

Newly-developed Cost Management System for Motorcycles

Seiji TSUCHIYA*

Katsunobu YAMAMOTO*

Masako WATANABE*

Takashi ARAMAKI*

ABSTRACT

A system, which efficiently totals the cost of the parts of a motorcycle throughout the model development process, from the concept stage through the prototype stage, to mass production, has been developed. This is characterized by the capability of introducing cost and supplier information into mass production parts and newly developed parts of the Development division, linking Manufacturing BOM (Bill of Material) managed by the Manufacturing division and the Development BOM managed by the Development division. This system has been developed with Java, one of the fundamental technologies of webification. Furthermore, such method has been introduced as lowering the maintenance cost and load on the network and PC clients, which reduces the concerns when considering overseas use. Consequently, this system makes it easier to see the cost changes of a new model being developed, and helps to improve operation efficiency in calculating model costs in overseas offices as well as offices in Japan.

1. Introduction

Cost management for motorcycle development was conducted largely using spreadsheet software and a host computer in the Purchasing Division. The number of models being developed has been increasing in recent years, which along with an increase in overseas production, has made it extremely time-consuming to collect the necessary information needed for such a cost management method, and also has made it difficult to contribute to shortening development time. It is also necessary to have the capability to manage locally-procured parts cost and supplier information together with the cost for Japan-supplied parts in order to manage total cost per unit for models produced overseas. Therefore, a cost management system that could link the engineering bill of material (BOM)⁽¹⁾ and the factory BOM became necessary. Consequently, a cost management system was developed that could be used efficiently by development, purchasing, and production divisions.

* Asaka R&D Center

2. Development Concept

2.1. Issues in Model Cost Management

Management was previously conducted independently by the divisions concerned, so that the engineering BOM would manage technical information relating to the development of a model, and the factory BOM would manage parts costs and suppliers involved in production. Cost management of a new model therefore necessitated obtaining the two separate BOMs from the development and production divisions, then expending a large number of person-hours in order to combine the two by hand. Obtaining the cost for locally-procured parts managed by production divisions overseas also required a great deal of time. Many hours and much manpower were required to calculate the model cost for these reasons. Furthermore, the same issues were faced at overseas offices as well as in Japan. These issues are summarized below:

- Engineering BOM and factory BOM are not managed in an integrated manner
- No tools are available for efficiently calculating new model costs
- A long time is required to obtain information on locally-procured parts from overseas offices

The new system was developed in order to overcome the above issues, and the objectives were set accordingly, as follows:

Provide information-sharing tools that can be used efficiently to share the interactive information required for model development both in Japanese and overseas offices as a cost management system for motorcycle parts aimed to establish common tooling.

2.2. System Requirements

The requirements for a system to deal with these issues are as follows:

- (1) Centralize engineering BOM and factory BOM
- (2) Provide function to calculate the single unit cost for a model
- (3) Share information on locally-procured parts at overseas offices
- (4) Implement global utilization (highly responsive, 24-hour operation, local-language support)

The new system was developed using a Web-based system architecture in order to meet the extremely large number of needs for overseas use.

3. Development Method

3.1. System Development Method

3.1.1. Change of development style

With the dizzying pace of change in recent years, there is demand for systems that are capable of handling multiple user needs. There are also increasingly complex architecture-related concerns, which arise repeatedly, such as distributed systems, parallel systems, replication, security, load balancing, and fault tolerance. The design of the system model (program structure and database structure) that forms its foundation has become a crucial element in system construction and maintenance. The reason for modeling a complex system is that it has become difficult to completely grasp the system as a whole. Just as the focus in computers has shifted from mainframes to personal computers and the Internet, the conceptual approach of system development methods is also undergoing change. It has become necessary to carry out system development with engineering drawings like these, which could not be represented using conventional system development methods by uniting incompatible system development methodologies, such as the Object Oriented Approach (hereafter OOA) and the Data Oriented Approach (hereafter DOA), in order to produce.

3.1.2. Development process and objectives

The aim with the new system was to unite the strengths of OOA and DOA to realize flexibility and reusability. The specific objectives of each phase in the development process are as follows:

- (1) User requirement definition phase
 - User requirement documentation (use case scenario)
 - Business flow (business flow diagram)
 - Prototype screen

(2) Analysis & design phase

- Class relationship diagram based on use cases (program structure)
- Database model
- Skeleton source code (Program template)
- Test specifications
- Manual of operating procedure

(3) Program development & testing phase

- Program source code
- Test report

The above summarizes the output yielded by each phase.

3.2. System Design Methods

3.2.1. User Requirement Definition

The business flow of the system (Fig. 1) was laid out in the initial phase of development in order to express the specific objectives below with regard to user requirement definitions.

Based on the business flow created here, prototype screens were created and the final image of the system was confirmed with users. This work was performed at the initial stage of development in order to prevent any deviation from occurring in the direction of system development.

- (1) Understand the organizational structure and its fluidity
- (2) Users and developers share a common understanding aimed at an efficient operational structure
- (3) Elicit system requirements
- (4) Reach agreement with users and concerned parties regarding necessary system requirements
- (5) Have program developers gain a firm understanding of system requirements
- (6) Clearly define the scope of system application

3.2.2. Database design by DOA

Data modeling⁽²⁾⁻⁽⁴⁾ was carried out with the intention of focusing on the forms used in the workplace, the spreadsheet data itself, and performing logical analysis and modeling in time series with the business flow. Commercially available tools were employed for the new system in order to ease the task of system modification while maintaining the logic of the database structure with respect to future changes in the business model.

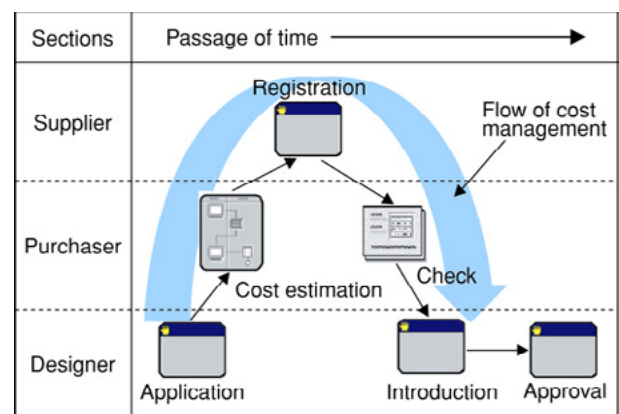


Fig. 1 Business flow diagram

3.2.3. Program design by OOA

During the design phase, it was calculated that this system included approximately 700 programs. In consideration of the relatedness, reusability, quality, and so on of the program as a whole, an object-oriented language was utilized for program development. This does not mean that measures to address the immediate issues were all developed from the ground up. Rather, programs used to address issues in the past were reused and such programs were accumulated within the system as whole. This method linked such programs together much in the way that a structure is built out of building blocks. Repeated trials of this kind of program reuse were carried out during the design phase with the building blocks further revised each time until the design was finally complete. It is also necessary to model the programs by a process similar to modeling data, as explained in section 3.2.2. In this process the structure of the program itself is defined and the roles of the business logic layer (Model), the screen display layer (View), and the control layer (Control) are clearly assigned. The group of programs used in common is extracted, and then the building blocks are made into parts⁽⁶⁾ so they can be reused. The system was designed in this way to give it the flexibility to deal with future changes.

4. System Construction

4.1. System Architecture

The new system utilized a system architecture as shown in Fig. 2. All processing demanded by clients is controlled and executed on an application server. Periodically demanded processing is managed by a job controller and executed at fixed times on the application server. Processing was concentrated in the application server to reduce the load on the clients and improve system maintainability. Processes

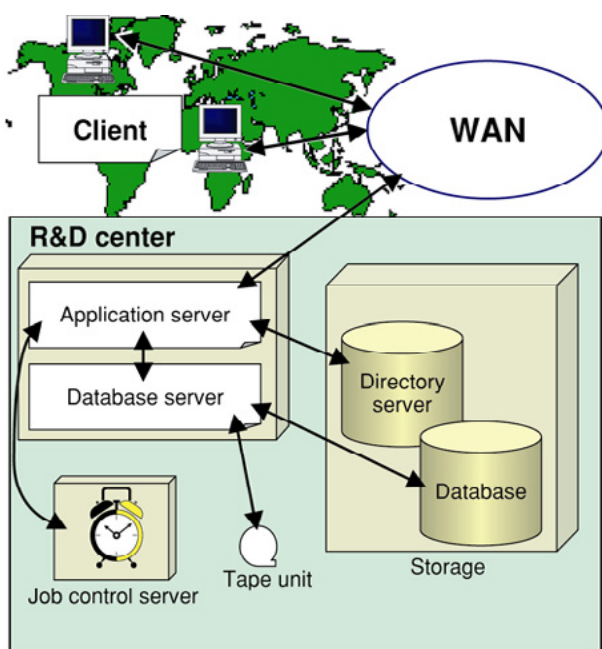


Fig. 2 System architecture

executed on the application server can access databases on the database management server to conduct queries and updates. The server is divided into an application server and database management server in order to distribute system load on the server.

4.2. System Construction Techniques

4.2.1. Centralized management of engineering BOM and Factory BOM

As shown in Fig. 3, the new system defines the engineering BOM and factory BOM, which have different management configurations, on the database. This associates the two BOMs and represents them as though they were a single BOM.

When a new drawing is issued, the drawing issue information is transferred from the engineering BOM system to this system. Cost information is also acquired from the factory BOM system. A setup was also developed for the new system whereby a single cost bill of material (cost BOM) can be generated by matching the information in the engineering BOM with the costs in the factory BOM.

This had made it possible to grasp both the centrally managed engineering BOM and factory BOM at the same time. Users can now automatically compute costs on the new system without having to read the two BOMs and then re-input that information into spreadsheet software as they did before. This has significantly improved work efficiency. Costs of production parts can also be registered and updated without having any effect on information in the engineering BOM. This has made it possible to conduct cost variation control that tracks changes in engineering specifications in development divisions.

4.2.2. Security management

The security-related logic for the new system is set up as shown in Fig. 4. It was designed in this way to enable user security to be restricted by specific product models.

The user management server (referred to below as the directory server) is set up in advance with information on staff members, and the database with information on model

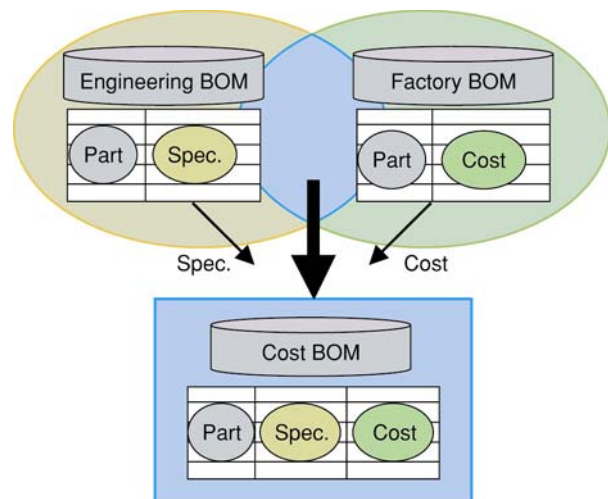


Fig. 3 Structure of cost BOM

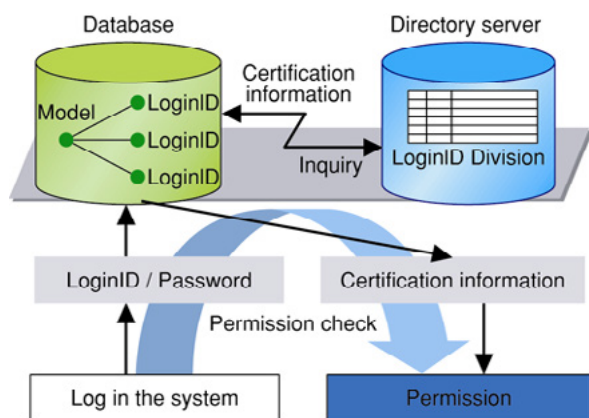


Fig. 4 Setting of authorization information

restrictions.

When users log into the systems, the directory server authenticates their log-in IDs and passwords. It then uses a model authentication program to query the database regarding model restrictions for the particular user logging in, after which it makes a final determination. This has made it possible to conduct authority control that differs for each individual user. This authorization set-up was designed not only to restrict updates of model data but also to restrict queries according to authorization. As a result, the new system has realized a high degree of security against leakage of cost information for offices or other models.

4.2.3. Summarizing the single unit cost for a model

The cost BOM in the new system is subject to transition history management for each part. This allows users to summarize the registered cost-related information for a single model unit at any point in time, and it also makes it very simple for them to find the costs at that point in the transition history.

The cost calculation method is to multiply the unit cost of a part by the share rate (the proportion ordered from each supplier), the fitting rate (a rate of use of that part), and the number applied. That quantity is then multiplied by the share rate, fitting rate, and number applied of all parts. By summarizing the costs for each part, this method yields the single unit cost for a model.

4.2.4. Dealing with globalization

The new system was designed with a view to deployment to overseas offices. It is a characteristic of motorcycle operations that most of the overseas offices tend to be small in size. It is necessary to assure a minimum level of network response for offices located in fragile network environments. It is also necessary to provide a user interface that allows simple operation by users in all offices when sharing information on locally-procured parts. The new system was designed to meet these objectives by running on Web browsers that have intuitive operating methods and that users at any office have experience using (Fig. 5).

The issues faced in deploying the new system to overseas offices include the following:

(1) Operational maintenance

Some small-scale offices do not have maintenance

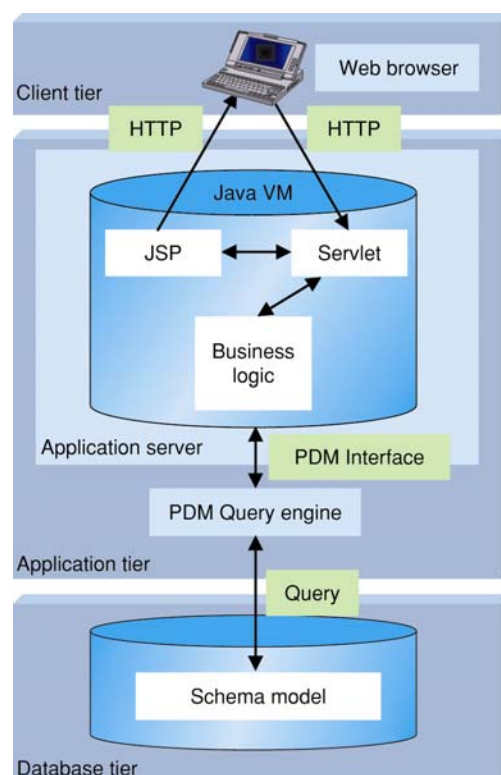


Fig. 5 Logical architecture

personnel to provide support during times the system is down or being upgraded. This causes a time lag in trouble shooting, program maintenance, and updates.

(2) Networks

There are many communication control units (hereafter referred to as routers) interposed on the networks between Japan and remote locations such as overseas offices. They are required in order to carry on communication, but when communication data is congested in the routers, delays are caused in proportion to the number of routers and their processing capacity.

A server-side processing approach (JSP/Servlet type)^{(6), (7)} was adopted for the new system in order to overcome the above issues. Compared to processing on the client (applet type), this limits consumption of both loads on the client and the network while assuring good response.

Elements of network load under the new system include not only transmission volume (traffic volume) but also the number of transmission times (number of packets). System development took these factors into account. Fig. 6 shows the number of transmission times (number of packets) per second for the same process performed on the client and performed on server side.

4.2.5. Program structure

An MVC model was adopted that divides the programs among a business logic layer (Model), screen display layer (View), and control layer (Controller).

The database access, data processing, and other such business logic processes of the system are written into the Model. This is the main body of the system. Screen display and data I/O components are handled by the View. The

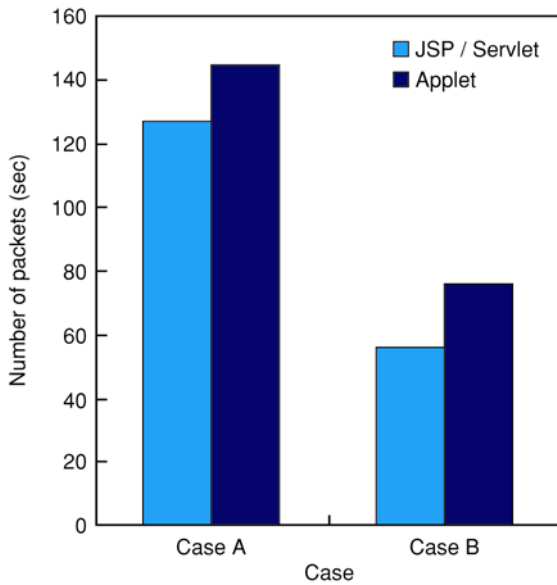


Fig. 6 Difference of the volume of packet transferred in retrieving BOM

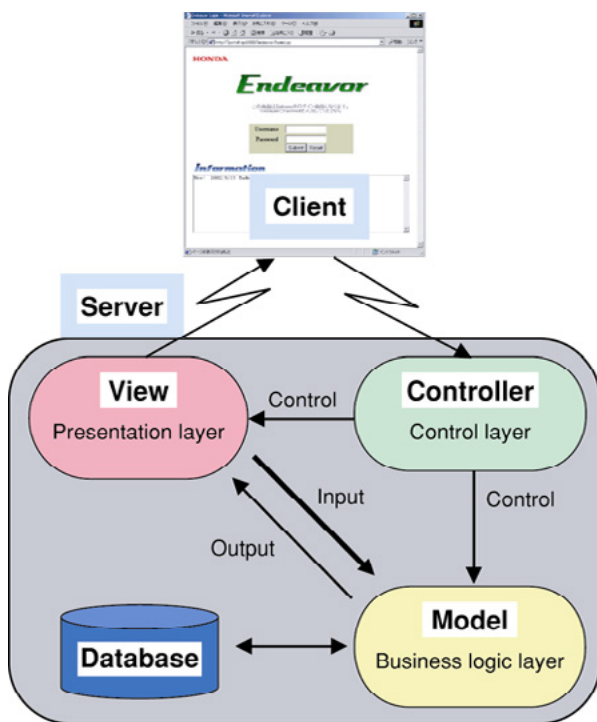


Fig. 7 MVC architecture

Controller controls the View and Model. The controller itself does not execute screen display or logic, but instructs the Model to execute the necessary logic according to the input, then passes the result to the View and causes it to display the result on the screen (Fig. 7).

Designing the system in accordance with this model

clarifies the separation of individual functions, and assures their autonomy. Having the functions clearly separated during development makes it possible to apportion the development work for each function. At the same time, this makes it possible to limit the sharing between layers to a minimum, so that the program structure can be made less susceptible to the influence of changes in other components. This enables a flexible development, so that when the database design is going to be changed, only the business logic needs to be altered and the database can be redesigned repeatedly in order to create further programs.

4.2.6. Support for multiple languages

When conventional methods were used to provide support for multiple languages, screens and systems were created in each of the languages. This reduced system maintenance efficiency because every screen required maintenance whenever the system was modified.

The new system avoids such issues by introducing program component groups that can support multiple languages. These program component groups can be used in combination, thus enabling programming with uniform quality and without dependency on the level of the developers. In the course of plotting out a number of program component groups, it was determined that superior system maintainability was afforded by Struts (a program component group for Web application development that was developed by Jakarta Project, a standardization organization)^{(8),(9)}, which was capable of display control and integration with the MVC model. This Struts framework manages display messages for each language, and it provides a method for switching between languages as necessary.

The use of Struts yielded the following four advantages:

- Messages are centrally managed in definition files, so that the text styles for every screen can be made uniform. It is not necessary to alter programs even when the display messages are changed.
- New languages can be supported by simply adding definition files for the messages translated into those languages.
- Conventional JSP tended to grow complex because HTML files included Java language. However, this could be organized by using the tags (screen program components) provided by Struts, thus enabling reduction of development person-hours.
- Screen transition information is managed centrally in XML files. This eliminated the need to create screen transition control processes in the business logic layer, thus improving program productivity and quality.

4.2.7. Load balancing and fault tolerance

The new system was given load balancing and fault tolerance (cluster configuration) capabilities by utilizing application servers (Fig. 8).

The system handles processing requests from multiple clients by tracking the load conditions and distributing the processing requests among application servers where the load is lower. This load balancing function realizes a uniform response to clients even when processing requests are concentrated together (Fig. 9).

The system was given fault tolerance by connecting multiple application servers together so that when a fault occurs in an application server, the system could switch to another application server that is connected. This enables system operation that does not need to shut down service to users even when faults occur in the application servers.

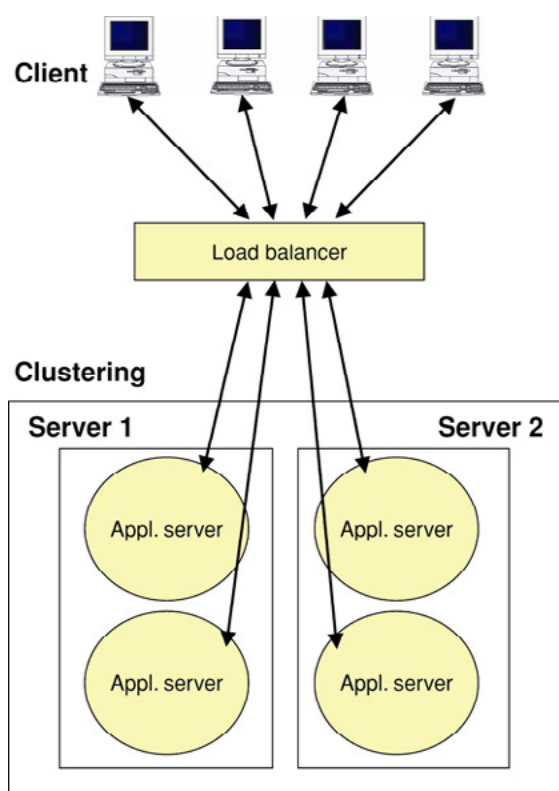


Fig. 8 Load balancing and clustering

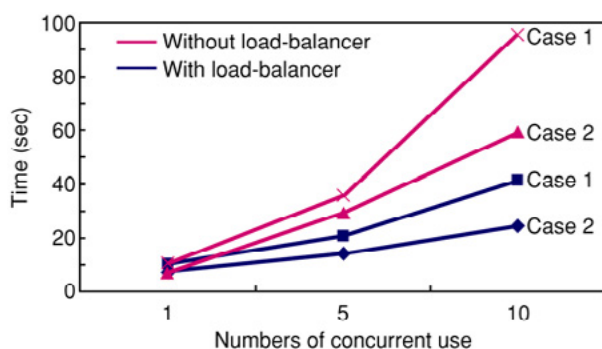


Fig. 9 Load balanced performance

5. Conclusions

The new system was developed as an effective system for the cost management of new models, and it has the features and yields the effects described below:

(1) Features

- Flexibility to handle the future addition of management items and increases in data
- Scalability to ease deployment overseas, support multiple language environments, and reduce network load

(2) Effects

- Enabled sharing of supplier information and management of the single unit cost of a model for individual development events, including models produced overseas
- Enabled clearly understandable look-up of costs and supply patterns for individual locally-procured parts used in models produced overseas
- Reduced the time required to calculate the single unit cost for a new model to two-thirds of the conventional method

Due to the above, the new system is expected to be utilized at offices not only in Japan but also overseas.

6. Future Developments

A dedicated user screen (portal screen) is going to be offered in the future. This will offer the information required by each division in an organized form, so that users can more efficiently track transitions in technical specifications and cost information. As deployment at overseas offices proceeds, overseas requirements will be incorporated into the system in order to enhance the system.

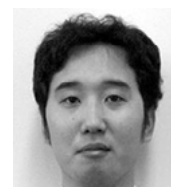
References

- (1) Yamamoto, M.: Development of Engineering BOM Data Management System, HONDA R&D Technical Review, Vol. 14, No. 1, p. 77-84 (2002)
- (2) Martin, J.: Information Engineering, Book I-III, TOPPAN (1991-1994)
- (3) Client/Server Database Design Techniques, Soft Research Center Inc. (1993)
- (4) T-Form ER-Diagram Database Design Techniques, Soft Research Center Inc. (1998)
- (5) Pree, W.: Design Patterns for Object-Oriented Software, TOPPAN PRINTING CO., LTD., p. 75-78 (1996)
- (6) Singh, I.: Designing Enterprise Applications with Java 2 Platform, Enterprise Edition, Second Edition, Sun Microsystems, p. 79-180 (2002)
- (7) Harada, Y.: Java Server-Side Programming, Gijutsu-Hyohron Co., Ltd., p.164-269 (2000)
- (8) Goodwill, J.: Mastering Jakarta Struts, SHOEISHA, p. 87-156 (2003)
- (9) Cavaness, C.: Programming Jakarta Struts, O'Reilly, p. 104-193 (2002)

■ Authors ■



Seiji TSUCHIYA



Katsunobu YAMAMOTO



Masako WATANABE



Takashi ARAMAKI