Regular Paper

Visual Design for Server-Side Programs and Program Generation

Takao Shimomura,† Muneo Takahashi,†† Kenji Ikeda† and Yoshio Mogami†

Client programs' graphical user interfaces have been developed using graphical components. The BioPro system this paper presents makes it possible to develop server-side programs by the combination of drag & drop operations and automatic generation of server-side program code. The BioPro system has the following features; (1) users can develop Web applications according to their image of what they want to develop, (2) they can easily verify the completeness of components that make up the applications and the consistency of those relationships, and (3) they can easily confirm what they have developed, regardless of which stage of development they are currently at.

1. Introduction

With the development of the information society, it has become necessary to release software early that satisfies users. Therefore, it has become important to develop software quickly so that users can try it, and give the developers feedback. Recently, instead of the conventional water-fall-model development, new development techniques such as aspect-oriented programming have been researched [8),10),14)]. The software development techniques that make use of graphics have also been researched in a variety of fields, which include the visualization of software requirements [5),6),15)], assistance for object-oriented programming with UML [19),22)], the development of language processors with the graphical representation of their behaviors [11),30)], automatic form generation by the combination of graphical components [18),27)], visual software development environments [3),20)], and as for database accesses, visual retrieval of structured Web information [17)], and the visualization of the contents of a database [9)]. In these researches, graphics assists users to design software requirements and program structures; visualizing program behaviors assists them to design program detail specifications; and visual representation of program components and data assists them to develop complicated software. Client programs' graphical user interfaces have also been developed using graphical components [21),25)].

This paper presents the visual programming method for server-side programs that uses graphics as a tool of designing software, and enables users to easily develop software according to their image of what they want to develop [26)]. It also describes the BioPro system that implements this method for Web applications. The BioPro system makes it possible to develop server-side programs by the combination of drag & drop operations and automatic generation of server-side program code. This system has the following features: (1) users can develop Web applications according to their image, (2) they can easily verify the completeness of components that make up the applications and the consistency of those relationships, and (3) they can easily confirm what they have developed, regardless of which stage of development they are currently at. The remainder of this paper is organized as follows: Section 2 shows how Web applications are designed according to users' image with the BioPro system. Section 3 presents visual programming for Web applications. Section 4 illustrates automatic generation of Web applications from visual design and programming. Section 5 describes the evaluation of the system. Section 6 discusses related work. Finally, Section 7 summarizes the paper.

2. Visual Design for Web Applications

2.1 Visual Design and Programming

Using the BioPro system, we first design the outline of each Web page by choosing Web components (hyperlinks, tables, text fields, buttons, etc.) from menus (or buttons), and pasting them into the windows that correspond to Web pages as in home page building tools [13)]. We next design the tables of a database (hereafter,

† Department of Information Science, University of Tokushima
†† Department of Control Engineering, Toin University of Yokohama

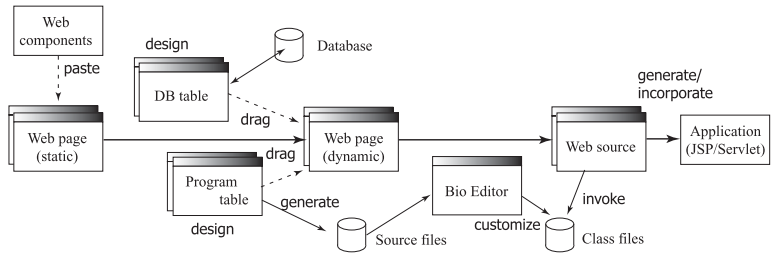


Fig. 1 Visual design and programming of Web applications.

described as DB tables) and the tables that are only used during the execution of the program (hereafter, described as Program tables). Then, we drag some of the fields of these tables into Web pages to create their dynamic pages. Finally, we write business processes in the Web source windows. The BioPro system automatically generates methods necessary for accessing the Program table data. In summary, we design the data access layer of the application using DB tables and Program tables, design its presentation layer using Web page windows, and write its business-logic layer using Web source windows. **Figure 1** illustrates how Web applications are designed, programmed and generated with the BioPro system.

2.2 Design of Web pages

In this paper, we consider, as a simple example, a program "onlineShop" that realizes Web-based online shopping and illustrate how a Web application can be developed with the BioPro system using this program. In this application, (1) we first enter our names, and (2) choose a fruit from a menu. (3) The purchased fruit is added to a shopping cart, and then we can change the number of the fruit or continue shopping. Finally, (4) when we check out, a purchase list is displayed. As shown in **Fig. 2**, this application consists of six Web pages, "entry", "shop", "order", "check", "toEntry", "noOrdersContinue", and two DB tables, "fruit", "orders", and a Program table "Cart". The "entry" page accepts a customer's name; the "shop" page displays a list of fruits; the "order" page shows the contents of a shopping cart; the "check" page displays a purchase list and records it in a database; and the "toEntry" page, which shows a link to the "entry" page with a message "Please enter from here!", warns customers when they do not enter from the "entry" page. If the shopping cart has no entry when the "order" page is displayed, the "noOrdersContinue" page will be displayed instead to show a link to the "shop" page.

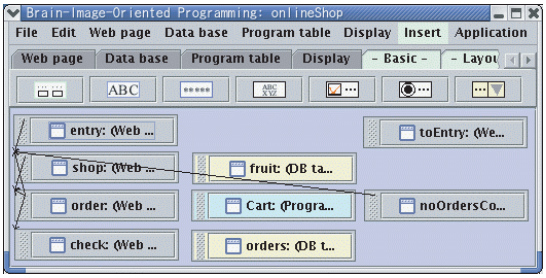


Fig. 2 Pages and tables of application "onlineShop".

2.3 Linkage with Database Tables

Most of Web applications use a database, and therefore, to make software development easy, it is important to facilitate the way of referring to the contents of a database. Let us explain how a database is treated in the BioPro system. **Figure 3** shows an example of design for the "order" page of the application, and **Fig. 4** shows an example of its execution. The "order" page shows the name, image, price, number, and amount of a fruit stored in a shopping cart. This page refers to the contents of DB table "fruit" that stores the names, images and the prices of various kinds of fruits. When we click the "Create DB table" button (or choose the "Create DB table" menu item) and specify "fruit" as a DB table name, a DB table window "fruit" is displayed. Although this window is iconified in Fig. 2, **Fig. 5** (a) shows its contents. If this table exists in the database, the contents of the table will be displayed in this window. If this table does not exist in the database, we can create and store it in the database by specifying the field names, types, and the data values for each row of the table in this window. Moreover, if we want to add, change, or delete some values, we can update the table in the database by adding, changing those values, or deleting corresponding rows in this DB table window.

In the "order" page, we arrange a table (Web table) with two rows and five columns. When

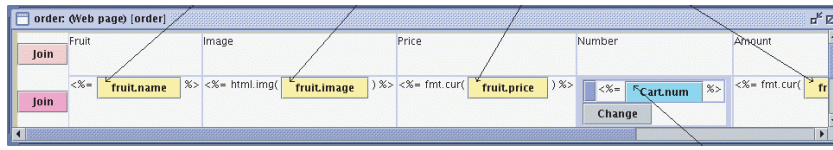


Fig. 3 Design of the "order" page.

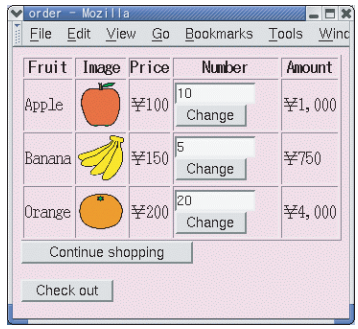
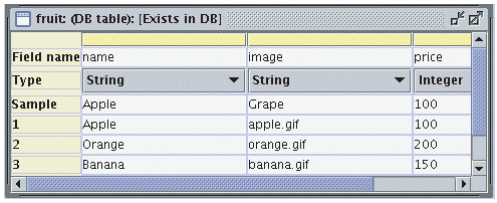


Fig. 4 Execution of the "order" page.

(a) DB table *fruit*



(b) Program table *Cart*

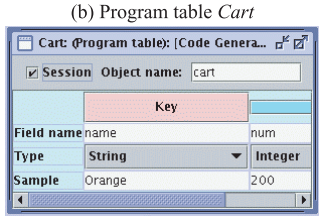


Fig. 5 DB and Program tables of application "onlineShop".

we drag and drop the "name" field of DB table "fruit" into a cell of the table to display the name of a fruit, a "fruit.name" button is automatically created and inserted into the cell. If this page is displayed when the application is executed, the table will be automatically extended so that it will have the same number of rows that DB table "fruit" has, and then the values of DB field "name" will be displayed in each cell where the "fruit.name" button is inserted. When we drag and drop the "image" field of DB table "fruit" into a cell of the table to display the image of a fruit, a "fruit.image" button is automatically created and inserted into the cell. To display the image of the image file, we choose an image-display method from the shortcut menu of the "fruit.image" button. As a result, a method call "html.img()" for displaying its image is inserted immediately before the "fruit.image" button. Similarly, we insert a "fruit.price" button and then insert a method call "cur.fmt()" to display its value in a currency form.

2.4 Linkage with Program Tables

The "onlineShop" program needs to store purchased fruit in a shopping cart. To deal with the cart, we create a Program table "Cart" as shown in Fig. 5 (b). From this Program table "Cart", the BioPro system generates a class "Cart" that defines some methods for the retrieval, addition, update, and deletion of the Program table data (described in detail in Section 3.2). Next, we arrange a text field in the "order" page, and drag and drop the "num" field of Program table "Cart" into the text field to display the number of a fruit stored in the cart.

Then, we arrange a submit button labeled "Change" for changing the number of the fruit. In order to submit the name of a fruit (to the "order" page itself) as a hidden field when the "Change" button is pressed, we drag and drop the "fruit.name" button into the "Change" button. As a result, when the "Change" button is clicked during the execution of this application, the value of this hidden field (whose default parameter name is "fruitName") is sent to the "order" page. As for the number of a fruit, we do not need to submit it as a hidden field because the value of the text field (the name of which is "num"), where the "Cart.num" button is displayed, is automatically submitted.

2.5 Procedure of Developing Web Applications

When we develop such a Web application that consists of multiple Web pages, it is important to verify the completeness of the Web pages and the Web components that constitute the application. In addition, it is also important to verify the consistency of those relationships. The BioPro system shows the relationships between Web components as colored arrows. If Web components exist that refer to undefined

ones, we decided to display them in a warning color. It is also possible to display only specific relationships such as page transfers (as shown in Fig. 2), field references (as shown in Fig. 3), and frame references, and to display the relationships concerning some specific components.

3. Visual Programming for Web Applications

3.1 User-defined Code for Displaying Web Pages

In the BioPro system, users can design Web applications according to their image by using graphics. Therefore, they almost do not need to write code when they develop such simple Web applications as display multiple Web pages among which a customer transfers, submit some values from one page to another, and retrieve some data from a database. For example, we never wrote any code to develop a simple program "selectProduct" [26], where it accepts a customer's name at the first page, and when the customer chooses a product at the next page, it displays the customer's name and the name and price of the purchased product at the final page. On the other hand, we need to write some code when we calculate values displayed in the rows of a table, specify retrieval conditions for database tables, update data values in database tables, or process Program tables. The BioPro system assists users to write such code by automatic generation of Java classes and Web source window facilities. When we develop some other business processes, we need to write Java code in the same way as in the current IDE's [21),25)].

We here describe how users specify retrieval conditions for database tables and execution code for table calculation. The "order" page of program "onlineShop" joins DB table "fruit" and Program table "Cart", and displays its result in a table. To do this, we specify a join condition ("fruit.name = Cart.name" in this example) in SQL in the "Change join condition" window, which is displayed by choosing from the shortcut menu of the "Join" button attached to the table. The "check" page displays the total amount of the purchased fruit. We specify one line of code ("total += fruit.price * Cart.num;" in this example) in Java in the "Change execution code" window, which is displayed by choosing from the shortcut menu of the "Exec" button attached to the table. Factors "fruit.price" and "Cart.num" are automatically replaced with appropriate variables, which are generated by the BioPro system, and then this code is executed repeatedly when each row of the table is generated.

The "check" page displays the purchase list and then records it in DB table "orders" that stores the customer's name, and the names and the numbers of the purchased fruits. To do this, we also write such execution code as "stm.executeUpdate("insert into orders values("'+ name +"', "'+ fruit.name +"', "+ Cart.num +")";" for the table calculation.

3.2 User-defined Code for Processing Program Tables

The "order" page stores the name and number of a fruit in Program table "Cart", and then shows the name, price, number, and amount of each fruit in it (as shown in Fig. 4). We need to update the contents of Program table "Cart" when a customer chooses a fruit at the "shop" page to add it to the cart or when the customer changes the number of a fruit stored in the cart at the "order" page. The BioPro system generates Java classes "Cart_sys" and "Cart" from Program table "Cart". Class "Cart_sys" defines some methods that perform the retrieval, addition, update, and deletion of data values in Program table "Cart". As shown in Fig. 5 (b), the field "name" of Program table "Cart" is selected as a primary key. As a result, a method "getRowByName()" for returning the row that contains a specified value as the value of field "name" is also automatically generated. Class "Cart" is a skeleton subclass of "Cart_sys", where a user can define necessary methods. The "onlineShop" application requires such methods as add a fruit to the cart and change the number of a fruit in it. Therefore, we define methods "addOrder()" and "setOrder()", respectively in class "Cart" using the methods of class "Cart_sys".

3.3 User-defined Code for Processing Page Transfers

When a customer transfers from the "shop" page to the "order" page, we need to add a fruit to the cart, and when the customer transfers from the "order" page to itself, we need to change the number of a fruit in the cart. That is, the "order" page needs to update the data values of the Program table "Cart" depending on the page which the customer transfers from. To assist users to write such processes that occur with page transfers, the BioPro system provides a Web source window for each page, where users can easily write these processes in Java.

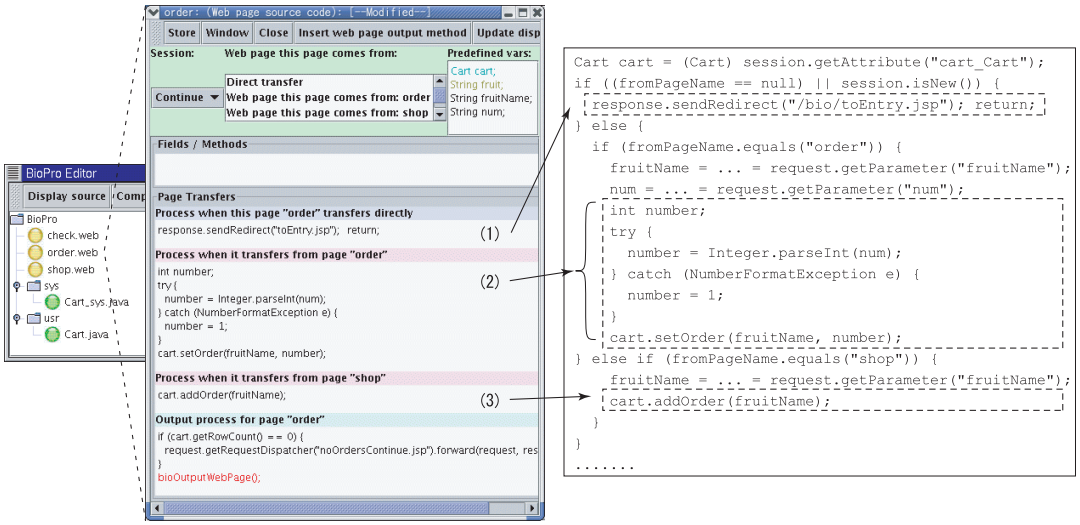


Fig. 6 Web source window for the "order" page and automatic code generation.

This makes it possible to design these processes separately from the contents of Web pages.

A Web source window corresponds to one Web page. As shown in **Fig. 6**, a "Web page source code" window (Web source window for short) displays the status of a session, a list of pages from which a customer transfers to this page, and a list of some automatically generated variables ("cart", "fruit", "fruitName", and "num" in this example) that can be referred to in the page. Below these, it also displays the name of a page from which a customer transfers to this page and a text area for that page, where users can write Java code for a process executed when this page comes from the corresponding page. For example, in a Web source window for the "order" page, such processes are written as follows: (1) when this page is directly accessed, it guides a customer to the "toEntry" page; (2) when this page comes from the "order" page, it changes the number of a fruit using automatically generated variables (variables "num" and "fruitName" in this example) in the "Predefined vars" list of the window; (3) when this page comes from the "shop" page, it adds a fruit to the cart using a predefined variable ("fruitName" in this example). Users can use the BioEditor invoked from the BioPro system. As shown in Fig. 6, it displays Web page source files (*.web) automatically generated from Web source windows and Java source files for processing Program tables (*.java). The users can edit these files and see their API specifications.

4. Automatic Generation of Web Applications

4.1 Automatic Code Generation

The BioPro system generates program code from designed Web pages, Program tables, DB tables, and Web page source files. The generated code includes code for displaying Web pages, receiving parameter values sent by other Web pages, retrieving data from a database, processing Program tables, and managing a session. The system incorporates these pieces of code with user-defined Web page source code to create a Web application, which consists of some JSP pages and Java class files. Figure 6 illustrates a part of generated JSP code for the "order" page. The code surrounded by broken lines is incorporated from the Web source window for that page.

4.2 Code Generation for Preview

In the visual design, it is important to be able to check the behaviors of the components of a program at any stage of development. The BioPro system makes it possible to check how Web pages will be displayed even if the whole program is not completed. When we choose a Web page and click the "Web preview" button, we can see the preview of that page. In the preview of the "check" page, for example, only the name of parameter "name" is displayed because its value has not yet been stored in a session. The table of this page shows the name, price, number, and amount of a fruit stored in a cart. In the preview, however, no fruit has yet been

stored in the cart. To display the contents of the cart even in the preview, a Program table provides a "Sample" row (See Fig. 5 (b)). If a user chooses one of data types for each field of a Program table, a suitable sample value will be automatically displayed in this row. The user may change those values. As shown in Fig. 5 (b), Program table "Cart" contains a "Sample" row, where the name of a fruit is "Orange" and the number of oranges is "200". The BioPro system initializes the Program table so that it will have these sample values.

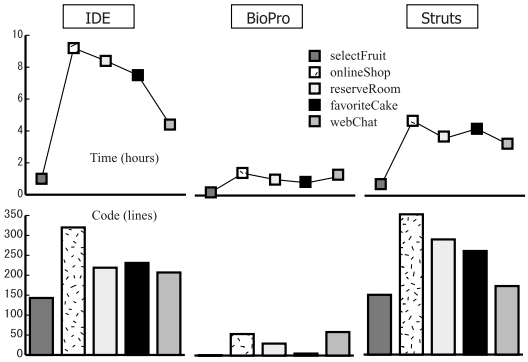


Fig. 7 Evaluation of time and code.

5. Evaluation

To evaluate the efficiency and ease of software development in the visual programming, we had experiments that compared the BioPro system with a current integrated development environment [21] and Struts [12]. Each of four programmers developed several sample programs with IDE, BioPro, and Struts in this order. Those sample programs include applications (1) "selectFruit" (selection of fruits), (2) "onlineShop" (online shopping using a Program table), (3) "reserveRoom" (reservation for meeting rooms using a DB table), (4) "favoriteCake" (a questionnaire program about favorite cakes), and (5) "webChat" (Web-based chatting using a frame set). **Figure 7** shows the lines of code and the time required to develop these applications on average. The lines of code indicate the total lines of JSP and Java code required to develop each application. The time indicates how many hours it took to make an application, test it, and make sure of its execution result. With Struts, we needed to write 50 to 60 more lines of code than the code shown in Fig. 7 for each application to set some configuration files such as web.xml and struts-config.xml. In this preliminary experiment, the BioPro system reduced the time required to develop these Web applications to 1/2 to 1/3 compared with the other systems.

This is considered to be because users need to write much less amount of code with the visual programming, and this programming method keeps the users from making mistakes during program development. Even if there is a fault in an application, they can easily detect it at an early stage of development by checking relationships between components (colored arrows are drawn between related components) and the preview of Web pages being designed. In this experiment, we considered several simple programs. Their main function is to create the contents of Web pages. In large-scale business applications that contain business processes other than those the BioPro system assists in, the effectiveness of the visual programming might not be this remarkable. However, even when users develop those Web applications, we believe that it is effective to provide facilities such as generating JSP code of Web pages, automatically sending and receiving parameter values between Web pages, automatically retrieving data values from a database, generating Java code for Program tables, incorporating user-defined code into processes for page transfers, and managing sessions.

6. Related Work

There are several systems that assist in the development of Web applications such as IDE's [21),25], Struts [12], and Zope [16]. An IDE is an integrated development environment that includes editors, compilers, debuggers, project management, various source code templates, and application servers. Struts provides a framework for building Web applications that consists of such components as views, controllers, and actions. Separately from business processes, users can easily write code for verifying form data and can specify target actions to which requests are forwarded. Zope is an application server with which users can easily develop Web applications using a Web browser that is connected to a Zope server. JavaServer Faces [28] simplifies building user interfaces for Web applications. It wires client-generated events to servere-side event handlers. Tapestry [1] is a framework for creating Web applications in Java, where a Web application is composed of a combination of a specification file in XML, an XHTML template and a Java

class. The template defines the XHTML document that includes dynamic contents, and the page components written in Java define the representation of the dynamic contents. FAR [4] is an end-user visual language to assist in the development of the Web applications that use spreadsheets. JWIG [7] provides a session model and a flexible mechanism for dynamic construction of XHTML documents. With JWIG, a web application can be written as a single thread using an extension of Java. TestWeb [23] contains a test case generation engine that determines the path expression from the model of a Web application, and generates test cases. The generated test cases are sequences of URLs which grant the coverage of the selected criterion.

IDE's assist users to write program code by providing an easy-to-use editor and source code templates that are used together with a wizard, and by incorporating and customizing various program components. In most of the systems described above, Web applications are developed using text-based languages such as XHTML, JSP, JSP tag libraries and Java. On the other hand, the BioPro system assists users by automatically generating program code from Web pages that are designed using visual Web components, Program tables, DB tables, and Web page source files. OPM/Web [2] introduces hierarchical state expressing and suppressing to model both structure and dynamics of Web applications. OOHDM [24] models a web application so that the navigation model can be separated from the conceptual model. WAST [29] specifies a navigational structure of Web applications and detects the inconsistency of parameter names between JSP pages and actions during the test execution. When users need to write code, the BioPro system shows them these parameter names as automatically generated variables in Web source windows. Even if they use a wrong parameter name, this error can be detected during the compilation time becaue that wrong variable is not declared.

7. Conclusion

This paper has presented the BioPro system that implemented a method to treat Web pages, DB tables, and Program tables as visual image, and by incorporating these components, enable users to easily develop Web applications. With the visual programming, what is being designed is intelligible, and errors are hardly involved in the design. This makes it easy to develop applications. We intend to further develop more Web components such as a variety of table generators and incorporating facility of Macromedia Flash movies.

References

1) Apache Software Foundation: *Tapestry*, http://jakarta.apache.org/tapestry/ (2003).
2) Berger, I.R., Dori, D. and Katz, S.: OPM/Web — Object-Process Methodology for Developing Web Applications, *Annals of Software Engineering*, Vol.13, pp.141–161 (2002).
3) Blackwell, A.F.: See What You Need: Helping End-users to Build Abstractions, *Journal of Visual Languages & Computing*, Vol.12, No.5, pp.475–499 (2001).
4) Burnett, M., Chekka, S.K. and Pandey, R.: FAR: An End-User Language to Support Cottage E-Services, *IEEE Symposium on Human-Centric Languages*, pp.195–202 (2001).
5) Castelló, R., Mili, R. and Tollis, I.G.: ViSta: A tool suite for the visualization of behavioral requirements, *Journal of Systems and Software*, Vol.62, No.3, pp.141–159 (2002).
6) Chen, D.J., Chen, W.C. and Kavi, K.M.: Visual requirement representation, *Journal of Systems and Software*, Vol.61, No.2, pp.129–143 (2002).
7) 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).
8) Cockburn, A.: *Agile Software Development*, Addison Wesley Longman (2001).
9) Cruz, I.F. and Leveille, P.S.: As You Like It: Personalized Database Visualization Using a Visual Language, *Journal of Visual Languages & Computing*, Vol.12, No.5, pp.525–549 (2001).
10) Elrad, T., Filman, R.E. and Bader, A.: Aspect-Oriented Programming, *CACM*, Vol.44, No.10, pp.28–32 (2001).
11) Glass, S., Ince, D. and Fergus, E.: Llun — a high-level debugger for generated parsers, *Software — Practice and Experience*, Vol.31, No.10, pp.983–1001 (2001).
12) Goodwill, J.: *Mastering Jakarta Struts*, John Wiley & Sons, Inc. (2002).
13) IBM Software: *WebSphere Studio Homepage Bldr Program Package 6.0*, IBM Software (2002).
14) Kiczales, G., Hilsdale, E., et al.: Getting Started with AspectJ, *CACM*, Vol.44, No.10, pp.59–65 (2001).
15) Kung, D.C.: An Executable Visual Formalism for Object-Oriented Conceptual Modeling, *Journal of Systems and Software*, Vol.31, No.1, pp.33–43 (1995).

16) Latteier, A. and Pelletier, M.: *The Zope Book*, Macmillan Computer Pub. (2001).
17) Li, W.S., Shim, J. and Candan, K.S.: WebDB: A System for Querying Semi-structured Data on the Web, *Journal of Visual Languages & Computing*, Vol.13, No.1, pp.3–33 (2002).
18) Mamrak, S.A. and Pole, S.: Automatic form generation, *Software: Practice and Experience*, Vol.32, No.11, pp.1051–1063 (2002).
19) Mellor, S., Balcer, M. and Balcer, M.J.: *Executable UML: A Foundation for Model-Driven Architecture*, Addison-Wesley (2002).
20) Mills, K.L. and Gomaa, H.: A Knowledge-Based Method for Inferring Semantic Concepts from Visual Models of System Behavior, *ACM Transactions on Software Engineering and Methodology*, Vol.9, No.3, pp.306–337 (2000).
21) Mogha, R. and Bhargava, R.: *Sun One Studio Programming*, John Wiley & Sons (2002).
22) Nentwich, C., Emmerich, W., Finkelstein, A. and Zisman, A.: BOX: Browsing objects in XML, *Software — Practice and Experience*, Vol.30, No.15, pp.1661–1676 (2000).
23) Ricca, F. and Tonella, P.: Testing Processes of Web Applications, *Annals of Software Engineering*, Vol.14, pp.93–114 (2002).
24) Rossi, G. and Schwabe, D.: Object-Oriented Design Structures in Web Application Models, *Annals of Software Engineering*, Vol.13, pp.97–110 (2002).
25) Shavor, S., Anjou, J.D., et al.: *The Java Developer's Guide to Eclipse*, Addison Wesley (2003).
26) Shimomura, T., Takahashi, M., Ikeda, K. and Mogami, Y.: Web application generator by image-oriented design, *ACM SIGSOFT Software Engineering Notes*, Vol.28, No.2, pp.14–19 (2003).
27) Stoecklin, S. and Allen, C.: Creating a reusable GUI component, *Software: Practice and Experience*, Vol.32, No.5, pp.403–416 (2002).
28) Sun Microsystems, Inc.: *JavaServer Faces*, http://java.sun.com/j2ee/javaserverfaces/ (2003).
29) Tai, H., Nerome, T., Abe, M. and Hori, M.: A Model-driven Development Support Environment for Web Applications, *IPSJ*, Vol.44, No.6, pp.1498–1508 (2003).
30) Zhang, K., Zhang, D.Q. and Cao, J.: Design, Construction, and Application of a Generic Visual Language Generation Environment, *IEEE Trans. Softw. Eng.*, Vol.27, No.4, pp.289–307 (2001).

(Received November 4, 2003)
(Accepted April 5, 2004)

Takao Shimomura is a professor of the Dept. of Info Sci. & Intel. Syst. at the University of Tokushima. 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 received a BS in mathematics from Kyoto University in 1973, an MS from Tohoku University in 1975, and a Ph.D. in computer science from Tokyo Institute of Technology in 1994. He is a member of IPS Japan, IEICE Japan and ACM.

Muneo Takahashi is a professor at the Toin Univ. of Yokohama. His research interests include software metrics and quality assurance. He has a BE from Chiba University and a Ph.D. from Kyushu University.

Kenji Ikeda is an associate professor at the Univ. of Tokushima. His research interests include adaptive control and system identification. He has a BE, ME and Ph.D. from the University of Tokyo.

Yoshio Mogami is an associate professor at the Univ. of Tokushima. His research interests include learning automata and reinforcement learning. He has a BE, ME from Kyoto Institute of Technology and a Ph.D. from Kyoto University.