Adaptation of Frameworks for Development of Public Office Systems

● Hitoshi Inoue ● Hiroki Nakanishi

(Manuscript received March 28, 2006)

Conventionally, the development of information systems in Japanese public offices has been based on mainframe computers. However, the development of information systems in public offices has recently mainly been Web-based in order to reduce system lifecycle costs, expand public services, and create a ubiquitous society. Web applications for public office must be developed quickly, at low-cost, and to a high quality. One approach for meeting these requirements is the use of frameworks. This paper shows how Fujitsu reduces the cost and development time of Web applications for public services by adapting a framework to enhance the functions of Interstage Application Framework Suite, which is a framework product that conforms to J2EE.

1. Introduction

The development of information systems catering for public offices in Japan to date has pursued the formation of mainframe-based systems. Recent years have witnessed a growing trend of downsizing systems to reduce lifecycle costs. In response to the expanding portfolio of public services, for example, e-Applications, e-Notifications, and the growth of ubiquitous computing, Web system development has become the mainstream approach to system development.

Experience gained from the development of the mainframe-based systems now dominant in Japanese public offices, however, will not ensure higher productivity and better quality when it comes to developing Web systems. Another obstacle is the chronic shortage of Java engineers and Web applications developers.

This paper reviews the implementation of frameworks that have been built to develop a large business system for use in a Japanese public office in order to increase productivity and quality based on the concept of SDAS.

2. System development requirements and design philosophy

The public office mentioned above requested a system that would computerize their paper-based applications and notifications. To ease the burden on their personnel, they specified certain requirements so the system could collaborate with existing backend systems managed by a mainframe or other large-scale computer. The requirements were as follows:

1) High quality
   This is a prerequisite when large sums of money are transferred automatically.
2) Availability of many screens of similar appearance.
3) High maintainability to enable modifications in response to legal reforms or other moves associated with the system's characteristics.
4) A three-layer system architecture consisting of a Web server, application server (AP server), and database server to meet security requirements.

To meet these requirements, the developers

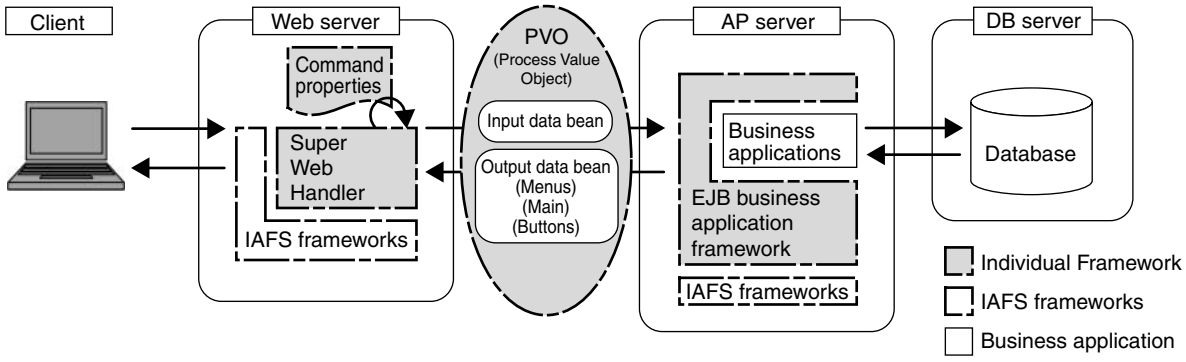

Figure 1
Outline of framework.

faced the task of efficiently designing a sophisticated system solution and completing the overall operational testing in only 18 months. The developers assumed the use of Interstage Application Framework Suite (IAFS),[1] which is a product oriented for the Japanese market and includes Interstage Application Server,[2] which uses a framework called Web Front Framework.[note] Based on this assumption, the following design goals were established for the basic Individual Framework:

1) Broaden the scope of the framework provided by IAFS.
2) Among the three layers, avoid the need to develop applications that will run on the Web layer.
3) Group the functions that have been individually developed by IAFS to make them easier for the interface developers to use and maintain.
4) Enable flexible control of multi-frame screens.

note) Web Front Framework is a framework that makes development and maintenance more efficient by separating screens, business logic, and data. It also improves the security of Web applications. Details are given in the paper, "Promotion of Development Efficiency by Using Frameworks," elsewhere in this special issue.

3. Individual Framework functions

This section summarizes information about the framework that was developed according to the design philosophy outlined above. The frameworks are functionally organized into the following blocks (**Figure 1**):

1) Super Web Handler

A typical three-layer system is built by configuring business applications on an application server. However, with applications (Web handlers) dedicated to making attribute checks on input data and exercising screen controls implemented on the Web server, the work entails numerous screens of similar appearance, so high-quality work is mandatory. Super Web Handler is a functionally enhanced version of IAFS Web Handler, designed to suppress increases in the amount of development per screen and quality variations from screen to screen.

Instead of implementing screen-specific Web handlers, Super Web Handler focuses on the creation of attribute check descriptions and other descriptions to provide the following functions (**Figure 2**):

- Attribute checking
- Moving information from DataBean to Enterprise JavaBeans (EJB) business applications

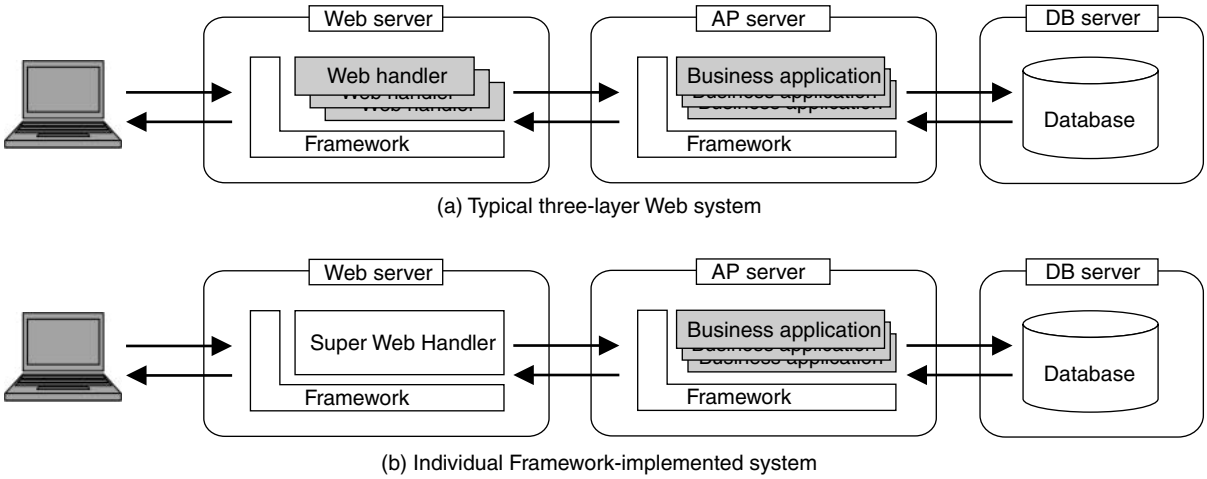

Figure 2
Super Web Handler.

- Invoking EJB business applications
- Screen controls

2) EJB business application framework

This framework provides a repertoire of superclasses necessary to build business applications and common components that are used in the creation of any business system. It also has the following functions:

- Connects to databases and disconnects from them
- Commits and rolls back databases
- Activates and enables logs

3) Process Value Object (PVO) interface

Because value objects are used to enable a Web server and application server to collaborate with each other, PVO has been developed as an interface between Super Web Handler and the EJB business application framework in the present context of Individual Framework.

4. Touring functions implemented

This section describes the system implementations of the framework functions.

1) Screen transitions

Because the system's menus, buttons, and other controls are laid out in fixed positions for greater handling ease, several screen frame patterns are used (**Figure 3**). To support screen transitions to these patterns, code that specifies the layout of each pattern is embedded in a property file implemented in Super Web Handler. This enables the specification of split-frame patterns at screen transitions.

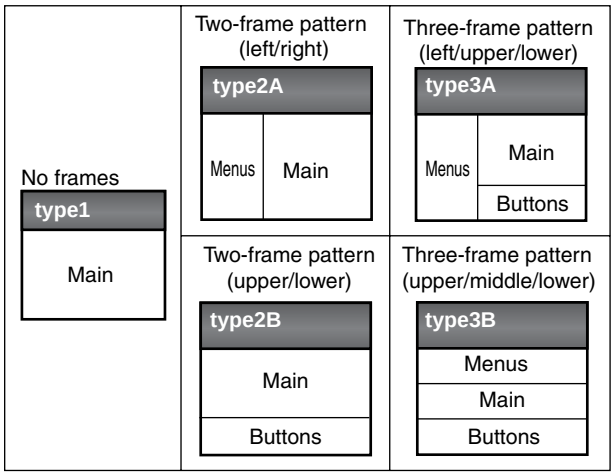

Figure 3
Frame patterns.

2) Simultaneous multi-frame updating

Once a multi-frame screen is designed, its frames must be updated simultaneously to synchronize their contents. The framework is implemented in the output data bean on the PVO interface to allow simultaneous updating of

multiple frames (**Figure 4**).

IAFS does not support simultaneous updating of multiple screens that contain PDF documents or eForms. However, it has been functionally enhanced to allow the framework to be easily implemented in the output data bean on the PVO interface while still allowing simultaneous updating of multiple screens. Application developers do not need to be aware of this process.

3) Screen transition information history management

The framework supports screen history management, which saves the PVO at screen transitions and generates screens from the saved PVO as needed. Because data beans of multiple frames are stored in the PVO, history screens can also be invoked on a multi-frame screen by working in the same way as for a single-frame screen.

4) Development/testing environment simplicity

If the members of a Web system application development run their shares of a development activity or test simultaneously, a significant impact can be anticipated. Therefore, individual development environments should be used. The development of a three-layer system in particular requires one Web server and one application server to be installed in each development environment. Keeping multiple development environments within a single server, however, would not simplify development and testing very much.

To overcome this problem, the framework makes simple modifications to the development processes so that business applications are deployed in the Web server (instead of an application server) to streamline the workflow for creating a system development environment. With these modifications, the name of an EJB application and the name of an action class invoked into a data bean are embedded in a framework-specific property file (commands.properties). When the Super Web Handler references the property file, it automatically invokes the business application associated with the data bean. This means that users simply specify the name of an EJB application as entered in the property file and allow Super Web Handler to automatically invoke the associated business application from the Web server. Consequently, the Web server is

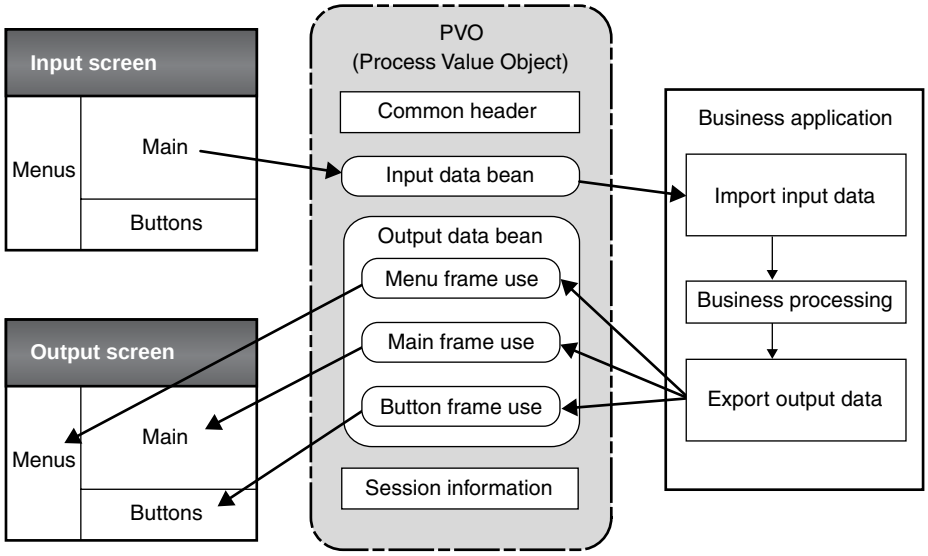

Figure 4
Simultaneous multi-frame update.

all that is necessary to develop and test an application. The result is greater simplicity in the development environment and elsewhere (Figure 1).

5. Implementation benefits

This framework implementation of the functions outlined above offers the following benefits:

1) Higher productivity

Use of the framework has trimmed the scale of system development by about 30% compared with the pre-implementation days of isolated development activities. This enables a big improvement in productivity.

2) Simplified workflow for creation of system development environment

By using the framework to combine a Web server, application server, and database server into a single three-layer system and make them collaborate with each other in the same development environment, the workflow of development and testing is streamlined.

3) Improved maintainability

The framework consolidates isolated development tasks into a single task to enable flexible responses to evolving changes (e.g., changes made to specifications after reviews), making it easier to comply with scheduled delivery dates.

6. Conclusion

This paper discussed how a framework designed to make up for the shortage of development engineers can be used to speed up and streamline the workflow for Web application development for a large-scale business system and improve application maintainability.

As the framework is developed and implemented to aid the development of a business system, it needs to be tailored to suit the background and expected deliverables of the system being developed. The framework is also designed to comfortably fit into the development of other kinds of systems. Further upgrades will be explored through the upcoming operational phases to standardize the merger of IAFS with this framework. These upgrades will therefore support the development of more systems in the future.

References

1) Fujitsu: Interstage Application Framework Suite. (in Japanese). *http://interstage.fujitsu.com/jp/framework/index.html*
2) Fujitsu: Interstage Application Server. *http://www.fujitsu.com/global/services/software/interstage/products/apserver/*

Hitoshi Inoue, *Fujitsu Ltd.*
Mr. Inoue received the B.S. degree in Mathematics from Sophia University, Tokyo, Japan in 1993. He joined Fujitsu Ltd., Tokyo, Japan in 1993, where he has been engaged in IT system development and its maintenance activities for the Japanese Government.

Hiroki Nakanishi, *Fujitsu Ltd.*
Mr. Nakanishi received the B. Com. degree in Commerce and Management from Hitotsubashi University, Tokyo, Japan in 2000. He joined Fujitsu Ltd., Tokyo, Japan in 2000, where he has been engaged in IT system development and its maintenance activities for the Japanese Government.