# Practical Combinatorial Testing Approaches: A Case Study of a University Portal Application

Mary Frances Moore and Sergiy Vilkomir
Department of Computer Science
East Carolina University
Greenville, NC 27858 USA
{moorema, vilkomirs}@ecu.edu

***Abstract*—Time and quality are important factors when determining the proper approach for software testing. A software program can often be used in various environments (different platforms, operating systems, browsers, networks, etc.) and require thorough testing to provide high quality and reliability in different configurations. Combinatorial testing is an effective approach to testing hardware and software configurations. However, testing resources are often restricted in real practice. Because business goals require different testing methods, there is no best one-size-fits-all testing approach. For this reason, we experimentally investigated and analyzed several combinatorial approaches based on Each Choice and pairwise methods (with and without the consideration of operational profiles) through the testing of an Adviser Scheduling application located in a university web portal. Test sets with various configurations were generated according to six different combinatorial strategies. The Advanced Combinatorial Testing System (ACTS) tool, which was provided by the National Institute of Standards and Technology (NIST), was used to generate pairwise test sets automatically. The case study software application was retested for each of the proposed testing approaches, and the results were compared after taking into account the number of test cases and the corresponding detected faults. Based on this analysis, we provide recommendations for the selection of testing approaches to align with different business goals. The recommendation chosen for the university web portal application allowed for improved quality and reduced time for software testing.**

***Keywords—Combinatorial testing; pairwise; Each Choice; Operational Profile***

## I. INTRODUCTION

A software program can often be used in various environments, such as different platforms, operating systems, browsers, networks, etc., and it requires comprehensive testing for many configurations to provide high quality and reliability. One of the best approaches in this situation is combinatorial testing [1, 2], which has been confirmed to be practical and effective [3-6].

Combinatorial *t*-way testing requires that any combination of values of any *t* testing parameters or configuration items should be included in some test case. This type of testing is often used for *t*=1 (Each Choice testing) or *t*=2 (pairwise testing) [7, 8]. Each Choice covers all values of all parameters, but it does not consider combinations of values. Pairwise testing covers all values *and* combinations of each value with all others, i.e., it covers all pairs of values. A larger value of *t* increases the effectiveness of *t*-way testing, but this requires more test cases, obliging testers to compromise between desirable effectiveness and available testing resources.

Testing resources (time, money, and human resources) are often restricted in real practice. A company's business goal might include improving the quality or effectiveness of testing while keeping the same degree of testing or even reducing the number of test cases, while still maintaining the appropriate level of testing quality. Because different business goals require different testing approaches, there is no "best" testing approach. Sets of different approaches should be considered to select one suitable for the current situation and the specific business goals.

In real practice, some configurations are more common than others. For example, when a particular software application is accessed by many users, Internet Explorer may be used more often than Chrome, Windows 8 more often than Windows Vista, etc. This can be described using an operational profile, which is a quantitative characterization of how a system or software will be used [9]. To achieve trustworthy test results, software testing should be performed according to the operational profile, namely, the proportion of tests for different configurations should approximately reflect the occurrences of these configurations in the software's real usage [10, 11]. However, combinatorial approaches treat all testing configurations equally. In order to reflect the operational profile, these approaches should be modified and the number of configurations extended.

In this paper, we analyzed several combinatorial testing approaches based on Each Choice and pairwise methods, with and without consideration of an operational profile. The paper is organized as follows: Section II explains the organization of our investigation and the proposed testing approaches. The Advisor Scheduling application, used as a case study for applying combinatorial testing, is described in Section III. Section IV contains detailed information on test configurations that were generated according to different combinatorial testing methods. Section V provides experimental testing results of these configurations and analyzes the effectiveness of the proposed approaches. The conclusions are presented in Section VI.

DOI reference number: 10.18293/SEKE2016-045

## II. Testing Approaches and Organization of the Investigation

Our experimental investigation included retesting the Advisor Scheduling application using six combinatorial approaches based on Each Choice and pairwise testing. This application, used as a case study and described in the next section, has already been deployed and tested in depth but without using combinatorial approaches. The first version of this application, which contained all the bugs found and removed in the original testing phase, was retested. Our results were compared to the original test results, taking into account the number of test cases and the corresponding bugs found for each combinatorial approach. Our goal was to find an approach comparable in size to the number of original test cases while providing greater effectiveness in fault detection.

We investigated the following six approaches:

- Each Choice
- Each Choice with consideration of the operational profile
- Each Choice with additional parameters
- Pairwise
- Pairwise with consideration of the operational profile
- Pairwise with additional parameters

The "pure" Each Choice and pairwise approaches used the same configuration parameters and their values as the initial testing. However, Each Choice required significantly fewer test cases. Pairwise required approximately the same number of test cases as the initial testing but provided much better coverage of parameter value combinations. Taking into consideration the operational profile, we suggested using the same method for Each Choice and pairwise. Usually, the precise reflection of operational profiles requires using probabilistic models and a large number of test cases [12], but this was not possible for practical testing in our case study. An approximate approach to consider operational profiles has been suggested by Kuhn et al. [1, pp. 61-64] by adding additional values of parameters. We suggested another simple approximate method of adding one or two additional test cases into the test sets with the most frequently used parameter values. This made the distribution of values in test sets closer to the real operational profile.

Because combinatorial approaches require a small number of tests, it was possible to add additional configuration parameters and still have a comparable number of test cases compared to those in the original testing. In turn, new parameters allowed testing new configurations that were not tested originally. Detailed analysis of the generated test cases (Section IV) and the results of testing for different approaches (Section V) allowed for the provision of practical recommendations for the selection of testing approaches to align with different business goals.

## III. A Case Study: Advisor Scheduling Application

### A. Description

Many universities use a web portal to provide students, faculty, and staff with access to valuable data and applications within one location. Access to a portal is usually granted with university credentials. These credentials validate the access the user should have. For instance, when an employee logs into a portal, he or she may have access to applications such as Pay History and Workplace Training. However, when a student logs in, he or she would not get access to those applications. Instead, other applications would be available such as GPA Calculator, Grade History, and Course Schedule. Frequently, applications are available to multiple user types, but each user is granted a different level of access based on need. For example, an employee user type could view announcements in an application, but an administrator could add, edit, and delete the announcements. Each application within a portal is customized to fit the needs of all users having access to the application.

Before each application within a portal is released to its users, it must be tested thoroughly to ensure that it works for all user types. Testing configurations should be created to include user access testing, along with other parameters such as browser, operating system, etc. Testing multiple configurations can ensure that the application is ready for all users who will access the application in different ways. This case study reviews the original testing process for an Advisor Scheduling application in a university portal and proposes and implements practical combinatorial testing approaches.

The Advisor Scheduling application is used for advising processes specified by an Academic Advising Collaborative. The application has four main functions: scheduling, accepting appointments, viewing academic information, and processing appointment outcomes. The scheduling feature and processing appointment outcomes feature is accessible by all users of the application. The accepting appointments and viewing academic information features are available only to advisors.

There are three different user permission groups in the Advisor Scheduling application: administrators, advisors, and delegates. Each group is given limited access to the application based on the permission group they have been assigned. For users to access the Advisor Scheduling application, they must first log into the university portal. Once they have done so, using their university credentials, they can find the Advisor Scheduling application in the portal's Tools section.

The application's interface allows advisors to manage and perform routine tasks with ease. After opening the application, the advisor will see their calendar, the advisee roster, and outstanding actions. The outstanding actions list contains appointment requests created either by delegates within the Advisor Scheduling application or by students using the student scheduler application (which is a separate entity from the Advisor Scheduling application). When an advisor accepts or rejects an appointment displayed in their Outstanding Actions list, the calendar and advisee roster is updated accordingly. After accepting an appointment, the calendar immediately displays the appointment in its allotted time slot. At the same time, the advisee roster adds the student for the newly created appointment to the top of the roster. If rejected, the appointment is removed from the Outstanding Actions list and not added to the calendar. Fig. 1 displays a portion of an adviser's calendar with student appointments.

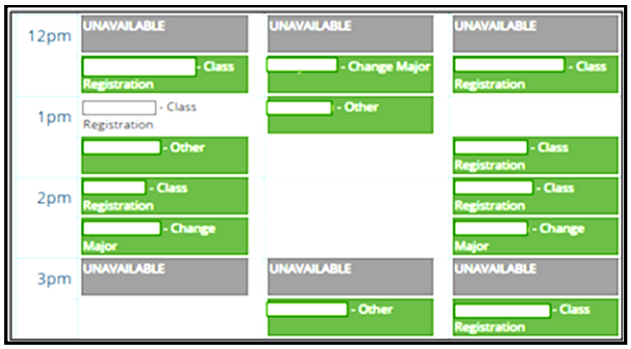


Figure 1. Adviser Scheduling Application – Redacted for Confidentiality

### B. The Original Testing Phase

When originally testing the Adviser Scheduling application, one test set consisting of 11 configurations of parameters was used, with a total of 121 tests completed for the application's 21 test cases. The configuration parameters and their values included:

- Browser: IE, Chrome
- User type: Faculty, Administrator, Delegate
- Operating System: Windows 7, OS X
- Network: On Campus–Secured, On Campus–Unsecured, Off Campus

These configurations were determined by the testing time allotted as the project's deadline approached, along with tester knowledge of the portal and experience of the types of bugs normally found. The original testing procedure was found to be adequate and thorough; however, there were not enough resources to provide mobile testing. Table 1 shows the configurations of the parameters listed above that were used in testing and the number of tests completed for each configuration.

TABLE I. ORIGINAL TESTING CONFIGURATIONS AND NUMBER OF TESTS COMPLETED

| Config | Browser | User Type | Operating System | Network | Tests |
|---|---|---|---|---|---|
| 1 | IE | Faculty | Win 7 | On Campus–Secured | 21 |
| 2 | IE | Admin | Win 7 | On Campus–Secured | 10 |
| 3 | IE | Delegate | Win 7 | On Campus–Secured | 12 |
| 4 | IE | Faculty | Win 7 | On Campus–Unsecured | 7 |
| 5 | IE | Faculty | Win 7 | Off Campus | 7 |
| 6 | Chrome | Faculty | Win 7 | On Campus–Secured | 21 |
| 7 | Chrome | Admin | Win 7 | On Campus–Secured | 10 |
| 8 | Chrome | Delegate | Win 7 | On Campus–Secured | 12 |
| 9 | Chrome | Faculty | OS X | On Campus–Secured | 7 |
| 10 | Chrome | Faculty | OS X | On Campus–Unsecured | 7 |
| 11 | Chrome | Faculty | OS X | Off Campus | 7 |

### C. The Operational Profile

According to the data collected by the Academic Technologies team, 42 percent of users access the Adviser Scheduling application with Internet Explorer. Faculty are the only user type to have access to all 21 of the application's test cases, while the other user types perform auxiliary functions, making the faculty user type the most common. Due to the fact that the standard operating system provided to university faculty is Windows 7 connected to the on-campus secured network, they are most commonly used to access the application. In addition, the data collected by the Academic Technologies team suggested that 16 percent of user traffic comes from mobile devices; however, that percentage has risen and will continue to rise due to mobile device and tablet popularity and convenience. When testing with the Each Choice and pairwise testing methods, the operational profile data can be considered while determining the test sets as a way of tailoring the test sets to the user majority.

## IV. APPLICATION OF THE TESTING APPROACHES

The two methods being explored through re-testing of the Adviser Scheduling application are Each Choice testing and pairwise testing, both with and without consideration of the application's operational profile. The Each Choice testing method provides test sets that allow for a significant reduction in the amount of testing while retaining or improving the testing quality. The pairwise testing method provides test set sizes similar to or slightly larger than the original testing approach.

With both the Each Choice and pairwise testing methods, variations can be made to the test sets to tailor the methods to testing needs. Three sets of testing approaches were used for both methods to test the Adviser Scheduling application. For each testing method, at least one approach aims to reduce the number of required test configurations, one takes the application's operational profile into consideration, and one aims to increase the test coverage and number of detected bugs.

### A. Each Choice Testing Approaches

The three test sets (Tables II, III, and IV) were created using the Each Choice testing method. The goal of the results from these tables is to maintain testing quality while significantly reducing the number of required tests.

Each Choice Test Set 1: This Each Choice testing table includes the application's original testing parameters shown in Table II. The table covers every parameter tested originally at least once using three configurations.

TABLE II. EACH CHOICE TEST SET 1

| Config | Browser | User Type | Operating System | Network | Tests |
|---|---|---|---|---|---|
| 1 | IE | Faculty | Win 7 | On Campus–Secured | 21 |
| 2 | Chrome | Admin | OS X | On Campus–Unsecured | 10 |
| 3 | IE | Delegate | Win 7 | Off Campus | 12 |

Each Choice Test Set 2: With the resources provided by the Academic Technologies team, the operational profile of the Adviser Scheduling application was determined and considered when testing with the Each Choice and pairwise testing methods. The data in Table III include the most common parameter values based on the operational profile. This extra configuration was added to Test Set 1 to consider the application's operational profile, making Test Set 2 consist of four configurations.

TABLE III. EACH CHOICE TEST SET 2

| Config | Browser | User Type | Operating System | Network | Tests |
|---|---|---|---|---|---|
| 4 | IE | Faculty | Win 7 | On Campus –Unsecured | 21 |

Each Choice Test Set 3: This Each Choice testing table introduces a new parameter and values and consists of five configurations (see Table IV). The aim of this test set is to reduce the number of required tests while increasing the number of bugs found by testing mobile devices, which were not a part of the application's original test range.

TABLE IV. EACH CHOICE TEST SET 3

| Conf | Brow-ser | User Type | Oper. System | Network | Mo-bile | Tests |
|---|---|---|---|---|---|---|
| 1 | IE | Faculty | Win 7 | On Campus–Secured | No | 21 |
| 2 | Chrome | Admin | Win 10 | On Campus–Unsecured | No | 10 |
| 3 | Firefox | Delegate | OS X | Off Campus | No | 12 |
| 4 | Safari | Faculty | iOS | On Campus–Secured | Yes | 21 |
| 5 | Stock | Admin | Android | On Campus–Unsecured | Yes | 21 |

### *B. Pairwise Testing Approaches*

To generate the test sets used for pairwise testing in this case study, the Advanced Combinatorial Testing System (ACTS) tool was used. Provided by the National Institute of Standards and Technology, ACTS is a free tool that assists users in generating *t*-way combinatorial test sets. Prior to using the ACTS tool, the user must identify the parameters that will be used in testing, along with their values and associated constraints. The tool eliminates any combinations that violate constraints between parameters as they were configured while creating the system, and then it allows the user to build and view the test set. For the Adviser Scheduling application case study, the ACTS tool was used to create pairwise (*2*-way) test sets for the application's 21 test cases. Three test sets were created using the ACTS tool.

Pairwise Test Set 1: This test set includes the parameters used in the original testing phase to show that less testing could be performed with a much higher level of coverage. This test set includes nine configurations of the original testing parameters, shown in Table V, which is two configurations fewer than used originally.

TABLE V. PAIRWISE TEST SET 1

| Config | Browser | User Type | Operating System | Network | Test |
|---|---|---|---|---|---|
| 1 | Chrome | Faculty | OS X | Off Campus | 21 |
| 2 | IE | Faculty | Win 7 | On Campus–Secured | 21 |
| 3 | Chrome | Faculty | Win 7 | On Campus–Unsecured | 21 |
| 4 | IE | Admin | Win 7 | Off Campus | 10 |
| 5 | Chrome | Admin | OS X | On Campus–Secured | 10 |
| 6 | IE | Admin | Win 7 | On Campus–Unsecured | 10 |
| 7 | Chrome | Delegate | OS X | Off Campus | 12 |
| 8 | IE | Delegate | Win 7 | On Campus–Secured | 12 |
| 9 | Chrome | Delegate | OS X | On Campus–Unsecured | 12 |

Pairwise Test Set 2: The second test set includes the parameters used in the original testing phase, the same nine configurations as in Test Set 1, and an additional two configurations that are derived from the operational profile. These additional two configurations are shown in Table VI.

TABLE VI. PAIRWISE TEST SET 2

| Config | Browser | User Type | Operating System | Network | Tests |
|---|---|---|---|---|---|
| 10 | IE | Faculty | Win 7 | On Campus–Unsecured | 21 |
| 11 | IE | Faculty | Win 7 | Off Campus | 21 |

Pairwise Test Set 3: The third test set introduces more parameters and parameter values to broaden the testing scope with the intent to find new bugs. This test set includes 19 configurations, listed in Table VII, based on five parameters. The new parameter introduced in this set is "Mobile" with values "Yes" and "No." Several new parameter values were added for parameters "Browser" and "Operating System."

## V. RESULTS OF THE INVESTIGATION

Fig. 2 and Table VIII display the results of the investigation, comparing the number of tests completed for each test set with the number of bugs found. In addition, Table VIII compares the number of parameters and their values and the number of configurations tested per test set. A total of 43 tests were performed while testing Each Choice Test Set 1. This test set provided the same quality of testing as the original procedure, while reducing the number of tests from 121 to 43, essentially cutting the number of tests by 65%. While testing Each Choice Test Set 2, 64 tests were performed that detected the same 53

bugs found during the original testing. Each Choice Test Set 2 consisted of 21 more test cases than Each Choice Test Set 1 to include consideration of the operational profile. Each Choice Test Set 3 consisted of 85 tests. This test set introduced testing coverage for mobile devices. A total of 75 bugs were found during testing. These bugs consisted of the same 53 bugs detected during the original testing of the Advisor Scheduling application, in addition to 22 new bugs that were specific to mobile devices and mobile browsers.

TABLE VII. PAIRWISE TEST SET 3

| Conf | Brow-ser | User Type | Oper. System | Network | Mo-bile | Tests |
|---|---|---|---|---|---|---|
| 1 | IE | Admin | Win 7 | On Campus–Unsecured | No | 10 |
| 2 | IE | Delegate | Win 10 | Off Campus | No | 12 |
| 3 | Chrome | Faculty | Win 7 | On Campus–Secured | No | 21 |
| 4 | Chrome | Admin | Win 10 | On Campus–Unsecured | No | 10 |
| 5 | Chrome | Delegate | OS X | Off Campus | No | 12 |
| 6 | Chrome | Faculty | Android | On Campus–Unsecured | Yes | 21 |
| 7 | Chrome | Admin | iOS | On Campus–Secured | Yes | 10 |
| 8 | Firefox | Delegate | Win 7 | Off Campus | No | 12 |
| 9 | Firefox | Faculty | Win 10 | On Campus–Secured | No | 21 |
| 10 | Firefox | Admin | OS X | On Campus–Unsecured | No | 10 |
| 11 | Firefox | Delegate | Android | On Campus–Secured | Yes | 12 |
| 12 | Safari | Faculty | OS X | Off Campus | No | 21 |
| 13 | Safari | Delegate | iOS | On Campus–Unsecured | Yes | 10 |
| 14 | Stock | Admin | Android | Off Campus | Yes | 12 |
| 15 | Safari | Faculty | iOS | Off Campus | Yes | 21 |
| 16 | IE | Faculty | Win 7 | On Campus–Secured | No | 21 |
| 17 | Stock | Faculty | Android | On Campus–Unsecured | Yes | 21 |
| 18 | Safari | Admin | OS X | On Campus–Secured | No | 10 |
| 19 | Stock | Delegate | Android | On Campus–Secured | Yes | 12 |

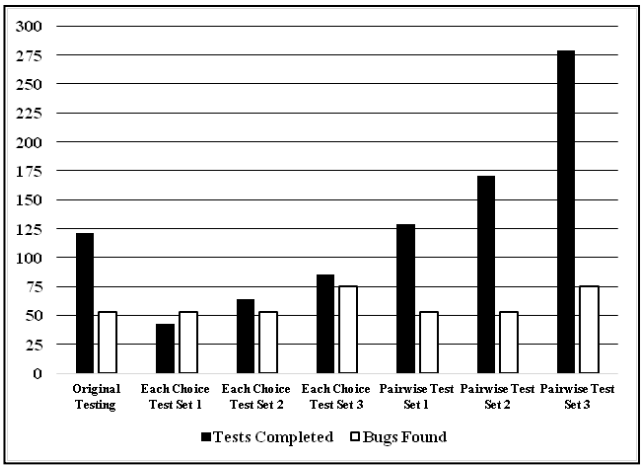


Figure 2. Comparison of the Number of Tests Completed to Number of Bugs Found

TABLE VIII. NUMBER OF PARAMETERS AND CONFIGURATIONS FOR EACH TESTING METHOD

| | Test Sets | Para-meters | Para-meter Values | Config. per Test Set | Tests | Bugs |
|---|---|---|---|---|---|---|
| 1 | Original test | 4 | 10 | 11 | 121 | 53 |
| 2 | Each Choice Test Set 1 | 4 | 10 | 3 | 43 | 53 |
| 3 | Each Choice Test Set 2 | 4 | 10 | 4 | 64 | 53 |
| 4 | Each Choice Test Set 3 | 5 | 16 | 5 | 85 | 75 |
| 5 | Pairwise Test Set 1 | 4 | 10 | 9 | 129 | 53 |
| 6 | Pairwise Test Set 2 | 4 | 10 | 11 | 171 | 53 |
| 7 | Pairwise Test Set 3 | 5 | 16 | 19 | 279 | 75 |

Pairwise Test Set 1 consisted of 129 tests. This test set provided the same quality of testing as the original procedure. However, it required slightly more test cases but fewer test configurations. The amount of testing was approximately the same as it was for original testing, but the provided level of coverage was much better. While testing Pairwise Test Set 2, 171 tests were performed that detected the same 53 bugs as found during the application's original testing. Pairwise Test Set 2 consisted of 42 more test cases than Pairwise Test Set 1 to include consideration of the operational profile. Pairwise Test Set 3 consisted of 279 tests to include testing coverage for mobile devices. A total of 75 bugs were found during testing, as with Each Choice Test Set 3. Each Choice Test Set 3 provided the same quality of testing as Pairwise Test Set 3, while reducing the number of tests from 279 to 85. While in our case study pairwise testing did not demonstrate additional benefits when compared with Each Choice, in other situations results could be different.

## VI. Conclusions

In conclusion, we found that the original testing of the application was thorough, but while good enough to detect the same bugs as the pairwise and Each Choice approaches, it was not systematic and did not allow time for mobile testing. The purely pairwise method and Each Choice method did not reveal any new bugs but reduced the number of configurations. Reducing these also reduces the effort required in testing. Although pairwise testing requires completing more tests than Each Choice, the latter cannot produce better results than pairwise testing, although it can reduce the testing time. The Each Choice testing method allowed for testing an extra parameter (mobile) without exceeding the time and resource restriction as originally faced in testing. While no new bugs were found when considering the operational profile, there is greater confidence in the test results that the application is ready for use by its main audience.

In this case study, pairwise was not the most beneficial testing method, but it is usually considered to be the better testing alternative (compared to Each Choice) when sufficient resources are available. Our case study is limited to one application and does not intend to compare these approaches in detail. However, even this one example demonstrates how practical combinatorial testing approaches can minimize the number of test cases and/or maximize the number of detected faults.

With the results of fewer tests, new configurations, and better detection rates, this study shows that the Each Choice testing method with the inclusion of the mobile device parameter can be implemented as the preferred testing approach for applications within the university's web portal.

## Acknowledgment

This work was performed under the following financial assistance award 70NANB15H217 from the U.S. Department of Commerce, National Institute of Standards and Technology.

## References

[1] D. R. Kuhn, R. Kacker, and Y. Lei, Introduction to Combinatorial Testing, Chapman and Hall/CRC, 2013, 341 pages.

[2] D. R. Kuhn, R. Kacker, Y. Lei, and J. Hunter, "Combinatorial software testing, " IEEE Computer, vol. 42, no. 8, August 2009, pp. 94–96.

[3] P. J. Schroeder, P. Bolaki, and V. Gopu, "Comparing the fault detection effectiveness of n-way and random test suites," Proceedings of the 2004 IEEE International Symposium on Empirical Software Engineering (ISESE'04), 19–20 August 2004, Redondo Beach, CA, USA, pp. 49–59.

[4] D. R. Kuhn, Y. Lei, and R. Kacker, "Practical combinatorial testing: Beyond pairwise, " IT Professional, 10.3, 2008, pp. 19–23.

[5] S. Vilkomir, K. Marszalkowski, C. Perry, and S. Mahendrakar, "Effectiveness of Multi-Device Testing Mobile Applications," Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems (MobileSoft 2015), May 16–17, 2015, Florence, Italy, pp. 44–47, in conjunction with the 37th International Conference on Software Engineering (ICSE'15).

[6] S. Vilkomir, O. Starov, and R. Bhambroo, "Evaluation of t-wise Approach for Testing Logical Expressions in Software, " Proceedings of the IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2013), 18–20 March 2013, Luxembourg, Luxembourg, pp. 249–256.

[7] J. Czerwonka, "Pairwise testing in Real World. Practical extensions to test case generators," Proceedings of 24th Pacific Northwest Software Quality Conference, Portland, Oregon, October 9–11, 2006, pp. 419–430.

[8] M. Grindal, J. Offutt, and S. F. Andler, "Combination testing strategies: a survey," Software Testing, Verification and Reliability, vol. 15, no. 3, March 2005, pp. 167–199.

[9] J. Musa, "Operational profiles in software-reliability engineering," IEEE Software, 10.2, 1993, pp. 14–32.

[10] C. Smidts, C. Mutha, M. Rodríguez, and M. J. Gerber, "Software testing with an operational profile: OP definition," ACM Comput. Surv. 46, 3, Article 39, February 2014, 39 pages.

[11] P. A. Brooks and A. M. Memon, "Automated gui testing guided by usage profiles," Proceedings of the 22 IEEE/ACM international conference on Automated software engineering, November 5–9, 2007, Atlanta, Georgia, USA, pp. 333–342.

[12] J. A. Whittaker and M. G. Thomason, "A Markov Chain Model for Statistical Software Testing," IEEE Trans. Softw. Eng. 20, 10, October 1994, pp. 812–824.