New Web Application Development Tool and Its MDA-Based Support Methodology

● Yasuyuki Fujikawa ● Takahide Matsutsuka

(Manuscript received February 11, 2004)

Web applications are ubiquitous on the Internet, and almost every type of business now needs to be able to quickly develop their own Web applications. This paper introduces a technology developed by Fujitsu that reduces the development period for Web applications and also improves their quality. The paper discusses a development style that enables iterative refinement of a specification and specification checking with customers.

1. Introduction

1.1 Problems in Web application development

The importance of the Internet is rapidly increasing, and Web applications are quickly gaining popularity. Especially, thin Web client applications, in which the content is described using HTML, are now quite common (**Figure 1**). However, developers have two major problems: the low productivity and low maintainability of Web applications compared to conventional client-server type applications. These problems occur for the following reasons:

1) We need to use many technologies at the same time, for example, HTML, Java, Java Servlet, JavaServer Pages (JSP), JavaScript, and JavaBeans. The quality of a system strongly depends on the technological skills of the developers. However, the developers might not have sufficient skills to develop a high-quality system.
2) Because the ranges of each technology overlap each other and there is no standard architecture, there are various ways of implementing applications. Such diversity makes development difficult in terms of reuse.
3) Frequent changes of business needs can be observed in many cases, and applications need to reflect such changes in a timely manner.

1.2 MVC architecture

To solve the problems mentioned above, several software frameworks have been proposed that are based on the Model-View-Controller (MVC) architecture (**Figure 2**). The MVC model is a technique for designing applications. It classifies the functions of an application into three types of components: models, views, and controllers.[1)]

Models carry the data and the application logic. Views display the information from a model in windows or browsers. Controllers receive requests from a view and dispatch the requests to the appropriate models or views.

1.3 Advantages of Web application frameworks

By using the Web application frameworks described above, developers can easily separate the model from the view based on the MVC framework. Because the MVC framework provides most of the controller's parts as a common functional-

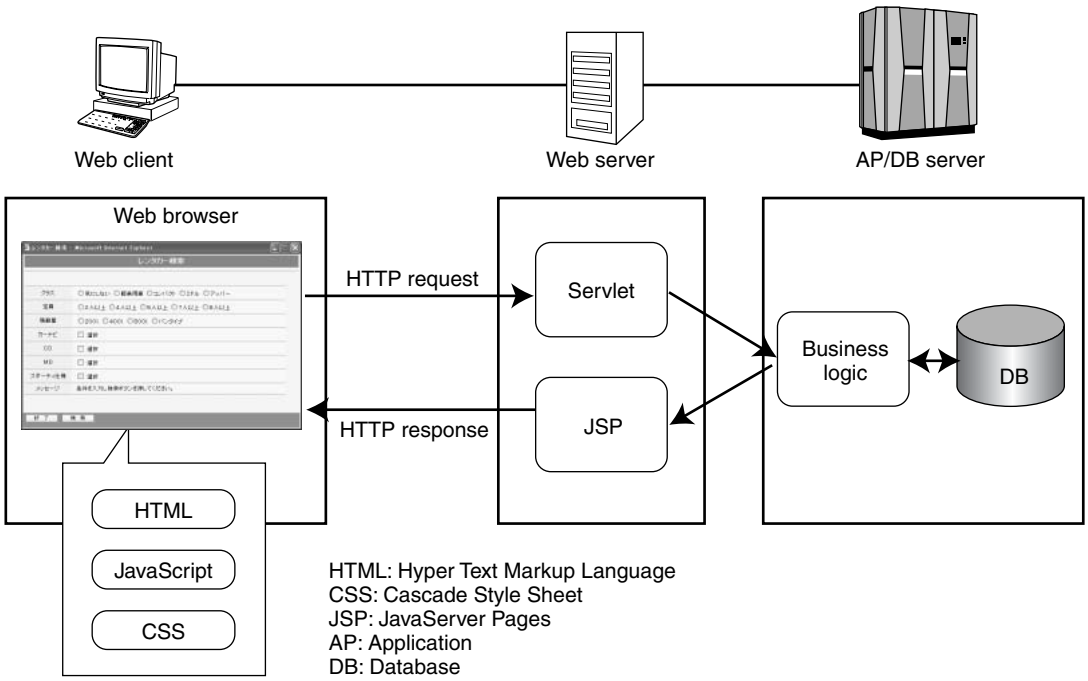

Figure 1
Thin client Web application.

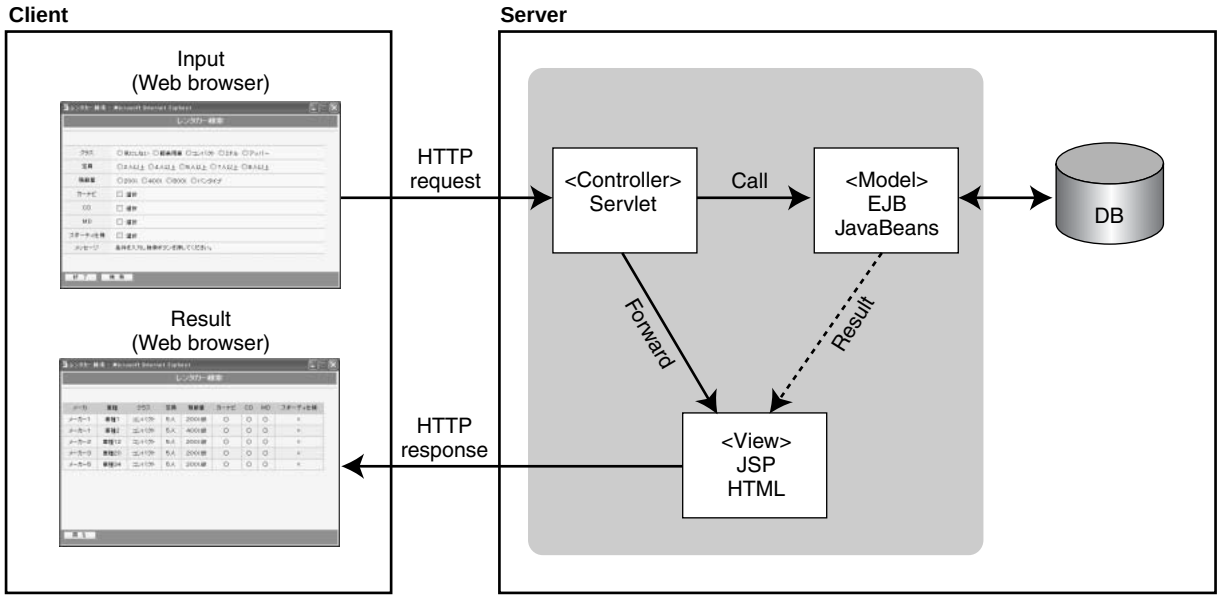

Figure 2
Model-View-Controller (MVC).

ity, developers do not need to implement the controller from scratch.

The Web application framework provides the following advantages:

1) The amount of development is reduced.
2) Developers can easily standardize the construction of applications.
3) Individual developers can focus on their specialties. For example, a designer can create views using HTML and a Java programmer can develop models using JavaBeans.
4) Developers do not need to consider low-level

layers such as network protocols.

Another strength of the framework is its conceptual nature, which consequently contributes to screen transitions. Because the framework provides definitions, it allows the development of screen transitions to reflect the conceptual specifications. In other words, the framework definitions contribute more to the development of screen transitions than code per se.

Judging from the above advantages, we can expect improved productivity, maintainability, quality, and reusability.

Fujitsu released a Web application framework called Webcoordinator[note 1)] in 2001. In 2002, Struts[2)] appeared as another Web application framework from the open source community. These initiatives indicate that frameworks are now more commonly used.

1.4 Problems of Web application frameworks

While frameworks solved some problems in Web application development, various new problems are found. For example:

1) Before developers can develop applications, they must spend a substantial amount of time mastering the given frameworks.
2) Because models and views are independent of each other, developers have to carefully organize the relationships between them by editing definition files that describe the relationships.

2. Our approach

2.1 Model Driven Architecture (MDA)

MDA[3)] is a software development architecture proposed by the Object Management Group. It enables developers to develop applications without knowing the underlying platforms. Because of this characteristic, MDA is expected to solve the problems described in the previous section.

We therefore adopted MDA[note 2), note 3)] as a basis on which we solve problems in Web application development. The concept of MDA is shown in **Figure 3**.

The key concepts of MDA are as follows:

1) MDA uses two models, called the Platform Independent Model (PIM) and the Platform Specific Model (PSM), to map a real-world problem to a computer system. The PIM reflects the conceptual model of the target system. The PSM describes a model that is specific to the target platform, for example, a relational database or Java 2 Platform, Enterprise Edition (J2EE).
2) MDA uses tools to convert a model between the PIM and PSM levels and generate source code from the PSM model.
3) MDA adopts Unified Modeling Language (UML)[4)] to describe the PIM and PSM models.

2.2 Problems with current MDA tools

Although MDA is a useful concept, the existing tools are not sophisticated enough to fully realize it. Many tools do not support complete conversion between PIM and PSM or between PSM and source code. As shown in **Figure 4**, most current MDA tools generate only a skeleton code

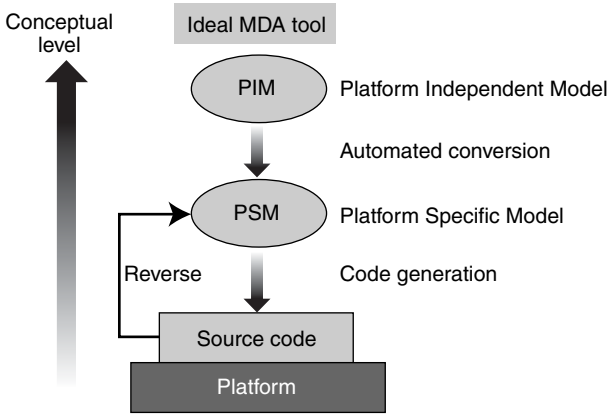

Figure 3
Concept of the Model Driven Architecture.

note 1) Webcoordinator has been renamed Apcoordinator.
note 2) The term "model" in MDA differs from that in MVC.
note 3) OMG and MDA are registered trademarks of Object Management Group.

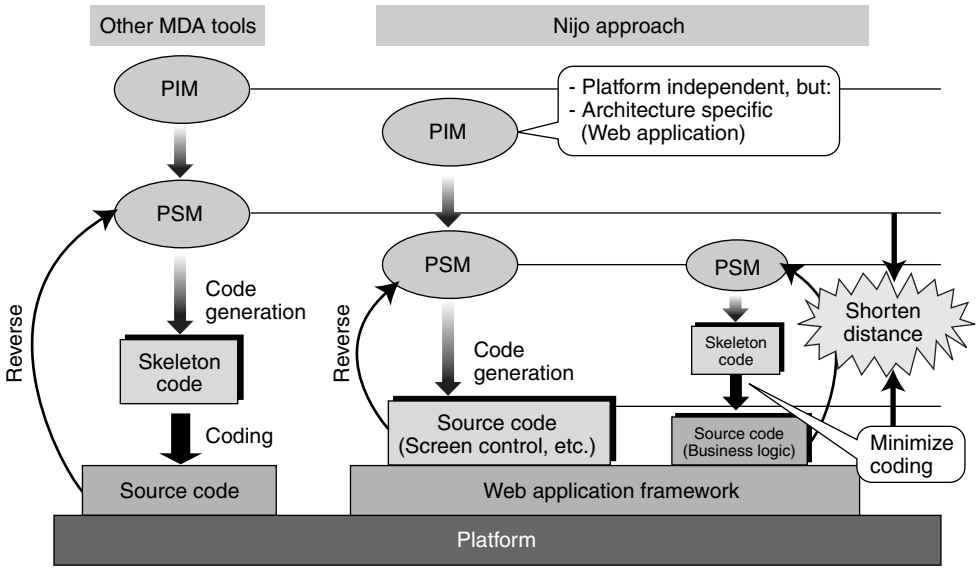

Figure 4
Model Driven Architecture and Nijo's approach.

from PSM. Therefore, developers still need to write most of the source code.

2.3 Nijo approach

We solved this problem by limiting the application areas to Web application. Figure 4 shows the difference between existing MDA tools and our Nijo approach. Nijo has been implemented using a standard UML extension mechanism, so users can describe a Web application as a PIM-level model.

Our approach is as follows:

1) A Web application is described as a model. Models are independent from specific platforms, but are specific to Web applications. To describe a model, we introduce a formal modeling language for Web applications. More detailed information about the language is given in the next section.
2) Models are automatically converted into models specific to the target framework. A converted model is similar to PSM at the conceptual level.
3) Our tool generates a set of source code from the PSM model.

We do not support certain portions of an application, for example, business logic. However, we provide connections between other PSMs and our models. Developers can independently define business logic as a PIM with other MDA tools and connect it to the Nijo model at the PSM level.

2.4 Language for Web application modeling

To describe a Web application model, we designed a modeling language. A model written in this language is called a Nijo model. This language is defined as an extension of UML. Selecting UML as our basis has the following advantages:

1) The language inherits UML's ability to express a model graphically and formally.
2) The language has an extension mechanism.
3) Models have a single, unambiguous interpretation, because they are defined formally in terms of semantics.
4) The popularity of UML promotes efficient user training.

Figure 5 shows the meta-classes of the Nijo model in a UML class diagram.

A Nijo model consists of the core part of the application and a definition part for screen

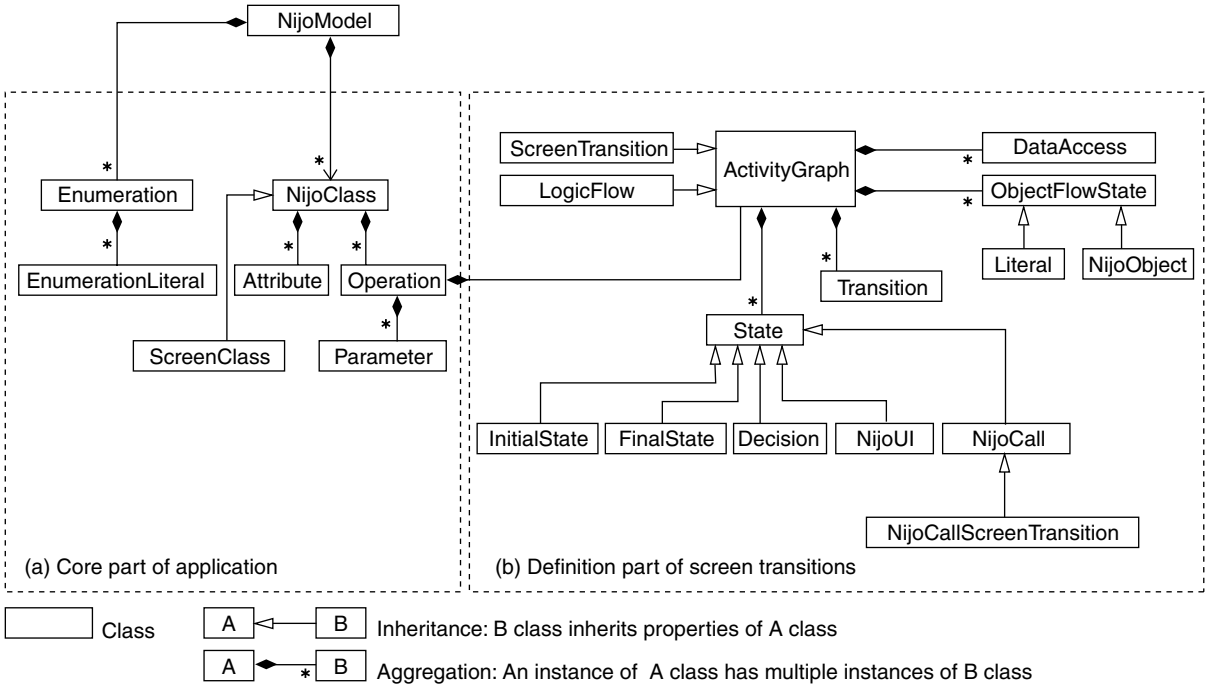

Figure 5
Nijo meta model.

transitions.

The application core part defines the main composition elements of the Web application, for example, classes, data items, and methods. This part is the same as a UML class diagram except for its ScreenClass. The ScreenClass represents the data structure of a screen, in which each data item has additional information such as the distinction between the input and output, the alignment style, and the number of characters. In short, the ScreenClass is an extension of Class in UML.

The definition part of the screen transitions is an extension of the UML ActivityGraph. It contains the following elements: a NijoUI that represents a screen, a NijoCall that represents an action that calls a procedure defined elsewhere, a Transition that indicates a change of state from one activity to another, a NijoObject that represents an object, and a DataAccess that represents the relationship between a NijoUI or NijoCall and a NijoObject to specify data accessed by the activity.

2.5 Development tool for Nijo model and graphical expression of Web applications

To make Nijo-compliant models, we developed a development tool called Apmodeler. It provides a GUI, with which all the information of a Nijo model can be manipulated.

Figure 6 shows an example of a screen transition diagram (STD), which can be edited through the STD editor of Apmodeler. In the diagram, NijoUI, NijoCall, and NijoObject are indicated by a window icon, two small squares, and a package, respectively. A solid line represents a Transition between NijoUIs and NijoCalls. A dashed line represents a DataAccess from a NijoUI or a NijoCall to a NijoObject.

This example is for a small application. It contains the following screens and operations: an input screen that accepts a search condition from the user, a search operation that accesses a database using the search condition, and a result screen that displays the result data.

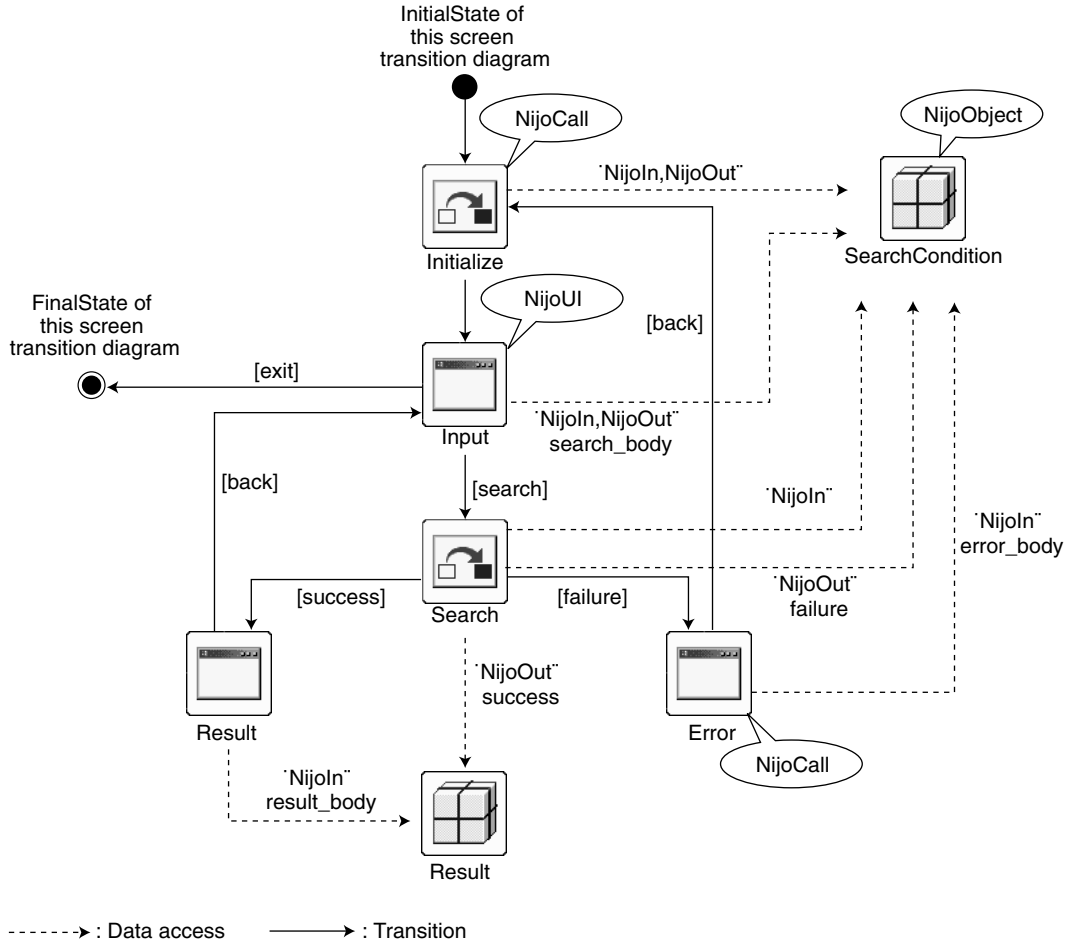

Figure 6
Screen transition diagram.

2.6 Apmodeler runtime environment

Apmodeler has been implemented as a plug-in of Interstage Apworks. Interstage Apworks is based on Eclipse, which is an open-source software development environment. Eclipse allows the Apmodeler plug-in to be well-integrated with the Eclipse environment, so Apmodeler can be used with other plug-ins of Eclipse.

3. Development process using Apmodeler

3.1 Problems with the conventional development style

The following is a conventional style of developing Web applications:

1) User interface definition

Systems engineers (SEs) and customers discuss the user interfaces, screen layouts, and data items in screens. Then, the SEs create HTMLs as a specification of this information.

2) Conversion of HTMLs to JSPs

Programmers transform the HTMLs to JSPs.

3) Development of the application

Programmers develop the application, which consists of parts such as the screen transition part, database access part, and input check part.

4) Confirmation of application behavior

The SEs check the application and deliver it to the customer to make sure the product meets the customers' requirements.

This process has the following problems:

1) The customer cannot check the behavior of a specification until the application has been developed.

2) The customer and SEs only check the static aspects of an application, for example, the screen layout. The dynamic part of a specification is developed by programmers at a later stage.
3) It is difficult for the customer to check the behavior of an application with a static HTML specification.

3.2 Development style using Apmodeler

To solve these problems, Apmodeler provides features to help the customer check the behavior of an application at an early stage of development. By using Apmodeler, we can take the development process shown in **Figure 7**.

1) User interface definition

Based on the customer's requirements, SEs define data items in screens and screen transitions. Then, Apmodeler generates an executable application with a screen layout (JSP). The SEs can run the application and discuss the user interfaces with the customer and show them how the application behaves. The customer can then check immediately whether the application meets their requirements.

2) Refinement of screen layout

SEs refine the generated JSP interactively based on the user's comments about the application's behavior.

3) Development of application

The user interface specification has already been developed. Programmers develop only the business logic parts, for example, the database access routines.

4) Confirmation of application's behavior

SEs check the application and deliver it to the customer if it is satisfactory.

This process resolves the problems of the conventional development style.

1) User interfaces can be decided at an early stage of development. The customer can check the behavior of the application directly by running it.
2) The customer can check not only the static layouts but also the dynamic behavior of the application.
3) SEs can define all of the user interface specification. In general, SEs can determine the

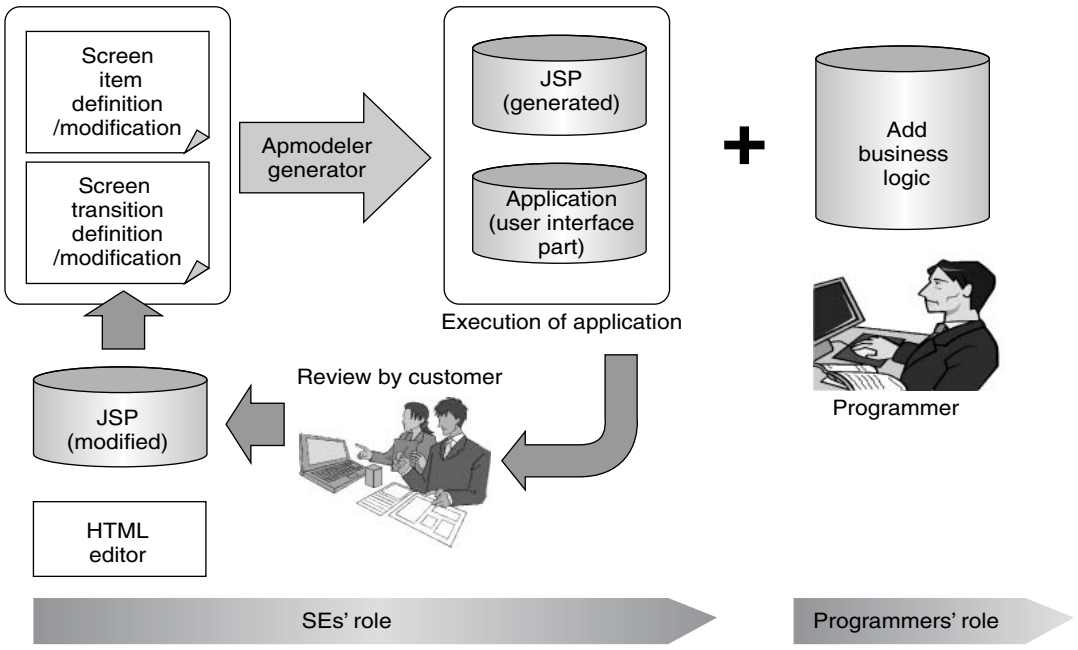

Figure 7
Development style using Apmodeler.

user's requirements, but may be less able to complete the programming. However, by using Apmodeler, SEs can develop the application's exterior without needing programming skills.

4) Programmers can concentrate on the business logic and the application's interior.

4. Discussion

4.1 Debugging

Automatic code generation tools may have shortcomings in the maintenance phase. When programmers find a bug, they tend to change the generated source code directly rather than changing the specification documents. They stop using the generation tools at this point, because the source code would be overridden if they used them again.

To address this problem, Apmodeler supports debugging functionality at the model level. Similar to the debugging functionality of popular programming languages, the developers can monitor the current location, set breakpoints, and inspect the value of objects. All operations can be carried out in the context of the screen transition diagram. Therefore, the developers do not need to debug the generated code.

4.2 Application parts not handled by Apmodeler

Nijo does not handle all parts of an application. As discussed in Section 2.2, the developers must connect the model developed by Apmodeler and other modeling tools at the PSM level.

Therefore, we may need to come up with a mechanism to connect the Nijo model with other models at the PIM level.

5. Conclusion

We introduced a graphical language that can model Web applications in platform-independent forms. Using this language, developers can develop Web applications without knowing specific platforms or frameworks.

We also proposed a new development style that makes it possible to check whether the specifications meet the customers' requirements at an early stage.

References

1) T. Matsutsuka et al.: An Architecture to Develop Presentation Logic for Enterprise Business Applications. Proceedings of the Evolve 2000 Conference, Feb. 2000.
2) The Apache Software Foundation, Struts. *http://jakarta.apache.org/struts/*
3) MDA Guide Version 1.0.1. *http://www.omg.org/docs/omg/03-06-01.pdf*
4) OMG Unified Modeling Language Specification Version 1.5. *http://www.omg.org/docs/formal/03-03-01.pdf*

Yasuyuki Fujikawa received the B.S. and M.S. degrees in Physics from Osaka University, Osaka, Japan in 1985 and 1987, respectively. He joined Fujitsu Ltd., Kawasaki, Japan in 1987, where he has been engaged in research and development of systems engineering methodologies and tools.

Takahide Matsutsuka received the B.S. and M.S. degrees in Computer Science from Tokyo Institute of Technology, Tokyo, Japan in 1994 and 1996, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1996, where he has been engaged in research and development of enterprise distributed systems and software architecture. He was a visiting researcher at Carnegie Mellon University in the 2001 to 2002 academic year, where he was engaged in research of pervasive computing. He is a member of the Information Processing Society of Japan (IPSJ).