NIPPON STEEL TECHNICAL REPORT N0. 76 MARCH 1998

UDC 681 . 3

Component-based System Generation and System Testing Suppot Tools

Nobuo TAKAYANAGI*(1) Chisato KOTANI*(1)
Shinya KUBO*(2) Masahiro NOGUCHI*(1)

Abstract:

The authors have developed an application generator and a system testing tool to support rapid development of application systems with graphical user interfaces. The former tool can generate application systems through component synthesis, adopting object oriented technology, while the later tool can realize automatic system test execution based on automatically generated test specifications. These tools enable software engineers to efficiently build application software, generate its specifications, and test it, by visually combining prepared program components (i.e. objects).

1. Introduction

The trend of down-sizing along with open standards of computer systems demands construction of a more user friendly system. With the current remarkable progress of hardware and software, several version upgrades a year are not so exceptional.

In this situation, the demand for software development has been further raised, increasing the technology and loads necessary for development such as shortening of developing periods, firm confirmation of specifications at initial stage, quality enhancement, and installation of easy-to-use and easy-to-understand GUIs (graphical user interfaces). Therefore, development support tools are very important, particularly for software development for GUI based application systems.

This report describes a component-based system generation*1 tool and a system testing support tool which were developed by the authors as well as collaborated matters. The former is a support tool for prompt and easy construction of GUI based application systems. The latter is a support tool for testing them efficiently and integrally.

2. Development of GUI Applications

Development of GUI applications usually has the following characteristics.

1) GUI components*2 have rather high reusability and its range can be expanded if the target domain of the system is further analyzed.
2) GUI design has a large degree of freedom with numerous alternations. This requires a repeated trial and error process.
3) An integration of GUI components and programs called therefrom can be patternized.
4) Integration work itself is very difficult and is likely to induce defects in it.

At development, if the system constructing elements can be converted into components in advance and its assembling method defined by system developers for automatic synthesis, the application system can be assembled promptly and with excellent results in quality. This makes for very effective development.

This enables the assurance of high quality because no human activity intervenes in the component integration logic. A component itself is raised in quality in repeated reuse. However, a system structure composed of high quality components does not always

*(1) Electronics & Information Systems Div.

*(2) Ohita Works and Electronics & Information Systems Div.

*1 System generation: The system in this case is a software system as a program assembly to realize any processing on a computer. System generation means automatic system generation (partial) using some tools.

*2 GUI component: Program components to produce basic elements for constructing a GUI such as buttons and menus.

assure high quality for the entire system. This is because some defects are found only after they are assembled. Therefore, system testing is requisite for those with high quality components assembled.

Meanwhile, system testing for the GUI-based system is very difficult. This is primarily a result of the characteristics of an event driven GUI mainly carried out in dialogue processing with users. In another words, the number of resultant test cases expands exponentially due to high freedom of GUI for user's operation. Therefore, a test tool is strongly required to provide automatic testing as far as possible.

Automatic testing greatly depends on whether to give the test target specification correctly and in detail. If the requirement is given at the system generation tool side, the automatic testing is drastically enhanced.

Component-based system generation and system testing support tools, if realized, allow construction of a high quality system in a short period. The authors, therefore, have developed a component-based system generation tool and a system testing support tool. These tools are aimed at developing applications which can operate on UNIX[*3] and X Window Systems[*4]. Linkage of both tools is expected for efficient operation from system generation to testing. These tools are described in the following sections.

3. Component-based System Generation Tool

3.1 Outline

The tool for software assembly in graphical environments (hereinafter referred to as SG) constructs the target software by assembling components on a GUI environment. The SG enables an operator to generate not only the component synthesis application system but also a source code[*5] and specification document automatically for prompt system generation under the GUI environment. The tool deals with two kinds of program components[*6] including a GUI structure component (GUI component) and an application logic component (application component). The tool is very different from a GUI builder in that GUI components are equally treated as application components. The tool has also a message delivery function, called a message link, to transfer between components. The function can define the system action by setting this message link. This method has an advantage that a layered structure is naturally constructed with excellent extensibility and maintainability because GUI components are separated from application components, and their connection is expressed by the message link. Additional characteristic not seen in commercial tools is the inclusion of a function to produce specification documents. **Fig. 1** shows an outline of the functions and processing flow.

Application range to be produced is difficult to explain because it depends on components to be used. For example, a factory automation system requires meter components and chart components as GUI components, and signal data input/output components and control logic components as application components. For instance, an office automation system requires text input/output components and fast search components, and a financial system needs label components and database connection components specified for date and money expression. It is important to prepare high quality components suitable for each application field. In fields where many wide use components are prepared, productivity improvements enhance more effectively. **Fig. 2** shows an example of SG monitor. SG itself is prepared by SG.

3.2 Function

The SG has four major functions: an editing function to select components to set messages, a component registration function, an automatic generation function of source codes, and an automatic generation function of specification documents. These functions are described below.

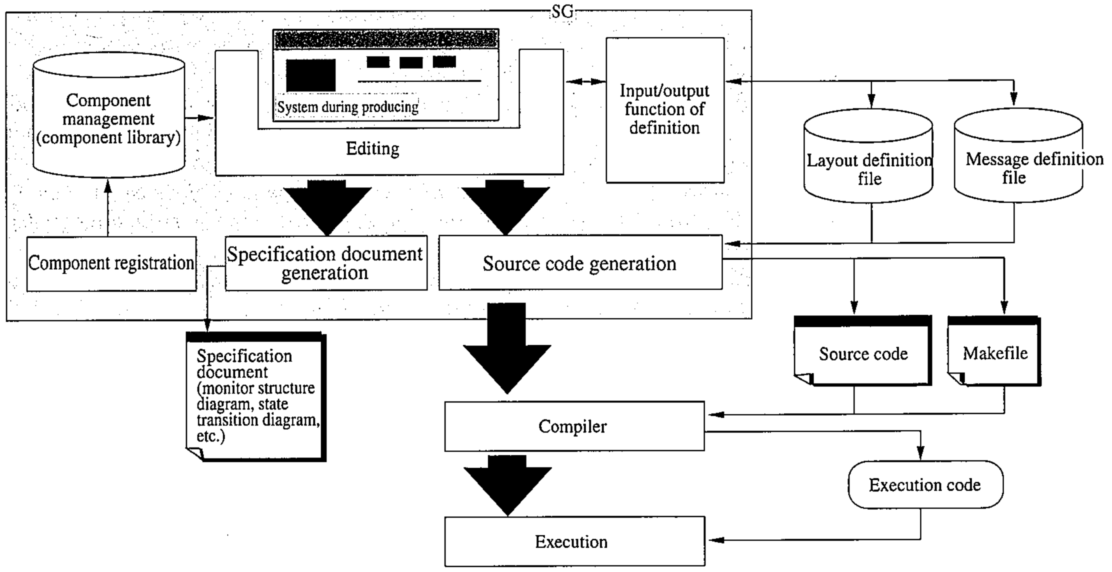
SG
Component management (component library)
System during producing
Editing
Input/output function of definition
Layout definition file
Message definition file
Component registration
Specification document generation
Source code generation
Specification document (monitor structure diagram, state transition diagram, etc.)
Source code
Makefile
Compiler
Execution code
Execution

Fig. 1 Outline of the function of SG and process flow

*3 UNIX is the trademark exclusively licensed by X/Open Company Limited applied in the United States and other countries.
*4 X Window System is the trademark of The Open Group.
*5 Source code: Program expressed by a programming language. It is written by a programming developer. The source code itself cannot be operated and must be converted to an executable format for a real operating program.
*6 Program component: Component using wide use elements or parts in the program.

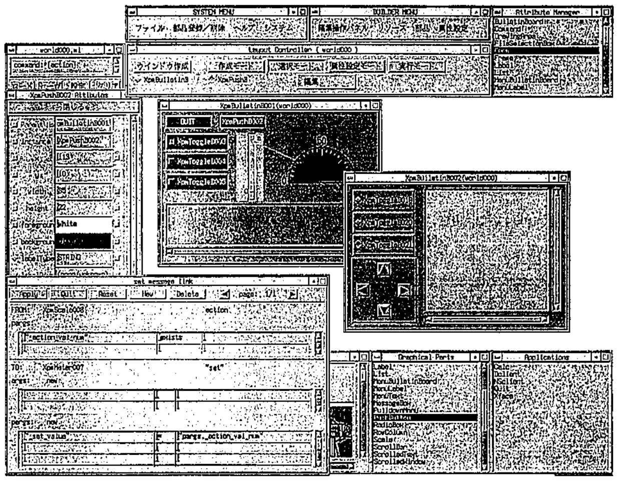

Fig. 2 Example of SG display

(1) Component synthesis and operational definition

As mentioned above, the system's structure and operation can be defined by assembling GUI components and application components and by setting message communication between components (**Fig. 3**). This enables construction of the system operating actually only by assembling existing components.

Practical operation lays out GUI components with a mouse on the monitor where each component is connected by a line to set messages between components. Almost all operations are available simply by mouse.

(2) Additional registration of components

Additional registration of components by users enables the accumulation of components in the target domain. For component generation support, an exclusive use generation support tool is provided for GUI component generation. For application components, a function for automatically generating an interface part to convert components is provided so as to deal with functions prepared by users as components.

(3) Operational check during generation

When generating a prototype, "generate as testing" is required first, so that its operation can always be checked anytime during application generation.

(4) Automatic generation of source code

Most general GUI builders have a function for generating only a source code skeleton by adding necessary parts by users as a prerequisite. The SG generates all the source code (C++[*7]) which can be compiled as it is instead of the skeleton. As an exceptional unique function, it can generate the source code removing the GUI part. This function is mainly applied for control system development. The control system has high reusability of control logic and can integrate components in the specified pattern, and makes prototyping at design important, so that this is an effective field for component-based system generation. Hence, the authors have aimed to make the control logic on SG extend from design to actual installation, and then in practical operation at generating a compact and high performance system by removing unnecessary GUI.

(5) Automatic generation of specification documents

The SG has a function for automatic generation specification document of the system. This enhances maintainability even in systems prepared on a trial and error basis. In addition, it also has a function for generating a test specification document for the system testing tool. This enables automatic ganeration of a test script for the system testing tool described in the next chapter.

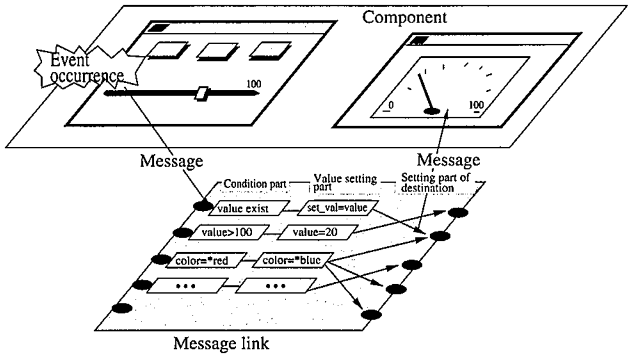

Fig. 3 Message propagation between components

4. System Test Support Tool

4.1 Outline

This system testing tool (hereinafter referred to as ST) is a tool to support effectively the GUI testing and system testing for the client program operating on X Window Systems to improve productivity and quality of the testing process.

The ST can be applied to the GUI testing and system testing in various shapes for individual purposes, and in particular it can exert its effectiveness at the stress testing[*8] and regression testing[*9].

Conventional commercial tools use the specified GUI library or are restricted to an operation on the X server expanded for testing support. The ST is characterized by excluding such restriction. It can automatically generate the test script for verification of speci-

[*7] C++: One of the programming languages.

[*8] Stress testing: A test to clarify the maximum load condition where normal operation of the test target system is not available. For example, it can access 100 or more terminals simultaneously for an issuing system which can be accessed from 100 terminals simultaneously.

[*9] Regression testing: A test to check whether the function available before modification is degenerated or not due to modification.

fications from the specification for system to be tested, which is an exceptional characteristic not observed in other commercial tools.

4.2 System structure

The ST is composed of four sections : a capture and play back section for capture and play back of mouse and keyboard operations and for verification of specifications ; a script generation section to generate automatically verification script for the specification on the basis of the specification to be tested; a specification browser to display the specification to be tested graphically on a CRT monitor and a script editing section available for an interactive operation. The capture and play back section is composed of three sections; a function for recording operations; a function for playing back operations and a GUI for the entire tool.

Input for the ST includes three kinds of specification files for the test target and the test script file. Output from the ST contains the testing result report. The intermediate generation files within the ST are the state transition diagram[*10] data file and structure diagram data file. Three kinds of test script files are used: a file based on the operation record; a file generated automatically and a file prepared by programming. The test script file is supplied from the operation record, automatically generated one and programming. The relation is shown in **Fig. 4** (script editing section and GUI section are not shown in the figure).

4.3 Functions

Main functions installed in the ST are described below.

(1) GUI operation capture and play back function

This function supports the testing work, which is first required in automatization of GUI testing. Concretely, the function records operations for the test target with a mouse and a keyboard as the test script and plays back them automatically at any time without any human operations. In order to play back, the test script is available directly prepared by programming instead of recording.

(2) Script automatic generation function for verification of specification

This function supports the testing design and testing case generation work, but it is left to the last in automatic GUI testing and the system testing because of the difficulty in realizing it. The ST realizes automatic generation of the test script, which includes all of the states and state transitions defined by the structure specification[*11], the state transition specification[*12] and the layout specification of the testing target. Concretely, on the basis of specification for application to be tested, this function causes the application to carry out an operation collecting the state transition. This enables study of the state of application during operation for verification whether it meets the specification.

(3) Testing target specification browsing function

This function displays graphically the structure specification and the state transition specification for the application system to be tested, allowing easy understanding by the test personnel. Concretely, it is a function for displaying the structure specification and state transition specification on the monitor as a graphical view. This enables a test personnel to build an image easily on how to construct an application, what components to use and how to transit the state.

(4) Interactive execution function

This function supports the testing data generation, testing execution operation, and the test result judgment by supplying visual information to help deepen the understanding of the test personnel during testing on the testing contents. With an exclusive test script editor, the script content can be extracted as an arbitrary unit during operation for interactive operation with the testing target.

4.4 Test script language

This tool understands a command sequence written in the test script to execute automatically the GUI testing and system testing according to the contents. It also enables the ST to conduct automatically all human operations to the testing target by combining appropriately tens of kinds of prepared commands.

The test script language developed for the ST is an expanded Tcl language[1] characterized by assembling the commands specified for GUI operation and testing operation and by improved readability and maintainability of the test script due to an object based notation.

Fig. 5 shows an example of the test script which can be carried out by ST.

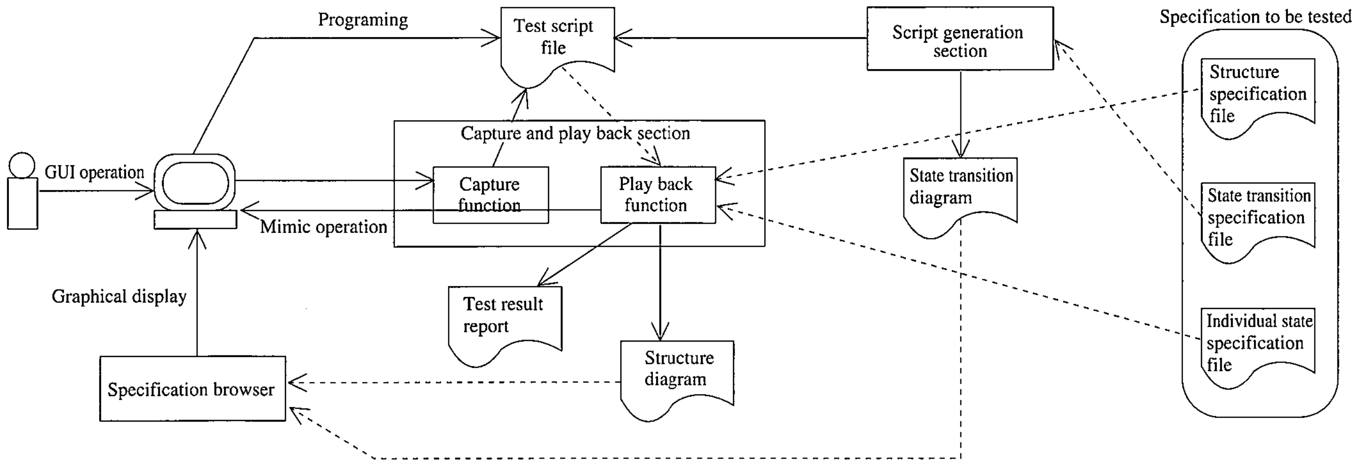

Fig. 4 Entire ST schematic diagram

*10 State transition diagram: A diagram expressing the state transition between states using nodes and edges.

*11 Structure specification: Specification in structure showing how to construct application and what program components to assemble.

*12 State transition specification: Specification showing that the application to be tested is transited by which input (event) from which state to which other state.

```
# Read data from a file to input to a text area.
MoveWin @(Sample) @(100,0) t100        # Setting of GUI initial layout
Raise  @(Sample)        t1000          # Set
set fd [open Sample.dat r]             # Open data file
set WTIME 100                          # Event occurring time interval
While {! [eof $fd] } {                 # File processing loop
  for {set i 1} {$i<10} {incr i}{      # Read a line to data
        gets $fd data
        if {$data == "Right"} {set product "R"}
        if {$data == "Left"} {set product "L"}
        # Click the ith botton in the window
        Click      @(Sample.button$i) t$WTIME
        # Type data string to the ith text of Sample window
        Type   '$data'  @(Sample.test$i) t10
  }
  •••
}
Closed $fd                               # Close the file
exec echo Sample_finished >@ stdout      # Finishing message output
```

Fig. 5 Example of test script file

5. Linkage between Component-based System Generation Tool and System Testing Tool

As mentioned above, assembling of components, each being of assured quality to some extent, does not always assure total quality of the assembled result. This requires, therefore, the system test to detect defects which can be found only after assembling.

In system generation by SG, information is held in the SG side not only on the GUI structure but also on state transitions, which delivers correct information on the system to the testing tool. Therefore, it is possible that the testing specification can be prepared automatically at the SG side based on which the test script is generated automatically at the ST side. Concretely, at the SG side as shown in **Fig. 6**, the structure specification, state transition table, and individual state specifications are generated automatically for transmission to the test tool side. This realizes automatic system testing for verification of the specification in the form containing the state transitions.

6. Applying Situation

The SG has been transferred so far to several platforms and used for construction of multiple real systems. For example, at the Yawata Works of Nippon Steel Corp., the SG has been utilized from the beginning of development. At present, the SG has been expanded to a control system constructing tool GOOD[2] which contributes to the control system development as a powerful tool. At the Ohita Works of Nippon Steel Corp., the ST was used to develop a thick plate shipping order generating system[3] to reduce the work load when shipping orders of thick plate and to improve the accuracy of order shipping. Since November 1995, this system has been operated satisfactorily and smoothly.

The ST is used for software development by the authors' group even during developing stage and executes not only quality assurance of the software but also detection of defects of commercial GUI libraries. Typical applications for practical projects include "Sumitomo Bank Off-balance project"[4], a large-scale system development with two million steps contracted by Financial Solutions Department of Electronics & Information Systems Div. This project uses repeatedly ST during GUI testing and system testing, contributing to detection of defects and shortening of the testing period.

7. Conclusions

This report described two tools for supporting construction of GUI-based application systems. The SG generates systems by the component-synthesis and ST automatically performs system testing in part. Linking both tools has successfully realized support from system generation to testing to some extent. These tools have been applied on a trial basis in Nippon Steel Corp. even during the development stage to reflect experience in actual system construction to the next stage with reasonable evaluation.

References

1) Ousterhout, J.: Tcl Overview. University of California at Berkeley, 1993
2) Sekiguchi, O. et al.: Shinnittetu Giho. (364), 19 (1997)
3) Miyanaga, Y., Kawabe, T.: CAMP-ISIJ. 9 (5), 957 (1996)
4) Nikkei Computer. (412), 156 (1997)

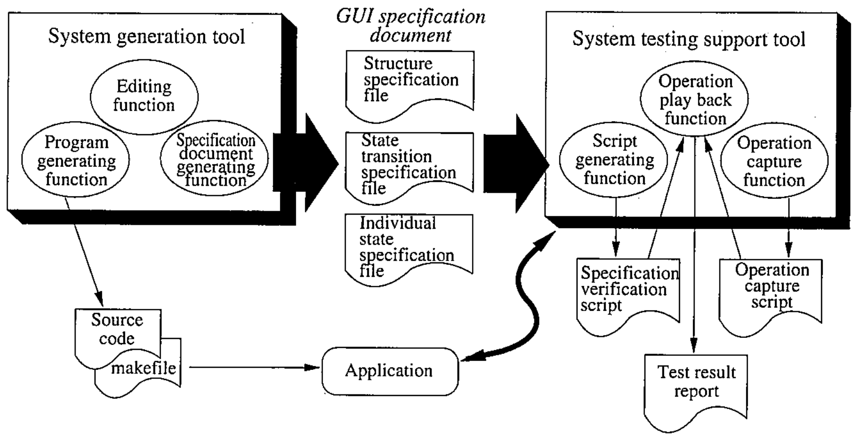
System generation tool
Editing function
Program generating function
Specification document generating function
GUI specification document
Structure specification file
State transition specification file
Individual state specification file
System testing support tool
Operation play back function
Script generating function
Operation capture function
Specification verification script
Operation capture script
Test result report
Source code
makefile
Application

Fig. 6 Automated system testing