Test Automation Technology to Reduce Development Costs and Maintain Software Quality

Haruto Tanno, Xiaojing Zhang, Keiichi Tabata, Morihide Oinuma, and Kazuhito Suguri

Abstract

As companies increasingly strive to reduce the cost of software while maintaining its quality, interest in technology supporting software testing is also increasing since testing is vital to maintaining quality, and it accounts for a large share of the total development cost. Currently a large part of testing is done manually, which is expensive and prone to mistakes and oversights. In this article, we introduce technologies that automatically generate comprehensive test cases and test data from software design documents, as a way to resolve these types of problems.

Keywords: software testing, integration testing, test design

1. Introduction

Software development is divided into processes, as shown in **Fig. 1**. Of these processes, testing is estimated to account for 50% of the total cost. However, most of this work is done manually, so it is an area where there is room for cost reductions. Further, faults that do not get exposed in the testing process will be released to market unresolved, so testing is really the last defense against poor quality. The fact that it is done manually, which results in errors and omissions, presents another problem in ensuring quality.

These issues could be resolved by raising the skill level of testing staff, by increasing the number of people involved, or by delaying delivery to provide additional testing time. As long as the work is done manually, though, errors and omissions cannot be entirely eliminated. In addition, these measures could lead to an increase in costs.

Given these circumstances, the NTT Software Innovation Center is working on a test automation approach that will use computers to replace work being done manually.

2. Current state of software testing support

The objectives of software testing are to verify that software has been implemented according to the design and specifications, and to reduce the number of defects. The testing process mainly consists of the five tasks shown in **Fig. 2**: test planning, test design, test execution, test reporting, and test management. In test planning, issues such as the time frame and allocation of resources for testing are decided based on the overall development plan. Test design involves clarifying the various tests that must be done and designing test cases comprehensively. In test execution, the test data are input for each of the test cases, and the software is run to check that it behaves according to design. In test management, real-time management of the state of test execution is carried out, and the test plan is revised if necessary. When all tests have been executed, the results are summarized in test reporting, completing the process.

We are working to support testing in the design and execution tasks, which are the main tasks in the testing process. Automation of these tasks can be very effective in reducing costs and ensuring quality.

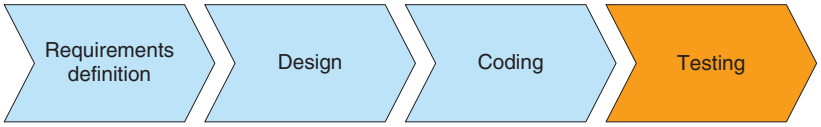


Fig. 1. Overall development process.

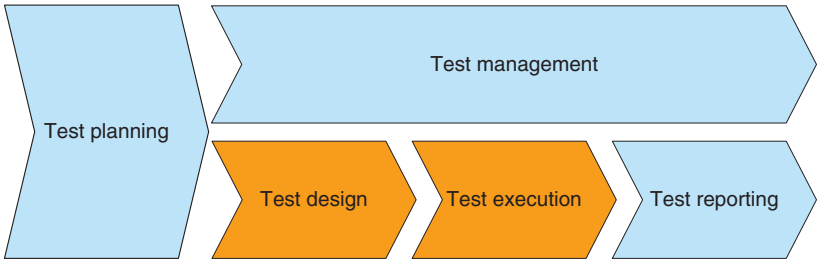


Fig. 2. Testing process.

Currently, a degree of automation has been achieved in test design and execution for unit testing of source code, with tools such as Parasoft Jtest*1 and JUnit*2. However, automation is still inadequate in integration testing, which checks the operation of software combining multiple modules, and in system testing, which tests entire systems.

Various tools are available to support the design of some functionality tests. For example, PictMaster (Microsoft) generates combinations of input values based on a combinatorial method called All-Pair, and Enterprise Architect (Sparx Systems) can generate test cases from module descriptions written in the UML (Unified Modeling Language) specification language. However, there are major barriers to introducing these tools in the development workplace. These barriers include the need for testing staff to have specialized knowledge of the tools and the testing technique which the tools use, and the need to write descriptions in an unfamiliar language.

There are also test execution tools such as Open2-Test, which automates testing of Web applications based on test scripts. However, even when using such tools, the test scripts and test data must still be prepared manually, and thus, there is still much room for further automation.

3. Research vision

Our vision is to reduce the cost and maintain software quality in the testing process by achieving automation of test design and test execution, as shown in **Fig. 3**. One feature of our approach is that it requires only the software design documents, which is the artifact of processes preceding testing, and no other new input needs to be specially created. The software design documents are input, comprehensive test cases are extracted, and test procedures, data, and scripts are generated automatically. This removes the need for testers to have specialized knowledge or to learn a new notation, so it should be possible to introduce it smoothly into the development workplace. Also, by using the test data and scripts generated during the test design, we can link tests with existing test execution tools to implement fully automated testing in one button click, from test design through to test execution.

4. Scope: Integration testing of enterprise applications

Our scope is integration testing of enterprise applications, which involves testing a system to check whether it operates correctly when a web browser, server-side processing, and database are integrated, as shown in **Fig. 4**. For example, when a search is performed using a search screen, the operations to be

*1 Parasoft Jtest: An automated testing tool from TechMatrix Corp.
*2 JUnit: A framework for automating unit testing.

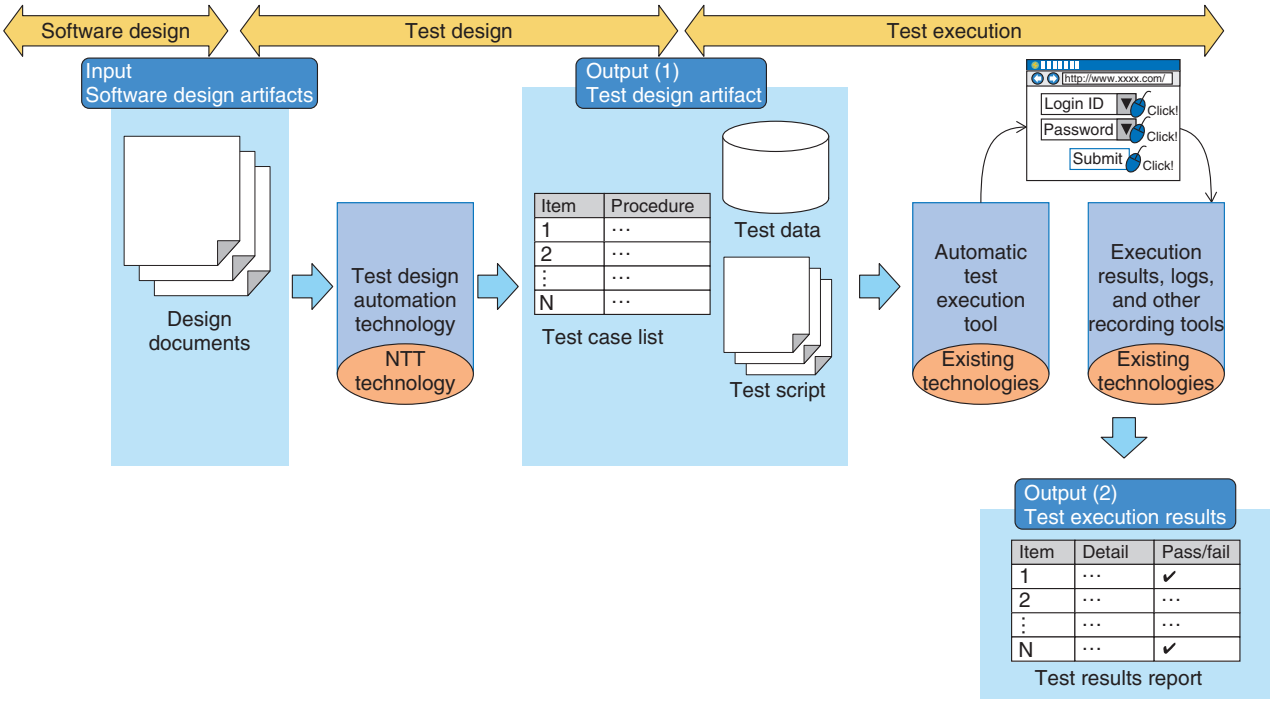


Fig. 3. Research vision.

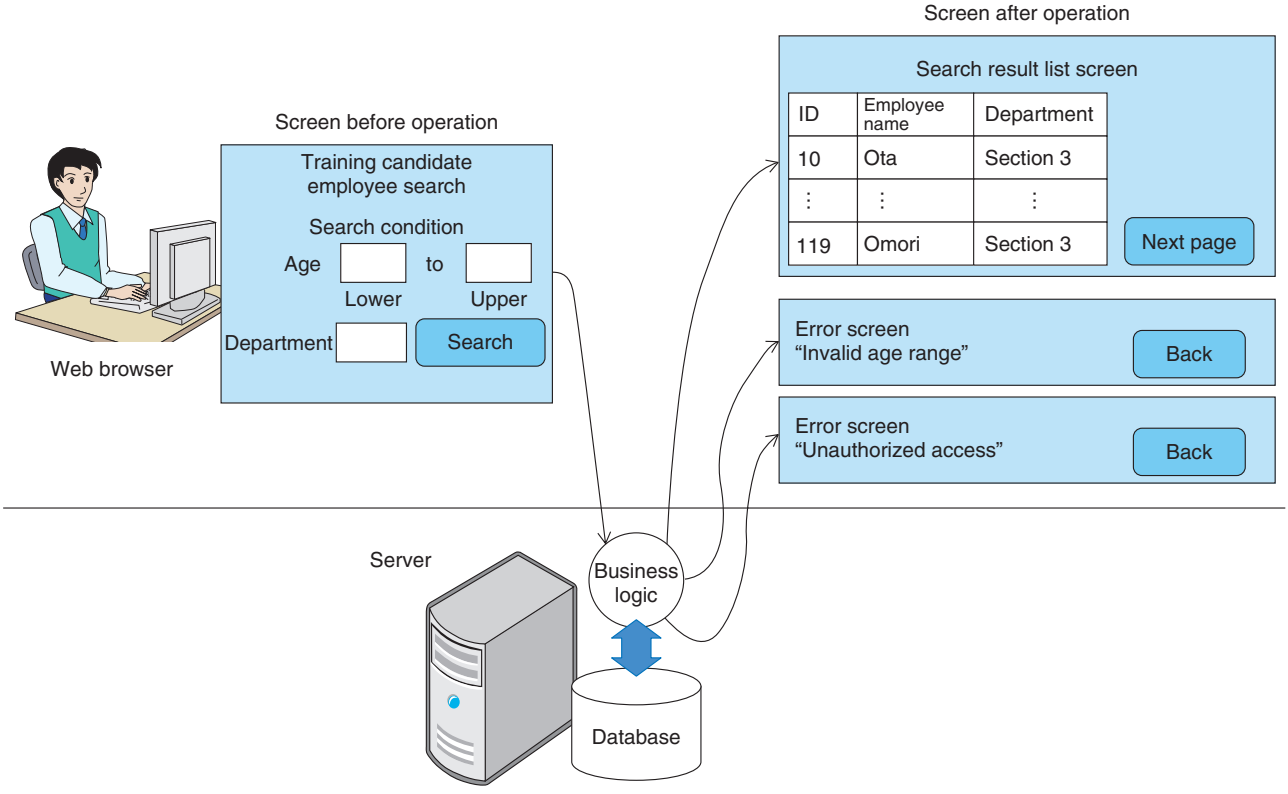


Fig. 4. Integration testing of enterprise applications.

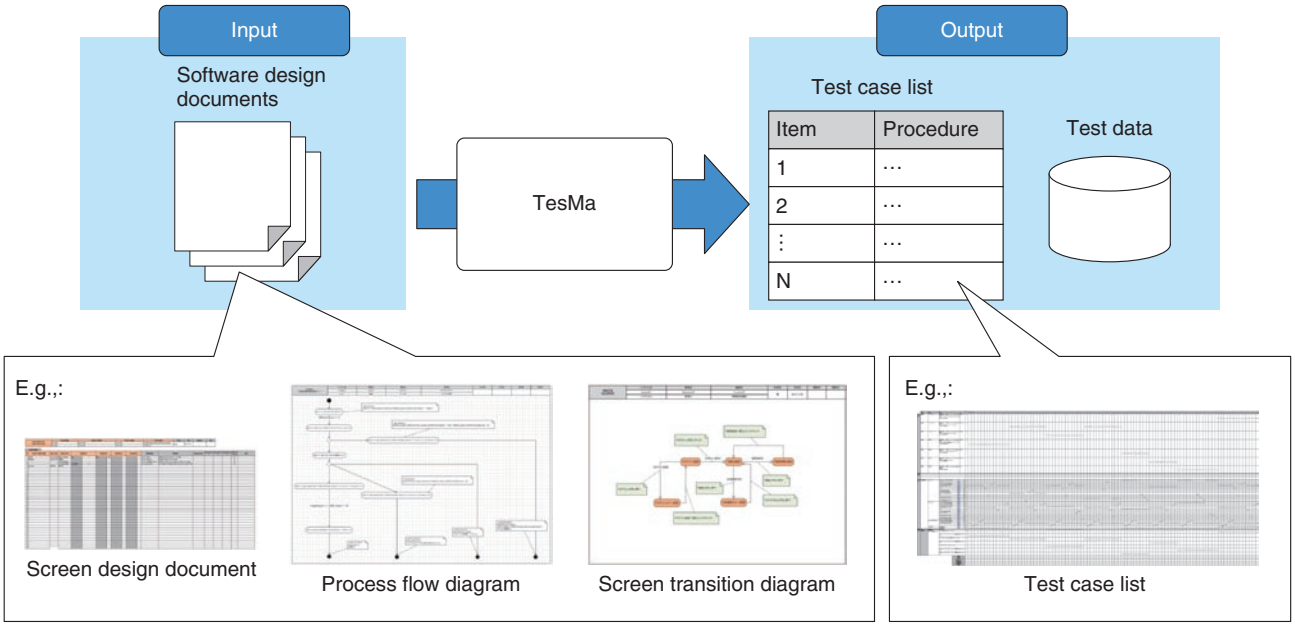


Fig. 5. Integration testing design support tool: TesMa.

checked include the search function to determine if the search occurred normally, as well as the transitions to the search results screen and to error screens. Therefore, in test design, test cases for all variations in operation and inputs, such as login IDs, must be covered without missing any test case which should be done. Test data include values to input on web browser screens and data to be inserted in the database beforehand for each test case, and these data must be created in the test design stage.

5. Integration testing design support tool: TesMa

We propose here an integration testing design support tool called TesMa, which automatically generates test cases and test data required for integration testing of enterprise applications from software design documents, as shown in **Fig. 5**. This capability can reduce the effort required for creating test cases and test data, which was previously done manually by testing staff, to the scope covered by TesMa. This tool extracts a comprehensive set of test cases, so it also avoids omissions that can occur when doing this work manually.

The tool has the following features:

- The input for the tool is a set of design documents written according to some set descriptive rules. These documents are the results of the design process, which is part of the existing development process. Thus, it has the advantage of being easy to introduce into the development workplace.
- The tool generates a comprehensive set of test cases [1] and test data [2], [3] based on processing patterns and input data variations. This helps to prevent omissions from occurring in manually generated test designs and also generates the test data required to execute each test case, so test execution is much easier.

These features enable the test tool to reduce the cost of integration testing, while maintaining software quality through a comprehensive test design.

6. Achievements and future prospects

TesMa technology is already being introduced in NTT Group companies, and its effectiveness in reducing costs while maintaining quality in comprehensive test design is becoming apparent. There are still some issues to be addressed involving the introduction of automatic test-data-generating technology into the development workplace, so some further improvements are needed. This will be a task carried out through consultation with the relevant companies. In the future, we will work on integrating our test

design automation technologies with automated test execution tools, and on achieving testing that is fully automated, from test design to execution. We intend to continue to advance our research and development step-by-step toward our goal of reducing costs while maintaining quality.

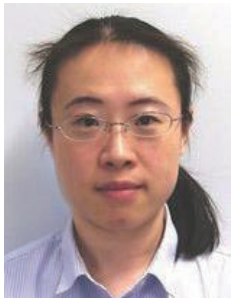
References

[1] X. Zhang, H. Tanno, and T. Hoshino, "Introducing Test Case Derivation Techniques into Traditional Software Development: Obstacles and Potentialities," Proc. of the 6th Testing: Academic and Industrial Conference—Practice and Research Techniques (TAIC PART), Berlin, Germany, 2011.

[2] X. Zhang and T. Hoshino, "Test Case Extraction and Test Data Generation from Design Models," Proc. of the 5th World Congress for Software Quality (5WCSQ 2011), Shanghai, China.

[3] H. Tanno, X. Zhang, and T. Hoshino, "Design-Model-Based Test Data Generation for Database Applications," Proc. of the 4th Workshop on Model-based Testing in Practice (MoTiP 2012), pp. 201–206, Dallas, TX, USA.

Haruto Tanno
Researcher, Software Engineering Project, NTT Software Innovation Center.
He received the B.E. and M.E. degrees in computer science from the University of Electro-Communications, Tokyo, in 2007 and 2009, respectively. He joined NTT in 2009. His interests include programming language and software testing. He received the Super Creator Award from the Information-technology Promotion Agency, Japan, in 2008 and the Distinguished Paper Awards from the Information Processing Society of Japan (IPSJ) in 2009 and 2013. He is a member of IPSJ.



Xiaojing Zhang
Researcher, Software Engineering Project, NTT Software Innovation Center.
She received the B.E. and M.E. degrees in computer science from Kyushu University, Fukuoka, in 2005 and 2007, respectively. She joined NTT in 2007. Her main interest is software engineering, especially model based development. She is a member of the Institute of Electronics, Information and Communication Engineers.

Keiichi Tabata
Researcher, Software Engineering Project, NTT Software Innovation Center.
He received the B.E. and M.E. degrees in computer science from Waseda University, Tokyo, in 2010 and 2012, respectively. He joined NTT in 2012. His interests include software engineering.

Morihide Oinuma
Senior Research Engineer, Software Engineering Project, NTT Software Innovation Center.
He received the B.E. and M.E. degrees in electrical engineering from Keio University, Kanagawa, in 1984 and 1986, respectively. He joined NTT in 1986. His current research interests include software engineering. He is a member of IPSJ.

Kazuhito Suguri
Senior Research Engineer, Supervisor, Software Engineering Project, NTT Software Innovation Center.
He received the B.E. and M.E. degrees from Toyohashi University of Technology, Aichi, in 1990 and 1992, respectively. Since joining NTT in 1992, he has contributed to the development of video-codecs, communication equipment, and content delivery platforms. His current research interests include software development processes, model-driven development, and software metrics. He is a member of IPSJ and IEEE.