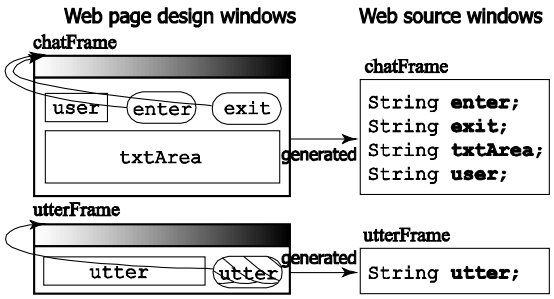


Fig. 9 Predefined variables for chatFrame and utterFrame

D.Framework for generating Web applications

The BioPro system is not based on any of the existing frameworks. It proposes a visual programming framework, where a generated application only uses Servlets and JSPs. To connect to a Postgresql database server, the generated application uses JDBC. The system generates Web application program code from the visual design of a Web application. As shown in Fig. 1, we first visually design the contents of each of Web pages in their *Web page design windows*. When we use database tables, we first visually design those database tables in *DB table design windows*. Instead, we may specify the names of existing database tables to display them in *DB table design windows*. We drag and drop the fields of a database table from a *DB table design window* to a *Web page design window* so that these fields will be displayed in the corresponding Web page. When we use a table in the program that exists only during the execution of the program, we visually design the contents of this table in a *Program table design window*. For example, we design a table of an online-shop cart in this *Program table design window*. The system automatically generates JavaBeans code *Web page*, for each Web page control transfers from, we can write a necessary action, which is executed when control transfers from the Web page to this Web page. From these resources, the BioPro system automatically generates Servlets, JSP pages [18], and Java classes that compose a Web application in a client side, and uploads them to the Web server together with other resources such as image files, and customized Java classes. To start the Web application, the system then runs a Web browser to make it send a request to the entry Web page of the Web application.

The system generates code for connecting a database server, retrieving records, receiving submitted data, and displaying Web components designed in a *Web page*. These processes will be run as threads. After these threads complete, it will start a Web browser to access the entry Web page of the Web application. The entry Web page is automatically determined as a Web page that does not have its preceding page.

Fig. 10 shows a method to automatically generate the program code of the Web-based chat system from the visual design created in Section 3B. The part of the program code printed in italics in Fig. 10 represents the code that was automatically generated by the BioPro system. We first designed a Frameset page chat in the *Frameset page design window*, and designed two Web pages that are displayed inside its frames in the *Web page design windows*. In the *Web source windows* of the corresponding Web pages, we then wrote the necessary actions that would be executed when control transfers to each of the Web pages by referring to predefined variables (for example, "utter" as shown in Fig. 8) that were automatically generated by the BioPro system.

The BioPro system generates the program code that composes the Web application from these pieces of design information. It generates a JSP page "chat.jsp" from the *Frameset page* chat, and as shown in Fig. 10, it generates a JSP page "utterFrame.jsp" from these *Program tables*. We next write the actions that are executed when control transfers to a Web page in the *Web source window* that corresponds to the Web page. Control may transfer to one Web page from multiple Web pages. In the *Web source window* of a by

synthesizing the utterFrame *Web page* and its *Web source*.

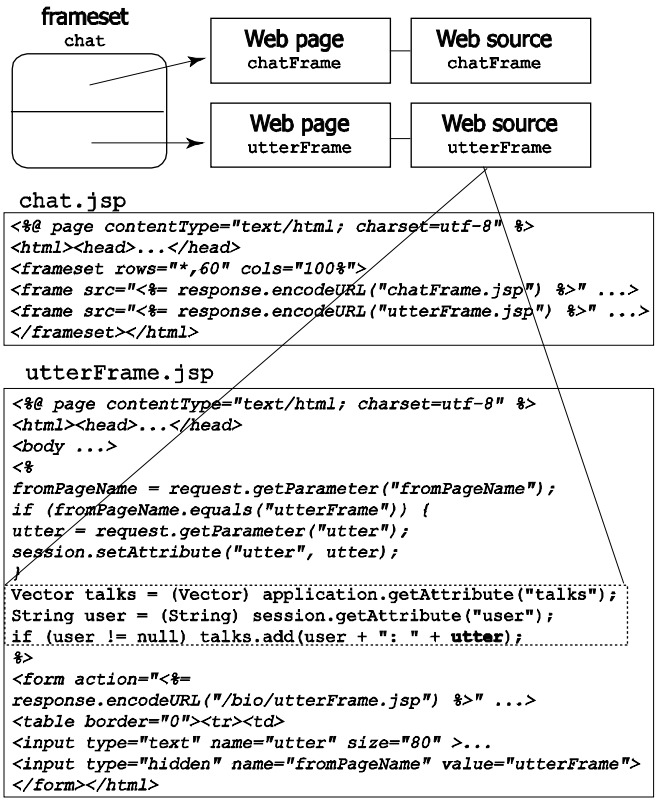


Fig. 10 Automatic generation of Web application program code

V. Evaluation

Using visual programming, we were able to greatly reduce the code that was required to develop Web applications. In the Web-based chat system which is a typical example of conventional development, it required 272 lines of code (10 for chat.jsp, 178 for chatFrame.jsp, and 84 for utterFrame.jsp). This method reduced it to 63 lines of code, which is about one-forth. Talking about the efficiency, it only took about fifteen minutes to design the chat Frameset, chatFrame Web page, and utterFrame Web page, which are shown in Fig. 5. The nisWeb application is another program that uses frame facilities. Since this is a quite large program [14], we evaluated only the code of its home page. The larger applications become, the less remarkable the effectiveness of the visual programming might be. It is because program logic will occupy a large part of the program.

As shown in Fig. 5, the structure of the Web application is displayed visually, and this makes it easier to understand the program and efficient to modify and debug the program. When we design framesets and the frames that the framesets contain, we can easily create them, and divide, delete, exchange frames by clicking or drag-and-dropping the mouse. In addition, we can create another frameset in a different *Frameset page design window*, and make the parent frame refer to the created child frameset. This simplifies the design of framesets and the frames that the framesets contain. We can also easily see the hierarchy of the frames in the *Frameset page design windows*, and see the contents of the frames in the *Web page design windows* that are pointed at from the corresponding frames in the *Frameset page design windows*.

Web source windows show some automatically generated variables in their predefined variable column. These automatically generated variables that have the values of the query data submitted to the server enable developers to easily write actions in the fields/methods columns and the page transfers columns of the *Web source windows*. This avoids a mismatch problem between variable names that are written in the form tags of JSP pages and in the methods of Servlets.

VI. Conclusion

With frames, we can scroll each frame independently and extend a frame by moving its border. Moreover, from the point of view of Web applications, each frame can be updated independently. This paper has presented a methodology that makes it possible to visually design and program Web applications that use frame facilities. Existing tools such as Homepage Builder [5], and Dreamweaver & Fireworks [1] provide a variety of GUI components and have sufficient facilities for editing Web pages. On the other hand, this system has facilitated the development of Web applications by providing Web components such as *Frameset pages*, *Web pages*, *DB tables*, *Program tables*, and *Web source windows*, where each Web component can easily refer to the definitions of the other components. As the next step, we are going to investigate how to visually incorporate rich components such as Flex, Flash, JavaScript, and Applets in the design phase.

References

[1] Adobe Systems Incorporated., Adobe Dreamweaver and Fireworks, http://www.adobe.com/products/dreamweaver/ (2007).

[2] The Apache Software Foundation, Struts, http://jakarta.apache.org/struts/ (2004).

[3] Christensen, A.S., Moller, A. and Schwartzbach, M.I., "Extending Java for High-Level Web Service Construction," *ACM TOPLAS*, Vol.25, No. 6, pp.814-875 (2003).

[4] .Comber, T. and Maltby, J., "Layout complexity: does it measure usability?," *Proc. of Interact '97 Conference on Human-Computer Interaction*, pp.623-626 (1997).

[5] IBM, WebSphere Studio Homepage Builder, http://www-306.ibm.com/software/awdtools/hpbuilder/ (2007).

[6] Iron Speed, Inc., Iron Speed Designer, http://www.ironspeed.com/ (2006).

[7] Lee, S.C. and Shirani, A.I., "A component based methodology for Web application development," *Journal of Systems and Software*, Vol.71, No.1-2, pp. 177-187 (2004).

[8] Leff, A. and Rayfield, J., "Web-application development using the model/view/controller design pattern," *Fifth International Enterprise Distributed Object Computing Conference*, pp. 118-127 (2001).

[9] MIT OpenCourseWare project, http://web.mit.edu/ocw/ (2007).

[10] Metsker, S.J. and Wake, W.C., "Design Patterns in Java," Addison-Wesley (2006).

[11] Pemberton, S. et al., "XHTML 1.0 The extensible hypertext markup language (Second Edition)," http://www.w3.org/TR/xhtml1/ (2002).

[12] Repenning, A., Ioannidou, A., Payton, M., Ye, W. and Roschelle, J., "Using components for rapid distributed software development," *IEEE Software*, Vol.18, No.2, pp. 38-45 (2001).

[13] Shimomura, T. and Isoda, S., "Linked-List Visualization for Debugger," *IEEE Software*, Vol.8, No.3, pp. 44-51 (1991).

[14] Shimomura, T., Chen, Q.L., Lang, N.S. and Ikeda, K., "Integrated Laboratory Network Management System," *Proc. of 7th International Conference on APPLIED INFORMATICS AND COMMUNICATIONS*, pp. 188-193 (2007).

[15] Shimomura, T., Ikeda, K., Quan, C.L., Nhor, L.S. and Muneo, T., "Visual Programming for Web Applications that Use HTML Frame Facilities," *Proc. of WSEAS International Conference on Computer Engineering and Applications*, pp. 384-389 (2007).

[16] Shimomura, T., Muneo, T., Ikeda, K. and Mogami, Y., "Visual Design for Server-Side Programs and Program Generation," *Transactions of Information Processing Society of Japan*, Vol.45, No.7, pp. 1737-1744 (2004).

[17] Sun Microsystems, Inc., Sun Java Studio Creator, http://wwws.sun.com/software/products/jscreator/ (2004).

[18] Sun Microsystems, Inc., JavaServer Pages Technology, http://java.sun.com/products/jsp (2006).

[19] The Eclipse Foundation, Eclipse, http://www.eclipse.org/ (2006).

[20] Van Schaik, P. and Ling, J., "The effects of frame layout and differential background contrast on visual search performance in Web pages," *Interacting with Computers*, Vol.13, No.5, pp. 513-525 (2001).

[21] WWW Consortium, Cascading Style Sheets, http://www.w3.org/TR/css3-roadmap/ (2007).

Quan Liang Chen was born in 1980. He is a student in the Course of Information Science and Intelligent Systems at the graduate school of the University of Tokushima. His research interests include rich clients, component-based development, visual programming, and visual maintenance techniques for Web applications.

Takao Shimomura was born in 1949 in Japan. He has a BS from Kyoto University, an MS from Tohoku University, and a PhD from Tokyo Institute of Technology. He is a professor of the Dept. of Information Science and Intelligent Systems at the University of Tokushima in Japan. He was a senior research engineer at NTT Software Laboratories from 1975 to 1995 and a guest associate professor at the Graduate School of Information Systems of the University of Electro-Communications from 1992 to 1994. His research interests include software design automation, program visualization, and automated debugging. He is a member of IPS Japan, IEICE Japan and ACM.