行政院國家科學委員會專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　※
※　　　三層應用程式的測試工具之研製　　　※
※　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：V 個別型計畫　　□整合型計畫
計畫編號：90-2213-E-009-147-
執行期間：90 年 8 月 1 日 至　91 年 7 月 31 日

計畫主持人： 王豐堅
共同主持人：

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊工程系

中　華　民　國　91 年　12　月　19 日

行政院國家科學委員會專題研究計畫成果報告

計畫編號：90-2213-E-009-147-
執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：王豐堅　國立交通大學資訊工程系
計畫參與人員：許嘉麟等研究生

摘 要

許多三層式軟體均為流程為基的應用軟體。舉例來說，辦公室自動化軟體、物流控制軟體、ERP、等。這些流程為基的控管軟體之開發有的來自於自動化的發展環境，有的則利用傳統程式技巧、網路技術、以及資料庫相關技術。由於它們是三層式與流程為基的軟體，它們的開發與傳統軟體的開發方式不同，因此，他們的測試技巧自然也與其他軟體的方法大不相同，本研究成果敘述此類軟體測試的過程，包括相關工具的研發。

Abstract

Many three-tier programs are workflow-based application software. Examples are office automation software, ERP software and etc. These integrated software systems, founded on Workflow Management System (WfMS), can be developed in a generation environment associated environment with database. They can also be coded with conventional techniques including object-oriented programming, web-techniques and database. Or, to be more effectively, these two approaches are adopted at the same times. However, due to the characteristics of three-tier and workflow architectures, the development processes of this kind of software are different from others. So are the testing processes. The report presents the testing process of this kind of software, including the development of related tools.

Keywords: Workflow, WfMS, Three-tier Application, Software Testing

1. Introduction

Many three-tier application software contains workflow, the automation of a business process in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules [1]. With the growing complexity of business processes, different WfMSs have evolved [2] for example, FlowMark/IBM[3], Staffware/Staffware Corp.[4], InConcert/TIBCO Software Inc.[5], Ultimus/Ultimus Inc.[6], AgentFlow/FlowRing Tech.[7], to help organizations define, execute, monitor and manage processes in various product areas.

Most WfMSs provide test tools to assist in testing workflow processes, and these tools are developed from capture/playback techniques [8], for example, FlowMark/IBM [9]. However, these tools are concentrated on simulation and little on the automation of testing to facilitate the construction of workflow applications. Quality Software Testing Solutions [10] provides testing solution to EAI-based workflow applications, such as iPlanet integration server and Microsoft BizTalk server. [11] discusses the concept of analyzing workflow application based on the process state transition during the process life-cycle.

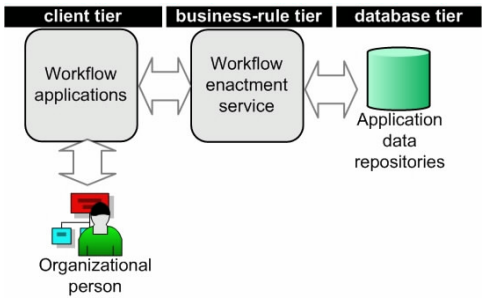

Figure 1. Three-tier architecture in generic WfMSs

According to Fig. 1, WfMSs are developed to comply with a three-tier software architecture: (1) workflow applications for the client tier; (2) the workflow enactment service for the business-rules tier; (3) and

audit/application data repositories for the database server tier. [12] presents a testing framework to test three-tier Web applications. The framework extends the architecture proposed in [13] and comprehensively evaluated [14] for traditional software test environments. The testing framework is also based on the architecture to enable the test designs to be reused. Therefore, this report presents a testing framework, based upon test designs for a three-tier software architecture, not only to test workflow processes, but also to automate the tests with little effort.

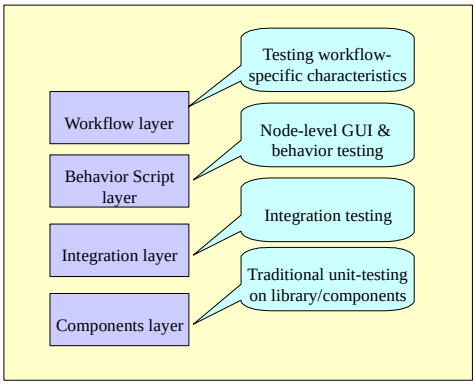

Figure 2. The layered structures of workflow application design.

As shown in Fig. 2, most of the integrated development environments for constructing workflow applications support the layered design structures. In the layered design structure, the workflow application designers model the business process in the top-down manner. The first one is workflow layer. In the layer, the developers draw the process flow diagram to denote the sequences, branch judgment or iteration of business processes. The second layer is about behavior scripting. In the layer, developers describe the work items for human or computer participants to perform on each process node. In the integration layer, the workflow application may interact with external application system through API-level integration. In the component layer, the developers reuse off-the-shelf software components or libraries to reduce the developing efforts. The correctness and reliability of the components or libraries are usually verified during their unit test phase.

In this testing framework, most of the features of workflow applications to be tested are only concentrated on the workflow layer and behavior script layer since the testing requirement in the integration and component layers are covered by traditional testing tools. Therefore, the testing framework is centered on an experiment on the behavior of workflow application with respect to their requirements, as for example on the workflow participant assignment.

Section 3 of this paper presents the testing framework and introduces the background and issues of testing workflow applications in Section 2. Section 4 describes the prototype implementation. Section 5 demonstrates a test experiment that uses the prototype. Section 6 concludes.

2. Background

2.1 General Concept of Software Testing

Traditional application development environments often provide a suit of typical test tools. Consider TestStudio, a suite of testing tools developed by Rational Software Corp., as for example in [15] and [8]. TestStudio supports implementation, execution, and evaluation of tests. The tools in TestStudio enable testers to create and execute GUI-based test scripts, focusing on the quality dimensions of reliability, function, and performance. TestStudio includes the following tools.

- Robot supports the implementation and execution of tests by enabling testers to create and play back GUI test scripts and compare actual results with expected results.
- LogViewer compares test results with expected results and presents reports to evaluate the test's execution.
- TestManager supports test planning, design, and evaluation and provides requirements-based test coverage and test

status reports.

- TestFactory supports reliability testing by automatically generating and executing test scripts and reporting on code-based test coverage.

2.2 Issues of Workflow Applications Testing

Compared with testing traditional application software, when testing a workflow applications, a testing engineers may perform the following steps:

(1) Select a process, instantiate the workflow process to be tested.

(2) Following the instruction of the test cases, perform the process-node-level testing. The work items defined in each process nodes may be performed through form-filling within a GUI screen. Therefore, testing in the step is just like testing traditional GUI application's operation.

(3) Complete the work items in a workflow node by submitting the task.

(4) Expect the occurrences of subsequent tasks caused by the task submission.

(5) Pick up the subsequent tasks to perform process-node-level testing.

In workflow application, the unit test of program behavior inside one workflow node may follow the strategy of that of conventional programs. However, in higher-level view, the goal of the workflow application testing aims to explore all of the task nodes or paths in the business process diagram.

3. The Testing Framework

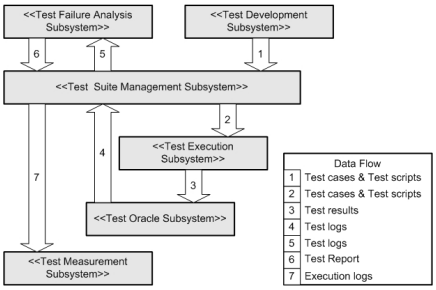

Figure 3. Dataflow in the testing framework

According to Fig. 3, the test suite management subsystem provides a warehouse to keep all test artifacts, and includes facilities to enable other subsystems to manipulate these artifacts. In the test execution subsystem, a test driver is constructed to drive process execution according to test scripts. In the test oracle subsystem, the test driver is equipped with several oracle functions to validate the test results at run time. These oracle functions can retrieve test results by querying the workflow engine via a workflow API, and compare them with expectations specified in the test scripts. The workflow API supported by the workflow engine includes many command sets, such as for session establishment, process control, process status, and data handling. Among the API, process status functions and data handling functions can be utilized to retrieve most test results, such as the details of a current process/activity instance, and application data. In the test development subsystem, engineers develop their test scripts and test cases according to the requirements to be assured in a workflow process. After test scripts are executed, the test driver validates the test results for test failure analysis. The test failure analysis subsystem will summarize the test results, and highlight detected errors in a test report. In the test measurement subsystem, path coverage is adapted to measure tests of workflow processes. In addition to these six subsystems,

this framework includes a subsection, suggested for generating test sequences according to the model-based test automation [16]. Herein, a test sequence refers to a series of defined test actions with test data and expectations to be executed in a workflow process. Such a set of test data and expectations will be referred to as a test case in the framework.

3.1 Test Execution Subsystem

In the test execution subsystem, engineers focus on implementing automation, and the construction of test tools to drive process execution. The critical problem lies in how to automate activities that involve human resources. As presented in Fig. 4, in such activities, workflow participants interact with workflow applications, each of which is specialized to assist participants to perform a specific task, such as accounting, inventory management, invoicing, or delivery scheduling.

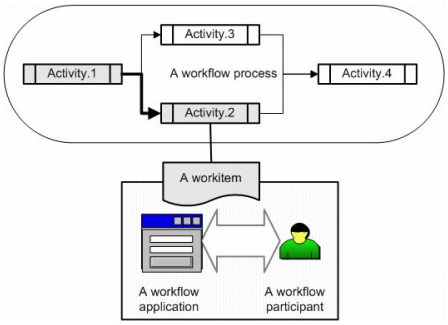

Figure 4. Scenario of activities that involve human resources

In test automation, human behavior in these activities is often simulated, and test tools have been developed to reproduce user interaction with tested applications according to test scripts, and have recorded pre-captured user operations, such as keystrokes and mouse activities. However, when tested applications change due to the addition of new features, the captured and associated test cases frequently fail. At this point, the tests are thrown into disarray and the capture must be performed again. Unlike such test tools, the test execution subsystem demands that workflow applications be scriptable, so that engineers have more flexibility in scripting their actions, and can more easily maintain test scripts, especially when requirements in workflow processes are frequently changed. Unfortunately, making ready-made workflow applications scriptable is hard, due to their GUI and redundancies. Engineers must implement a surrogate, referred to as a shell in the testing framework, for their workflow applications, and bind the surrogate with a script engine ensure its scriptability.

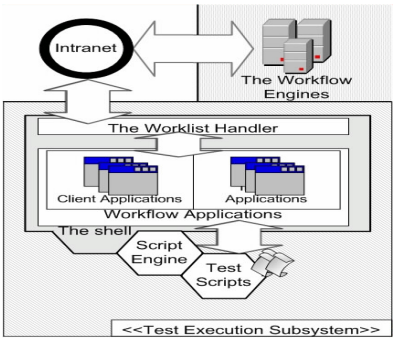

Figure 5. Relationships between components in the test execution subsystem

As discussed above and presented in Fig. 5, the test execution subsystem includes five components. From bottom to top, they are test scripts, script engine, shell, workflow applications, and worklist handler. Apart from the test scripts which were described under the test development system, other components are detailed in the following.

Before engineers construct such a test execution subsystem, they must prepare their experimental environment first. Engineers must add virtual members and virtual departments into their existent organizational model, or create a new organizational model, since interfering with the work of real users is unacceptable. According to Fig. 6, the API can be classified into two types - workflow API and application API. The former, offered by the workflow engines, supports access by workflow applications, and includes many sets of commands for operations on individual or collective process/activity

instances, and for manipulating worklists[2]. For example, process control functions enable the shell to create, start, and terminate an individual process. Many state-of-the-art WfMSs, including FlowMark and InConcert, have a complete application programming interface (APIs) [17][18] which allows everything that can be done through the user interface also to be done via an API: the API can be used to introduce tools that meet specific application requirements, for example the test-specific application.

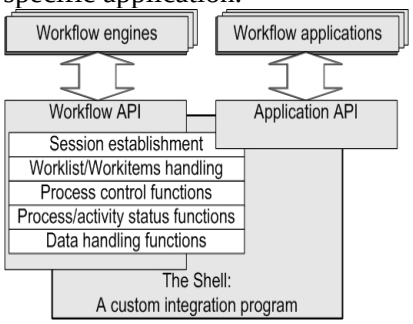

Figure 6. Application integration through application programming interface

Workflow applications can be driven without a front-end user in charge, simply by binding the shell with an instruction interpreter and defining a set of instructions and requisite arguments, each of which is mapped to a method of the shell. Given a text file that includes rows of instructions and arguments, the interpreter will drive workflow applications to request facilities and services from a workflow engine as usual.

However, the command-line mode of an interpreter is not flexible and is far from the control logic required to handle intricate or unexpected conditions encountered in process execution, so the instruction interpreter is replaced with an object-based script engine such as the JavaScript engine. Besides interpreting and running a script, the script engine can wrap an arbitrary object or class in a scriptable object with its fields and methods reflected as properties of the scriptable object.

There are two scenarios encountered in test execution and associated test strategies.

Scenario 1. As presented in Fig. 7, tested workflow process involves concurrent or parallel activities.

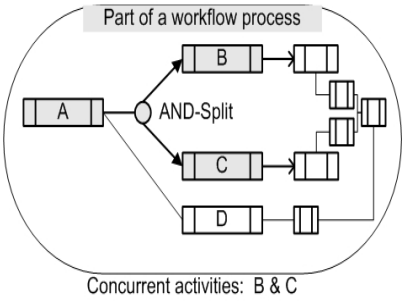

Figure 7. Part of a workflow process with concurrent activities

Process status functions, a command set in the workflow API, can be employed to query details of a process/activity instance, and can be used in the shell well. During test execution, engineers may query the workflow engine about which activities are successful after an activity is completed, and determine which is to be conducted first. Notably, the workflow engine can keep waiting activities in a stack or a queue. In any case, engineers have many strategies to determine which activity is to be exercised, such as always taking the one at the top of the waiting list, or performing a depth-first search. Alternatively, engineers can simultaneously drive multiple workflow applications to handle parallel activities.

Scenario 2. Tested workflow process involves an activity, which is randomly assigned to a participant.

Similar to scenario.1, during test execution, engineers may query the workflow engine, via process/activity status functions, to find out who is the chosen workflow participant for the selected activity, and may then begin this activity with the participant's identification. At this point, to facilitate test execution, engineers should unify the passwords of members in the experimental organization model.

3.2 Test Development Subsystem

In the test development subsystem, the primary task of engineers is to prepare test cases according to the requirements to be assured in a workflow process. During test automation, a common approach is to abstract test cases from test scripts, and articulate both only when test tools are executing tests. In script languages, a multi-dimensional array is well suited to keeping a test case, since its value assignment can be postponed until the test begins to be executed.

3.3 Test Oracle Subsystem

Automating the test actions is only half the battle. Engineers also require an automated method to determine whether the application works correctly. A test oracle is a method involving a function that determines whether the tested application has behaved correctly in response to a test action [16].

A test execution subsystem is constructed according to the shell; likewise, a test oracle subsystem can be constructed with the shell. Oracle functions must be added to the shell so that the shell can be used as engineers' eyes. Each oracle function specially examines the results or effects one particular action, such as the signing on or out of workflow participants, or the completion of an activity. Oracle functions may request the workflow engine, workflow applications, or even the organizational model via workflow API or application API, to retrieve adequate data, and determine whether these data are as expected. For example, data handling functions help engineers to retrieve workflow relevant data or application data from the workflow engine, and workflow relevant data are the most important data because they are used to determine the behavior of workflow processes.

3.4 Test Suite Management Subsystem

The test suite management subsystem stores and manages test cases, execution paths, test results, test reports, and other items. This subsystem also provides engineers with access interfaces and other subsystems to create, manipulate, query, or delete the above items. Now that WfMSs have collected information on the state transitions of a process instance into their audit data repositories for historical records, the test suite management subsystem needs only take care of the management of test scripts, test cases, test logs, and test reports. Restated, a test script with all its associated test cases, test logs, and test reports may be regarded as a test suite for a workflow process.

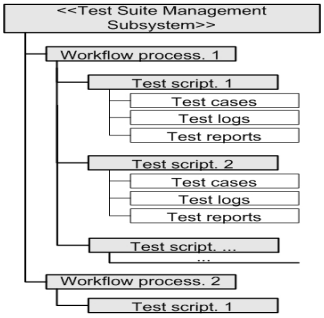

Figure 8. Test artifacts and their relational structure in the test suite management subsystem

According to Fig. 8, the test suite management subsystem can be simply implemented using a particular directory as a warehouse, and the test-related artifacts can be categorized into subdirectories according to their target workflow processes and their relationships. Engineers can specify an input directory to test the execution subsystem and an output directory to test the oracle subsystem. Test cases in the input directory will be exercised with the designated test script, and their test logs will be put into the output directory for test failure analysis.

3.5 Test Failure Analysis Subsystem

The test failure analysis subsystem, or the test failure analyzer, analyzes test logs, and then generates a test failure report for engineers, not only to determine the fraction of test cases that pass the test

oracle, but also to summarize which test cases or which test data failed in the test oracle.

Initially, the test failure analyzer requests test logs from the test suite management subsystem. Test failure analysis is easy because a test log is programmed in a tabular format. A simple approach is to check the validation column in the test logs. Based on these summarized errors, engineers can amend their workflow processes and proceed to subsequent test cycle.

3.6 Test Measurement Subsystem

The test measurement subsystem includes test coverage measurement and analysis. Each test coverage measure reports whether and how much a test criterion is adequately satisfied [8]. For example, the coverage is the percentage of statements covered by a set of test cases if the statement coverage and all-statements criterion are applied.

Path coverage must be altered to fit into the particularities of workflow. The workflow transition includes four routing confunctions such as the parallel routing that typically commences with an AND-Split and concludes with an AND-Join, and the sequential routing that typically commences with an OR-Split and concludes with an OR-Join. These four terms are outlined below and illustrated in Fig. 9.

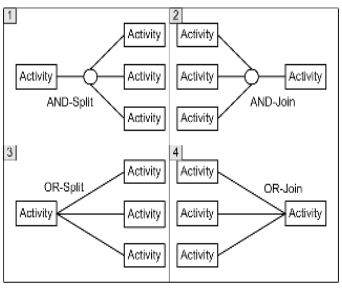

Figure 9. Simple flow diagrams of AND-Split, AND-Join, OR-Split, and OR-Join

1. AND-Split is a point in the workflow where a single thread of control splits into two or more threads, which are executed in parallel. Restated, AND-Split allows multiple activities to be executed simultaneously.
2. AND-Join is a point in the workflow where two or more parallel executing activities converge into a single common thread of control.
3. OR-Split is a point in the workflow where a single thread of control makes a decision regarding which branch to take when multiple alternative workflow branches are encountered.
4. OR-Join is a point in the workflow where two or more alternative workflow branches re-converge to a single common activity as the next step in the workflow.

Except the two conjunctions described above, the third conjunction that commences with an OR-Split but concludes with an AND-Join will certainly end with a fault in the process execution, reported to engineers in the test failure analysis subsystem. However, the fourth conjunction that commences with an AND-Split but concludes with an OR-Join will not be detected by most WfMSs. Although harmless to the sequential routing, the fourth conjunction will cause a blind spot in the path coverage when the processes involve the fourth routing rather than the expected parallel routing. Imagine that a process involved in an unwanted fourth routing could be completed by reporting one erroneous compound path, and that 100% path coverage could be mistakenly though to be satisfied since possible paths are visited and reported. This situation would clearly constitute a severe fault in the path coverage, and would be hard for engineers to detect. From this point of view, the testing framework refers to the fourth conjunction as erroneous parallel routing, in the remaining sections.

3.7 Generation of Test Sequences

In contrast to the capture/playback a set of test tools used to record test sequences

verbatim, model-based tests require testers to identify the state model of the tested application's behavior, enabling test tools to recognize the state in driving tested application, and thus determine what test actions are possible and what outcome is expected. Consider files in a windows folder as an example. Fig. 10 shows its state model, which is tabulated in Table 1.

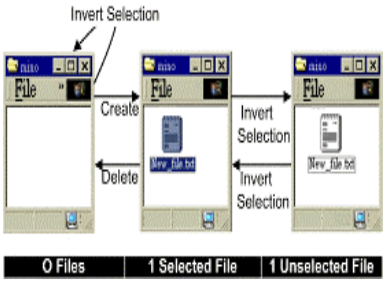

Figure 10. State model of files in a windows folder

Starting State	Action	Ending State
0 Files	Invert Selection	0 Files
0 Files	Create	1 Selected File
1 Selected File	Invert Selection	1 Unselected File
1 Selected File	Delete	0 Files
1 Unselected File	Invert Selection	1 Selected File

Table 1. State model of files in a windows folder

Model-based tests can examine the application's behavior in relation to the model's predictions. As the production cycle proceeded, developers write new features into the application. Testers can quickly update the model, and the tests continue to run.

Engineers need only to enact control logic to generate possible human behavior and test data/expectations in test scripts because workflow processes, which applications are being tested, are already well-defined state models with pre-defined transition sequences in WfMSs.

4. Experiment with the Test Framework

An intra-organization workflow process, named vacation application, is tested to show the feasibility of the testing framework. As shown in Fig. 11, the vacation application involves four workflow branches and no parallel routings, so that path coverage can be adopted as test coverage. Four kinds of participants are involved in the vacation application. As shown in Table 2, they are applicant, applicant's deputy, department manager, and personnel staff.

Workflow Participant	Activities
Applicant	Applicant requests, Rejection, Applicant acknowledges
Applicant's deputy	Approval of Agent
Applicant's department manager	Approval of Dept. Manager
Staff of the personnel department	Approval of Personnel Dept.

Table 2. Participant assignment for vacation application

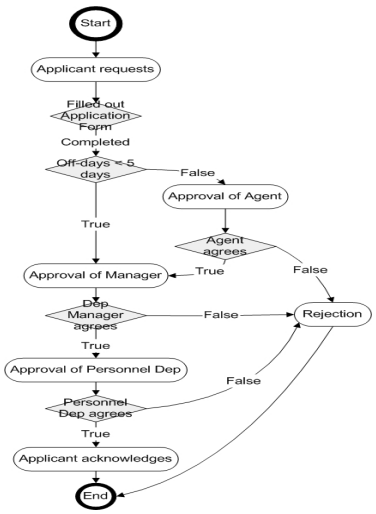

Figure 11. Transition graph of vacation application

Fig. 12 shows the document template for the vacation application. Tables 3 and 4 give test actions and expectations to be enacted in test scripts, respectively.

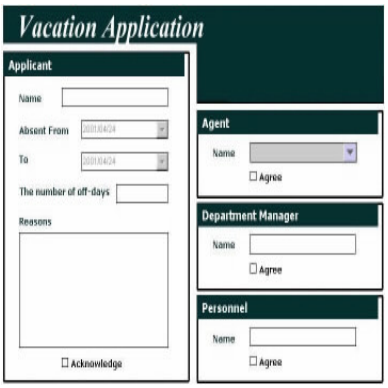

Figure 12. Document template for vacation application

Activity	Test actions
Applicant requests	1. The applicant gives his name, the starting and ending dates, the number of off-days, and reasons for vacation. 2. If the number of off-days > 5, the applicant needs to select a colleague as his agent.
Approval of Agent	Agent agrees or not.
Approval of Dept Manager	Dept Manager agrees or not.
Approval of Personnel Dept	Personnel Dept agrees or not.
Rejection	Do nothing.
Applicant acknowledges	The applicant acknowledges.

Table 3. Test actions to be enacted in test scripts for vacation application

Activity	Expectations
Applicant requests	**3 Check data integrity**
Approval of Agent	1. Check data integrity 2. If the number of off-days > 5, check whether participant assignment is in accordance with the field of agent's name in the document.
Approval of Dept Manager	1. Check data integrity 2. Check whether participant assignment is in accordance with the field of manager's name in the document.
Approval of Personnel Dept	1. Check data integrity 2. Check whether participant assignment is in accordance with procedural rules in vacation application.
Rejection	Nothing to do
Applicant acknowledges	Check data integrity

Table 4. The expectations to be enacted for test actions in Table 3

In the test experiment, a test script, enacted to generate test sequences randomly, achieves 100% path coverage in 6 minutes after being executed over 50 iterations. For large and complicated workflow processes, engineers may make test scripts generate test sequences according to the situations encountered during process execution, such as passed activities, or they may use test scripts with more client terminals to improve performance. As shown in Fig. 13, a test tool for path coverage reports the thoroughness of the tests of vacation application and tabulates the execution path of the selected process instances.

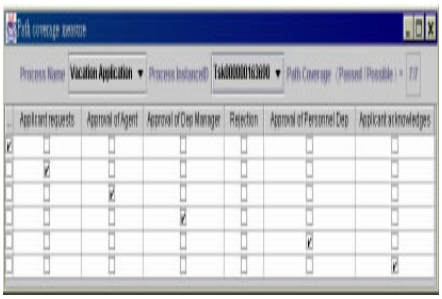

Figure 13. Path coverage of tests of vacation application

5. Conclusion

This report presents a testing framework for workflow processes based upon test designs for three-tier software architecture. The testing framework utilizes the properties of workflow processes, the resources supported by WfMSs, the facilities of the script engine, and the script language to automate the tests with little effort. The framework also helps engineers to maintain test artifacts especially when the requirements of the workflow processes frequently is changed. Besides, an experiment with the framework was done to indicate its feasibility. Moreover, based upon the testing framework, engineers can develop other test tools for performance testing and stress testing, which development left to future work.

[1] The Workflow Management Coalition Specification, "*Workflow Management Coalition Terminology & Glossary,*" Workflow Management Coalition, Feb 1999.

[2] The Workflow Management Coalition Specification, "*Workflow Management Coalition The Workflow Reference Model*," Workflow Management Coalition, Jan 1995.

[3] F. Leymann, D. Roller, "*Workflow-based applications*," IBM Systems Journal - Application Development, Vol. 36, No.1, pp. 102, 1997, http://researchweb.watson.ibm.com/journal/sj36-1.html

[4] Staffware Corp., http://staffware://www.staffware.com/.

[5] TIBCO Software Inc., http://www.tibco.com/.

[6] Ultimus Inc., http://www.ultimus.com/.

[7] Flowring Technology Corp., http://www.flowring.com/

[8] P. Kruchten, "*The Rational Unified Process: An Introduction,*" 2nd ed., Addison Wesley, 2000, pp. 205 – 206.

[9] F. Leymann, D. Roller, "*Workflow-based applications*," IBM Systems Journal - Application Development, Vol. 36, No.1, pp. 102, 1997, http://researchweb.watson.ibm.com/journal/sj36-1.html

[10] Quality Software Testing Solutions. http://www.classiq.com/

[11] T. Andersson, A. Andersson-Ceder, and I. Bider, "*State Flow as a Wbay of Analyzing Business Processes - Case Studies.*" (http://www.ibissoft.se/English/ExpReport.htm). Will appear in Logistics Information Management, Vol. 14, MSB University Press, 2002.

[12] J.-L. Huang, "*An Architecture for Web Application Testing Environment*," National Chiao-Tung University, Master Thesis, 1999.

[13] D.J. Richardson, "*TAOS: Testing with Analysis and Oracle Support,*" International Symposium on Software Testing and Analysis, March 1994, pp. 138—153.

[14] N.S. Eickelmann, D.J. Richardson, "*An Evaluation of Software Test Environment Architectures*," International Conference on Software Engineering, March 1996, pp. 353—364.

[15] Rational Software White Paper, "*The Rational Approach to automated Testing*," Rational Software Corp., http://www.rational.com/products/whitepapers/100581.jsp

[16] H. Robinson, "*Intelligent Test Automation,*" Software Testing & Quality Engineering Magazine (http://www.stqemagazine.com), Sep/Oct 2000, http://www.geocities.com/harry_robinson_testing/Intelligent_Test_Automation.htm & http://www.io.com/~wazmo/qa/#test_automation

[17] A. Dogac, L. Kalinichenko, M. T. Qzsu, A. Sheth, "*Workflow Management Systems and Interoperability*," the first edition, NATO ASI Series, 1998, pp. 384—385.

[18] L. Fischer, "*Workflow Handbook 2001,*" 1st ed., Future Strategies Inc., 2001, pp. 225—240.

Acknowledgment: The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC 90 – 2213 – E – 009 – 147.