Knowledge-Based Software Engineering
A. Čaplinskas, H. Pranevičius, T. Nakatani (Eds.)
Technologija, 2010

VISUAL MODELING AND PROGRAM GENERATION FOR END-USER-INITIATIVE DEVELOPMENT

Takeshi CHUSHO[1] and Noriyuki YAGI
Department of Computer Science, Meiji University, Kawasaki, Japan

Abstract. The development of Web applications should be supported by business professionals themselves since Web applications must be modified frequently based on their needs. This paper describes end-user-initiative application development based on visual modeling and program generation. Visual tools are necessary for development of Web applications with three-tier architecture of user interface, business logic and database. Furthermore it is necessary that program codes are generated automatically from simple description of business logic. These systems are implemented and the effectiveness is confirmed.

Keywords. End-user computing, visual modeling, automatic program generation.

Introduction

The number of Web applications which end-users have access to has been increasing. Most of these applications are developed by IT professionals. Thus, attempting to achieve automation is limited to particular tasks which calculate profit over the development cost. Furthermore, it is difficult to develop applications quickly. Primarily, Web applications should be supported by business professionals themselves since Web applications must be modified frequently based on users' needs. Therefore, end-user- initiative development has become important in the automation of end-users' fulfilling their own needs.

There are several approaches for the end-user-initiative development. That is, the UI-driven approach makes it possible to develop applications for the UI-centered front-end subsystems easily. It is strengthened by using framework technologies. The model-driven approach makes it possible to develop applications for the workflow-centered back-end subsystems easily. It is strengthened by using a visual modeling tool. Furthermore, the form-driven approach must be easier than the aforementioned two approaches for business professionals since they are familiar with forms in daily work. It is strengthened by the form-to-form transformation and Web service integration.

Terms for end-user computing (EUC) and papers on EUC often came out in 80's. Some papers describe definitions and classifications of EUC [7] or the management of EUC [2]. A recent paper summarizes the trends of end-user development without IT professionals' assistance [20].

There are some other works related to EUC. In the programming field, the technologies for programming by example (PBE) [14] were studied. The PBE implies that some operations are

[1] Takeshi Chusho and Noriyuki Yagi are with the Department of Computer Science, School of Science and Technology, Meiji University, Kawasaki, 214-0033, Japan.

This work was supported in part by KAKENHI of Grant-in-Aid for Scientific Research.

automated after a user's intention is inferred from examples of operations. The non-programming styles for various users including children and for various domains including games were proposed. In the database field, the example based database query languages [19] such as QBE (Query-By-Example) were studied. QBE implies that a DB query is executed by examples of concrete queries. User-friendly inquiry languages were proposed in comparison with SQL.

Our research target is different from these technologies and is for business professionals and business domains. The user's intention is definitely defined as requirement specifications without inference as business professionals with domain expertise develop software which executes their own jobs.

Therefore, this paper pays attention to a Web application in which the user interface is a Web browser because most users are familiar with how to use the Internet. Furthermore, the three-tier architecture is supposed, which has been popular recently. Generally, there are three approaches corresponding to the user interface (UI), business logic and database (DB). In our studies, application frameworks, visual modeling tools based on components and form transformation tools for Web service integration were developed for EUC. The tile programming especially is tried to be used for description of the business logic.

This paper presents Web application development technologies in Section 1, issues on EUC in Section 2, abstract forms and form transformation in Section 3, and modeling by tile programming in Section 4.

1. Web Application Development

1.1. Basic Approaches

Our approach to Web application development is shown in Figure 1. The business model at the business level is proposed by those end-users who are business professionals. Then, at the service level, the domain model is constructed and the required services are specified. At the software level, the domain model is implemented by using components. In this approach, the granularity gap between components and the domain model is bridged by business objects and application frameworks. The semantic gap between the domain model and end-users is bridged by form-based technologies.

The approaches to the end-user-initiative Web application development methodologies based on the three-tier architecture are classified into the three categories of UI-driven, model-driven and data-driven processes by first focusing on any one of the UI (user interface), the model (business logic) or DB. These approaches are described in this section.

1.2. A UI-Driven Approach

Recently, a UI-driven approach has emerged as Web applications are increasing. A typical example of this approach is the Struts framework [1] which is an open source framework for building Web applications in Java. The visual forms are defined first and then components for business logic and access to the DB are defined. In this approach, it seems to be easier for the end-user to define the UI in comparison with definitions of the model or the DB.

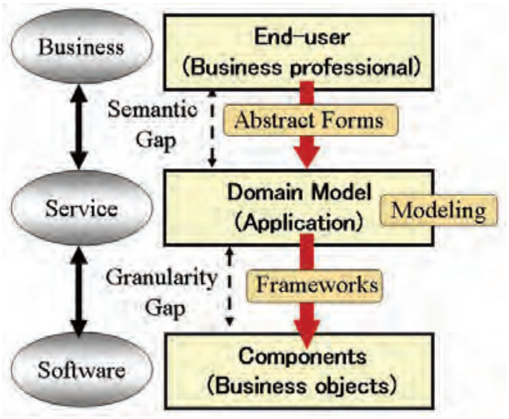


Figure 1. Technologies for end-user computing.

We have also been studying this approach for several years [5]. The UI-driven approach is proposed for the front-end subsystem based on CBSE (Component-Based Software Engineering) [3], [8]. The systems are constructed by using UI-centered frameworks [10] and agent technologies [12]. The effectiveness of the UI-driven process is confirmed through experiences with the development of frameworks.

Business professionals define requirements for an application to be developed by using the framework as follows:

1. Service definitions: Services at the counter are defined.
2. Form definitions: Forms for these services are defined with navigational information.
3. Registration: These form definitions are registered into the corresponding servers.

Intelligent navigation by agents is implemented in XML. The metadata for a window is described in an RDF (Resource Description Framework) style. While forms are defined in HTML in the conventional way, the semantics of forms are defined in RDF style also.

However, this framework for a service counter does not support the back-end subsystem with the workflow and DB. When another framework for a reservation task such as a room reservation system was developed, a visual tool for defining the DB table easily was developed simultaneously. Although end-users can use this tool, the target DB table is limited to a simple reservation table.

In our UI (user-interface) driven approach, the forms are defined first and the framework is used. The business logic depending on the application is defined by the form definitions. The other business logic is embedded into the framework.

For example, the framework for booking was developed and applied to a meeting room reservation system for our department. A total of 23 forms are developed in this system for users and the administrator. Among them, 20 forms are dynamic Web pages which are defined by our visual tool. The UI transition diagram on the forms for users is shown in Figure 2. Almost all functions are directly mapped into some forms. The functional sufficiency and the usability can be evaluated and improved easily since all use cases are simulated on the forms and the UI transitions.

However, end-users may need an IT professional’s assistance for the UI to be implemented in JSP, components to be newly developed and a complicated DB management system.

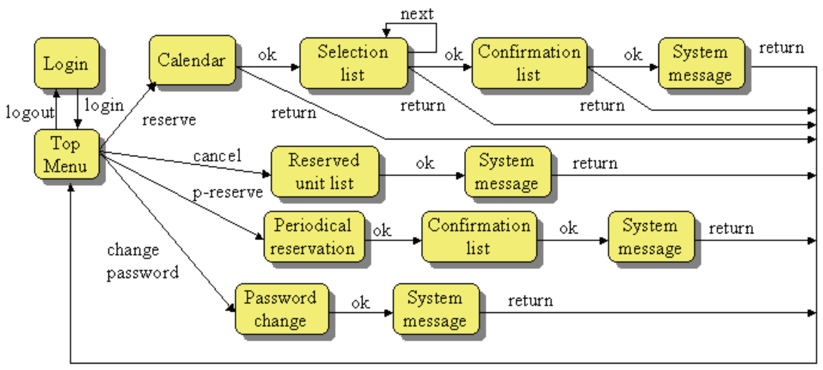


Figure 2. An example of a UI transition diagram.

1.3. A Model-Driven Approach

Around the 90's, object-oriented analysis and design (OOAD) technologies came out and have become the major methodologies. Some of them match the waterfall model and others match the iterative and/or incremental development process [11], [13]. In the OOAD methodologies, the unified modeling language (UML) [18] is used for definitions of the system model. OOAD is a model-driven approach. In addition, UML2.0 requires more rigorous definitions of models for automatic generation of program codes based on the model-driven architecture (MDA) [17].

We have also been studying this approach for several years [4]. The model-driven approach based on CBSE is proposed for the back-end subsystem which the main part is a workflow. Our solution is given as a formula of “a domain model = a computation model.” This formula implies that one task in a domain model of cooperative work corresponds to one object in a computation model based on an object-oriented model. Therefore, it is not necessary for end-users to convert a domain model into a computation model with application architecture. The domain model is considered as the requirement specifications. This process requires necessarily the fixed architecture and ready-made components such as business objects.

Our approach is different from most conventional object-oriented analysis and/or design methods which need defining an object model on static structure of objects prior to a dynamic model on interactive behavior among objects. At the first stage, the system behavior is expressed as a message-driven model by using a visual modeling tool while focusing on message flow and components. At the second stage, a user interface is generated automatically and may be customized if necessary. Then the transition diagram of user interfaces is generated automatically and used for confirmation of external specifications of the application. Finally, the system behavior is verified by using a simulation tool.

This component-based development process was confirmed by feasibility study on a given problem of the IPSJ(the Information Processing Society of Japan) sigRE group. The problem is how to define requirement specifications for a program chair’s job of an academic conference. A dynamic model was constructed while introducing eleven kinds of objects as shown in Figure 3. These objects are defined by drag-and-drop operations from the palette of icons. A message between objects is defined by drawing an arrow from the source object to the drain object.

In addition, branch conditions are described in rule expressions. For example, in the “produce” method of the CFP Production object, the following rules are described for branch conditions:

- if Printer = yes then print;
- if CFP Distribution = yes then distribute;

Furthermore, simulation is executed for validation of the requirement definitions both on the domain model and on the sequence diagram, while displaying traces of the message flows.

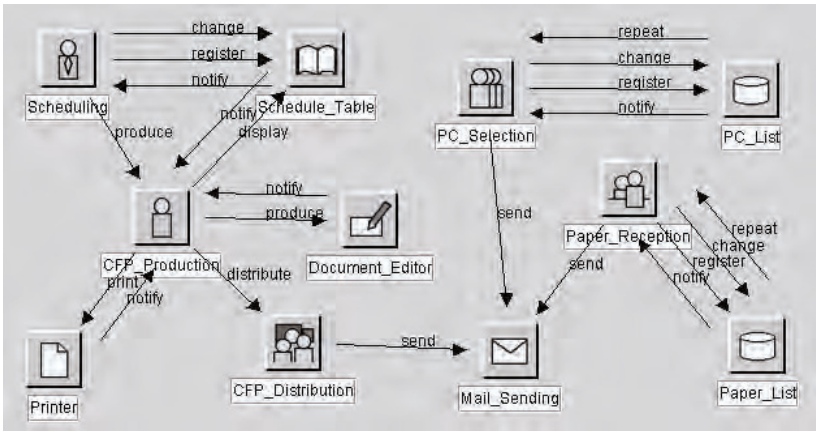


Figure 3. An example of a domain model

1.4. A Data-Driven Approach

As for a data-driven approach, a data-centered or data-oriented approach was introduced in the 80's. In this method, the data flow diagrams (DFD) are sometimes used for a definition of the workflow. The data model is defined with entity relationship diagrams (ERD). In many mission-critical applications, the DB design is the most important. Since data structures are more stable than business logic, the data model is defined prior to business logic.

However, the design of a large-scale DB is difficult for end-users. In our UI-driven approach and/or the model-driven approach, it is supposed that DB components are used. As for a small-scale DB, visual tools are introduced for defining the DB tables.

2. Issues on End-User Computing

In our experiences, sometimes end-users needed IT professionals' assistance. It is difficult for end-users to develop new components, to modify ready-made components for complicated business logic and to implement user interfaces in JSP.

In the business world, the external specifications of application software are recently considered as services as shown in keywords such as ASP (Application Service Provider), Web service, SOA (Service-Oriented Architecture) and SaaS (Software as a Service) [9], [15], [16]. Our new approach to end-user computing is that end-users develop Web applications by service integration for both the front-end subsystem and the back-end subsystem because end-users consider their applications as a level of service, not as a level of software.

That is, the service counter is considered as a metaphor to describe the interface between a service provider and a service requester for Web services. Such a service counter is not limited to the actual service counter in the real world. For example, in a supply chain management system, data exchange among related applications can be considered as data exchange at the virtual service counter.

Generally, the service counter receives service requests from clients as shown in Figure 4. Forms are considered as the interface between them. That is, the following concept is essential for our approach:

”One service = One form.”

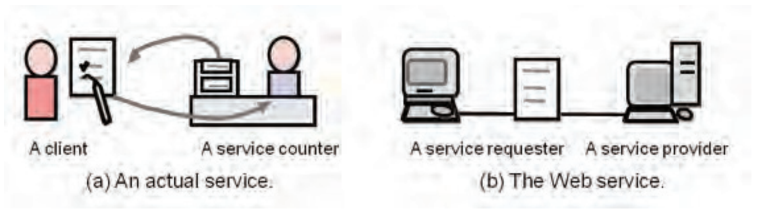


Figure 4. A service counter as a metaphor for Web service.

The integration of some individual Web services is considered as transformation from some input forms into some output forms. Although most of these forms are not visual forms, end-users can consider this form as a visual form for the requirement specification. Such a form is called an abstract form in this paper. Since end-users can consider such Web service integration as the workflow with visual forms which they are familiar with, IT skills are not required of end-users.

Furthermore, our previous two approaches are unified by these concepts. The UI-driven approach with frameworks for front-end subsystems is considered as the special case that a part of abstract forms are actually visual forms for interaction between the system and the external world. The model-driven approach with visual modeling tools for back-end subsystems is considered as the special case that the message flow is used instead of the form flow as the workflow. That is, cooperative work at an office is expressed by using a form flow model with the abstract forms.

3. Abstract Forms and Transformation

3.1. Form Transformation in XSLT

The desirable solution is that end-users can get application software by form definitions and form-to-form transformation definitions. An application which generates individual examination schedules for each student has been selected for applying our solutions to practical Web service integration. Actually, the university supports the individual portal sites for each student. The student gets the examination schedule in PDF and the individual timetable for classes in HTML. In our experiment, an actual examination schedule in PDF can be transformed into an XML document manually.

The target application generates an individual examination schedule for each student from the individual timetable for classes and the examination schedule. The form transformation is shown in Figure 5.

One input is the individual timetable for classes in HTML which is extracted from the individual portal site for each student. This document includes information about subjects for each student, that is, subject names, instructor names and class numbers. This HTML document is transformed into an XML document by using the wrapping technology. The other input is the examination schedule in XML, which includes information about subject names, instructor names, dates and periods, and room numbers. These two XML documents are merged into the individual examination schedule in XML format for each student.

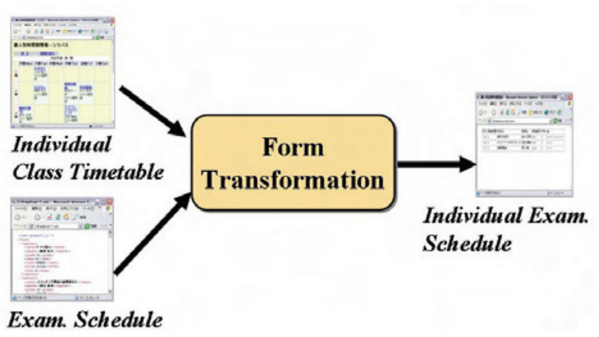


Figure 5. Form transformation for Web service integration.

This individual examination schedule in XML is transformed into an HTML document which can be displayed on the Web browser of each student. There are some conventional tools used for this transformation. The XSLT stylesheet for this application is generated by using one of the conventional tools.

The key technology of this system is the form-to-form transformation from two XML documents into an XML document. The system administrator of this application is not an IT professional but a clerk in the university office. Such an end-user does not have the ability to perform programming, but needs to modify the system when the inputs change.

For the solution of this problem, basically, the procedure of this application is described in a scripting language. Furthermore, a visual tool supports the end-user. The system generates the XML document as the output by extracting classes which are included in the both input files. The early opinions on this approach are described in detail in [6].

The basic merger transforms input abstract forms in XML into output abstract forms in XML with simple business logic. Next, this merger is extended for dealing with complicated business logic. As an example, we chose a system to advise students on graduation requirements. This system advises a student to take specific subjects necessary for graduation from inputs of both his/her school report and a manual on graduation requirements. The school report shows a list of subjects and the credits the student has already completed. The manual shows conditions for graduation, which are considered as complicated business rules. That is, subjects are categorized into several groups which compose a hierarchy, where each category has individual conditions.

For dealing with such complicated business logic, the multistage merger is introduced. In the multistage merger, generally, the intermediate output, is transformed into the next intermediate output. Finally, in this application, the seven stages were necessary for the confirmation of conditions required for graduation.

3.2. Form Transformation by Mapping

One solution to the problem of the form transformation in XSLT is the form transformation by mapping from input forms to output forms. The end-users do not need to learn XML and XSLT technologies since they can define the form transformation procedure by only mouse manipulations to relate items in input forms to items in output forms. After the definition of this procedure, the form transformation from input forms into output forms is executed as shown in Figure 6.

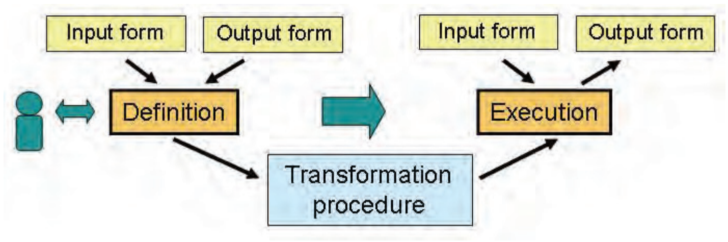


Figure 6. Form transformation by mapping.

For this study, a Web application for the reuse of laboratory equipment was selected. In the School of Science and Technology which we belong to, a lot of secondhand equipment such as PCs is thrown away although many of them can still be used. If the reuse site is open, the available but unnecessary equipment is registered there and someone can find and receive the reusable equipment easily. Therefore, our end-user initiative method is applied to this system which has the following main functions:

- The donor registers the unnecessary equipment.
- The donor replies to the donee.
- The donee receives the equipment for reuse.
- The donee requests the necessary equipment.
- The donee inquires about the registered equipment.
- The donee searches a list of the registered equipment.

The user interfaces (UI) and the UI transition diagram are designed. The business logic is specified by the form-to-form transformation (FTFT) with abstract forms. Figure 7 shows a part of form flows and form-to-form transformations. The three forms of left-hand side are visual forms for actual user interfaces corresponding to Login, Menu and List windows. The four forms of the right-hand side are abstract forms for end-users support, which are not displayed visually at the application execution time. Such an abstract form is used under construction of an application by end-users. M:N of FTFT implies the transformation from M input forms to N output forms. First three transformations are 1:1 and the last transformation is 2:1.

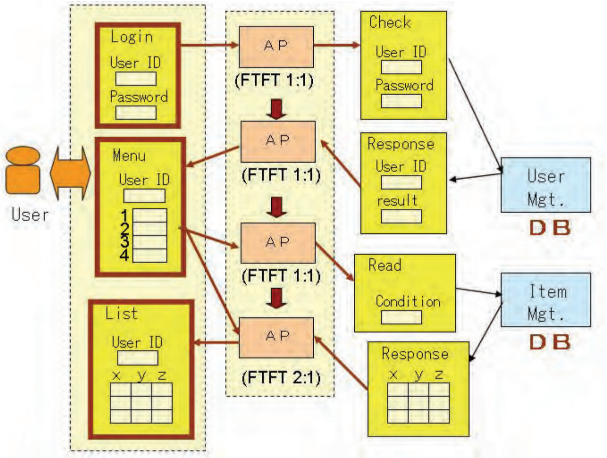


Figure 7. A part of form-to-form transformations and database accesses.

The Login form is transformed into the abstract form of ‘Check’ and it is sent to the user management DB. Next, the abstract form of ‘Response’ is transformed into the Menu form for selection of the next operation. If the user selects a display of a list of the registered equipment, the menu form is transformed into the abstract form of ‘Read’ and it is sent to the Item management DB. Then two inputs of the Menu form and the Response form are merged and transformed into the List form.

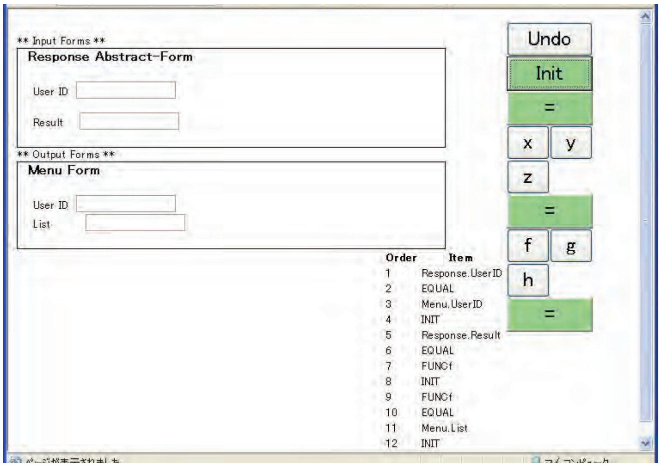
3.3. A Visual Tool for FTFT

A tool for defining the form-to-form transformation was developed. The user interface was implemented in HTML and JavaScript. The generated procedure in XML is sent to the server and stored there. The interpreter of this procedure in XML was implemented in Java.

Figure 8 shows an example of the form-to-form transformation. The input form and the output form are displayed on the left-hand side. The palette with buttons for operation items is displayed on the right-hand side. Whenever a column of forms or an operation item of the palette is clicked, the order and the name of the clicked item are displayed below for confirmation.

The transformation from the Response form into the Menu form in Figure 8 is defined as follows:

1. Response.UserID
2. EQUAL
3. Menu.UserID
4. INIT
5. Response.Result
6. EQUAL
7. FUNCf
8. INIT
9. FUNCf
10. EQUAL
11. Menu.List
12. INIT



Order	Item
1	Response.UserID
2	EQUAL
3	Menu.UserID
4	INIT
5	Response.Result
6	EQUAL
7	FUNCf
8	INIT
9	FUNCf
10	EQUAL
11	Menu.List
12	INIT

Figure 8. An example of FTFT by using a visual tool.

The first four operations define that the value of the User ID column in the input form is copied into the User ID column in the output form. The INIT operation implies the initialization as the previous execution result is not used. Next, one of the functions of f, g and h is used for complex business logic. That is, the following four operations define that the value of the Result column in the input form is the input of the f function while clicking the mouse button in order of {Response.Result, =, f, INIT}. The last four operations define that the output of the f function is assigned to the Menu column in the output form likewise.

The body of the function, f, will be implemented in a scripting language later. The variables of x, y and z are used for temporary stores of execution results.

4. Modeling by Tile Programs

4.1. Code Generation by Model Transformation

The best solution is that end-users can get application software by visual modeling which defines forms and form-to-form transformation. Furthermore it must be easier for end-users to develop Web applications if almost all forms are visual forms as actual user interfaces and form-to-form transformations are user interface transitions.

Our model-driven approach with model transformation is shown in Figure 9. End-users develop a Web application as follows:

1. A Web application model is defined by using a modeling tool.
2. The Web application model is transformed into a design model of a Struts2 model by a model transformation tool.
3. The design model is transformed into Java codes of the Web application by a code generation tool.

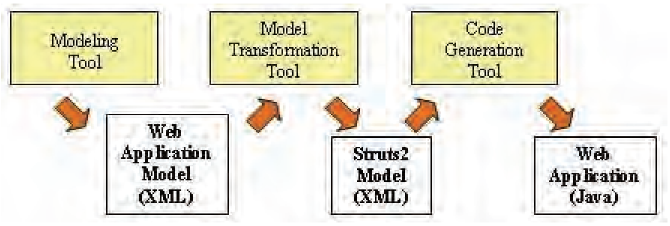


Figure 9. Model-driven approach.

4.2. Visual Modeling

For constructing the visual modeling tool, an event management system as a sample was developed. The system has the main functional specifications of registration of a user, registration of an event and recording of a reply to a request to the user for attending the event. General components with respect to user interfaces and their transition relations such as form widgets and link widgets were extracted from this application. The template commands for tile programming were extracted also.

The Figure 10 shows the usage of the visual modeling tool in Japanese when a application for a lending library is developed. The left side shows a palette of general components. The page at the upper left is the top page of the application. The page at the lower left is the form for register of a book. The page at the lower right is the form for definition of business logic of the register of a book. The page at the middle is the form for announce of termination of a register of the book. The page at the upper right is the form of database manipulation.

The top page is displayed when the Web application is initiated. The form for register of a book is displayed when the link of the register of a book is selected at the top page. When the properties of a book are entered and the button of "send" is clicked, the page for the business logic is executed and the book information is registered to the database. Then the form for announce of completion of a register of the book is displayed. The top page is displayed again when the link of "return to Top page" is selected.

4.3. Business Logic in Tile Programming

One of main problems for end-user-initiative development is how to describe business logic. In our past studies, some scripting languages and rules were tried. However, in these methods, end-users are required to learn some programming concepts. Therefore, tile programming was adopted for the visual modeling tool. The system prepares some templates for instruction statements as shown in Figure 11. End-users construct the business logic by combining these templates.

The lower right page in Figure 10 shows that business logic for book registration is described by tile programming. For example, the bottom line of the form of database manipulation, which is at the upper right in Figure 10, is labeled "Tile list." When this line is clicked, the templates for database manipulation are displayed as shown in Figure 11. The first template is the statement of variable declaration. This template is dragged and dropped on the form of business logic for book registration. Then the second template is the tile of a new book to be registered, and it is dragged and dropped on the blank column of the previous statement for variable declaration.

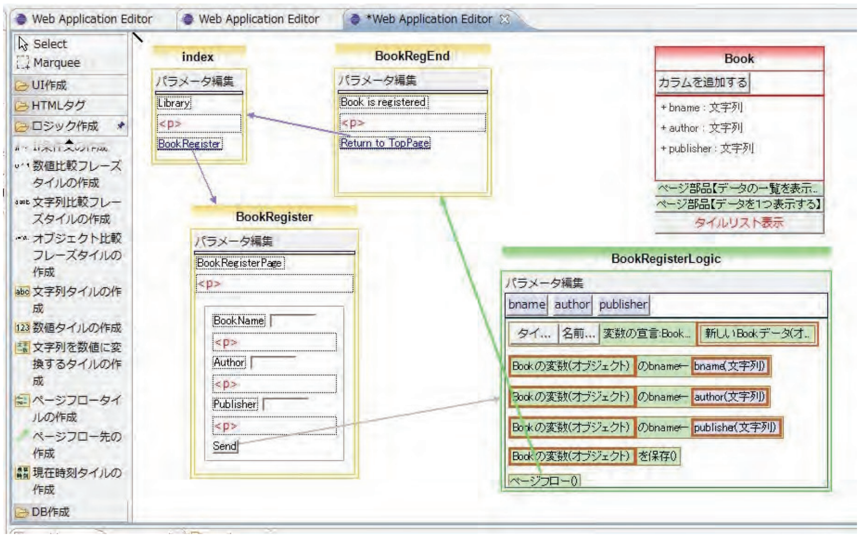


Figure 10. A Web application model by using the visual modeling tool.

Next, the seventh template is the statement of access to data in the database, and it is dragged and dropped below the statement for variable declaration. This template is displayed as "[__] <- [__]" and it implies that the right-hand value is assigned into the left-hand variable. Therefore, after the variable declaration statement is clicked, the left-hand blank column of this template is clicked. Furthermore, after the first parameter of the parameter list of "bname,"

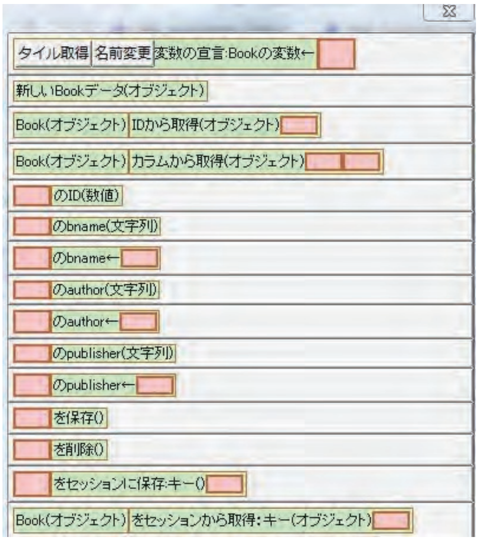


Figure 11. Templates for tile programming (in Japanese).

"author" and "publisher" is clicked, the right-hand blank column of this template is clicked. This statement implies that the title of the new book, which is passed from the form for register of a book at the lower left in Figure 10, is assigned into the new record of the BOOK database via the form of database manipulation at the upper right in Figure 10.

4.4. Model Transformation Tool

The Web application model which the end-user defines by using the visual modeling tool, is independent of the particular platform or the particular architecture and is described in a logical level. Therefore, the Web application model is transformed into the design model under the condition of the particular platform of the Struts 2 framework by using the model transformation tool.

Examples of mapping from the Web application model to the Struts 2 model are shown in Figure 12. Some of them are simple as one page is mapped into one JSP document. Others are complex as the business logic is mapped into Java classes and a Struts.xml document. This transformation program is described in XSLT since the both models are stored in XML in the system.

4.5. Code Generation Tool

The code generation tool generates the source codes for the application, which include Java classes with class names, properties and methods, and JSP files. On the other hand, a set of files which are common to Web applications, are not included in the design model, and are appended to the source codes by the code generation tool.

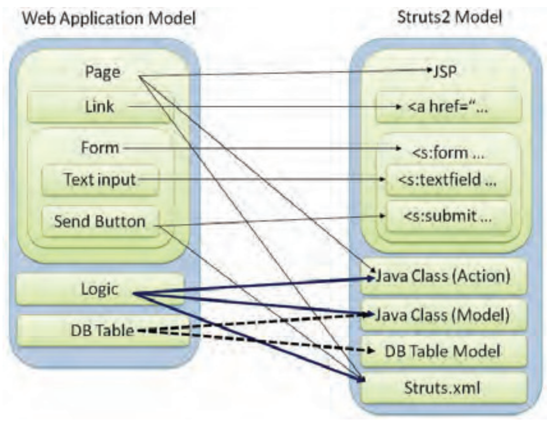


Figure 12. Mapping from the Web application model into the design model.

4.6. Feasibility Study

This visual modeling tool was applied to a library management system development [21]. The main functions are register and deletion of a member, register and deletion of a book, lending and return of a book, and a display of a list of book.

The Web application model is composed of 25 form definition pages, 8 business logic definition pages and 3 database table definitions. It is confirmed that it is easy for end-users to construct the Web application model.

Conclusions

The end-user-initiative application development approaches based on visual modeling and program generation were studied for Web applications with three-tier architecture of user interface, business logic and database. Based on results of the UI-driven approach and the model-driven approach, the form-base approach was tried. Furthermore, the modeling by tile programs was developed for easy description of business logic. The systems were implemented and the effectiveness was confirmed.

References

[1] The Apache Software Foundation, Struts. [Online] Available from: http://struts.apache.org/ [Accessed 16 April 2010].

[2] J. C. Brancheau, and C. V. Brown. The management of end-user computing: status and directions, *ACM Computing Surveys*, vol.25, no.4, 437–482, 1993.

[3] A. W. Brown(Ed.). *Component-based software engineering*. IEEE CS Press. 1996.

[4] T. Chusho, H. Ishigure, N. Konda, T. Iwata. Component-Based application development on architecture of a model, UI and components, *Proc. APSEC2000*, IEEE Computer Society, pp.349-353, 2000.

[5] T. Chusho, H. Tsukui, K. Fujiwara. A Form-base and UI-driven approach for enduser-initiative development of Web applications. *Proc. Applied Computing 2004*, IADIS, pp.II/11-II/16, 2004.

[6] T. Chusho, R. Yuasa, S. Nishida, K. Fujiwara. Web service integration based on abstract forms in XML for end-user initiative development. *Proc. The 2007 IAENG International Conference on Internet Computing and Web Services(ICICWS'07)*, pp.950-957, 2007.

[7] W. W. Cotterman, and K. Kumar. User cube: A taxonomy of end users. *Communications of the ACM*, vol.32, no.11, pp. 1313-1320, 1989.

[8] I. Crnkovic, et al. Specification, implementation, and deployment of components. *Communications of the ACM*, vol. 45, no. 10, pp.35-40. 2002.

[9] A. Elfatatry. Dealing with change: components versus services. *Communications of the ACM*, vol. 50, no. 8, pp. 35-39, 2007.

[10] M. Fayad, D. C. Schmidt, (Ed.). Object-oriented application frameworks. *Communications of the ACM*, vol.39, no.10, pp. 32-87, 1997.

[11] J. Jacobson, et al. *The Unified software development process*. Addison-Wesley, 1999.

[12] N. R. Jennings. An Agent-based approach for building complex software systems. *Communications of the ACM*, vol. 44, no. 4, pp.35-41, 2001.

[13] C. Larman. Introduction to object-oriented analysis and design and the unified process. Prentice-Hall, 2002.

[14] H. Lieberman, (Ed.). Special issue on programming by example. *Communications of the ACM*, vol.43, no.3, pp.72-114, 2000.

[15] T. Margaria, (Ed.). Guest editors' introduction: Service is in the eyes of the beholder. *IEEE Computer*, vol. 40, no. 11, pp. 33-37, 2007.

[16] O. Nano, A. Zisman, (Ed.). Guest editors' introduction: Realizing service-centric software systems. *IEEE Software*, vol. 24, no. 6, pp. 28-30, 2007.

[17] OMG. OMG model driven architecture. [Online] Available from: http://www.omg.org/mda/ [Accessed 16 April 2010].

[18] OMG, Unified modeling language. [Online] Available from: http://www.uml.org/ [Accessed 16 April 2010].

[19] G. Ozsoyoglu, H. Wang. Example-based graphical database query languages. *IEEE Computer*, vol.26, no.5, pp.25-38, 1993.

[20] A. Sutcliffe, N. Mehandjiev. (Guest Ed.). End-user development. *Communications of the ACM*, vol.47, no.9, pp. 31-32, 2004.

[21] N. Yagi and T. Chusho, A method for Web application development by end-users based on model transformation (in Japanese). *IPSJ SigSE Technical Report 2009-SE-163*, p.81-88, 2009.