Test-activity Analysis for Efficient Iterative Testing

Haruto Tanno, Hiroyuki Kirinuki, Takahiro Kawaguchi, Masaki Tajima, Morihide Oinuma, and Tatsuya Muramoto

Abstract

There is a growing demand for the early release of software while holding down costs. Software testing, which makes up a large portion of overall development costs and is essential to ensuring a certain level of quality in software, can be viewed as the cornerstone of quality, cost, and delivery in the development process. The NTT Software Innovation Center has developed technology that dramatically improves the efficiency of software testing and has made it available as open-source software. In this article, we introduce this technology.

Keywords: software testing, exploratory testing, test script

1. Importance of software testing

The software-development process is summarized in **Fig. 1**. In this process, software defects that could not be removed during testing are released in that state to the user, so testing is critical to software quality assurance. However, attempting to do all testing manually can be extremely costly. Users' needs have been changing rapidly, and software and hardware that constitute the operating environment have likewise been evolving at a rapid pace. This makes it necessary to revise software quickly as the need arises and release software updates frequently at short intervals. However, to conduct software releases frequently while maintaining a certain level of quality, testing cannot be limited to just the new additions. It is also necessary to conduct regression testing with respect to all existing features at every release to check whether they have been adversely affected by new features or the like, and this can also incur high costs. The NTT Software Innovation Center seeks to revolutionize testing—the cornerstone of quality, cost, and delivery in software development—and achieve a quantum leap in productivity in the software-development process.

2. Current state of software testing

The purpose of software testing includes checking that the software is behaving appropriately and reducing the number of software defects. The main tasks in the testing process are test design and test execution. Test design involves identifying test variations that should be carried out, exhaustively designing test

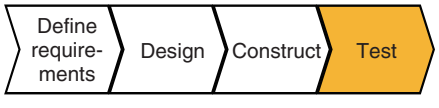


Fig. 1. Software-development process.

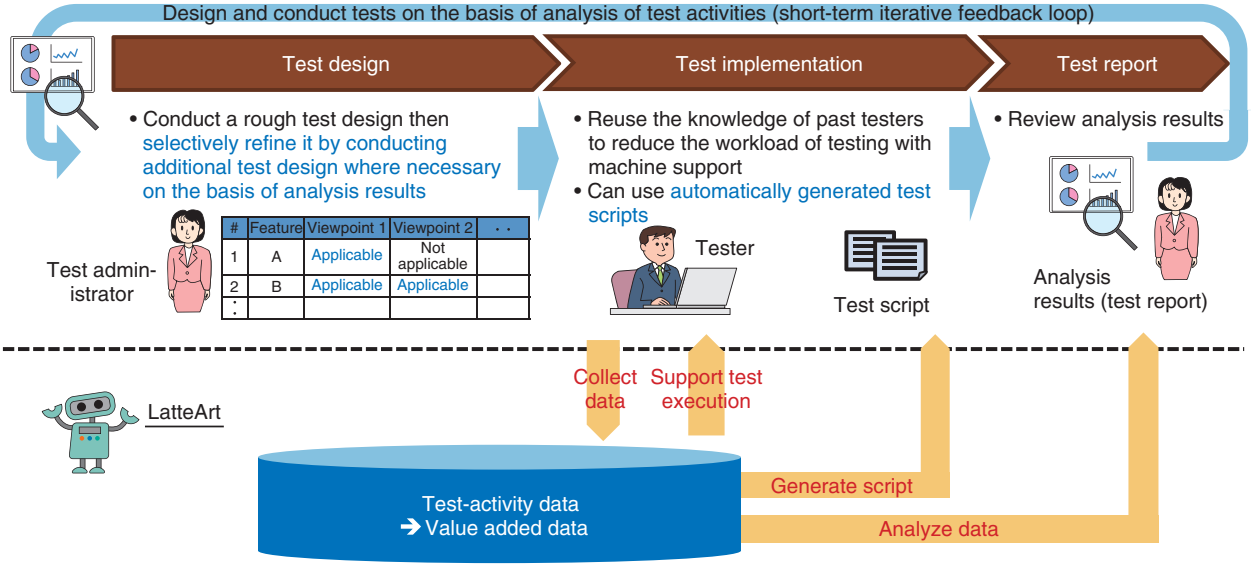


Fig. 2. The world we are aiming for.

cases, and for each test case, refining a specific procedure for executing the test and creating a script for automatic execution. Next, test execution involves providing input data for each test case, running the software, and checking whether the software is behaving as expected. We believe that there are three major issues with current tests.

The first issue is the high cost of designing exhaustive and detailed tests on the basis of conventional pre-planned index values. The pursuit of completeness to improve quality can take a massive amount of time. Moreover, to ensure that these tests are reproducible and auditable, a great deal of time and effort must be expended to create long-winded test-case tables containing intricate descriptions of the test procedures and other details of each test case.

The second issue is that executing tests requires skill and huge amounts of processing time. For each of the many test cases, it is necessary to follow the set procedure to manipulate and input information to the application under test then check the results. It takes much time to implement each test case in accordance with the correct procedure.

The third issue is the costs incurred in automating regression testing, which is required when issuing releases frequently in short cycles. Although many frameworks and libraries, such as JUnit and Selenium, are currently available for automatically executing tests, scripts must be created for such automatic execution, which can be very time consuming. To make it worse, a completed script does not mean that no more work is needed since it must be revised together with any revisions made to the software targeted for testing. This type of maintenance work is also labor intensive.

3. The world we aim for

To achieve ultra-high-speed development that can handle increasingly diverse and vague business requirements and rapidly evolving businesses, the approach taken by the NTT Software Innovation Center is to establish artificial intelligence (AI) development technology that can substitute or even excel in some human work and develop software through human-AI cooperation. To implement testing that facilitates ultra-high-speed development, we are aiming to achieve the world shown in **Fig. 2**.

In our approach, instead of haphazardly pursuing completeness, we select the locations that should be tested and concentrate our efforts there. In addition, we make successive judgments as to what locations to select and concentrate on by collecting and analyzing test-execution conditions and results. This approach solves the problems surrounding traditional exhaustive testing and achieves a quantum leap in testing efficiency. Since the collected data on the test-execution status and results includes detailed test procedures, it is possible to ensure that the test results are reproducible and auditable (addressing the first

issue). Furthermore, by collecting test-activity data and using them to automatically generate easy-to-maintain test scripts, it becomes easier to automate regression testing, which enables prompt releases after making software updates (addressing the third issue).

Finally, once a large quantity of test-activity data has been accumulated, it will be possible to reuse the knowledge of previous testers. This knowledge will not only be of assistance in the automatic execution of procedures that are repeated during test execution but will also make it possible to provide test recommendations to improve test quality. Therefore, we aim to achieve dramatic labor savings in the execution of tests (addressing the second issue).

The NTT Group develops many business applications that use web applications as front ends and tests these applications through integration testing. In this article, we describe LatteArt, our technology for integration testing, which requires efficiency improvements, and how it addresses the first and third issues mentioned above.

4. LatteArt: Analyzing test-activity data for efficient iterative testing

A business application has many use-case scenarios, features, and screens, and each screen may have many combinations of input patterns. This incurs high costs in traditional exhaustive testing. While there are tools that support test design through automatic design of exhaustive testing based on the type of model (e.g., software-design model), they have the drawback of incurring additional costs for the creation of software-design models. In addition, automating the testing of web applications requires the creation of test scripts for executing screen operations automatically. Such test scripts can be created by means of, for example, capture & replay tools such as SeleniumIDE without requiring advanced skills, but creating scripts in this manner still requires work. Moreover, because test scripts recorded using capture & replay are not modularized, they are difficult to modify and have low maintainability.

To solve these problems, we developed a test-activity data-analysis technology called LatteArt that implements efficient iterative testing. LatteArt has the following features.

(1) **Test-activity data collection**: Automatically accumulates test-activity data consisting of the tester's web application logs and screenshots as well as information input by the tester while executing tests, including test objectives, discovered bugs, and other findings.

(2) **Analysis of test-activity data**: Analyzes test-activity data that have been collected automatically and allows them to be visualized with various data models [1, 2]. For example, these data can be visualized using the models shown in the sequence diagram and screen-transition diagram of **Figs. 3** and **4**, respectively. This feature makes it possible to check the sufficiency of tests from time to time and supplement the minimum necessary tests to ensure that testing is conducted effectively. In addition, based on these test-activity data, it automatically generates test-case tables and information equivalent to test results, eliminating the need for long-winded test-case tables containing intricate descriptions of the test procedures that have thus far been created.

(3) **Test-script generation for regression testing**: This feature automatically generates a modular and highly maintainable test-script template based on page object patterns from test-activity data [3]. It also automatically generates documentation [4] to accompany and improve the readability of test scripts when editing. This documentation also includes screen-transition diagrams to show what screen transitions the test will execute and screenshots that show the elements being manipulated. Examples are shown in **Figs. 5** and **6**. This makes it easy to implement and maintain test scripts and allows regression testing to be automated with little effort.

5. Achievements and future outlook

We are currently evaluating the application of LatteArt through joint experiments with NTT Group companies and are receiving positive responses from development sites. LatteArt has been well received in presentations at academic societies in Japan and has received a number of awards. Since short-term iterative testing using LatteArt is a new testing method, we believe it is important to create a platform with which this new testing method can be understood and widely accepted by stakeholders (developers and system owners) both inside and outside the NTT Group. To this end, we have launched a LatteArt open-source software (OSS) community and released its source code on GitHub [5].

Going forward, we will leverage the data obtained

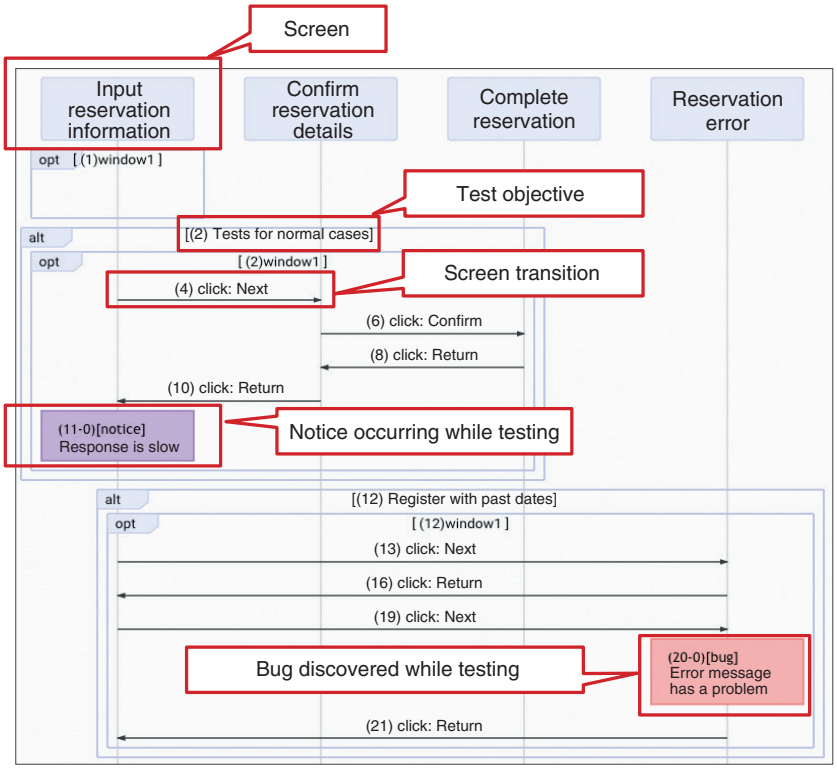


Fig. 3. Sequence diagram.

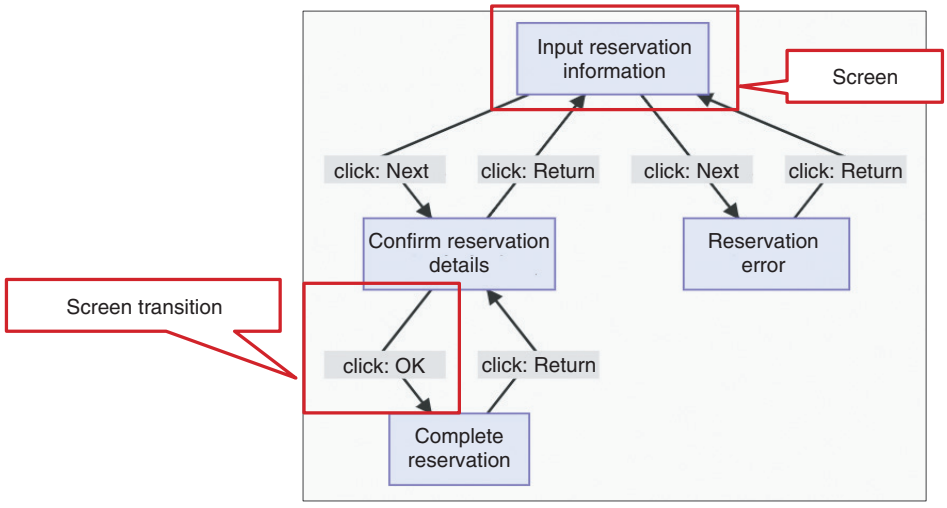


Fig. 4. Screen-transition diagram.

in practice to extract the knowledge of past testers from the large amount of accumulated test-activity data. By doing so, we aim to provide test recommendations and automate the preliminary groundwork that is repeatedly executed during testing, making testing much less labor-intensive and further improving its overall efficiency. In addition, we intend to develop a test-innovation ecosystem centered on the

```
doGotoNext({
  reserveMonth,
  // ・・・・・
}) {
  this.reserveMonth.setValue(reserveMonth);
  this.reserveDay.setValue(reserveDay);
  // ・・・・・
  return new ReservationDetailsConfirmation( );
}
```

Fig. 5. Test script.

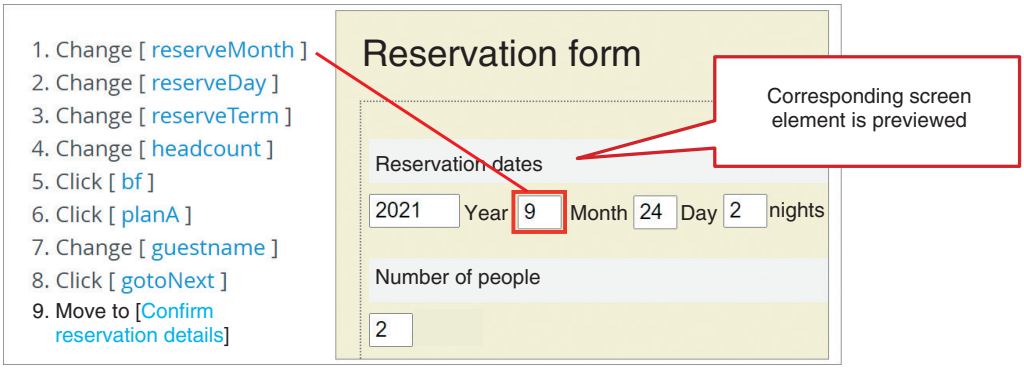


Fig. 6. Test-script documentation.

LatteArt OSS community through collaborations not only within the NTT Group but also with external organizations such as companies and universities to provide access to diverse knowledge from industry and academia. To achieve the world we are aiming for, we hope to continue making steady advances in our research and development one step at a time.

References

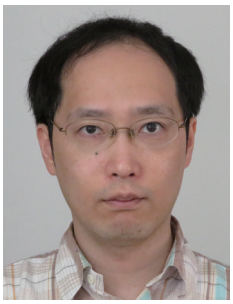
[1] I. Kumagawa, A. Mineo, H. Tanno, H. Kirinuki, and T. Kurabayashi, "SONAR Testing, New Testing Method that Ensures Both Efficiency and Objectivity," Software Quality Symposium 2019, Tokyo, Japan, Sept. 2019 (in Japanese).

[2] H. Kirinuki, T. Kurabayashi, H. Tanno, and I. Kumagawa, "Poster: SONAR Testing – Novel Testing Approach Based on Operation Recording and Visualization," Proc. of the 13th IEEE International Conference on Software Testing, Verification and Validation (ICST 2020), pp. 410–413, Porto, Portugal, Oct. 2020.

[3] H. Kirinuki, T. Kurabayashi, H. Tanno, I. Kumagawa, and K. Nagata, "Novel Testing Approach Based on Exploratory Testing and Operation Recording," IEICE Tech. Rep., Vol. 119, No. 56, KBSE2019-7, pp. 43–48, May 2019 (in Japanese).

[4] M. Tajima, H. Kirinuki, and H. Tanno, "Automatic Generation of Documentation to Facilitate Comprehension of End-to-end Test Scripts," Fundamentals of Software Engineering 28, Japan Society for Software Science and Technology FOSE 2021 (Lecture Notes/Software Science), pp. 177–178 (in Japanese).

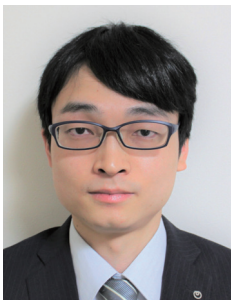
[5] LatteArt on GitHub, https://github.com/latteart-org/latteart/



Haruto Tanno
Senior Research Engineer, Software Engineering and Operation Technology Project, NTT Software Innovation Center.
He received an M.E. in 2009 and a Dr.Eng. in 2020 from the University of Electro-Communications, Tokyo. He joined NTT in 2009. His research interests include software testing and debugging. He is a member of the Information Processing Society of Japan (IPSJ).

Hiroyuki Kirinuki
Researcher, Software Engineering and Operation Technology Project, NTT Software Innovation Center.
He received an M.E. from Osaka University in 2015 and joined NTT the same year. His research interests include software testing and empirical software engineering. He is a member of IPSJ.

Takahiro Kawaguchi
Researcher, Software Engineering and Operation Technology Project, NTT Software Innovation Center.
He received an M.E. from Wakayama University in 2015 and joined NTT in 2021. His research interests include software testing.



Masaki Tajima
Researcher, Software Engineering and Operation Technology Project, NTT Software Innovation Center.
He received an M.E. from Shinshu University, Nagano, in 2017 and joined NTT in 2019. His research interests include software testing and automatic programming.

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

Tatsuya Muramoto
Senior Research Engineer, Supervisor, Software Engineering and Operation Technology Project, NTT Software Innovation Center.
He received an M.E. from University of Tsukuba, Ibaraki, in 1996 and joined NTT the same year. His current research interests include software engineering.