Automatic Generation of Tutorial Systems from Development Specification

Hajime Iwata[1], Junko Shirogane[2], and Yoshiaki Fukazawa[1]

[1] Department of Information and Computer Science, Waseda University,
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
{hajime_i, fukazawa}@fuka.info.waseda.ac.jp
[2] Tokyo Woman's Christian University,
2-6-1 Zempukuji, Suginami-ku, Tokyo 167-8585, Japan
junko@lab.twcu.ac.jp

Abstract. Recent complicated software functions have made it difficult for end users to operate them. Thus, it becomes important to learn how to operate them easily and effectively. Employing a tutorial system is the most suitable approach for learning how to operate software functions. A tutorial system demonstrates the how to operate using the actual software. As a result, end users can learn the usage as if they were actually using the software. However, development of tutorial systems requires much time and costs. Therefore, we propose a method of generating tutorial systems based on use case diagrams, sequence diagrams and test cases. In our method, a generated tutorial system shows function names extracted from use case diagrams, the how to operate along with sequence diagrams, and text string input and item selection using data from test cases. The generated tutorial system is then added to the source code for use in AOP (aspect-oriented programming).

1 Introduction

Recently, computer usage has become widespread, and various tasks have been computerized. Many kinds of software have been developed with many functions. Therefore, software usage tends to be complicated due to these many functions. It becomes more difficult for end users to learn how to operate software. It is important to provide support methods by which end users can learn the operation of the software. Now, there are some support methods for end user learning, such as online manuals, help systems, animated demonstrations and tutorial systems. In this paper, among the many end user learning methods, we particularly focus on tutorial systems.

A tutorial system has been used by end users for learning how to operate software. A tutorial system demonstrates the usage on the running software. When end users learn the usage of the software using a tutorial system, they can learn the sequences of operations as if they were actually using the software without interruption.

There are some advantages of a tutorial system over other many support methods:

L. Baresi and R. Heckel (Eds.): FASE 2006, LNCS 3922, pp. 79–92, 2006.

– End users can easily understand the purpose of each operation and the relationships among the operations.
– End users can interactively learn the software operation with the tutorial system. Thus, they can understand software operation using the tutorial system better than using an animated demonstration system.

However, currently, there are few built-in tutorial systems for software. Because the structures of software have now become highly complicated and have many functions, the development of tutorial systems places a heavy burden on software developers, and the cost is high.

In this paper, we propose a method for automatically generating a tutorial system based on use case diagrams, sequence diagrams [1] and test data provided by developers. These diagrams and data are made in the software development process. A generated tutorial system is woven into the software using AspectJ [2]. Using our method, it becomes easier to develop a tutorial system.

2 Tutorial System in Our Method

As an example, a tutorial system of address book software is shown in Figure 1. The right-hand window is a window of the tutorial system generated by our method. The left-hand window is a window of the address book software.

The software into which a generated tutorial system is woven is called target software in this paper.

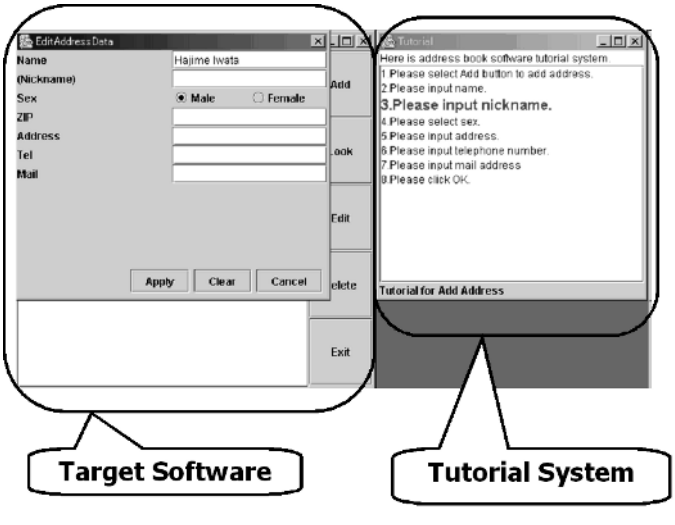

Fig. 1. Tutorial for address book software

In our method, the generated tutorial system, has the following features:

Showing Software Functions and Operations

In our method, the use case names are extracted from the use case diagrams for use as software function names in the tutorial system. The names of functions available in the software are shown in the tutorial system, and end users select the function they wish to learn. Then, the sequences of required operation steps extracted from sequence diagrams are shown step by step. The sequence of operation steps is described using itemized sentences in the tutorial system. End users can watch how the software function is operated, as well as the reaction of the software, continuously without any interruption. The sentence that explains the current operation is changed to have a vivid color and a large character size in the tutorial system window. Thus, it is easy to understand how to operate the software.

Showing Examples of Software Usage

The tutorial system shows a mouse pointer on a widget (Each part of a GUI, such as buttons, is called a widget.) in the target software at the same time highlighting the sentence explaining the current operation. At the same time, the required mouse operations are shown, such as clicking a button and choosing the radio button. End users can easily understand the sequences of operation steps required for the function by watching the software demonstration without interruption.

Demonstrating Input Texts

When it is necessary to input text for text fields, the tutorial system demonstrates an actual text input. The input text is not meaningless text but suitable

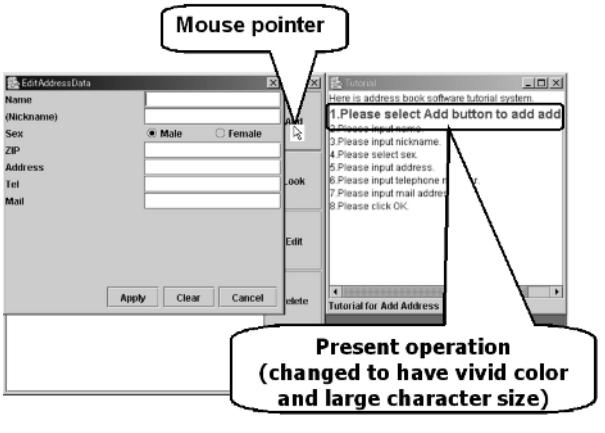

Fig. 2. Tutorial "Add Address"

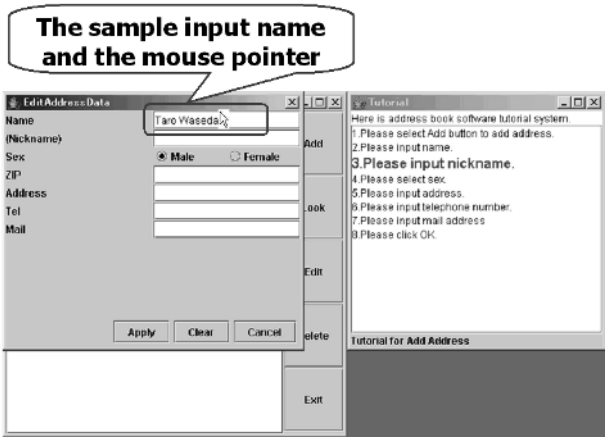

Fig. 3. Tutorial for sample input name

text extracted from the test data. We assume that the input text is the typical data in using the function. Therefore, end users can understand examples of actual data and how to input text for pertinent text fields.

In Figure 2, a user selects the tutorial "Add Address" is shown and how to operate for adding a new address.

The sequence of operations is displayed in the tutorial window as itemized sentences describing the steps in sequential order, and each step is numbered. When the end user selects the tutorial "Add Address", the explaining text which is "Please select Add button to add address." in Figure 2 is changed to have a vivid color and a large character size in the tutorial system window. Similarly, "Taro Waseda", the sample input name extracted from the test data, is entered automatically by the tutorial system. Also, the mouse pointer is moved to this text field. This example is shown in Figure 3.

3 Features of Our Work

3.1 Development Specification for Generated Tutorial System

In this paper, we propose a method for automatically generating a tutorial system for target software. Generating a tutorial system allows end users to efficiently learn how to operate the target software. The tutorial system is generated automatically on the basis of these specifications, which are use case diagrams, sequence diagrams and test data. Use case diagrams are used in the analysis phase of object-oriented software development, to describe software functions. Sequence diagrams are used for describing interaction between objects in order. An example of a use case diagram for the address book software in section 2 is shown in Figure 4, and an example of a sequence diagram in the case of the "Add Address" function is shown in Figure 5.

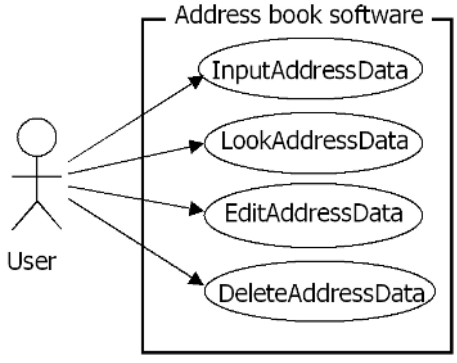

Fig. 4. Example of use case diagram

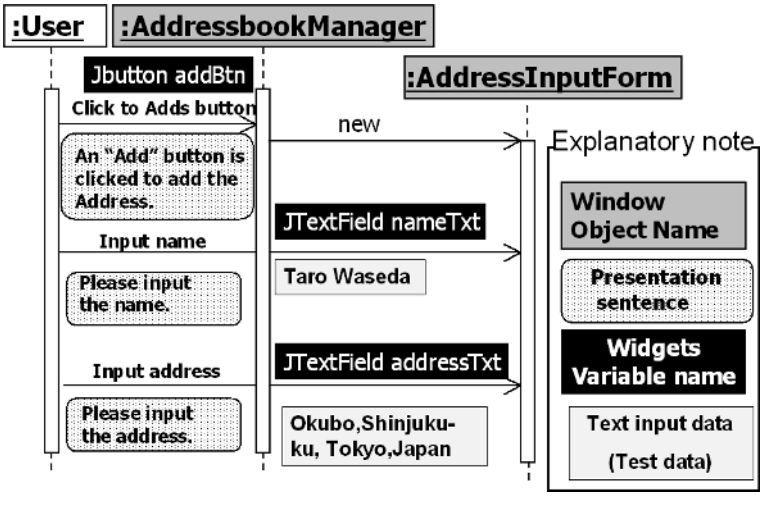

Fig. 5. Example of sequence diagram

In our system, diagrams described by the XMI (XML Metadata Interchange) [3] format are used. The XMI format is the standard file format for describing UML diagrams. Using the XMI format, use case diagrams and sequence diagrams provided by the existing UML modeling software can be generated for a tutorial system. Examples of UML modeling software supporting the XMI format are Rational Rose [4] and IIOSS [5].

In our research, the flows of procedures within use cases are described using sequence diagrams, and extra descriptions are added in the sequence diagrams. Target software developers combine the widget name, presentation sentence, and test data for input text, with the messages of sequence diagrams, and specify the window objects of sequence diagrams.

Window Objects

Target software developers are required to specify objects, which have widgets on GUI windows in the sequence diagrams. Thus, our system can specify the widgets that correspond to the objects, and our method can operate the tutorial system for the software. These objects are called window objects in our method. For example, ":AddressbookManager" and ":AddressInputForm" are the window objects seen in Figure 5.

Presentation Sentences

Target software developers are required to add extra descriptions to the messages of sequence diagrams. The extra descriptions are texts, which explain to users how to operate software functions. These extra descriptions are displayed to end users as explanations of the usage when the tutorial system is generated. For example, 'Click the "add" button to add the address' and "Please input the name" are the extra descriptions seen in Figure 5. These texts are called presentation sentences in our method.

Widget Variables

The names of the widget variables in a source code must be added to the messages of a sequence diagram. These widgets are operated by the end user, such as input text and select an item.

In our method, we assume the Java Swing set to be widgets. Typical widget variables that must be added are buttons (JButton), menus (JMenuItem), and check boxes (JCheckBox). For example, "addBtn", "nameTxt" and "addressTxt" are shown in Figure 5. However, widgets that are not operated by the end user, such as labels (JLabel) and panels (JPanel), do not need to be added to messages. Using a widget name, the coordinates of each widget in a window can be calculated. A demonstration of mouse pointer movement based on the coordinates of each widget is performed.

Also, each widget has a method that substitutes for the end user operation. For example, "JButton" has a method called "doClick()" which substitutes for a mouse click of the end user, and "JTextFields" has a method called "setText()" which substitutes for a data input of the end user. The tutorial system calls these widget methods for widget operation.

Test Data for Input Text

When widgets require text data, the tutorial system should show an example of the input. For this purpose, existing test data is used. In our method, we assume that the required test data is typical data for using the function. In this paper, we use the test data of JFCUnit [6]. JFCUnit is a software testing tool for Java Swing. The test data of JFCUnit is described as the correspondence a widget variable and a text datum. Therefore, the tutorial system can use the input text data by checking widget variables in sequence diagrams and in test data.

3.2 Added Tutorial System

The generated tutorial system is described using the AspectJ code. AspectJ is the one of the software tools that use aspect-oriented programming (AOP) [7]. To implement the tutorial system, the following two processes need to be added to the target software:

- Get the coordinates of widgets on the screen for showing the mouse pointer.
- Operate each widget, such as a mouse click and a text input

These two processes have to be added to each widget in the software. That is, they cannot be created as one class. They have to be added to many classes which process widget generation. Therefore, these two processes can be considered as crosscut concerns of AspectJ. Thus, these two processes are generated as AspectJ code in our method. It is possible to weave a tutorial system into target software without modifying the source code. Therefore, developers can easily maintain software.

4 System Architecture of ACTS

The system architecture of our system, called ACTS (Automatic Creation of Tutorial System), is shown in Figure 6. ACTS consists of the following five steps.

1. Adding extra description to sequence diagrams
2. Extracting function name from use case diagrams
3. Extracting operation information from sequence diagrams
4. Generating tutorial system automatically
5. Weaving tutorial system into target software automatically

4.1 Adding Extra Description to Sequence Diagrams

Target software developers output the use case diagrams and sequence diagrams described using various UML tools. Outputted data is written in the XMI format. ACTS creates a file in which required descriptions for generating a tutorial system are added to XMI-formatted sequence diagrams. The required descriptions are the widget variables, presentation sentences and test data for input text, and combined with the messages of sequence diagrams. Also, the window objects are specified in sequence diagrams. These extra descriptions are added by the target software developers. And ACTS adds test data of JFCUnit as input text to sequence diagrams.

4.2 Extracting Function Name from Use Case Diagrams

From the described use case diagrams, ACTS extracts the use case names of software functions used by the tutorial system. End users can watch an explanation of usage for a function by selecting the function name.

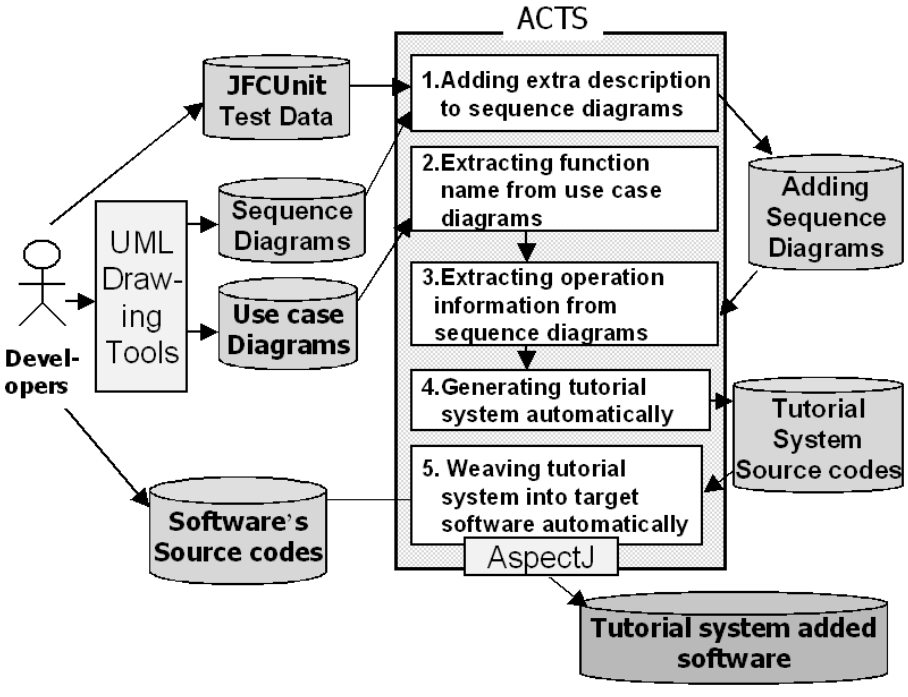

Fig. 6. System architecture of ACTS

4.3 Extracting Operation Information from Sequence Diagrams

ACTS extracts window object names and widget names from the described sequence diagrams. By using a widget name, the coordinates of each widget in a window can be calculated. A demonstration of mouse pointer movement based on the coordinates of each widget is performed. In our method, a mouse pointer is created as a visual image, and it is moved to each widget. Thus, our system can implement a demonstration of mouse pointer movement virtually.

The order in which the operations of the function are carried out is extracted from the flow of messages described in the sequence diagrams. By adding the explanation sentences of operations to the sequence diagrams, a suitable timing for displaying the explanation of an operation can be extracted.

4.4 Generating Tutorial System Automatically

A tutorial system is automatically generated by extracting function names from use case diagrams and information on how to operate the functions from sequence diagrams.

The procedure is shown as follows:

1. A process for displaying the function based on the extracted function name is generated for a tutorial.

2. An operation procedure based on the flow of the operation information in the messages of the sequence diagrams is constructed.
3. According to the constructed operation procedure, a process for the demonstration of mouse pointer movement from the location of the GUI widgets and end user operation is generated.
4. A process for displaying the explanation of the operation following the demonstration of mouse pointer movement is generated.

Generated source codes based on above the procedure include following contents.

- ACTS generates Java source codes creating a window. This window consists of buttons for representing function names of the target software and a text area for displaying the presentation sentences for the tutorial system.
- ACTS generates AspectJ source codes which demonstrate mouse pointer movement virtually based on the coordinates of widgets in the target software.
- ACTS generates Java source codes which display presentation sentences based on the operation procedure extracted from sequence diagrams.

4.5 Weaving Tutorial System into Target Software Automatically

Target software developers can add tutorial system to the target software by compiling generated Java source codes and weaving generated AspectJ source codes into the target software.

Widget names described in sequence diagrams are written in these AspectJ source codes. AspectJ software searches the target software for the same widget

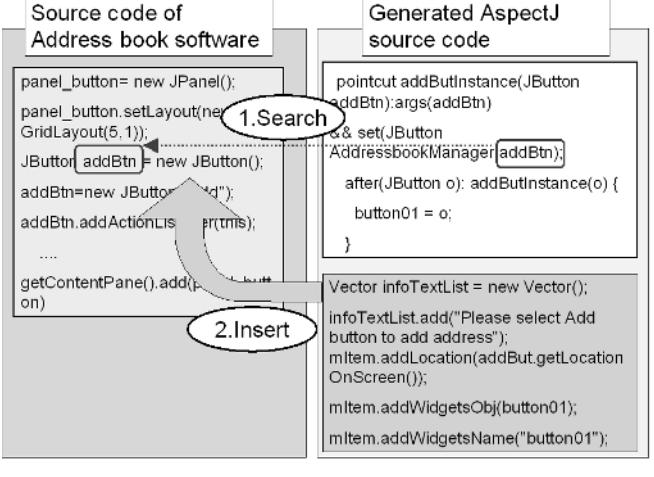

Fig. 7. A summary of clicking the button of "Add Address"

names as the widget names written in generated AspectJ source code, and weaves necessary codes into the target software, such as the procedure of getting the coordinates of widgets and displays the presentation sentences.

The summary of weaving generated AspectJ source code into target software is shown in Figure7. This is an example of clicking the button of "Add Address" in address book software.

In the example of address book software, AspectJ searches the widget name "addBtn" from the target software, because "addBtn" is written in the Pointcut of generated AspectJ source code(Figure7). AspectJ software adds following procedures to the found widget "addBtn" button.

- Get the coordinates of this button on the screen
- Show a mouse pointer on this button
- Highlight the presentation sentences when the current operation step of demonstration is about this button.

5 Evaluation

5.1 Add Some Extra Descriptions

Using the sequence diagrams of the following three types of software, we evaluate the burden placed on developers by the requirement of generating tutorial systems.

- Address book system software
- Mail-order system [8]
- Video rental system [9]

We counted the number of use cases, sequence diagrams, objects in each sequence diagram, and messages in each sequence diagram. The results are shown in Table 1. In this table, "Address" represents address book software, "Mail" represents mail-order system, and "Video" represents video rental system.

In our method, developers are required to add some extra descriptions to sequence diagrams. The required descriptions of sequence diagrams are the following three

- Specifying window objects
- Adding widget variables and explanation sentences of operations
- Adding text input examples for widgets

As shown in Table 1, "Window objects" indicates the number of specified window objects. "Extra descriptions" indicates the number of messages added as explanation sentences of operations and widget variables. "Extra text" indicates the number of messages added as examples of input text.

We verified that the rate of specified window objects per total number of objects is about 50% - 60%. The number of messages with extra descriptions added per total number of messages is about 50% - 60%. In these evaluations,

Table 1. Number of descriptions

	Address	Mail	Video
Use cases	4	15	7
Sequence diagrams	6	19	31
Objects	19	50	165
Window objects	13	28	95
Messages	53	102	379
Extra descriptions	32	51	138
Extra text	12	14	39

required extra descriptions for generating a tutorial system are of the same rate for other software tutorial systems.

Moreover, the number of messages with text input per total number of messages is about 10% - 20%. Input text is extracted from test data. Developers only select suitable test data. Therefore, developers are not required to create new input data for our method. The target software developers' burden does not become so big.

5.2 Correctness of Software Operation

Using the address book system software and video rental system, we evaluated whether the tutorial system generated by our method performs software operation correctly.

While executing the tutorial system, the following use cases were always performing correctly.

- "Add address data" in address book system software
- "Record new video" in video rental system
- "Record new cast" in video rental system

However, when the software did not have saved data, some use cases were not performed. For example, the following use cases were not performed.

- "Edit address data" in address book system software
- "Search video" in video rental system
- "Search cast" in video rental system

Therefore, when a use case needs some saved data, it is necessary to prepare the saved data for tutorial execution, or to specify the order in which tutorial execution is performed.

Next, we evaluate the correctness of performing the operations. The result is shown in Table 2. "Widgets" indicates the number of widgets, "Operating widgets" indicates the number of widgets operated by end users, "Performed operating widgets" indicates the number of widgets performing the operations correctly.

While executing the tutorial system, typical operations, such as clicking buttons (JButton), checking check boxes (JCheckBox) and selecting radio buttons (JRadioButton), were always performed correctly.

Table 2. Number of widgets and performed widgets

	Address	Video
Widgets	26	234
Operating widgets	14	77
Performed operating widgets	14	74

For the widgets that require input text data from end users, for example, text fields (JTextField) and text areas (JTextArea), the text data was automatically entered into the widget.

However, for the dialog boxes using JOptionPane, the tutorial system could not operate the widgets in the dialog boxes. The reason for this is as follows: Software with a GUI extended objects such as JFrame and JDialog often creates widgets in the constructor. That is, an instance of the extended object for displaying a GUI window on the screen is created.

At the same time, to display a dialog box on the screen an instance of JOptionPane is created. On JOptionPane, at least, some buttons are prepared as part of JOptionPane. The tutorial system cannot operate these prepared buttons on JOptionPane. Therefore, we should consider methods for operating JOptionPane.

6 Related Works

There are some existing methods for supporting the process of learning how to operate software. These include the user navigation system, the support system for learning how to operate software on the basis of operation logs, the GUI cover system, and other tutorial generation systems.

User Navigation System

The support system for learning how to operate software on the basis of operation logs takes the logs of user operations and uses them to understand the target function. This system analyzes the users' operations according to the goal. The next time the user uses the software, this system can show operations for that goal.

The following study is proposed for this method: development of a system which stores a log of a user's operations on a HCI (Human Computer Interaction) server, creates a model of the operations from this log, and then shows the operation [10].

The support system for learning how to operate software on the basis of recorded operation logs has the advantage of being able to allow user operation habits to be reflected in the next learning support session.

However, since it cannot determine which operation should be performed for the entire set of end users, this support system cannot perform an analysis using the end users operation log. Moreover, it is not necessarily that other users' operation logs are suitable for end user learning.

That is, this method is for users using the software many times; it is unsuitable as a learning aid for end users who use the software for the first time.

GUI Cover System

The GUI cover system [12] is a system that creates a new GUI screen which is different from the software's original GUI, and which makes operation possible on the new GUI. The new GUI screen is presented by hiding the software's original GUI; the new GUI screen, which functions as a substitute for the software's original GUI, link end user operations to software functions.

The GUI cover system involves preparing a GUI that can only be used for basic functions, even if it is for on advanced piece of software. The end users can change the GUI according to their preferences.

However, when a GUI cover is removed, it is noted as a disability that end users cannot operate the equivalent software function. That is, for end users who do not know how to use a new piece of software, a GUI cover system can reduce the burden on end users for learning the operations. However, when end users must use the software's original GUI, the learning effect is found to be inadequate.

Jedemo

Jedemo is a demonstration-authoring tool for Java applets [11].

In this method, developers add event-driven functions to a Java applet. The recorded event-driven functions operate the software automatically. Moreover, the event for automation is recordable from an operation. This method creates the help functions for Java applet software.

These are the advantages Jedemo:

- Since animated help functions can be created, new examination texts that software developers have to write are reduced.
- End users only need to push one button to see a demonstration of a concrete operation method.

Although these advantages do fulfill the aims of this method, the following points are noted as disadvantages. In order to generate a tutorial system, it is necessary to describe a new rule for introducing the concept of event-driven functions. In our method, a tutorial system is generated on the basis of use case diagrams and sequence diagrams, and these diagrams have already been described in the development stage. Therefore, our method imposes a smaller burden than this method.

7 Conclusions

In this paper, we propose a method for enabling developers to automatically weave a tutorial system into existing software, on the basis of use case diagrams,

sequence diagrams and test data. Using the tutorial system, end users can easily learn the operation of the software.

The following subjects remain for future work:

- Improvement of operation replication. A generated tutorial system in our method will support more GUI widgets.
- Supporting more test data formats. In this paper, at this moment, we use the test data for JFCUnit. We will support more GUI test tools.

References

1. G. Booch, I. Jacobson and J. Rumbaugh: "The Unified Modeling Language User Guide", Addison-Wesley (1998)
2. "aspectj Project", http://www.eclipse.org/aspectj/
3. "XML Metadata Interchange (XMI)": http://www.omg.org/technology/documents/formal/xmi.htm
4. T. Quatrani: "Visual Modeling With Rational Rose 2002 and UML", Addison-Wesley (2002)
5. "The IIOSS Project", http://www.iioss.org/index-e.html
6. "jfcUnit User Documentaiton": http://jfcunit.sourceforge.net/
7. G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.M. Loingtier and J. Irwin: "Aspect Oriented Programming", PARC Technical Report, SPL97-008P9710042 (1997)
8. G. Schneider and J.P. Winters: "Applying Use Cases: A Practical Guide", Addison Wesley Longman, Inc. (2000)
9. J. Shirogane and Y. Fukazawa: "A Method of Scenario-Based GUI Prototype Generation", 3rd ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 02) (2002)
10. L.M. Encarnacao and S. Stoev: "An Application-Independent Intelligent User Support System exploiting Action-sequence based on User Modelling", Proc. of the Seventh International Conference on User Modeling (1999)
11. M. Miura and J. Tanaka: "Jedemo: Demonstrational Authoring Tool for Java Applets", Proc. of the Fifth World Conference on Integrated Design and Process Technology (IDPT2000) (2000)
12. H. Okada and T. Asahi: "GUI Navigator/Cover: GUI Transformation Systems for PC Novice Users", Proc. of the 10th International Conference on Human-Computer Interaction (HCI International 2003) (2003)