

# Utilizing SDAS to Implement Financial Solutions

● Hiroshi Ojima ● Seigo Kuroda

*(Manuscript received April 18, 2006)*

Fujitsu's customers in the financial industry have restructured and expanded their systems. Now, these customers not only need the reliability and quality they enjoyed in the past but also need this reliability and quality much more quickly. Especially these days, it is necessary to use the latest technology and development methods to keep up with the constant changes in the business environment and IT techniques. To achieve this, the previously used development methods are becoming inadequate, and it is now necessary to establish and develop financial solutions that adopt SDAS. In this paper, we first explain the requirements of the solution architecture for the leasing industry and the design concept of a solution for lease system construction that covers both the business side and system side. Next, we introduce an approach that covers the design concepts on the system side. By repeatedly using solutions that adopt SDAS, we have achieved high-quality, short-term system development.

## 1. Introduction

Customers in the financial industry have increasingly sought to rebuild their huge, critical mainframe systems and construct new systems in an open environment. Further, these customers desire swifter development than ever, without risking the reliability and quality that have already been established. Financial solutions designed to fulfill these needs blend the development technologies provided by SDAS with working expertise and project management concepts.

In 1999, lease system construction solutions based on SDAS, which builds on AA/BRMODELLING,<sup>note 1)</sup> were developed to target

note 1) Abbreviation of Application Architecture/Business Rules Modeling. A technique that defines the notation and semantics of design documentation (modeling) on the basis of development methodology, focusing on data-oriented modeling techniques. The formats defined by Application Architecture/Business Rules Modeling are supported by the SDAS integrated CASE tool, AA/BRMODELLER.

the leasing business segment of the financial industry in Japan. On the basis of these solutions, critical systems for numerous leading leasing firms have been reconstructed in recycling short-term development processes. Lease system construction solutions are classified into two broad types — business or system — according to their design philosophy.

This paper identifies what is required of a system of industry-specific solutions and defines the two kinds of design philosophies mentioned above. It then looks at the concepts for implementing the system design philosophy of frameworks; namely, the organization and standardization of application structures, concealment of platforms, approach to achieving enhanced performance, approach to achieving Web system implementation through recycling, and benefits of recycling industry-specific solutions.

## 2. Requirements for industry-specific solutions and design philosophy of lease system construction solutions

This section outlines what is required of a system of industry-specific solutions and introduces the two types of design philosophy pertaining to lease system construction solutions.

### 2.1 Requirements for system of industry-specific solutions

Any system of industry-specific solutions should fulfill the following requirements:

- 1) Creation of business models based on industry-specific expertise

Business models visualize the workflow of business practice in system implementation and define the basic data structure of the business data required.

- 2) Creation of SDAS standard frameworks augmented with industry-unique characteristics

Each framework delineates a development environment that encompasses a set of components (e.g., communications control, screen control, and business specifications) of those functions that are operable in that environment.

- 3) Standardization of development techniques, environments, and tools

Development techniques, from requirements definition to testing, are standardized to iron out skill gaps among developers.

- 4) Creation of development standards, project management standards, and tools

Documents are organized and standardized to ensure a uniform quality of documentation. Further, the workflow of documentation and review management is visualized by the use of tools.

Once industry-specific solutions are developed and deployed, there are two important requirements. First, they must be put to cyclic use in the subsequent implementation of projects, with cutting-edge technologies and development techniques occasionally factored in. Second, for enhanced development productivity and quality,

information derived from their use must be fed back to help build a development organization that has know-how about creating and blending components on the basis of a knowledge of frameworks.

### 2.2 Design philosophy of lease system construction solutions

Lease system construction solutions are classified into two broad types — business or system — according to their design philosophy:

- 1) Business design philosophy: modeling the leasing business

The leasing business deals in a wide assortment of items and is open to complex changes in the status of the deals during the lease periods, for example, modifications to contract terms and cancellations. Because the terms of payment for items and fixed assets and the terms of lease collection are also subject to change in synchronization with the changing status of deals, a firmly organized system collaboration of various individual functions needs to be formed. To fulfill these complex requirements, the basic workflow of the leasing business, from commodity trading to paperwork handling, can be represented in fully integrated models (business, data, and logic models) to identify customer requirements and define the areas of addition and modification that are required.

- 2) System design philosophy: separation of business specifications and controls

The functional roles, from applications down to platforms, are broken down into 10 layers to separate the business specifications and computer controls. These layers are classified by the two major blocks of applications and platforms and the intervening software components (**Figure 1**).

The development of an application that builds on a leasing business model would encourage higher ratios of business and data component recycling. This would facilitate addition and modification to business and data components over an organized/standardized interface, even if

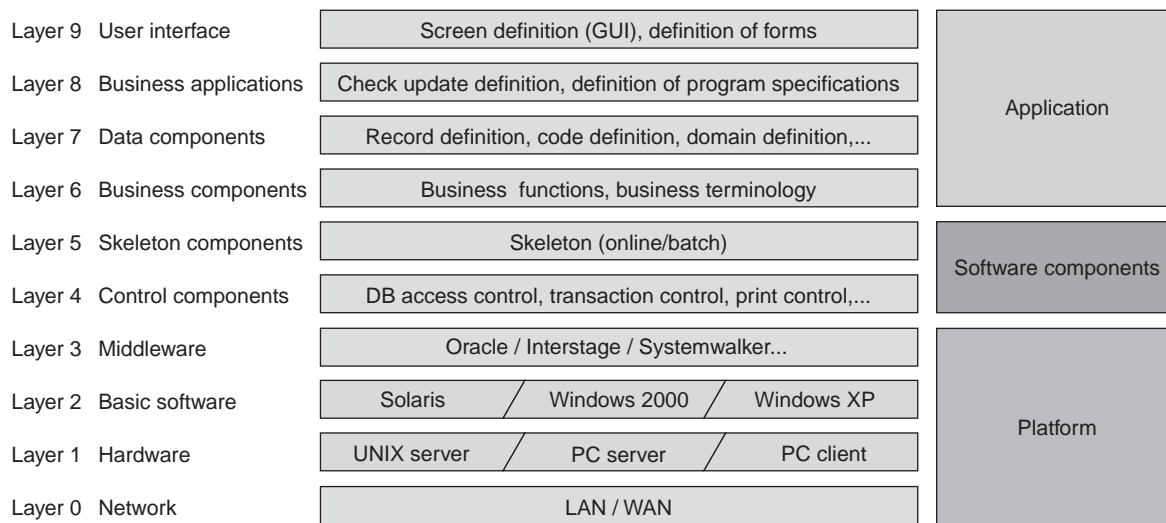


Figure 1  
Separation of business specifications and computer control.

there are no existing components. The interfaces between the applications and platforms (hardware and middleware) concealed and separated by components also promise additional increases in the ratios of business and data component recycling.

### 3. Organization and standardization of application structures

By recycling an application structure compatible with a given application under development over an organized/standardized interface, we can reduce design and construction lead-times and therefore prevent degradation of the quality of control logic. Further, the level of description of design documents can be enhanced by organizing and standardizing them in an organized/standardized application structure.

Applications fall into three broad groups: online programs, batch programs, and subprograms. Among these, online and batch programs have their program structures organized into 22 standardized skeletons<sup>note 2)</sup> (e.g., simple editing, distribution, summary, matching, database table editing, and form editing) according to their modes of processing.

These skeletons have components for transaction control, DB access control, error processing, log processing, message processing, and recovery processing assembled into them.

Online program skeletons come with an additional repertoire of components (screen control, event control, item control, and message control). For subprograms, components such as business, data, and common components are provided as skeletons to be created over a standard interface.

To ensure efficient online/batch/subprogram skeleton utilization, the standards of program blocking and each set of skeleton specifications are defined at the start of system structure design (SS) phase and then described in design documents. By using these design documents, programmers can economize on the program-writing period and still maintain control of the processing quality.

To realize such application structures, an SDAS integrated CASE tool called AA/BRMODELLER is used to provide the function implementations of skeletons and control compo-

note 2) Prototypes tailored to the individual sets of business specifications, which are used to identify business patterns with reference to the business workflow.

nents so as to separate the business specifications and computer controls and conceal them.

#### 4. Concealment of platforms

Hardware and software are renewed at such a swift pace in the modern development environment that in order to use applications over many generations, they must be kept ready to accommodate changes as they occur. To achieve this, the platform interface must be concealed from applications.

Although application structures are organized and standardized to conceal platforms from the view of skeletons and control components, the developers of batch program tests still need to be platform-conscious because the shells that control execution use different operating systems (e.g., Windows or Solaris).

As a solution to this problem, file assignments and program runs are coded using Job Control Language (JCL) as in the mainframe era.

For example, “Sort” can be used to designate a sort utility and “Delete” can be used to signify a process for deleting a file. OS-dependent contexts, for example, branches, are allowed to expand automatically to achieve the concealment of platforms (**Figure 2**).

In addition to the concealment of platforms as an automatic development function of the shell script (hereafter called a job) that controls execution, the following factors have also been taken into account:

- 1) Allowance for rerunning jobs.
- 2) Automatic judgment of return codes generated by commands such as programs and utilities.
- 3) Ability to automatically delete unnecessary files at the end of a job.
- 4) Readiness of jobs used in unit testing to pass as operational jobs.

Once jobs are completed, the next step is to register them with a scheduler so they can be

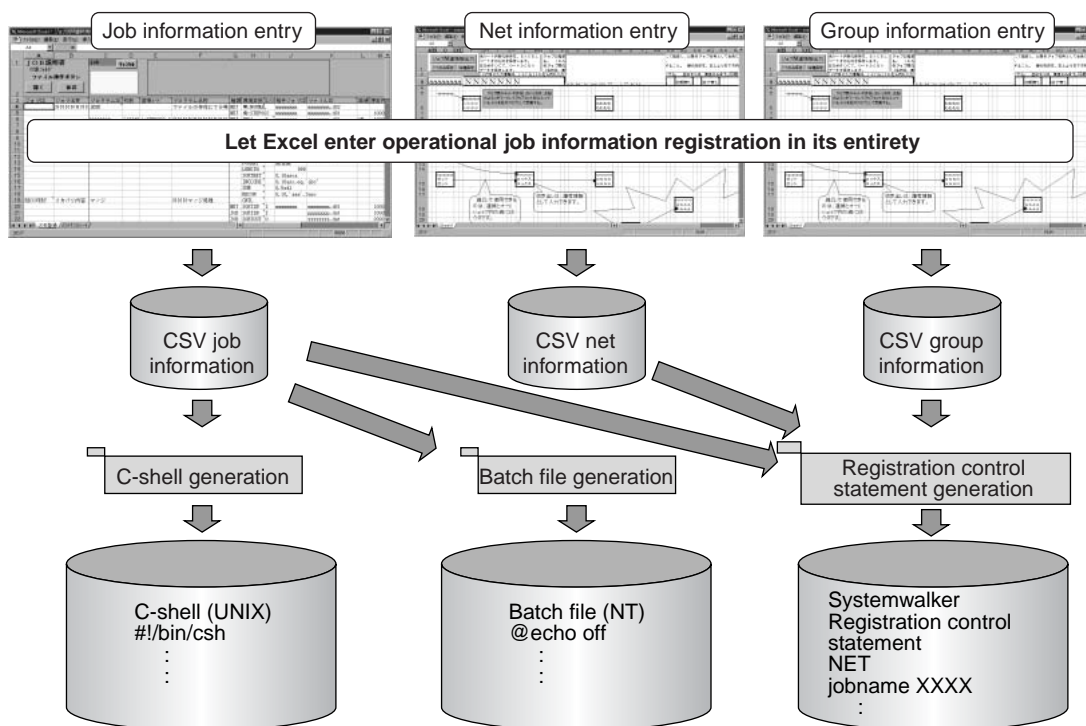


Figure 2  
Registration of operation jobs without considering platform.

launched in sequence. However, taking operation into consideration, procedures such as library management and change management are necessary. Therefore, because individual developers cannot directly operate the scheduler, an easy-to-use tool for entering scheduler information is provided.

When entering job net information, cells (preceding and succeeding job names) are linked with lines and connected to provide a visual definition of the job relationships. This definition is extracted and generated as a text and then registered with a scheduler.

The tool's scheduler registration functions enable the following:

- 1) Managing job relationship information in a library,
- 2) enabling data entry without using a server (scheduler),
- 3) allowing job relationships to be coded in a visual format, and
- 4) checking information (such as job loops) before it is registered with the scheduler.

This tool is presently used in conjunction with the scheduler Systemwalker OperationMGR.

## 5. Approach to achieving enhanced performance

Performance glitches account for one aspect of degraded quality. They are often uncovered in a system test (ST) phase in which large volumes of data are used. Because it is difficult to review applications at this stage, performance glitches should be identified in their early stages of occurrence.

- 1) Evaluate all SQL statements

Performance glitches are primarily traceable to SQL statements: one of the most likely causes being real access to data (or access to physical pages). For example, a process that used to take 10s of minutes before might take only several seconds after tuning because the tuning eliminated many physical accesses. To avoid performance glitches resulting from SQL statements, direct

calls to SQL from business applications have been totally prohibited, with database accesses being implemented using the access components provided. Skilled managers have created and managed access components and evaluated all the SQL statements that have been filed by the applications developers.

- 2) Collect response throughput information automatically

A scheme for automatically collecting performance information has been constructed into skeletons to automatically collect and build performance information from the integration testing stage onwards. Under this scheme, performance information is iteratively fed back to the business developers in each test cycle to support performance enhancement. Final system tests/operational tests subsequently performed on the cumulative performance information have confirmed that this scheme enhances performance.

- 3) Collect Oracle trace information about specific transactions

Because a server online application has each process communicating simultaneously with multiple clients, trace information about a selected transaction is not collectable until the ongoing process is terminated. Even if trace information about a transaction is collected, transactions from multiple clients would be intermixed in the trace information, making it difficult to analyze a particular range of trace information. This difficulty can be avoided by collecting information about a given transaction when no other process is being executed or collecting information using a dedicated environment.

As a solution, if a reference to information (e.g., a login ID or program ID) in a transaction control component identifies a target transaction, trace collection start and end instructions are issued to collect transaction-specific Oracle trace information. As a result, problem SQL statements can be identified faster, thereby trimming the survey time.

## 6. Approach to achieving Web system implementation through recycling

Lease system construction solutions have evolved from client-server system solutions into Web system implementations with the support of the growing popularity of the Internet. This section introduces key aspects of the approach to enabling Web system implementation of lease system construction solutions established on a client-server system basis so they can be recycled more easily.

The most important goal is to maximize recycling of the application servers of lease system construction solutions solely by constructing frameworks on the Web server. The second is to restrict business application developers to a single language.

While Web systems are operable on either a Servlet or operability-oriented Applet basis, we constructed a Servlet lease framework for the sake

of the development lead-time and business requirements (Figure 3).

With the Servlet framework, business application developers simply create HTML as a screen design and attribute information about screen I/O record definitions related to the Web server. When the designs and information are input to the tool, it automatically generates JavaServer Pages (JSP) for each screen. The newly developed Servlet framework controls the screen-processing operations (e.g., basic attribute checking, I/O item editing, length checking, and common code pull-down), which eliminates the need to code their business logic.

Dynamic information I/O is accomplished by specifying the information required for the Servlet framework from an application.

Consequently, because the Servlet lease framework comes with a complete set of implemented screen controls, it removes the need for business application developers to develop them

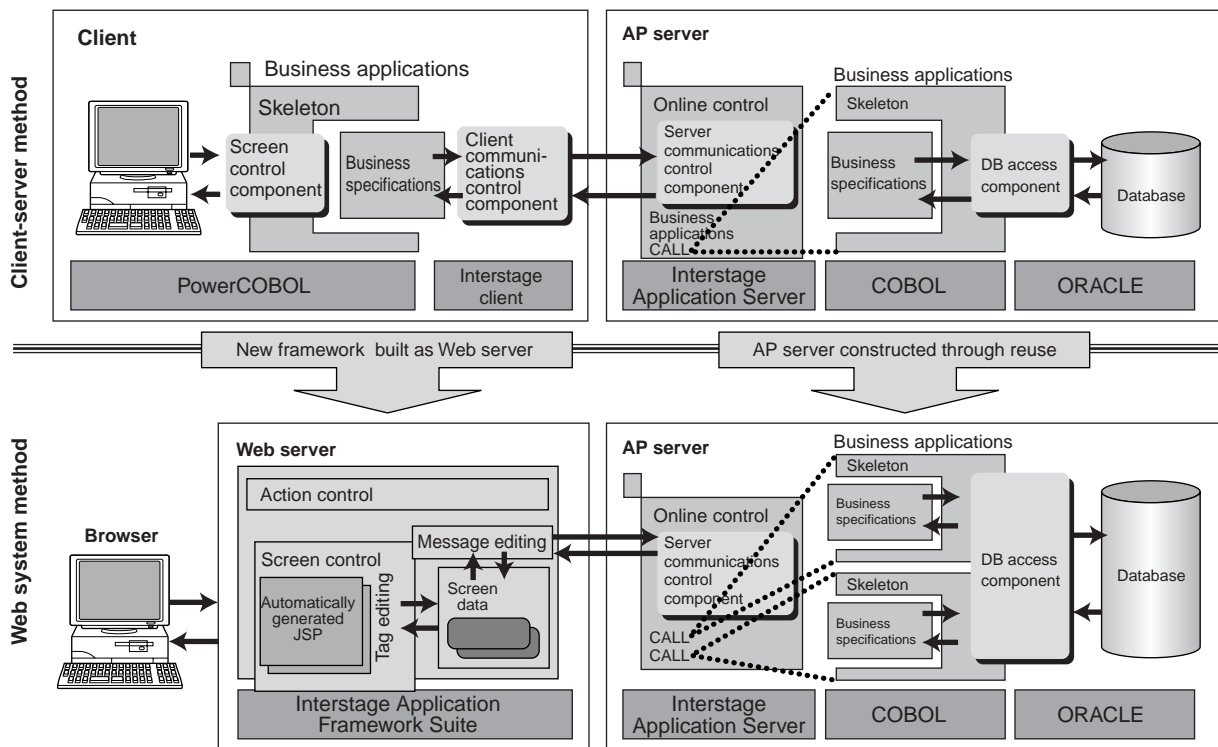


Figure 3 Lease framework of Servlet version.

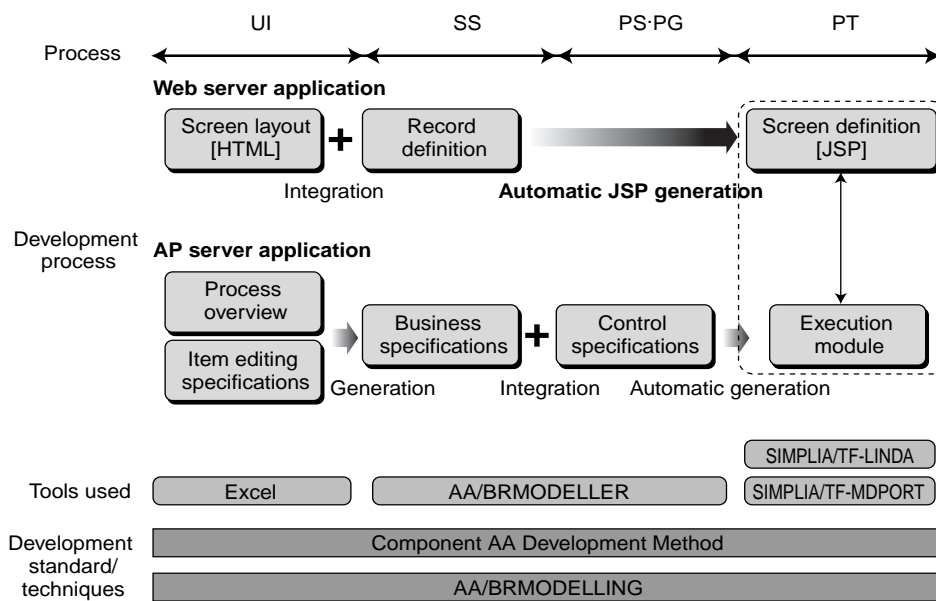


Figure 4 JSP automatic generation of Servlet version.

in Java or any other language (Figure 4).

At present, an Applet lease framework is being constructed based on a similar design philosophy.

## 7. Benefits of reusing industry-specific solutions

This section introduces the benefits of using established industry-specific solutions in the implementation of a lease project.

From a business standpoint, lease system construction solutions are recycled to implement "fit/gap analysis"<sup>note 3)</sup> based on models of the leasing business. Through this analysis, requirements are identified and customer needs are translated into specifications to achieve an overall reduction of mistakes and omissions.

From a system standpoint, application structures are organized and standardized and then

concealed by the platforms so that business and data components can be easily recycled in conjunction with the solutions. When using industry-specific solutions, it is also important to maintain a permanent staff of technicians having know-how of leased system construction solutions, including expertise in their recycling. Component usage and continued staff presence are considered extremely instrumental in trimming development lead-times and assuring enhanced quality.

This paper focused on the issues of system frameworks, but a comprehensive scheme of project implementation — inclusive of progress management, quality control, error recovery, specification change management, problem and task management, and configuration management — is a vital requirement for trimming development lead-times and enhancing quality.

In one of our leasing projects, by recycling lease system construction solutions, we reduced the development lead-time by 30 to 50% in relation to development scale (Figure 5).

note 3) This analysis compares the functions provided by the ERP package and resolves mistakes and omissions based on a lease business model.

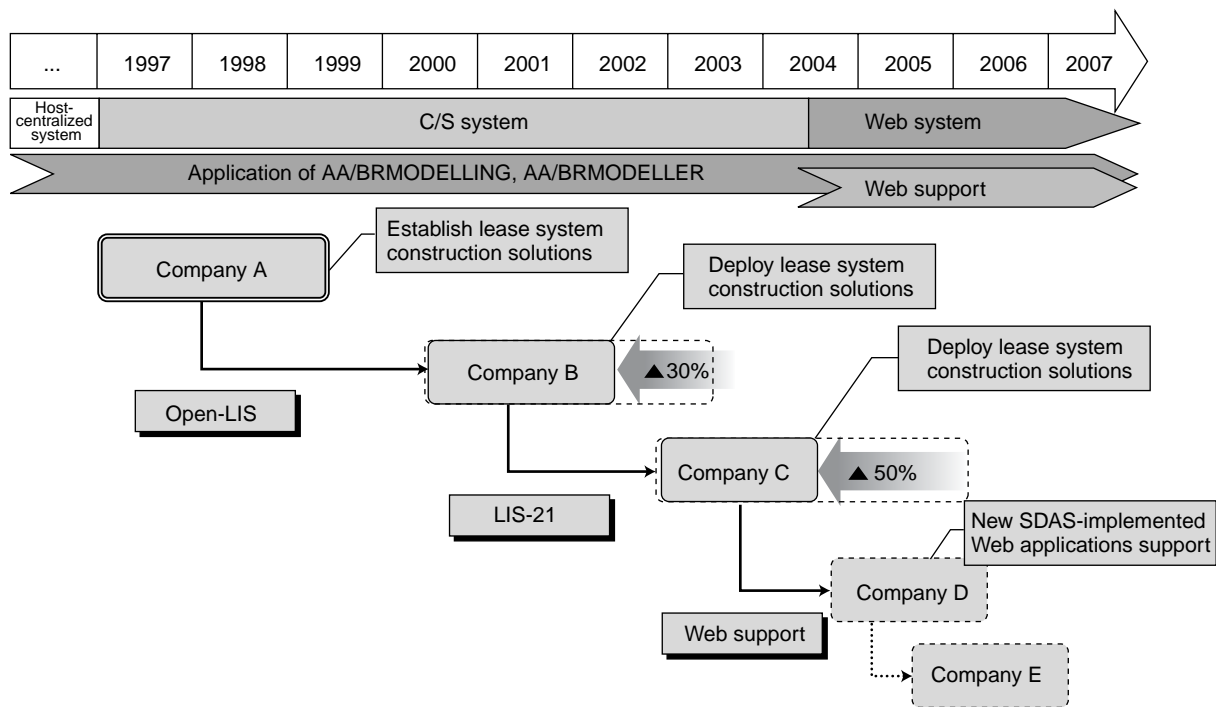


Figure 5  
Expansion of lease system construction solution.

## 8. Conclusions

This paper introduced how the recycling of leasing business solutions designed to serve the financial industry has led to shorter development lead-times and enhanced quality. For a project to successfully yield these benefits, development technology alone is insufficient. Facilitating problem determination and remedial action is a major impetus to trimming lead-times and enhancing quality, and this can be achieved by efficiently controlling the execution of a project through rules and tools that visualize the workflow.

A comprehensive financial services portfolio is needed to keep pace with the growing diversity of interdisciplinary merchandise and business lines in the financial industry. This need can be met by developing industry-specific solutions that address the need to transform financial business models in the wake of a broadening range of channels and openly collaborate with other industry segments.

Fujitsu has been working in Japan to refurbish its package of next-generation financial

solutions that address the changing Japanese financial business climate, focusing on the component implementation of services, collaboration between channels and services that builds on Service-Oriented Architecture (SOA), and standardized interfaces.



**Hiroshi Ojima, Fujitsu Ltd.**  
Mr. Ojima received the B.S. degree in Mathematical Sciences from Osaka Prefecture University, Osaka, Japan in 1981. Later that year he joined Fujitsu Ltd., Tokyo, Japan, where he has been engaged in the development of financial systems.

E-mail: ojima.hiroshi@jp.fujitsu.com



**Seigo Kuroda, Fujitsu Ltd.**  
Mr. Kuroda graduated from Hyogo Prefectural Aioi Industrial Senior High School, Aioi, Japan in 1979. Later that year he joined Fujitsu Ltd., Tokyo, Japan, where he has been engaged in the development of lease systems since 1987.

E-mail: s.kuroda@jp.fujitsu.com