UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Andrus Lall

Lab Package: Combinatorial Testing

Bachelor's Thesis (9 ECTS)

Supervisor: Dietmar Pfahl

Tartu 2018

Lab Package: Combinatorial Testing

Abstract:
The purpose of this thesis is to create lab materials for the University of Tartu's course "Software testing (MTAT.03.159)" on the subject of combinatorial testing. The thesis gives background information on the course and on combinatorial testing, describes the materials created and their execution, presents the results of the students feedback questionnaire and its analysis, along with the improvements made. The lab materials were introduced in the spring semester of 2018.

Keywords:
Software testing, combinatorial testing, lab package

CERCS:
P170 Computer science, numerical analysis, systems, control

Praktikumimaterjal: Kombinatoorne Testimine

Lühikokkuvõte:
Käesoleva bakalaureusetöö eesmärgiks on luua õppematerjali Tartu Ülikooli kursusele "Tarkvara testimine (MTAT.03.159)" kombinatoorse testimise teemal. Töö kirjeldab kursuse ja kombinatoorse testimise tausta, koostatud õppematerjali, esitab tudengite tagasiside tulemused ning nende põhjal tehtud analüüsi ja parandused. Õppematerjal võeti kasutusele 2018. aasta kevadsemestril.

Võtmesõnad:
Tarkvara testimine, kombinatoorne testimine, praktikumimaterjal

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Contents

Introduction

Software testing is the process of assessing software in order to determine its quality and whether it meets the needs of the people it concerns. The University of Tartu's course "Software Testing (MTAT.03.159)" is designed to teach students the fundamentals of testing, introduce important testing techniques and much more.

"Software Testing" is a mandatory course for bachelor's degree programme students studying informatics. As of 2018, the course has a volume of 3 ECTS points and consists of 7 labs and lectures. The labs are designed to introduce different testing techniques while also introducing tools that can be applied to aid the use of these techniques.

The purpose of this thesis is to create materials for the new lab on the subject of combinatorial testing. Before this year, the subject has only been taught in the form of lectures, but there became a need for new labs because the course is expected to increase in volume. The materials were created for both the lab session and the homework assignment. They contain instructions for the lab session and homework assignment, two applications, created as a part of this thesis to be the systems under test (SUT), and a tool called Automated Combinatorial Testing for Software[1] (ACTS) by the National Institute of Standards and Technology[2], that is crucial to testing the two applications created.

Combinatorial testing is a technique used to systematically test combinations of a software's input values. It is mainly used in critical systems with a multitude of different configurations and parameters, where thoroughness is important due to the possibility of defects causing harm.

This thesis is composed of five chapters. The first chapter gives an overview of the software testing course and combinatorial testing. Lab schedule, materials, tasks and grading is presented in the lab design chapter. The third chapter details the execution of the lab. Feedback, its analysis and future improvements are presented in chapter four. The last chapter concludes the thesis by summarizing the materials created and the results of their implementation.

[1] https://csrc.nist.gov/Projects/Automated-Combinatorial-Testing-for-Software
[2] https://www.nist.gov/

1 Background

The following subsections give an overview of the course "Software Testing" that this lab package was created for and the subject of combinatorial testing.

1.1 The University of Tartu's course "Software Testing (MTAT.03.159)"

As of 2018, "Software Testing (MTAT.03.159)" is a 3 ECTS course which, according to the course outline [1], covers the fundamental concepts of software testing, introduces various testing strategies and types of testing, while also giving an overview of different software defects, software defect management, and organizational aspects of software testing. It takes place during the spring semester and consists of 7 lectures and labs. The following is a list of topics for each lab:

1. Debugging
2. Black-Box & White-box Testing
3. Combinatorial Testing
4. Mutation Testing
5. Automated Web-Application Testing
6. Static Code Analysis
7. Document Inspection and Defect Prediction

1.2 Combinatorial testing

Combinatorial testing terminology

- t-way interactions/combinations - each possible value combination between t parameters is represented at least once
- Covering array - an array of test cases that as a whole covers all t-way interactions
- Interaction strength - the degree of a covering array's interaction strength, i.e. t

Combinatorial testing is a black box testing technique for systematically testing t-way interactions of configuration or input parameter values. t-way interactions represent all possible combinations of values between t different parameters. For example, if one had 3 boolean parameters - A, B and C - all 2-way interactions can be found by finding all

possible value combinations between A and B, then A and C, and finally B and C. As can be seen from figure 1, all of these combinations could be represented in a covering array containing only 4 test cases. Because there are 3 parameters, 3-way testing would be exhaustive of all combinations and would require 8 test cases - found by multiplying the number of values per parameter.

A	B	C
0	1	1
1	0	1
1	1	0
0	0	0

Figure 1. All 2-way interactions between boolean type parameters A, B and C

A higher degree of interaction strength is important, because empirical data has shown that testing only 1-way interactions (each value used at least once in isolation) will likely not discover all or in some cases even most of the defects in a system [2]. On the other hand, exhaustive testing is resource intensive, which is why the interaction strength is selected based on resources and risks. One study that analyzed 329 error reports of large distributed systems found that over 95% of failures were triggered by the interaction of 4 parameters and that no new failures required the interaction of more than 6 parameters [3]. However, increasing the interaction strength amplifies the test oracle problem of combinatorial testing. As the amount of test cases increases significantly when interaction strength is increased, the amount of expected outputs one needs to find for the test cases increases at the same rate. Several solutions to this problem include crash testing, embedded assertions and model based test generation [4].

Combinatorial testing is typically used to test architecturally complex systems with many input or configuration parameters, some cases even both. However, it can be useful for testing any system with a number of input parameters or configurations, e.g. mobile applications that are required to run on different operating systems and hardware or have many parameters with several values [5]. It is especially relevant to critical systems, like for example traffic collision avoidance systems, where software defects can cause harm and thoroughness is highly valued. As the adoption of more software systems with a multitude of configurations rises, so does the need for more effective test data sampling.

Automated Combinatorial Testing for Software (ACTS)

While manually creating *t*-way test cases for a two or three parameters with a few values may not be a problem, doing this for more complex systems is typically not feasible. Combinatorial testing with the help of tools like the ACTS (formerly known as FireEye)

allow for a fast generation of t-way interaction test cases that one can use to systematically evaluate a system under test, saving time and minimizing user error.

For this lab package, the ACTS tool was chosen due to its features, but also because it is free to use, has a graphical user interface and is quite easy to learn. The creators of the tool have also conducted extensive research to create the algorithms it uses and are one of the leading researches in the field of combinatorial testing [6][7].

Using the ACTS tool, one can create a combinatorial testing system (CT) system that reflects the input or configuration parameter values of their software. One then has the ability to build covering arrays using a desired interaction strength. There also exists a mixed strength feature, which allows one to specify a higher or lower degree of interaction strength for a chosen subset of parameters. Constraints can also be defined in order to rule out specific value combinations. The ACTS tool's general view of the graphical user interface can be in from figure 2. Defined parameters can be found on the left hand side under "System View", while the covering array (if one has been created) can be found on the right, under "Test Result".

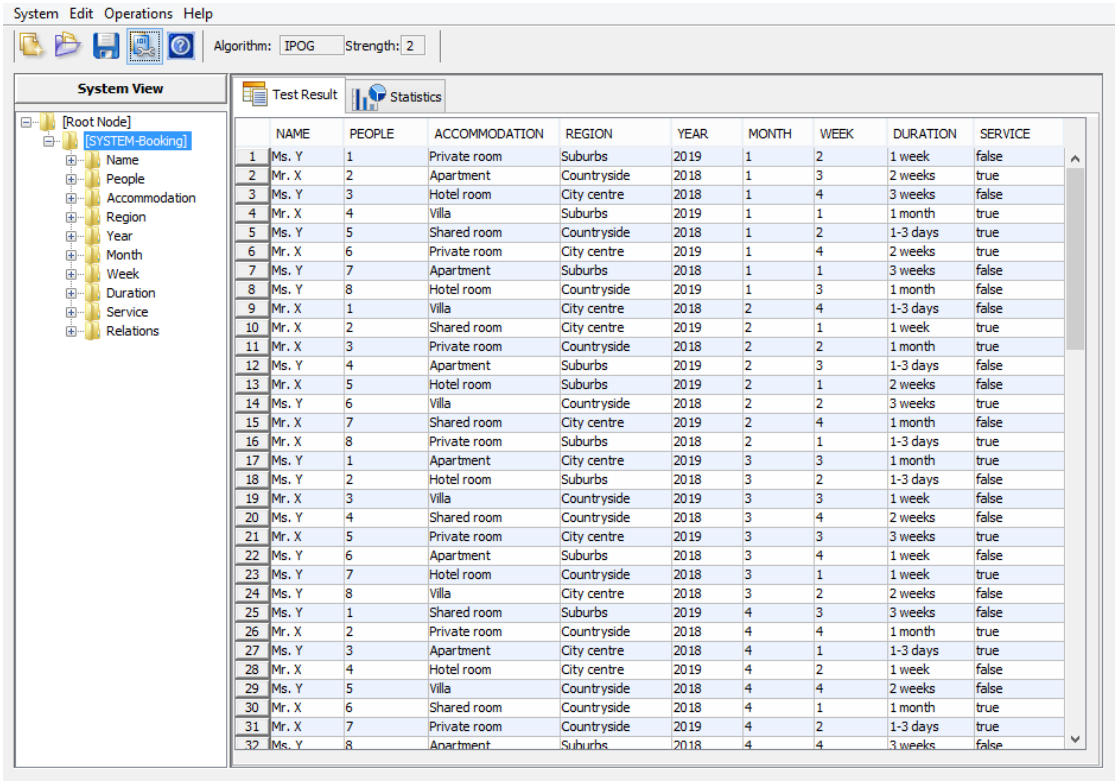


Figure 2. Graphical user interface of the ACTS tool

2 Lab Design

The following subsections describe what the lab materials consist of and explain the design choices made when creating them.

2.1 Lab Schedule

It is expected that each lab takes approximately 6 hours to complete, not including the lab session itself. The schedule for the lab session is as follows:

- 70 minutes - Lab session tasks: introduction to combinatorial testing, learning to use the ACTS tool and its features, testing the Marriage Checker application
- 20 minutes - Overview of the homework tasks and the Booking application, questions

2.2 Lab Materials

The lab materials created as a part of this thesis, which can be found in Appendix II, are divided into two parts - lab materials for students and for teaching assistants.

The lab materials for students consist of a PDF file containing the instructions for the lab session and homework assignment, an application named Marriage Checker, created to be used as the system under test (SUT) in the lab session, and an application named Booking, created to be the SUT for the homework assignment. A link to the ACTS tool is also provided, since it is an essential part of the lab.

The lab materials for teaching assistants consist of a PDF file containing instructions for the execution of the lab session, expected results for the homework tasks, a more detailed grading scheme and an example of a correct solution along with the necessary files.

2.2.1 Marriage Checker application

The Marriage Checker application is an application that was created as a part of this thesis, in order to be used as the SUT during the lab session. Being more comfortable and having more experience programming in Java than in Python for example, it was chosen as the language in which both the Marriage Checker and Booking applications were written.

It is presented to the students as a toy application that is used to check whether two people are married or not. To make things more interesting, the introduction doesn't give the complete test oracle, but it reveals that the application should return that the two

people are not married if they are people of the same sex. It is said that this is because the application uses test data based on a country that doesn't permit such marriages. An image of the Marriage Checker application's user interface can be seen in figure 3.

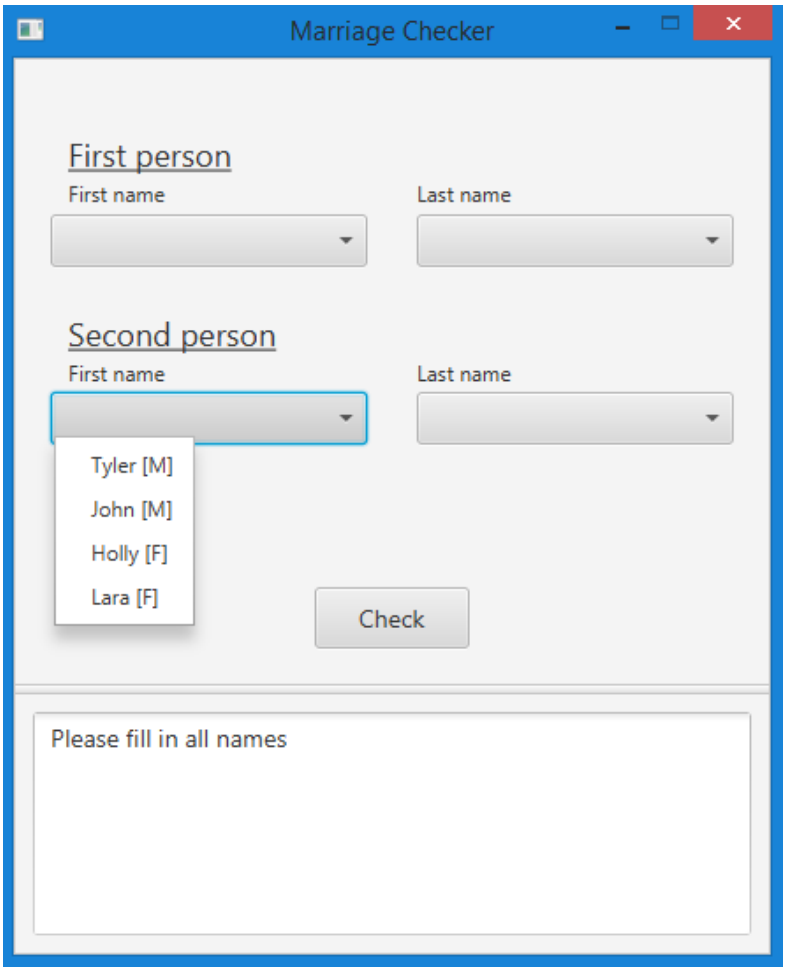


Figure 3. User interface of the Marriage Checker application

This simple application was created to be used in the lab session, so the students would be able to familiarize themselves with the ACTS tool and its features, while already feeling the difference in results between manually creating test cases and the test cases that the ACTS tool creates systematically.

Two different types of failures have been seeded into the application. The first type of failure triggers an incorrect result (as in not in accordance with the introduced rules) in the text area displayed at the bottom of the application. The second type of failure emulates the program crashing by displaying a pop-up window. To help show the importance

of systematic testing, the first type of failure requires less specific values to trigger than the second type of failure, making it highly unlikely that the second type of failure can be found when entering input values randomly. The exact failure triggering values and an example of the pop-up window can be found in the teaching assistant instructions, see Appendix II.

The possible first and last name values for both people are different, but both first name parameters include two male and two female first names. As such, one could divide the approaches to test all 1-way interactions (each value is used at least once) by doing one of the following:

- Choosing all combinations of values chronologically, by executing a test case containing the first value of each parameter, then the second value etc. This would be the least thought through approach and would only test for results of one type (corresponding to same sex combinations).
- Testing male + female and then female + male combinations. This may seem like a better approach than the previous one, but it too would test for results of only type.
- Equivalence class partitioning - executing test cases that check for both types of results. This would be the most rational approach.

Since there are four parameters and each parameter has four values, all of the approaches described above would require only four test cases to test all 1-way interactions. The values in the application that trigger failures were seeded in such a way, that each approach would likely trigger at least one, but not most of the failures.

2.2.2 Booking application

The Booking application is a Java application that was created as a part of this thesis in order to be used as the SUT for the homework tasks. It is introduced as a toy application that should display the amount of listings corresponding to a search criteria. Three rules are also given that should make the application display 0 results, if they were to be broken. An image of the Booking application's user interface can be seen in figure 4.

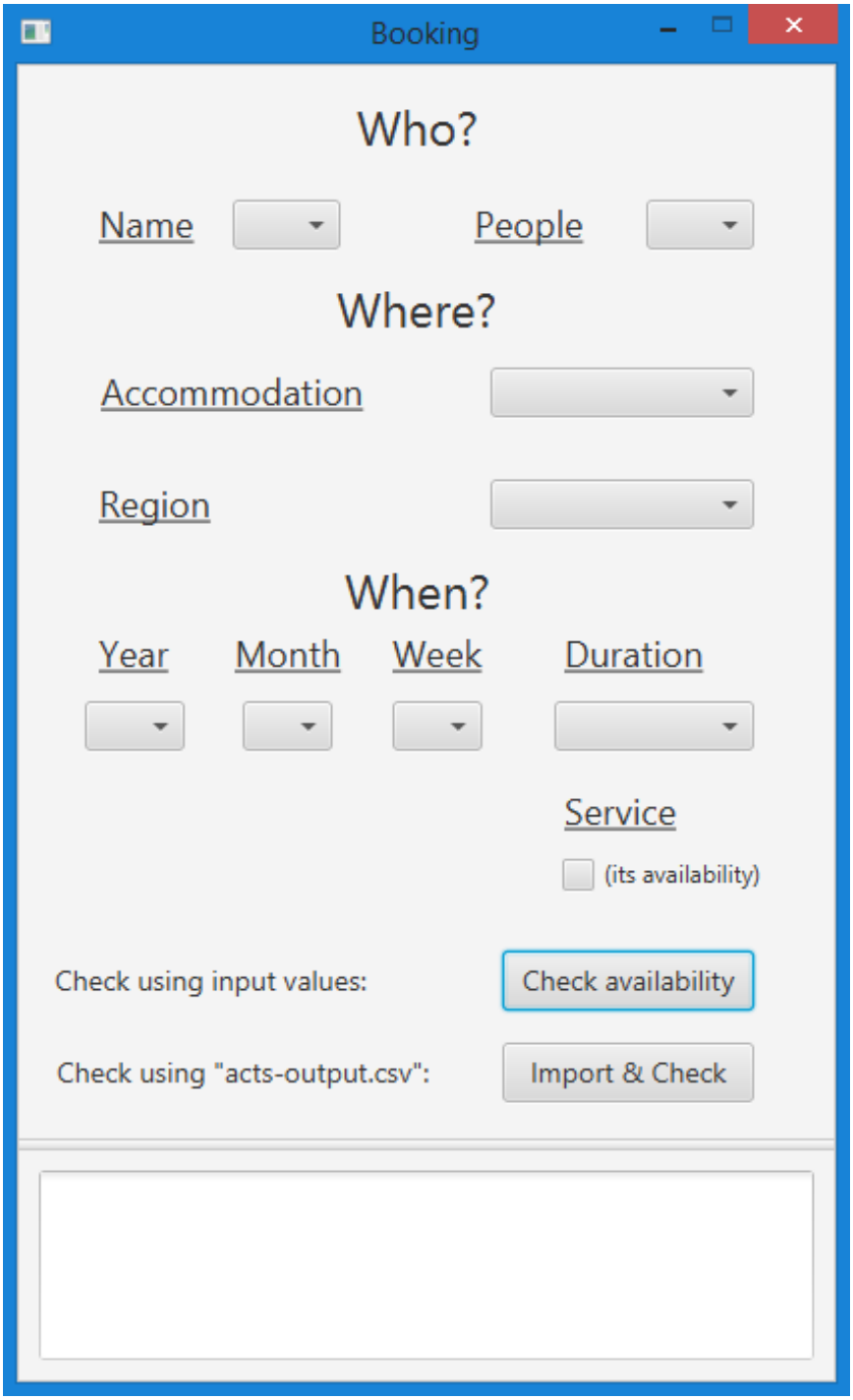


Figure 4. User interface of the Booking application

This application is more complex and contains a combination of parameters and input values that is more true to life. It has 9 parameters, each containing 2 to 12 different options to choose from.

It soon became apparent when first developing the application, that manually executing the test cases generated by the ACTS tool would be unfeasible, because it would be too time consuming for the tasks in mind. Due to this, the 'Import & Check' functionality was implemented to automate the process of executing test cases. This functionality is something that developers should be able to implement for any application with a set of input parameters and an output, to increase its testability.

The ACTS tool enables users to export their covering arrays as a CSV (comma separated values) file. The 'Import & Check' button can then be used to choose the file exported by the ACTS tool, after which it will run each test case one by one, generating the result of each test into a new CSV file. The result is added at the end of the corresponding test case, so the final file can be opened in a spreadsheet program (e.g. Excel) and analyzed.

This import functionality is flexible in that it doesn't matter in which order the parameters were added, when the corresponding combinatorial testing (CT) system in the ACTS tool was created. It is also case insensitive. However, it does require that each test case contains values from all parameters and that all values are written as they are in the application itself.

2.3 Lab Session Tasks

The aims of the lab session:

- To reiterate the combinatorial testing technique principles and begin applying them on the Marriage Checker application
- To introduce the ACTS tool and let the students become familiar with it
- To give an overview of the homework assignment and the Booking application

The lab assistants begin the lab by once again introducing the combinatorial testing technique. Students are then instructed to download the Marriage Checker application from the course webpage [8]. This application is then used as a SUT for the lab session tasks in the instructions.

After the students have downloaded the application, they learn how to find the amount of test cases in order to test all 1-way interactions. The students then execute test cases

to cover all 1-way interactions and note any failures. Next, it is demonstrated how to find the upper and lower bound for the amount of test cases required in order to test all possible combinations, followed by 2-way and 3-way interactions.

Afterwards, students are instructed to download the ACTS tool and used it to create a CT system, which would contain the parameters and respective values of the SUT. Using the system they had just created, they would then learn how to build the covering arrays that would contain test cases which systematically cover all t-way interactions. They can compare the amount of test cases the ACTS tool generates to the amounts they found earlier.

Students then execute the created test cases which cover all 2-way interactions and observe the results. They should find more bugs, compared to the test cases they had executed earlier.

The Mixed Strength feature of the ACTS tool is then introduced. It is explained how to determine the parameters that should be covered with an increased interaction strength. Students need to build the covering array using mixed interaction strength and compare their results with the results from their previous tests. They also have to analyze how the feature affected their covering array - how much more efficient or effective it is and whether it is something that they could always use successfully.

2.4 Homework Tasks

The homework assignment has several aims for the students:

- To become proficient at and realize the importance of using the ACTS tool
- To learn how to apply combinatorial testing wisely - e.g. how to determine what subset of parameters should be assigned a higher interaction strength, so to increase the failure detection power
- To give the students a chance to be creative in their approach to solving the test oracle problem
- To realize that there is no silver bullet to finding more failures, but that combinatorial testing can be used in order to become more effective, efficient and methodical at testing

The homework assignment contains three tasks. Because of the test oracle problem of combinatorial testing, tasks 1 and 2 revolve around failures that are easy to spot - error messages in the output of the Booking application. However, task 3 gives the students a

chance to solve the oracle problem how they see fit.

Task 1 requires the students to create a CT system in the ACTS tool that reflects the parameters and possible values that the Booking application has. Then, using the ACTS tool, they need to build and export all covering arrays from interaction strength 1 to 9. This is required for both the major optimization algorithms the ACTS tool offers - IPOG and IPOG-F. They are then required to fill a table with their results and construct a graph in order to visualize the difference between the cumulative amount of unique failures found at each interaction strength for both algorithms used.

Looking at the table they filled or the graph, it should become apparent that 1-way or 2-way interaction testing would have not found even a quarter of the failures seeded into the program. No new failures are found at interaction strengths higher than 6. These tables and graphs can also help them in task 2, so they can see which algorithm was more or less effective at each interaction strength.

The aim of task 1 is to further hammer home the importance of testing more than 1-way or 2-way interactions, as they can see from the graphs and tables that it wouldn't have been enough to fully uncover all the different failures. It was also created so that the students have all of the necessary files for the rest of the tasks.

Task 2 makes things more interesting - it presents a realistic limit on the amount of test cases the students can execute, further increasing the proficiency of the students in using the ACTS tool. They are required to use only the ACTS tool's features in order to create as effective of a covering array as they can, all the while staying inside this set limit, forcing the students to more wisely use combinatorial testing. It is mentioned that the students should recall the features introduced in the lab session to solve this task.

Choosing the parameters which have a higher degree of interaction can be deducted from the business rules given in the introduction of the Booking application. Since the parameters mentioned imply having more back-end logic, it is likely that choosing them would net more failures. They can also view the graph constructed as a result of task 1 and deduct that the IPOG-F algorithm found more failures between interaction strengths 2 and 3, meaning they should prefer this algorithm.

Students who have built a covering array that triggers more than the required amount of failures earn can earn a bonus point. This can be achieved by analyzing the results of the covering arrays of task 1 to find out which values might be the cause of errors triggering. One can then use the Constraints feature in the ACTS tool and constrain the ACTS to generate certain combinations of values. Thus, the aim of this bonus task is to compel

the more curious students to try out features of the ACTS tool that were not introduced in the lab.

Task 3 requires the students to search for new sorts of failures. When in task 1 and 2 they were looking at failures easy to spot, they now have to analyze the largest output file of the SUT and figure out if it violates any of its business rules. They are also required to present the test cases that triggered these rule violations in a separate file. Due to this and the fact that the number of test cases in the largest output file, which contains all possible input combinations, is far too high to manually analyze, they need to figure out a way to automate the process of checking for incorrect output, solving the test oracle problem.

This task was designed to be more lax in terms of how the students solve it, in order to give them a chance to be creative in their approach. It is more akin to a real world testing task in that they don't have to check for system crashes or obvious faults, but instead are concerned with failures against the specification (business rules) of the application.

2.5 Grading

Students can get a total of 10 points plus one bonus point for this lab. The marks are divided in the following way:

- 1 point for participating in the lab
- Up to 4 points for Task 1
- Up to 3 points for Task 2, plus 1 bonus point
- Up to 2 points for Task 3

A more detailed grading scheme can be found in the student lab instructions, see Appendix II.

3 Lab Execution

In the spring semester of 2018, the "Software Testing" course had a total of 7 labs, with the combinatorial testing lab being the 3^{rd}. There were 5 lab groups, so a total of 5 sessions were held for this lab, taking place on the 20^{th} and 21^{st} of March.

The subject of combinatorial testing was also covered in the lecture, so students who had attended were already familiar with it before the lab session. For this thesis, one lab session was observed to determine time consumption and to quickly react to any unexpected issues or questions that might arise.

However, the observed lab session went quite smoothly and according to the teaching assistants, so did the other labs sessions. The tasks took a bit longer than an hour and were completed without issues. Due to this, only one lab session was examined.

The lab assistants then began to give an overview of the homework tasks, describing the general workflow and various presentables that students were required to submit. There was some confusion regarding some column labels in the table for homework task 1, namely "Set of unique failures" and "Cumulative no. of unique failures up to strength *t*". Examples were then added to the instructions in order to resolve this.

The lab session lasted about 80 minutes in total, leaving time for questions from the students.

Figure 5 below shows the frequency distribution of the marks that students received for the lab. For comparison, this year's Black-box & White-box testing lab and last year's Static Code Analysis lab marks have been included as well, because both of them have been used for more than 2 years.

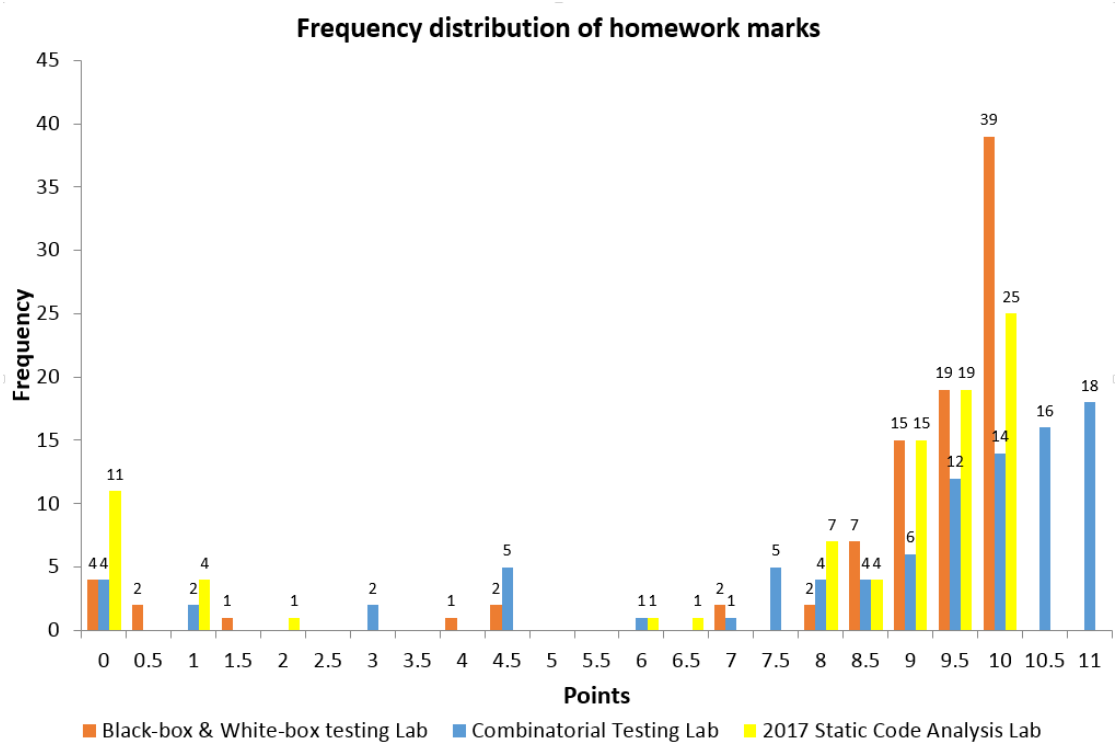


Figure 5. Frequency distribution of homework marks

The graph rounds points up, e.g. when a student received a mark between 7.0 (exclusive) and 7.5 (inclusive) points, it is accounted in the graph as 7.5. Both labs that are displayed for comparison had an upper limit of 10 points, while for the combinatorial testing lab students could receive 1 bonus point as well.

Overall, the distribution is similar to the other two labs. Although the amount of students who received 10 or more points is higher than the other two labs, the amount of students who got maximum points (10 points + 1 bonus point) is lower. This indicates that it was easier to score high points, but more difficult to score maximum points. However, this was discussed with the supervisor beforehand and was to be expected.

4 Feedback & Analysis

The following subsections describe the way feedback was collected and the results of the analysis of said feedback. It also states improvements already made based on the collected feedback and suggests future improvements.

A total of 91 students participated in the lab, 15 of whom answered the online feedback questionnaire.

4.1 Feedback collection

Students were asked to fill in a feedback questionnaire using SurveyMonkey[3], which is an online survey tool. The questionnaire had two parts: a set of statements, which could be rated on a fixed choice Likert scale [9], and two optional questions. There were a total of nine statements:

1. "The goals of the lab were clearly defined and communicated"
2. "The tasks of the lab were clearly defined and communicated"
3. "The materials of the lab were appropriate and useful"
4. "The ACTS tool was interesting to learn"
5. "If I have the choice, I will use the ACTS tool again"
6. "The "Booking" application was interesting and useful to test"
7. "The support received from the lab instructors was appropriate"
8. "The grading scheme was transparent and appropriate"
9. "Overall the lab was useful in the context of this course"

[3] https://www.surveymonkey.com/

For each statement, the students had to choose from five different options to reflect their agreement with the respective statement. The five options were:

1. "Completely disagree"
2. "Somewhat disagree"
3. "Neutral"
4. "Somewhat agree"
5. "Completely agree"

After the statements, the students were asked the following qualitative question: "What did you like or not like about the lab? Please try to be constructive."
The third and last question asked the students to provide any additional feedback they thought would be useful.

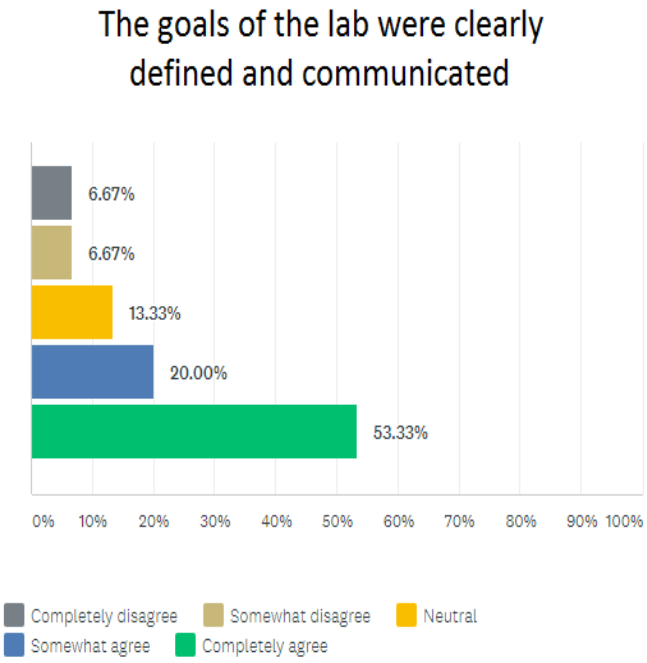
4.2 Analysis

A majority of the students answered positively to the statements 1-9. Seven out of fifteen students also answered the first qualitative question, each providing positive feedback on the lab, while two students also described some confusing aspects concerning the instructions. Only three respondents gave additional feedback regarding the final question.

4.2.1 Positive Aspects

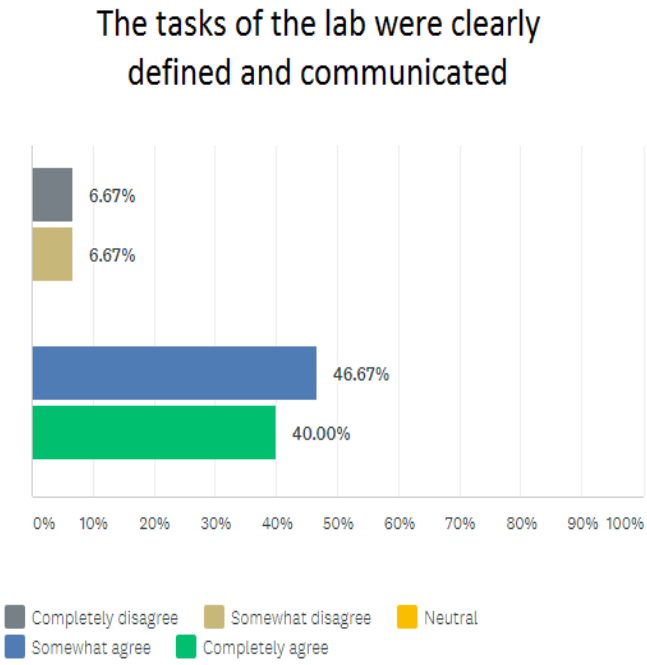
Over 73% of respondents concurred that both the goals and the tasks of the lab were clearly defined and communicated. More than 93% also found the lab materials appropriate and useful. A majority of the students thought the ACTS tool was interesting to learn and, if they had the choice, would use the tool again. About two-thirds of the students found the Booking application interesting and useful to test, although the subset of respondents who somewhat agreed was more than twice in size of those who completely agreed. A large majority agreed that the grading scheme was transparent and appropriate, with none of the respondents disagreeing with the statement. Finally, over 86% of students found the lab useful in the context of this course. Figures 6-14 show the frequency distribution of the results corresponding to each statement. For a summary of all of the statistics regarding statements 1-9, see Appendix I.

All qualitative feedback received for the question "What did you like or not like about the lab?" included positive feedback. Students mentioned that they were happy to be able to try out and learn the ACTS tool. The automation aspect of the lab and the ability to write scripts in order to optimize work was also praised.



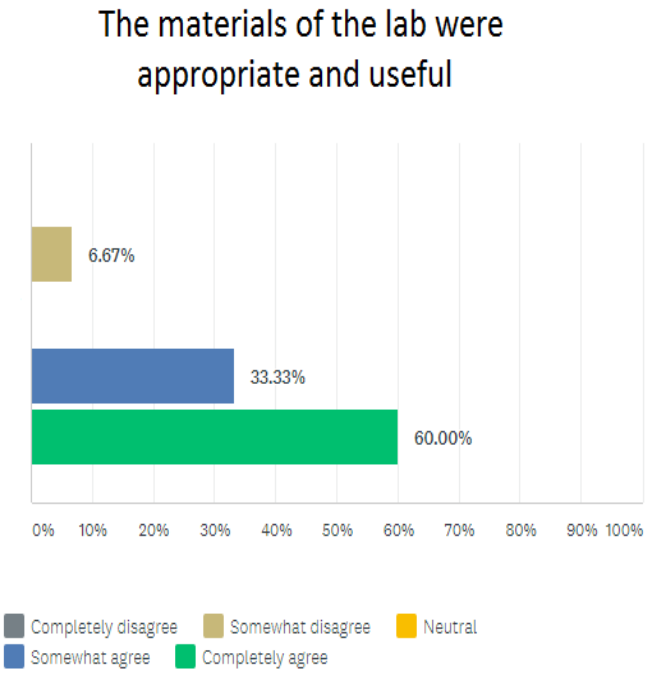
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The goals of the lab were clearly defined and communicated	6.67% 1	6.67% 1	13.33% 2	20.00% 3	53.33% 8	15	4.07

Figure 6. Statistics of "The goals of the lab were clearly defined and communicated"



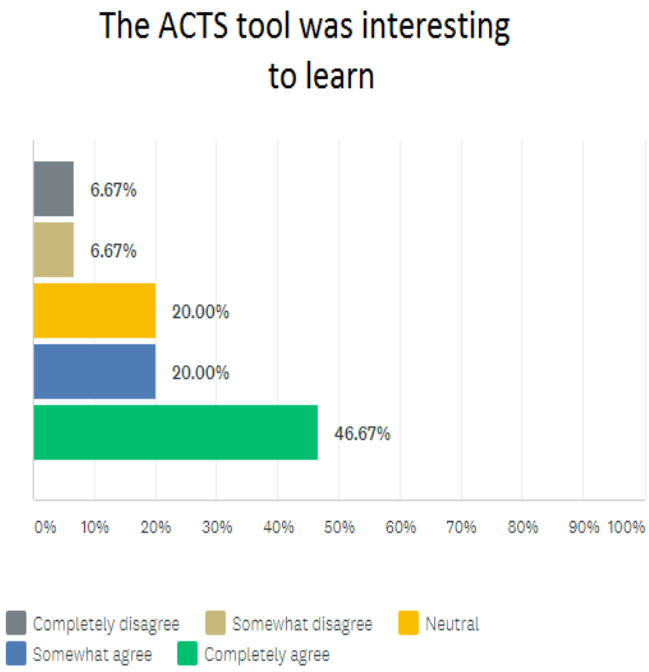
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The tasks of the lab were clearly defined and communicated	6.67% 1	6.67% 1	0.00% 0	46.67% 7	40.00% 6	15	4.07

Figure 7. Statistics of "The tasks of the lab were clearly defined and communicated"



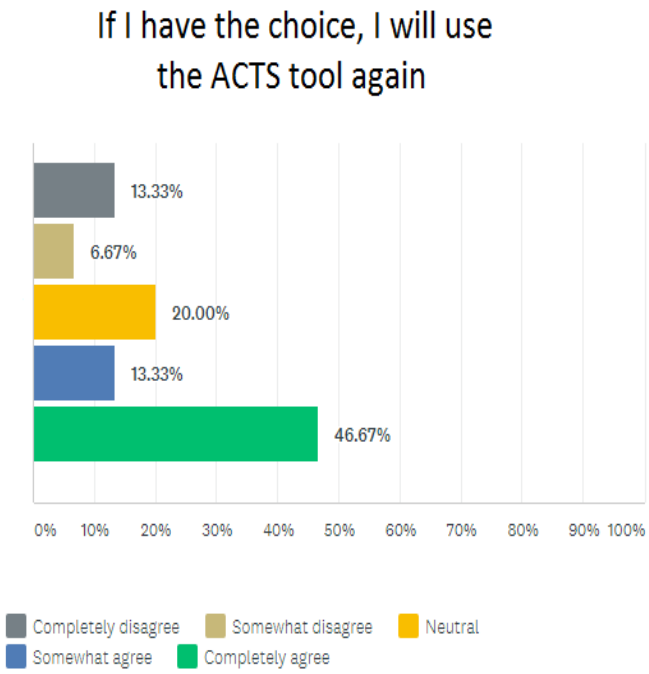
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The materials of the lab were appropriate and useful	0.00% 0	6.67% 1	0.00% 0	33.33% 5	60.00% 9	15	4.47

Figure 8. Statistics of "The materials of the lab were appropriate and useful"



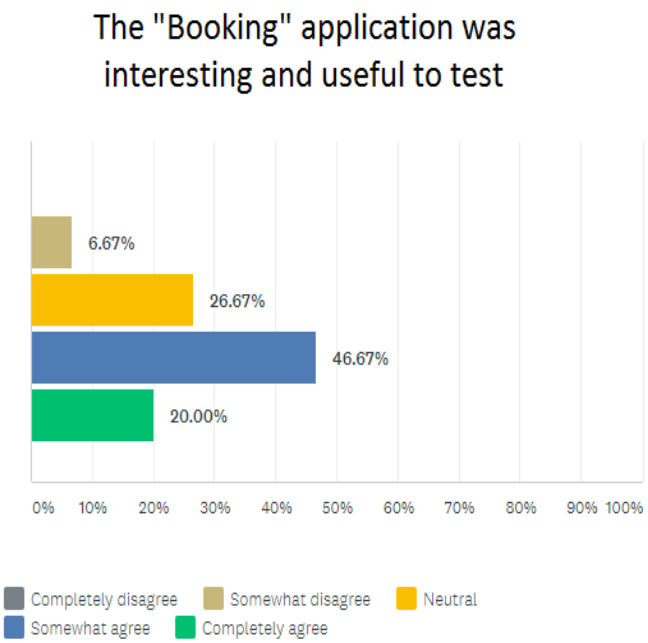
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The ACTS tool was interesting to learn	6.67% 1	6.67% 1	20.00% 3	20.00% 3	46.67% 7	15	3.93

Figure 9. Statistics of "The ACTS tool was interesting to learn"



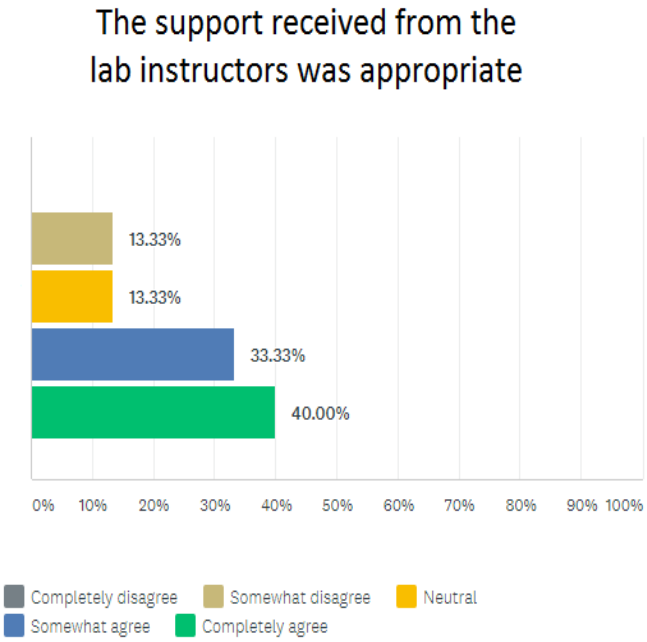
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
If I have the choice, I will use the ACTS tool again	13.33% 2	6.67% 1	20.00% 3	13.33% 2	46.67% 7	15	3.73

Figure 10. Statistics of "If I have the choice, I will use the ACTS tool again"



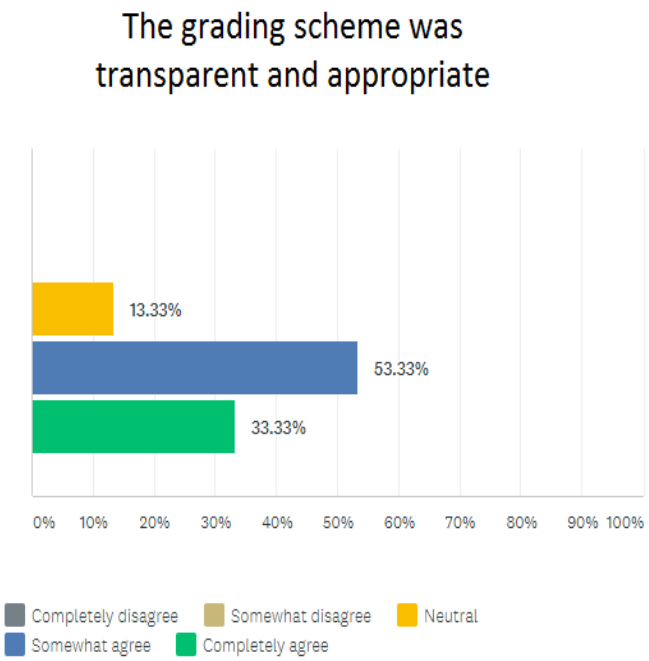
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The "Booking" application was interesting and useful to test	0.00% 0	6.67% 1	26.67% 4	46.67% 7	20.00% 3	15	3.80

Figure 11. Statistics of "The "Booking" application was interesting and useful to test"



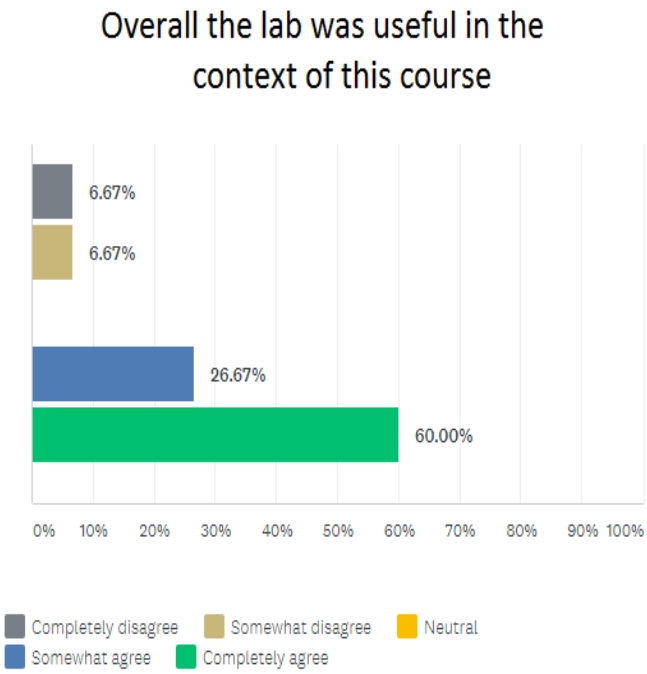
	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The support received from the lab instructors was appropriate	0.00% 0	13.33% 2	13.33% 2	33.33% 5	40.00% 6	15	4.00

Figure 12. Statistics of "The support received from the lab instructors was appropriate"



	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The grading scheme was transparent and appropriate	0.00% 0	0.00% 0	13.33% 2	53.33% 8	33.33% 5	15	4.20

Figure 13. Statistics of "The grading scheme was transparent and appropriate"



	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
Overall the lab was useful in the context of this course	6.67% 1	6.67% 1	0.00% 0	26.67% 4	60.00% 9	15	4.27

Figure 14. Statistics of "Overall the lab was useful in the context of this course"

4.2.2 Negative Aspects & Improvements Made

Feedback from the online questionnaire and the lab session itself revealed that some students were confused by the terms "set of unique failures" and "cumulative no. of unique failures up to strength t". Examples for both terms have since been included in the student lab instructions.

There was some confusion regarding task 3 of the homework assignment, after which some additional functionality was implemented in the Booking application that more explicitly reflects the business rules of the application. The original version of the application was not fully consistent with the third business rule of hotels offering service by default. The updated version now ticks the Service checkbox by default, should the user choose "Hotel room" as the Accommodation type.

A student mentioned having trouble fitting the booking application on their university laptop's screen. Although the application was tested to be suitable on university laptops with a recommended screen size of 1600 by 900 pixels, the height of the application has been reduced, so as to fit on screens even smaller than common laptop screens (1366 by 768 pixels).

It was also reported by one student that the workload was somewhat lower than for the previous two labs, saying that it didn't take them the whole day to finish.

Two teaching assistants gave oral feedback regarding the teaching assistant instructions. They found their lab session instructions lacking in some detail, namely how the upper and lower bounds for the amount of t-way combinations are found and how to present this to the students. The document was then improved with more detailed calculations and instructions, for example how to find the upper and lower bound for the amount of test cases needed to cover all t-way interactions.

4.3 Future Improvements

As seen in figures 9 and 10, a fifth of the students remained neutral towards statements 4 and 5. This indicates that some tweaks to the lab session instructions could make the introduction and learning of the ACTS tool's features more interesting, while also increasing its usefulness in the eyes of the students.

The neutrality towards statement 6 is also quite high at over 26%, indicating that some changes to the existing homework tasks or adding one or two additional tasks could make the testing of the Booking application more interesting and useful. For example, a new task could be created that revolves around solving the test oracle problem by introducing

model-based test oracles. D.R. Kuhn, R. Kacker and Y. Lei have provided a thorough example of how to do this in their paper titled "Practical Combinatorial Testing" [10] using an application called NuSMV[4].

As standard screen sizes of university laptops might increase in the future, the text size of the Booking application might also need to be increased in order to be legible.

The potential distribution of the files which are required from students for the grading process should also be considered. To discourage or maybe even catch such academically dishonest behaviour, it is suggested that the exact values which trigger failures in homework tasks 1-3 should be changed every couple of years.

[4]http://nusmv.fbk.eu/

5 Conclusion

The purpose of this thesis is to create lab materials, to be used in the new combinatorial testing lab, for The University of Tartu's course "Software Testing (MTAT.03.159)". Combinatorial testing is a software testing technique for systematically testing t-way interactions of configuration or input parameter values.

As a result of this thesis, lab materials were created and implemented in the 2018 spring semester lab on combinatorial testing. The lab materials consist of instructions for the students and for teaching assistants. Two applications were also created to be the systems under test for the lab session and homework tasks. Feedback was also collected in the form of an online questionnaire.

Although the feedback was mostly positive, there was feedback due to which some improvements have already been made, but future improvements are suggested as well. Some of these suggested improvements combat issues that haven't manifested yet, but could in the future. For example how to prevent or catch academic dishonesty, should the files required for submission in this lab become distributed and reused by students that have not actually done the work themselves. Other improvements are suggested to make the homework tasks more interesting and useful, for example the introduction of a model-based test oracle.

References

[1] *Software Testing course outline*. URL: https://courses.cs.ut.ee/MTAT.03.159/2018_spring/uploads/Main/swt2018-outline-v1.pdf. (08.03.2018).

[2] D.R. Kuhn; R. Kacker; Y. Lei; J. Hunter. "Combinatorial Software Testing". In: *Computer* (Aug. 2009), pp. 94–96. DOI: 10.1109/MC.2009.253.

[3] D.R. Kuhn; D.R. Wallace; A.M. Gallo. "Software fault interactions and implications for software testing". In: *IEEE Transactions on Software Engineering* (Aug. 2004), pp. 418–421. DOI: 10.1109/TSE.2004.24.

[4] D.R. Kuhn; R. Kacker; Y. Lei. "Practical Combinatorial Testing". In: *NIST Special Publication* (Oct. 2010), pp. 9–10. URL: https://courses.cs.ut.ee/MTAT.03.159/2018_spring/uploads/Main/SWT_comb-report.pdf. (14.04.2018).

[5] D.R. Kuhn; R. Kacker; Y. Lei. "Practical Combinatorial Testing". In: *NIST Special Publication* (Oct. 2010), pp. 13–15. URL: https://courses.cs.ut.ee/MTAT.03.159/2018_spring/uploads/Main/SWT_comb-report.pdf. (14.04.2018).

[6] Y. Lei; R. Kacker; D. Kuhn; V. Okun; J. Lawrence. "IPOG: A General Strategy for T-Way Software Testing". In: *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)* (Mar. 2007), pp. 549–556. DOI: 10.1109/ECBS.2007.47.

[7] Y. Lei; R. Kacker; D. Kuhn; V. Okun; J. Lawrence. "IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing". In: *Software Testing, Verification, and Reliability* (Sept. 2008), pp. 125–148. DOI: 10.1002/stvr.381.

[8] *Course webpage*. URL: https://courses.cs.ut.ee/2018/SWT2018/spring/. (08.03.2018).

[9] Rob Johns. *Likert items and scales*. 2010. URL: https://www.ukdataservice.ac.uk/media/262829/discover_likertfactsheet.pdf. (11.04.2018).

[10] D.R. Kuhn; R. Kacker; Y. Lei. "Practical Combinatorial Testing". In: *NIST Special Publication* (Oct. 2010), pp. 46–54. URL: https://courses.cs.ut.ee/MTAT.03.159/2018_spring/uploads/Main/SWT_comb-report.pdf. (14.04.2018).

Appendix

I. Feedback Questionnaire

	COMPLETELY DISAGREE (1)	SOMEWHAT DISAGREE (2)	NEUTRAL (3)	SOMEWHAT AGREE (4)	COMPLETELY AGREE (5)	TOTAL	MEAN
The goals of the lab were clearly defined and communicated	6.67% 1	6.67% 1	13.33% 2	20.00% 3	53.33% 8	15	4.07
The tasks of the lab were clearly defined and communicated	6.67% 1	6.67% 1	0.00% 0	46.67% 7	40.00% 6	15	4.07
The materials of the lab were appropriate and useful	0.00% 0	6.67% 1	0.00% 0	33.33% 5	60.00% 9	15	4.47
The ACTS tool was interesting to learn	6.67% 1	6.67% 1	20.00% 3	20.00% 3	46.67% 7	15	3.93
If I have the choice, I will use the ACTS tool again	13.33% 2	6.67% 1	20.00% 3	13.33% 2	46.67% 7	15	3.73
The "Booking" application was interesting and useful to test	0.00% 0	6.67% 1	26.67% 4	46.67% 7	20.00% 3	15	3.80
The support received from the lab instructors was appropriate	0.00% 0	13.33% 2	13.33% 2	33.33% 5	40.00% 6	15	4.00
The grading scheme was transparent and appropriate	0.00% 0	0.00% 0	13.33% 2	53.33% 8	33.33% 5	15	4.20
Overall the lab was useful in the context of this course	6.67% 1	6.67% 1	0.00% 0	26.67% 4	60.00% 9	15	4.27

II. Lab Materials

Student Lab Instructions

A1.1 "Lab instructions - Combinatorial Testing Lab Material by Andrus Lall", PDF file
https://courses.cs.ut.ee/2018/SWT2018/spring/uploads/Main/SWT2018-lab03v8.pdf

A1.2 "Marriage Checker application", ZIP file
https://courses.cs.ut.ee/2018/SWT2018/spring/uploads/Main/SWT2018-lab3-MarriageChecker.zip

A1.3 "Booking application", ZIP file
https://courses.cs.ut.ee/2018/SWT2018/spring/uploads/Main/SWT2018-lab3-Booking.zip

Teaching Assistant Instructions

A1.4 "TA Instructions - Combinatorial Testing Lab Material by Andrus Lall", PDF file

A1.5 "Solution files", ZIP file

For confidentiality reasons, teaching assistant materials are not made available in the thesis, but will be made available on request.

Source code for applications

A1.6 "Marriage Checker project", ZIP file containing Intellij[5] project files for the Marriage Checker application

A1.7 "Booking project", ZIP file containing Intellij project files for the Booking application

For confidentiality reasons, source code for the applications created as a part of this thesis are only made available upon request.

[5]https://www.jetbrains.com/idea/

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Andrus Lall**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Lab Package: Combinatorial Testing

   supervised by Dietmar Pfahl

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2018