Improvement of Software Quality and Productivity Using Development Tools

● Hideo Abotani ● Tomoki Shiratori ● Kouji Sasaki
● Masaki Tonomura

(Manuscript received March 24, 2006)

Information systems, which successively meet the requirements of on-site management and staff, have become enormous and complicated after many years of modification. To make IT investments efficient, it is essential to reuse current software assets effectively, quickly develop systems, and reduce testing costs. To help customers achieve these goals, Fujitsu provides a systematic development environment called the System Development Architecture & Support facilities (SDAS). This environment covers the entire lifecycle of application development. This paper introduces some systematic development tools for SDAS-based Web application development. Specifically, it introduces Interstage Apworks, which is an integrated development environment based on open standards; the SIMPLIA series of testing support tools; the PROSPECS reverse engineering tool; and the NetCOBOL multi-platform COBOL compiler.

1. Introduction

In recent years, in order to rapidly solve the problems of top management and on-site staff, it has become necessary to apply modifications to enormously large and complicated information systems. Also, it is becoming necessary to reduce the volume of development and test work-hours, by assessing methods of implementing additional features on the new optimized systems and the extent of modification that are required.

Moreover, tools are needed that increase software quality and productivity, for example, by enabling effective reuse of current resources, speeding up system development through small-scale projects, and increasing efficiency in document maintenance to keep up with the faster pace of development.

While J2EE-compliant Web applications and the frameworks that offer them are drawing attention due to the acceleration of open systems, a tide of industry standards is also pouring into the area of development tools, as can be seen in the use of open sources represented by Eclipse.[1)]

In this changing environment surrounding development tools, and taking into account the whole life-cycle of development from the design, implementation, testing, and maintenance of an application, providing a systematized range of products supporting a development language and multiple platforms will lead to the improvement of software quality and productivity.

This paper introduces the systematized development tool used in the Web application development by SDAS by introducing the following core products.

1) Integrated development environment Interstage Apworks
2) SIMPLIA series of testing support tools
3) Reverse engineering tool PROSPECS
4) NetCOBOL for various platforms

Note that the products introduced in this paper are all Japanese versions.

2. Interstage Apworks

Due to the shortening of the development period and the growing complexity of systems in recent years, demand for improved productivity using an integrated development environment has grown further. There are two main methods of improving productivity using an integrated development environment.

1) Automatic generation of routine processes

For code that executes routine processes, code with assured testing quality can be reused to reduce the volume of development and test workhours. By automatically generating codes with an integrated development environment, productivity can be improved. The most effective way of use is to standardize the application structure using a framework and automatically generate standardized code based on the structure.

2) Reducing development steps

It is preferable not to spend time and effort on repeatedly executed operations such as building and debugging. A basic requirement for an integrated development environment is to be able to shorten the total work-time by enabling such operations to be executed with fewer steps.

There is also a demand to shorten the period for learning the integrated development environment itself in order to improve productivity. In recent years, the open-source Eclipse is rapidly gaining recognition as a development environment and has already established its position as an industry standard. To shorten the learning period, there is a strong demand to adopt these industry standards and provide users with the operability they are accustomed to. Also, by adopting industry standards, companies become more attractive to third-party software houses.

What is currently required of an integrated development environment is to improve productivity and provide industry standard operability. This section introduces how these requirements are fulfilled in the case of Interstage Apworks,[2] which is the integrated development environment of SDAS. Note that this paper has been written based on Interstage Apworks V7.0.

2.1 Integrated development environment: Interstage Apworks

Interstage Apworks (hereafter called Apworks) is an integrated development environment that adopts the industry standard Eclipse (**Figure 1**).

Based on Eclipse, Apworks provides features for developing J2EE-compliant applications such as Web applications and Enterprise JavaBeans (EJB) as well as Java client applications such as applets (**Figure 2**).

In terms of the development lifecycle, Apworks has the following features for each phase of development.

- Unified Modeling Language (UML) modeling tool for requirements analysis and design
- HTML/JavaServer Pages (JSP) editor, Graphical User Interface (GUI) builder, and electronic form designer for visual editing
- Debugging features of J2EE applications optimized for Interstage Application Server and deployment features for the operation environment
- Development features of COBOL applications (COBOL editor, COBOL debugger)

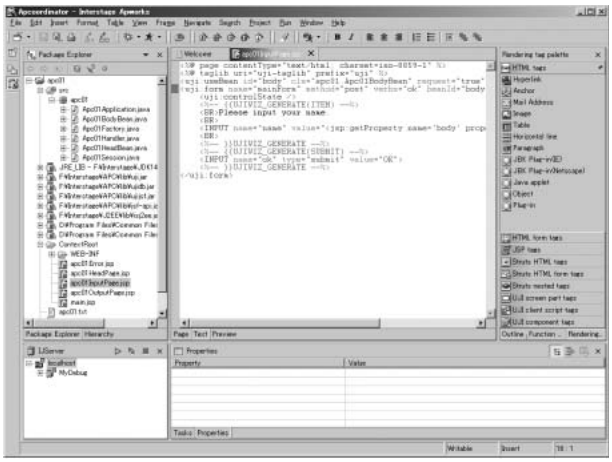

Figure 1
Interstage Apworks.

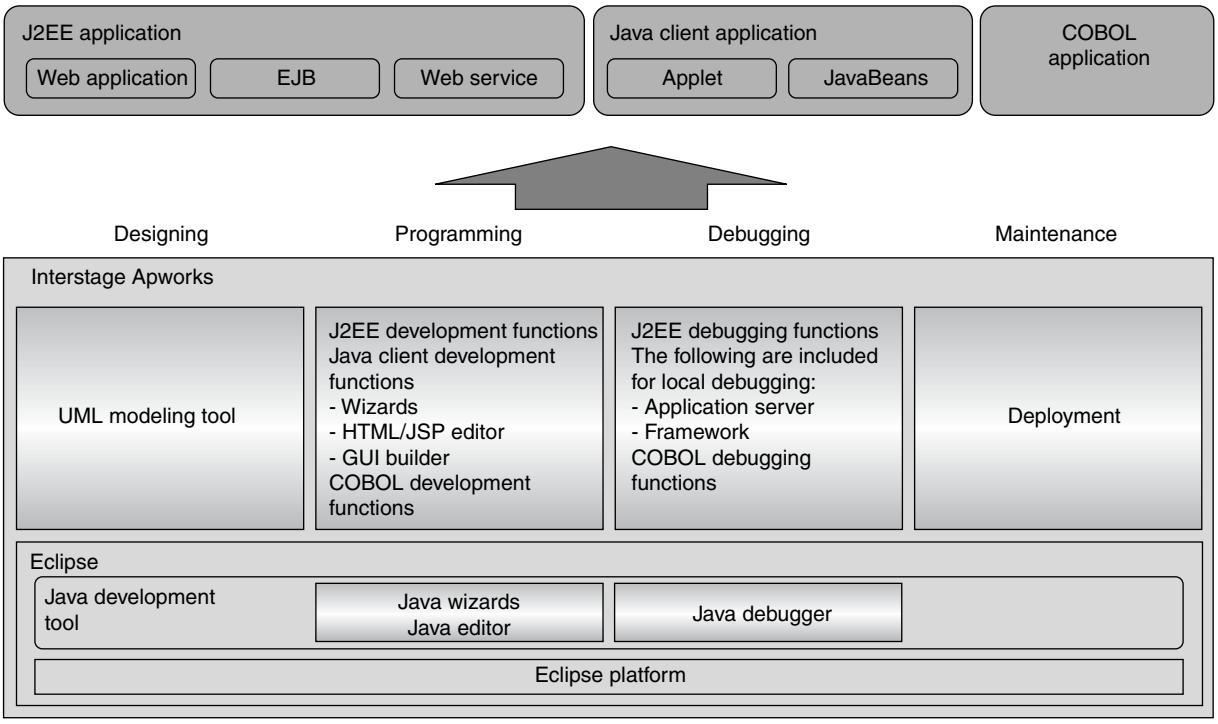

Figure 2
Key features of Interstage Apworks.

Among the various features provided by Apworks, features related to automatic generation and reduction of task moves are introduced here.

1) Wizards

Eclipse comes with a wizard that creates Java packages and class templates. Apworks also provides other wizards that automatically generate the following.

- Templates of Web applications and EJB
- Templates of applets
- HTML files and JSP files
- Various definition files used in J2EE applications

These wizards also support the SDAS framework (Interstage Application Framework Suite), and using this framework further improves the productivity of development.

2) Template

A template is code registered in advance to perform a certain process. With the insertion of a template, code can be embedded at a given position within the source file. By making and using templates of a process that is often executed, the user saves time and work for code description, and the time needed for coding and preventing simple input errors will also be shortened. Eclipse comes with an if statement and a for statement, which are templates that enable easy input of Java syntax. In addition, Apworks provides templates of processes that are frequently used in business, for example, the reference process of EJB and the routine process of COBOL. Users can also register original templates.

3) Efficient debugging

Apworks comes with an application server for debugging. The installation and various settings of this application server are set up automatically during the installation of Apworks. Although the deployment of application assets will become necessary when executing Web applications and EJB, debugging will be performed automatically by the debugger. There is no need for the user to perform any troublesome tasks, and

Web applications and EJB can be debugged just by clicking the debugging button.

Based on the industry standard Eclipse, Apworks enhances the features for improving the efficiency of business application development and aims to provide value as an integrated development environment that matches the requirements at the development site.

2.2 Support for latest open standard

Currently, the Eclipse 3.x series is becoming the predominant version of Eclipse. The next version of Apworks is also planned to be based on the Eclipse 3.x series. By supporting the latest industry standards and ensuring sufficient testing quality, we intend to offer various features to improve productivity and enhance the product so it becomes an integrated development environment that can be used comfortably by users.

3. SIMPLIA series

Since its mainframe times, SDAS has been providing a group of development support tools called the SIMPLIA series[3] by creating and generalizing several tools that are required in development projects. This section introduces an automated testing tool, particularly from the viewpoint of how it improves efficiency in testing tasks.

This section assumes that a Java Servlet/JSP screen Web application based on the Model-View-Controller (MVC) model is being developed (**Figure 3**). The tools used for each block of the application are described below.

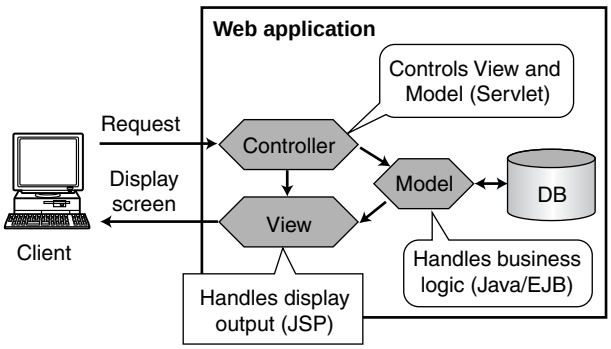

Figure 3
MVC model.

1) Testing the contents of each block: the Unit test

Two products are offered for testing the parts where programming was performed in the Controller block and Model block.

SIMPLIA/JF Kiyacker provides supporting features for creating coding rules, checking the rules of the Java source, and reviewing bugs and performance. Although in general, most of the common tools in the market simply offer checking functions, when they are used in a project, the method of keeping development staff informed of the rules becomes a matter of concern. Users of SIMPLIA/JF Kiyacker can customize the rules that the tool provides as standard and output customized rules as an HTML file. By distributing the rules, development staff can be kept up to date.

SIMPLIA/JF JudgePruefer offers a feature to automate the testing of Java programs by automatically generating drivers (calling programs created to operate the program that will be tested) and stubs (replacement programs that replace the uncompleted program called by the program to be tested). This is also possible with Junit,[4] which is an open-source framework for unit tests; however, its linkage with test specifications is poor, so additional thought is needed to improve it in applications. This tool can create test specification sheets and generates drivers and stubs. It also automatically executes and determines the results of tests based on this test specification, which makes it easy to review test details and manage test results, and enables support to be in line with the development style.

Linkage of the above testing tool with Eclipse is being enhanced continuously so it can be applied using recent development styles based on Eclipse.

2) Testing by connecting blocks: the integration test

SIMPLIA/JF JudgePruefer is applied in the server-side test that integrates the Controller

block and Model block. SIMPLIA/TF-WebTest is provided as a testing support tool for situations when all MVC blocks are integrated. This tool can automate a regression test from the Internet Explorer screen. An automated testing tool for Web applications that supports testing from the integration test specification sheet will also be developed starting in July 2006.

Details of SIMPLIA/JF JudgePruefer and the automated testing tool for Web applications are introduced below.

3.1 SIMPLIA/JF JudgePruefer

SIMPLIA/JF JudgePruefer has the following functions for automating unit tests and server-side tests.

1) Execution function

This function automatically generates drivers and stubs and automatically executes tests from test cases created by the user. The user can create a test case by setting arguments, the return value of a method, or the status of a file or DB before and after the test.

2) Verification function

After executing a test, this function compares the test results with the return value of the method and the status of the parameter, file, or DB that was set when creating the test case and then determines whether the test succeeded.

In the unit test, this tool generates drivers and stubs automatically, so there is no need for the user to create them. In addition, by verifying statement coverage, test cases that are lacking from the white-box viewpoint can be added. In the integration test, the automatic generation function of the driver of this tool, together with a function that automatically sets up the DB status before the test and the automatic verification function of the DB status after the test, enables automatic execution of server-side tests whose results depend on the DB status, which used to be a difficult task.

As explained above, the user can execute tests automatically simply by creating test cases. This enables the user to focus on creating test cases, which helps improve the quality of testing.

SIMPLIA/JF JudgePruefer was applied to a 3-month project, starting from the user interface designing process and ending in the system testing process. In this example, a nearly 100% statement coverage was realized for unit tests, excluding exception processing, and degradation (function setback) was prevented by automating regression tests during the specification changes after the integration test.

3.2 Automated testing tool for Web application

Except for loading tests, it is difficult to support the testing of tasks on the screen using tools, and testing tasks are mostly executed manually as in the past. Other companies offer products for automating regression tests from the client. However, these products do not improve efficiency in the initial test and are effective only after the second regression test. In addition, these products do not provide supporting features for creating test specification sheets and require test cases to be extracted manually, making the quality of testing dependent on personal rules. To solve this problem, a tool for automating Web application tests from the client is being developed.

It is necessary to consider the business specifications of a function test in an integration test. However, in most cases, business specifications are described in sentences, and because there are a huge variety of document formats, analysis is difficult to do with tools. However, with the screen transition diagram and screen item definition, it is relatively easy to extract information about screen transitions and items from the tools they were written with or from files, and it is also easy to generate screen transition patterns that become test cases. This tool generates a test case of a screen transition test from a screen transition diagram and screen item definition saved in the format of Rational XDE, which is a UML modeling tool from IBM, and can create a test

specification sheet in Microsoft Excel format (**Figure 4**).

The test case generates the test specification from test viewpoints, based on the information in the screen item definition within the path from start to finish in a screen transition diagram. The generated test specification can add or delete test cases on an Excel spreadsheet. The test viewpoints used for the generation are as follows.

1) Input data (normal/abnormal)
2) Number of items displayed on the screen for repeated items such as charts (0/1/*N*)
3) Number of loop executions within the screen transition diagram

Value candidates are generated automatically from the form of each item in the screen item definition of each test case. Items with definite codes such as gender, in particular, can be entered just by selecting from the defined code value displayed.

During a test based on a test specification sheet created in the above manner, a screen called the Test Execution Navigator (hereafter called the Navigator), which navigates the test and supports the testing tasks of a single test case, is displayed. Navigator extracts the test case from the test specification sheet and displays the tasks that must be performed before testing, the input operations of the testing screen, and the verified test results. The user performs testing tasks by following the instructions of the Navigator and enters the test results into the Navigator screen. Because these test results are automatically reflected on the test specification sheet, this test specification sheet can be used as a result report at the same time.

In addition, this tool can be linked with SIMPLIA/TF-WebTest, so testing operations can be recorded and saved and then retested during regression tests.

4. PROSPECS

In current application development, a development environment supported by features such as UML, a framework, and source program automatic generation is generally used. The productivity of source programs has improved dramatically due to the use of these development environments. However, at the same time, there

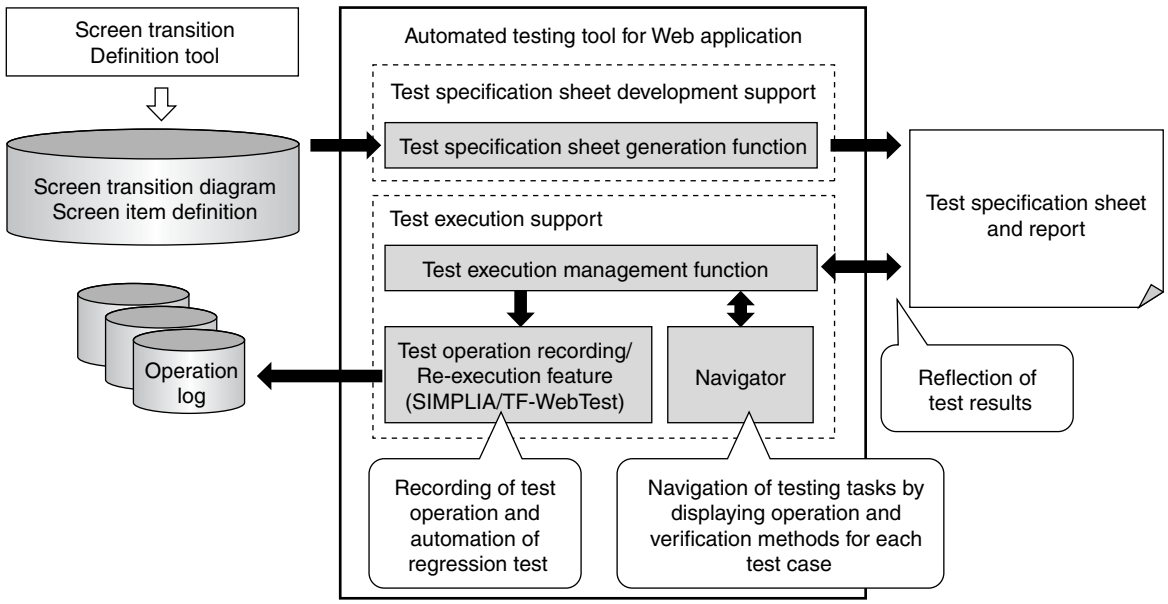

Figure 4
Automated testing tool for Web applications.

has been a troubling increase in the number of undocumented programs. This increase is due to the following problems with document creation.

1) Because of the shorter development periods, priority is put on modifying programs that are currently being worked on, and document creation is left for a later date.
2) Because of frequent changes in specifications, it is difficult to maintain the synchronization of source programs and documents.

As a result, document creation tends to be considered less important in system development.

Documents are required more during maintenance after the system has begun operation than during the system's development. In maintenance, it is necessary to understand the existing source program because the maintenance is based on modifying the existing source program. Moreover, as time passes, development personnel and maintenance personnel often change, so maintenance personnel will struggle to understand the source program. Furthermore, in most cases, the available documents about the source program have not been updated in synchronization with the source program.

A reverse engineering tool solves these problems concerning documents. There are three benefits in using a reverse engineering tool.

Firstly, such a tool enables the creation of accurate documents. Because the input is based on the currently operating source program, the reliability of the generated document is high.

Secondly, it enables documents to be created from just the source program, which is useful for understanding in-house source programs and open-source programs created by others when there is no other documentation.

Thirdly, it saves time when creating documents. This is a major benefit when creating delivery goods or determining internal specifications when the source program is being created, for example, as in prototype development.

4.1 Reverse engineering tool PROSPECS

There are a number of reverse engineering tools, most of which base input on the source program and automatically generate documents reflecting the comments of the source program based on the extraction rule for comments.

Fujitsu has developed and is merchandising a reverse engineering tool called PROSPECS.[5] The specifications of PROSPECS are as follows.

1) Input language: Java, C/C++, C#, Visual Basic (hereafter called VB)
2) Generation types: 47 to 69 types (depending on the language used)
   - Definition information: class list, method list, etc.
   - Description: class description, method description, etc.
   - Reference information: class reference list, parameter reference list, etc.
   - Structure information: class hierarchy diagram, method calling diagram, etc.
   - Metrics information: file metrics, method metrics, etc.
   - Difference information: file difference, method difference, etc.
3) Generation forms: printout, HTML, Word, CSV

PROSPECS can be used for creating documents and understanding the source program. It features a form editor and a source browser for supporting tasks. In most cases, users are standardizing the formality of documents. This is done using PROSPECS' form editor, which defines the format of a document and is a drawing tool for laying out rectangular areas, lines, comments, and source analysis results with a mouse. This tool is easy to operate and makes it easy to obtain a printed image (**Figure 5**).

The source browser is used to reference a source program. It consists of a window displaying reserved words, identifiers, and comments of the source program in different colors; a window displaying definition lists of classes and methods; and a window displaying reference and structure

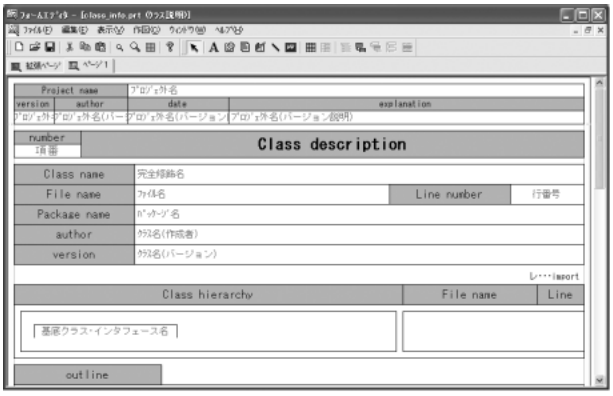

Figure 5
PROSPECS Class description.

information. This tool improves the readability of the source program. Moreover, by using the comment keyword setting, extracted keywords and comments reflected in the document can be displayed in different colors. This tool makes it easy to define the comment to be extracted from the source program.

4.2 Aiming for abstraction of logic

Software development can be broken down into processes such as analysis, designing, implementation, and testing. The documents generated by the current reverse engineering tool have a quality close to that of the source program created in the implementation process. One of our goals for the future is to generate documents having the same quality as documents in the analysis and designing processes. We can achieve this goal by establishing a technology to extract meaningful logic from the source program and generating highly abstract documents.

5. NetCOBOL

When constructing the base of an open system, there is always a debate regarding whether to use Java and VB or to use COBOL as a programming language, and to resolve this debate, the features of each language must be considered. Java supports the Internet, which enables flexible system construction; whereas with COBOL, existing assets operating on mainframe systems and office computers can be reused. In future system construction, it will be important to use the right language for the situation based on current existing assets and the objective of the system to be constructed. The number of cases that use Java for front operations is growing, and COBOL is used for constructing business logic using existing assets.

NetCOBOL,[6] which is a COBOL development environment offered by Fujitsu, provides integrated support for development using COBOL, from design to testing. It also ties up with Interstage Apworks, which is an integrated development environment that adopts the industry standard Eclipse. These two features of NetCOBOL enable quick and efficient development of a system using Java and COBOL. In addition, in a system construction based on Microsoft's .NET Framework, a system can be developed in the same way as VB and C# by using Visual Studio.NET, which is the integrated development environment of the .NET Framework. The NetCOBOL integrated development environment supports all the phases of the development cycle, for example, designing, programming, testing, and maintenance. This section introduces the basic features offered by the NetCOBOL integrated development environment, for example, the compiler, debugger, screen and print-form editing, testing support, and document creation support features.

1) Project Manager

Project Manager is the main tool of the NetCOBOL development environment. It controls the dependencies of development assets and enables operation linkage with various support tools, for example, the compiler and debugger. It can efficiently develop applications, including applications with conventional specifications and applications with object-oriented COBOL specifications. Some Project Manager screenshots are shown in **Figure 6**.

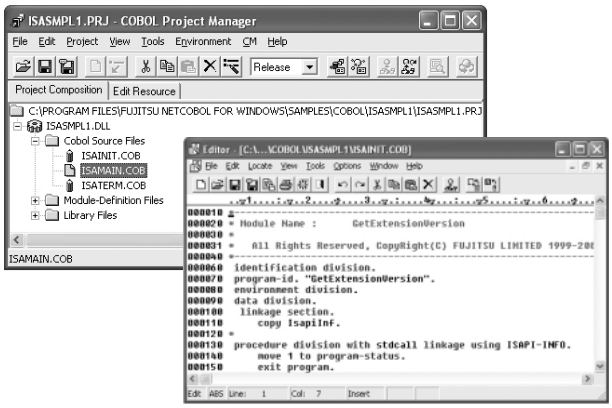

Figure 6
Project Manager screenshots.

2) Compiler

This complies with international standards and has object-oriented features that have been adopted in the new COBOL2002 standard. It also realizes major industry-standard specifications and the Fujitsu-standard specifications in common with Fujitsu mainframes and office computers.

Japanese cannot be expressed in the character-code system in a single byte, which can represent one of 256 characters. Therefore, multiple character-code systems such as Shift-JIS, EUC, Unicode, and EBCDIC-JEF are provided. The support of character-codes is important, so NetCOBOL supports the Japanese Shift-JIS, EUC, Unicode, and EBCDIC-JEF character-codes.

3) Debugger

NetCOBOL's rich interactive debugging features make it possible to improve efficiency when testing tasks and obtain quick solutions to problems. Some examples of these features are:

- Flexible breakpoint setting
- Monitoring of data contents
- Breaking at data modification points
- Testing of just a sub-program

It also has an analysis report feature that outputs detailed debugging information during operation and can speed up troubleshooting investigations.

4) Screen and print form editing

A detailed screen and print form application can be created in COBOL by defining three fields: the screen definition field, which defines the type and screen location of data to be entered/displayed; the print form definition field, which defines the type and location of data to be printed; and the overlay definition field, which overlaps the print form definition field. In addition, COBOL programmers can define the screen and print form definition field in the same way as a normal file by using the familiar WRITE and READ statements, which shortens the construction time.

5) Development and maintenance support (SIMPLIA/COBOL support kit)

This provides various features that support development and maintenance. These features include the following:

- Testing support

The conversion tool and interactive editor simplifies the creation of test data and enables data to be created more efficiently by using existing data.

- Document creation support

Maintenance documents (e.g., screen and ledger design, file design, application structure design, and module design documents) can be automatically generated from development assets.

- Support for migration between platforms

Support for migrating data and source code between mainframes, office computers, and open systems is provided.

6. Conclusion

In Fujitsu, application development using Java and COBOL is increasing year by year, and a systematic development tool set that supports multiple platforms without depending on technologies of a specific vendor is becoming increasingly important.

By using the information of the design process for implementation and testing, considering new technologies such as Model-Driven Architecture (MDA) and development methods based on

Service-Oriented Architecture (SOA), and through other means, Fujitsu intends to offer more practical development tools and advance in its pursuit of technological superiority.

References

1) Eclipse.
http://www.eclipse.org/
2) Fujitsu: Interstage Apworks.
http://www.fujitsu.com/global/services/software/interstage/products/devsuite/
3) Fujitsu: SIMPLIA series. (in Japanese).
http://software.fujitsu.com/jp/simplia/
4) Junit.org.
http://www.junit.org/
5) Fujitsu: PROSPECS. (in Japanese).
http://jp.fujitsu.com/fst/services/frontier/kobo/
6) Fujitsu: NetCOBOL.
http://software.fujitsu.com/jp/cobol
(in Japanese)
http://www.netcobol.com/

Hideo Abotani, *Fujitsu Ltd.*
Mr. Abotani graduated from Akashi National College of Technology, Akashi, Japan in 1983. He joined Fujitsu Ltd., Numazu, Japan in 1983, where he was engaged in development of programming language compilers on mainframes. Since 1994, he has been engaged in development of the COBOL compiler on open systems.

Tomoki Shiratori, *Fujitsu Ltd.*
Mr. Shiratori received the B.S. and M.S. degrees in Computer Science from the University of Tokyo, Tokyo, Japan in 1992 and 1994, respectively. He joined Fujitsu Ltd. Kawasaki, Japan in 1996, where he has been engaged in development of middleware products for Java programming language, for example, Graphic User Interface (GUI) library and its development environment.

Kouji Sasaki, *Fujitsu Software Technologies Ltd.*
Mr. Sasaki graduated from Hamamatsu Industrial High School, Hamamatsu, Japan in 1987. He joined Fujitsu Software Technologies Ltd. (formerly Fujitsu Shizuoka Engineering Ltd.), Hamamatsu, Japan, in 1987, where he has been engaged in development of source code analyzers.

Masaki Tonomura, *Fujitsu Ltd.*
Mr. Tonomura received the B.E. degree in Electric Engineering from Kinki University, Osaka, Japan in 1989. He joined Fujitsu Ltd., Kawasaki, Japan in 1989, where he has been engaged in development of Computer Aided Software Engineering (CASE) and support tools for software development.