UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Kert Prink

Lab Package: Automated GUI Testing

Bachelor's Thesis (9 ECTS)

Supervisor: Prof. Dietmar Pfahl

Tartu 2019

Lab Package: Automated GUI Testing

Abstract:

The goal of this thesis is to create lab materials about a way of GUI testing for the course "Software Testing" (LTAT.05.006) at the University of Tartu. The thesis gives an overview of the motivation of this lab, introduces the created materials, analyzes feedback and makes suggestions for future improvements for the created materials. This lab was carried out in the 2019 spring semester.

Keywords:

GUI, software testing, lab package

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

Praktikumimaterjal: Automatiseeritud graafilise kasutajaliidese testimine

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärk on praktikumimaterjalide loomine graafilise kasutajaliidese testimise kohta Tartu Ülikooli kursuse "Tarkvara Testimine" (LTAT.05.006) jaoks. Töös kirjeldatakse praktikumimaterjalide eesmärki, töö käigus loodud materjale, analüüsitakse tagasisidet ja tehakse ettepanekuid materjalide paremaks muutmiseks tulevikus. Loodud materjale rakendati 2019 kevadsemestril.

Võtmesõnad:

Graafiline kasutajaliides, tarkvara testimine, praktikumimaterjal

CERCS: P170, arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Table of Contents

1. Introduction

Software testing is a part of software development process, which determines software quality. There are several different ways of software testing, some of which are introduced in the course "Software Testing" (LTAT.05.006) at the University of Tartu. This thesis focuses on a way of GUI (graphical user interface) testing, where test scripts are generated automatically by the tool.

"Software Testing" is targeted at second year Computer Science Bachelor curriculum students. To teach the students creativity and open-minded attitude for determining software quality, different ways of software testing are covered in the labs. Until now, GUI testing with automatically generated scripts, has not been covered before.

GUI testing is a process to ensure the GUI meets its specifications. This is normally done with test scripts developed by testers. If software changes, testers must perform regression testing in order to ensure that the previously tested software still meets its requirements. This can be time consuming, because it might be necessary to develop new test scripts. To reduce the time spent for fixing and redeveloping the tests, automatically generated scripts can be used.

The purpose of this thesis is to create lab materials for teaching a way of GUI testing that can help find critical GUI faults with automatically generated scripts. Created materials were used in a lab session, followed by a homework assignment. The tool used for automated testing within the created materials, is called Testar[1].

This thesis consists of five main chapters. Chapter two gives an overall background information about the subject. Chapter three describes the lab and the materials created. Fourth chapter describes the usage of the created materials. In the fifth chapter, analyzation of the feedback, given by students, is done. The final, sixth, chapter is a summary of the thesis.

[1] Testar tool. https://testar.org/

2. Background

Chapter two gives an overview of the thesis background. First section describes the course[2], which this thesis was based on. Section 2.2 gives an overview of the current situation in GUI testing. The last section, 2.3, describes the tool, used for testing, within the created materials.

2.1 "Software Testing" Course

"Software Testing" (LTAT.05.006)[2] is a 6 ECTS course taught in spring semester at the University of Tartu. The course is a part of Software Development (24 ECTS) specialty module for students, who want to continue their studies in the Master's program or who want to begin their career as a software developer after graduation. The course outline states: "The course addresses the essential concepts of software quality control and testing and introduces various testing strategies and types of testing. It will also give an overview of different software defects, software defect management, and organizational aspects of software testing [1]." In 2018/2019 the course consisted of 13 lectures and 11 labs. The topics of the labs were as follows:

1. Debugging
2. Basic Black-Box Testing
3. Combinatorial Testing
4. Basic White-Box Testing
5. Automated Web-Application Testing
6. Automated Integration Testing
7. Web-Application Testing In The CI/CD Pipeline
8. Automated GUI Testing
9. Mutation Testing
10. Static Code Analysis
11. Document Inspection and Defect Prediction

The purpose of this thesis is to give materials for lab 8 – Automated GUI Testing, which has never existed before.

[2] Software Testing course in 2018/19 spring. https://courses.cs.ut.ee/2019/SWT2019/spring

2.2 Automated GUI Testing

GUI testing is a process to ensure that GUI meets its specifications. Tanja E.J. Vos along with other authors [2] have made an overview of a current situation in GUI testing.

Today, state of the art GUI testing tools are Capture & Replay and Visual testing tools. Capture & Replay tools record sequences, that user executes manually and it can be replayed later in regression testing as an automated test case. Problem is that these tools assume that the UI (user interface) structure does not change, and when it does, the scripts break and that causes a maintenance problem. To solve the maintenance problem, Visual testing tools use image processing to simulate the operations testers carry out manually. Problem with these tools is that they rely on the graphical stability of the UI, which means that changes to the UI threaten to break the scripts.

2.3 Testar

To solve the maintenance problem, Testar[1], a tool for automated GUI testing, introduced a "scriptless" way of testing [3]. The "scriptless" way actually means that the scripts are generated and executed automatically by the tool, which means, there is less maintenance issues regarding to test scripts, if the UI changes [2].

Testar[1] uses the operating system's Accessibility API[3] to derive possible actions from the UI, then it selects and executes one of the derived actions, which brings the SUT (system under test) into a new state, and after that, oracles are used to determine the correctness of the state [2, 4]. The illustration of the Testar[1] test cycle can be found in Figure 1 [4].

[3] Windows Automation API. https://docs.microsoft.com/et-ee/windows/desktop/WinAuto/windows-automation-api-portal

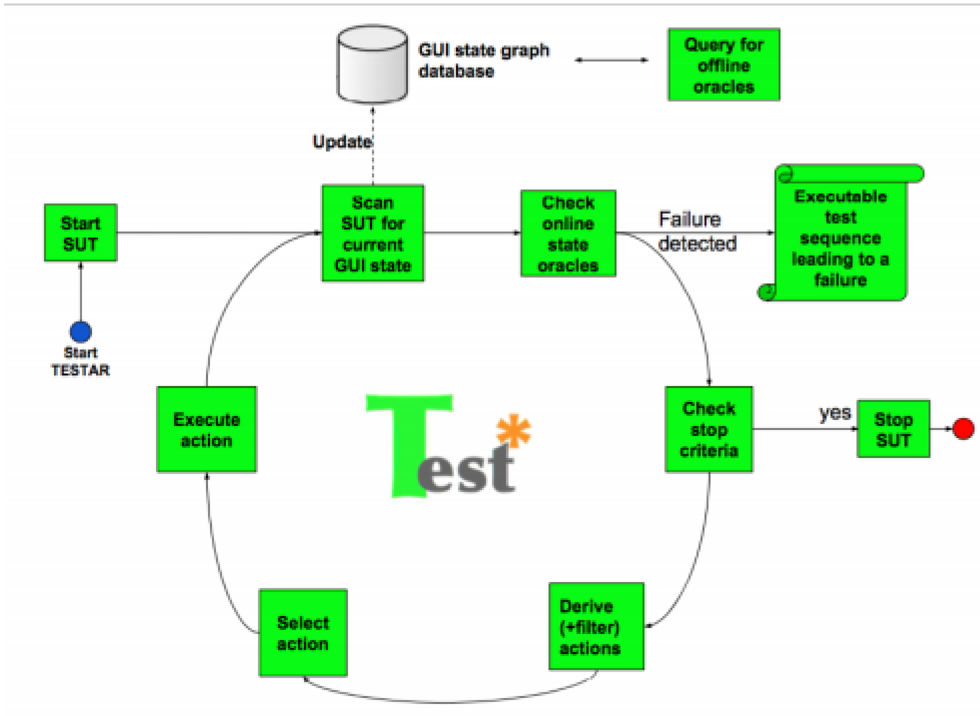

Figure 1. Testar[1] test cycle [4].

Figure 1 describes the test cycle after Testar[1] have started the SUT [4]:

- Scan the SUT for the current state of the GUI
- Check the oracles for failure
  - If oracle is matched, report a sequence leading to failure
  - If not, continue without reporting
- Check the stop criteria
  - If the amount of actions per test sequence is reached, then stop exit
  - If not, continue
- Derive possible actions from the GUI and filter out forbidden ones
- Select and execute an action
- Repeat the cycle

Testar[1] records the tests it generates, which means that every test can be manually inspected, if something suspicious is found [5].

2.3.1 Industry Test Results

Testomat Project[4], conducted testing of an actual product with Testar[1] in a company called ClaveiCon and found 10 previously unknown critical faults of their Enterprise Resource Planning system [2]. After that, they tested the tool with another company, called Softeam, where, in order to measure FDR (fault detection rate), existing faults were re-injected into the SUT [6]. Table 1 shows the comparison of Testar[1] and manual testers: preparation time was almost the same; Testar[1] took 1.5 hours more after testing; in terms of FDR, manual testers had 22% better results; Testar[1] took 76 hours more on testing, but it was autonomous; manual testers had 16% more code coverage [2][6]. Table 2 shows results from a company called Gap Gemini / ProRail: preparation time was almost the same on both sides; Testar[1] took 45 hours more time on testing, but it was autonomous; Testar[1] required 3 hours more time after running the tests; Testar[1] had 7% more code coverage; Testar[1] found 4 critical faults, while manual testers found none [6]. These results show that Testar[1] can have usage in real world UI testing.

Table 1. Testar[1] results at Softeam [6].

	Testar	Manual
Preparation time	40 hours	36 hours
Testing time	77 hours	1 hour
Post testing time	3.5 hours	2 hours
Fault detection rate	61%	83%
Code coverage	70%	86%

[4] Testomat project. https://www.testomatproject.eu/

Table 2. Testar[1] results in Cap Gemini / ProRail [6].

	Testar	Manual
Preparation time	44 hours	43 hours
Testing time	51 hours	6 hours
Post testing time	5 hours	2 hours
Critical faults found	4	0
Functional coverage	80%	73%

3. Lab Design

The following chapter gives an overview of the lab and the materials provided for it. Before the materials were declared as ready for in lab usage, the author, acting as a student with no clue about the solutions, solved the assignment and measured the time it took. Time, allocated for the lab (90 min), was filled with the lab assignment, and the homework task showed the desired level of difficulty was met, as the author took about two hours to solve it. Solving the homework assignment met two goals: desired difficulty was confirmed and unknown problems were found and fixed.

3.1 Schedule

The course allocates approximately 8 academic hours for each lab – 2 hours (90 min) for the lab itself and 6 hours (270 min) for the homework task. The assignments could be solved alone or with in pairs of another student from the same lab group.

3.2 Tasks

The lab consisted of two assignments. For the in-lab task, a faulty calculator application was developed in Java[5] and executable jar[6] file was provided for the students. Similarly, a faulty ATM (automated teller machine) application was provided for the homework task. Both tasks are explained in 3.2.1 and 3.2.2. The students are expected to solve the tasks and fill in the corresponding reports.

Reports and sample solutions were provided with the lab instructions. The in-lab task was meant to be used as an introduction and first hands-on experience. Because of the report forms were mostly the same, the in-lab task was also used to teach how to report the homework task. Only the homework task was graded.

Aim of this lab was to teach the students how to test the GUI with automatically generated scripts. In this way of testing with Testar[1], tests cannot determine if the system functions correctly, instead they will determine the robustness of the system. This means that Testar[1] cannot determine if 2+2 equals 4. It can detect critical failures like crashes, hangings and if some string matches user specified oracles. If something matches the oracles, it is reported as a 'suspicious', because it can

[5] Java. https://www.java.com/en/

[6] Java Archive (JAR) file format. https://docs.oracle.com/javase/tutorial/deployment/jar/index.html

either be false or true positive[7]. After the testing, students had to check manually the test sequences, which Testar[1] reported, determine the true positives[7] and fill in the report.

To help the students for writing the oracles and determining false and true positives[7], documentations about the SUT's were provided along with the lab instructions as appendixes. These appendixes described how the systems should behave and which were allowed information messages, that the system could show. If some of those allowed messages were reported as a failure, it was considered as a false positive[7].

To grade the homework task, students needed to fill in the report provided with the lab instructions. In that report they needed to describe the settings used in Testar[1], the failure itself, how to regenerate the failure step by step, and which combination of settings allowed to find the concrete failure. In addition to that, they needed to provide a picture of a graph to show how Testar[1] detected this failure. The reporting was made that way in order to detect possible plagiarism.

3.2.1 Lab Task

The lab started with introduction to the subject, after which the students received their first task. For the first task, a simple calculator application and instructions how to use Testar[1], were provided. The aim of this task was to teach the students how to use Testar[1] and what can be done with this kind of tool. The knowledge gained from this task helped the students to solve the homework task after the lab.

The calculator was programmed in Java[5] and failures were implemented within the code. There was a total of nine failures implemented. If the failure occurred, a message was shown at the input field, seen in Figure 2. In order to detect the failure, a matching oracle must have been configured. The oracles are written as regular expressions[8] and they are used to detect suspicious messages that can be failures [4]. An example matching oracle to the failure displayed in Figure 2, is shown at Figure 3.

[7] Classification: True vs. False and Positive vs. Negative. https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative
[8] Regular Expression Language. https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference

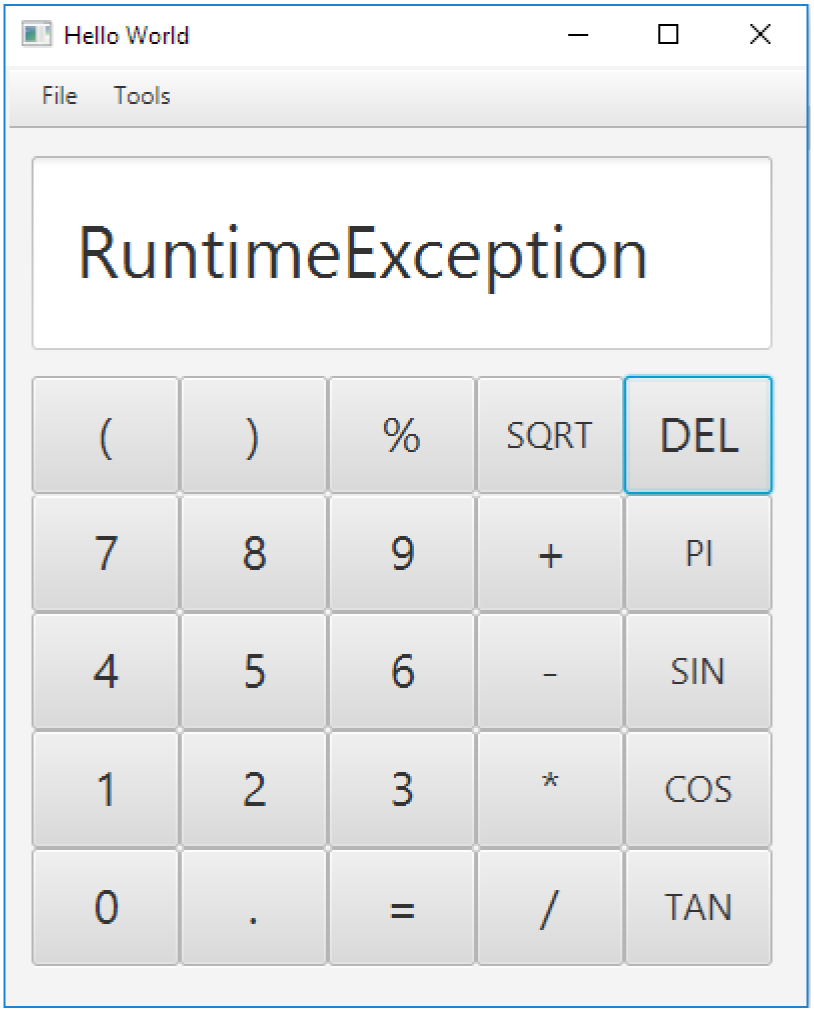

Figure 2. Calculator input field showing failure message.

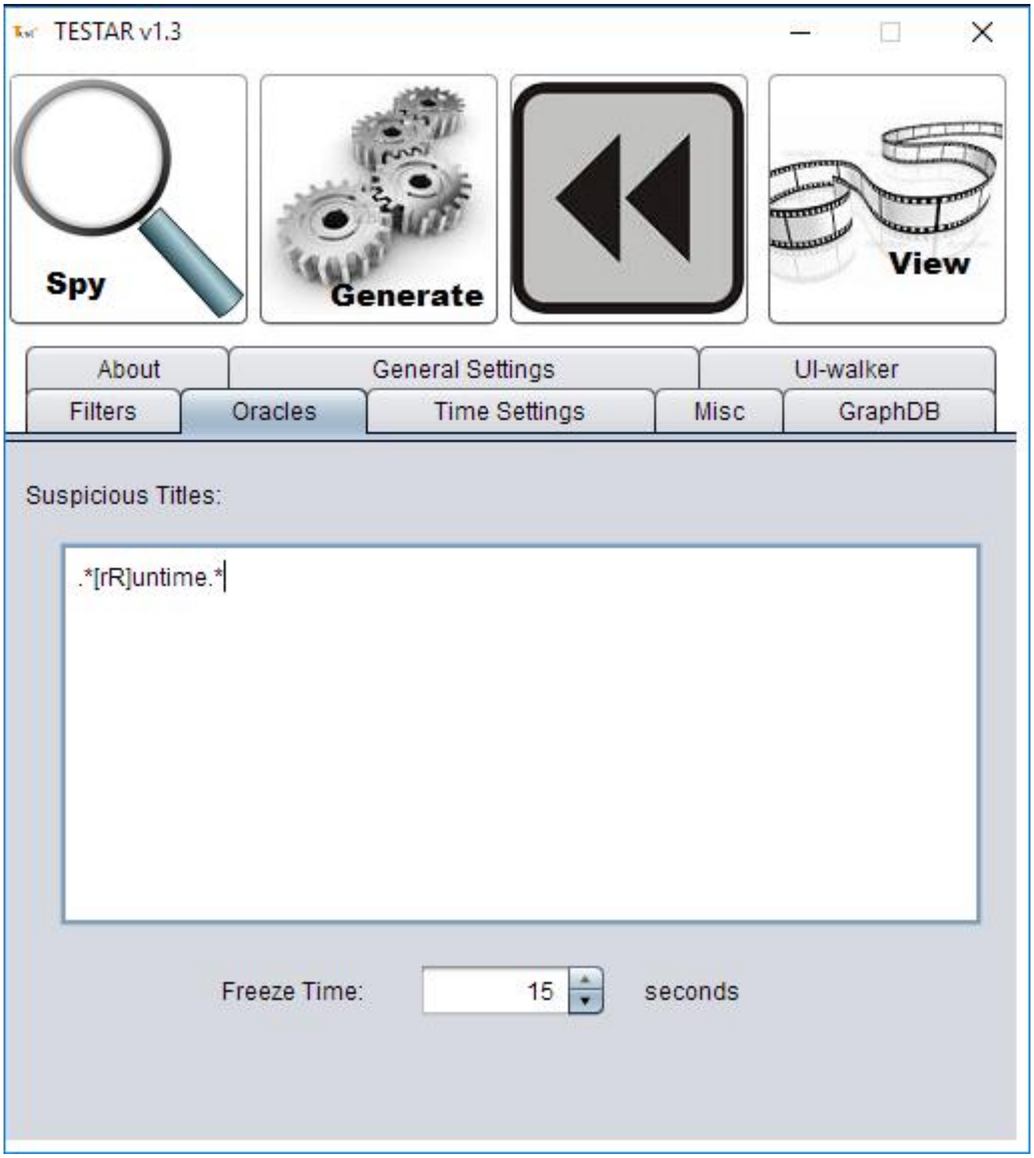

Figure 3. Example matching oracle to failure in Figure 2.

Lab materials provide step-by-step instructions configuring Testar[1] in order to find failure, shown in Figure 2. However, it was not certain that students would find this failure with first runs, because of the randomness how Testar[1] generates sequences of actions. After the step-by-step instructions, the students would have time to play around with the tool, learn how to configure different settings and how to fill in the failure report. The failure report for this task was not graded, it was used to teach how to report the homework task.

3.2.2 Homework Task

The SUT for the homework task was an ATM. The reporting structure was the same as for the lab task. The aim of this task was to teach, what can be tested and what are the pros and cons with this way of testing. As a side effect, it also taught how to use the tool efficiently. The task would make the students to do more work, while analyzing output of the tool after running the tests, if they do not use the tool in a smart way.

The ATM had five different account types, which some had some unique functionality and some had different rules for the same functionality. The failures implemented to the application had different detection difficulty. This means that some of the failures could be found with few clicks and some required specific state before they could appear. The desired difficulty for the homework task was achieved by combining the number of failures and how specific state the program should have before the failure could be triggered.

3.3 Materials

The materials created within the context of this lab can be divided into two parts: free access and restricted access materials. Free access materials contain all the data which is not used for grading and restricted access materials contain information for the lab assistants.

An overview of the materials division can be seen in Figure 4. These materials can be found in Appendix I.

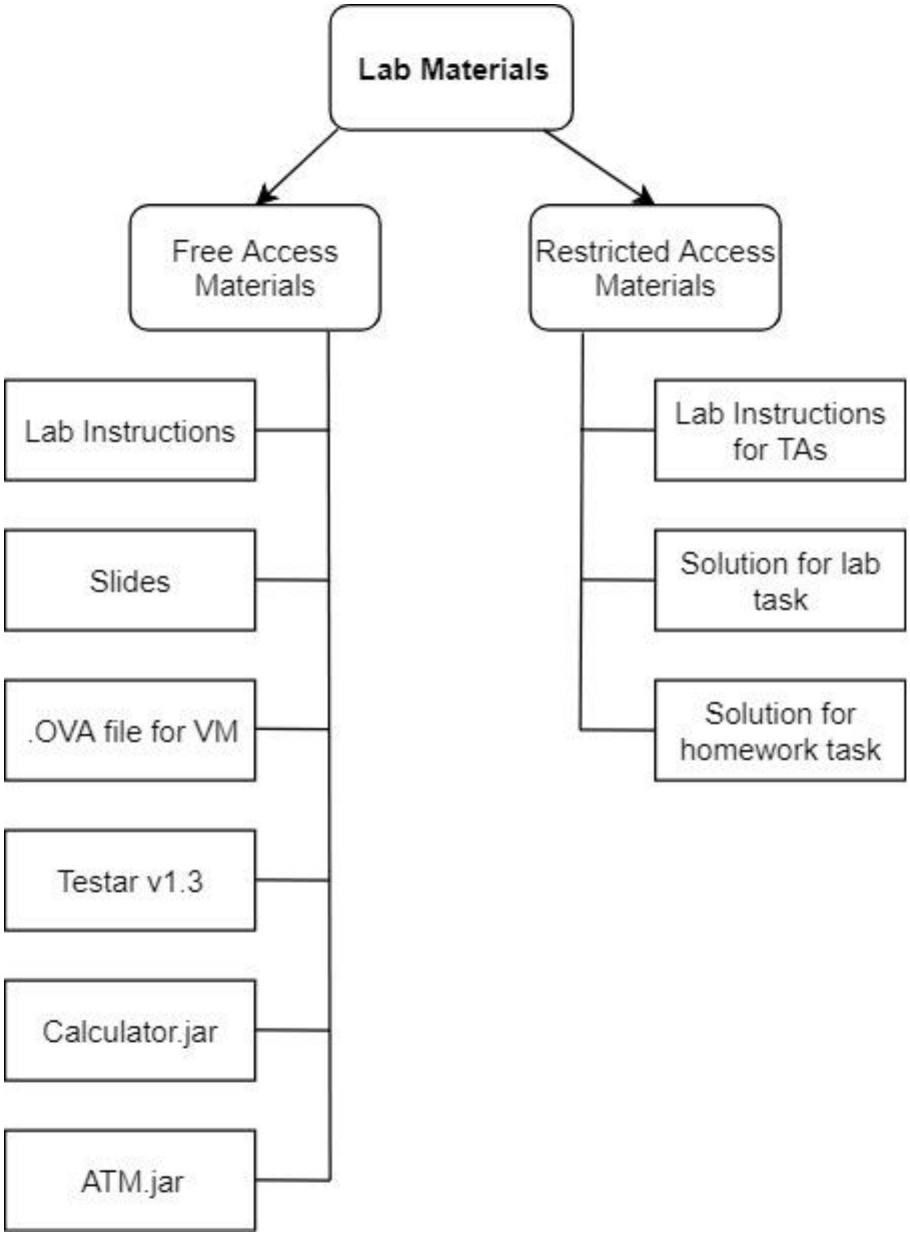

Figure 4. Structure of the materials.

3.3.1 Free Access Materials

"Lab Instructions" is a guide for the whole lab. It starts with introduction to the topic to explain the purpose of the lab, vocabulary and to give background information. The second chapter contains step-by-step instructions of how to use the tool, which is followed by a task, meant to give the students first individual experience of the topic. Third and fourth chapters contain homework and grading information. The last, fifth, chapter is used for links and references. Appendixes for the lab and homework task follow after the last chapter.

“Slides” are meant for the TAs (teaching assistants) to support the start of the lab and cover the introduction part of the “Lab Instructions”. The slides start with introduction to the topic, followed by an introduction of Testar[1] and its usage. The slides also give hints and guidance for the tasks in the lab.

“.OVA file” is an open virtualization format[9] file, which contains Windows 10 virtual machine[10]. The virtual machine[10] contains Testar[1] and the two SUTs: one for the lab and one for the homework task. Virtual machine[10] is used for two reasons: to allow students with non-Windows OS to complete the lab; and to increase safety of the actions to students’ personal computers, because Testar[1] selects and executes actions, it derives from the Windows Automation API[3], and on some cases, it can happen, that Testar[1] executes an action, not related to the SUT.

“Testar v1.3” is Testar[1] version 1.3, which was the newest version when the lab materials development begun. At the time of writing this thesis, version 2 is available as a development version[11].

“Calculator.jar” and “ATM.jar” are executable jar[6] files, used for lab and homework tasks.

3.3.2 Restricted Access Materials

“Lab Instructions for TAs” contain information about the lab execution in addition to the information from the “Lab Instructions”. “Solution for lab task” and “Solution for homework task” are included to the according report forms. These materials also include hints to give for the students and a conclusion for the solution of the homework task the author executed while testing the complexity of the lab.

[9] About OVA file. https://fileinfo.com/extension/ova
[10] About virtual machine. https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/
[11] Testar repository. https://github.com/TESTARtool

3.4 Grading

Students could get maximum of 10 marks for each lab. One lab, Automated Integration Testing, gave additional 2 bonus marks. Distribution of marks for this lab were as follows:

- 1 mark for attending the lab.
- Up to 9 marks from homework task:
  - Up to 2 marks for describing the settings
  - Up to 7 marks for discovered failures (each unique failure 1.4 marks)

Submission structure must have been followed, in order to get full marks. A sample solution was provided for easier understanding what was expected. Reporting the sample, did not gave any marks. TAs could apply penalties for incorrect submission and the decision for the size of the penalty remained to the TAs.

4. Lab Execution

The created materials, were used in the "Software Testing" course[2] lab 8 in April 2019. The course had 97 students registered and they were divided into five groups, whom each was guided by a lab assistant.

The students had to be familiar with regular expressions[8], taught in "Automata, Languages and Compilers" course[12], and with virtual machines, taught in "Operating Systems" course[13]. "Software Testing" course[2] is scheduled parallel with the "Automata, Languages and Compilers" course[12] and after the "Operating Systems" course[13]. This means that most of the students were familiar with regular expressions and virtual machines, but still, installation guide for the virtual machine and link for regular expression introduction were provided. It was required to prepare for the lab and download all the materials, because downloading the materials in the lab would waste time. It was said to the students in a lecture before the lab. Nevertheless, lab assistants had portable storage devices with materials for students, who did not prepare.

The author, participated in two labs, for observation and support. In addition to physical participation, online help by email was also provided during the lab week. It was seen in the labs that the lab assistants performed well and only few students had issues, which were solved on spot.

This lab was different than the usual labs in the course[2] – it required usage of a virtual machine and it started with a theory, which was taught in a form of a lecture. Overall the usual time schedule of the lab was about: introduction and lecture – 40min; solving the lab task – 30 min; questions and introduction to homework – 20min.

All in all, the average points, gained from the lab, were 7.8 out of 10. The average points were calculated excluding students, who did not submit their solution. In total, 14 students did not submit, and about half of them did not plan to complete the course[2].

[12] Automata, Languages and Compilers course in 2018/19 spring. https://courses.cs.ut.ee/2019/AKT/spring
[13] Operating Systems course in 2018/19 spring. https://courses.cs.ut.ee/2019/opsys/spring

5. Feedback

The following section gives an overview of the feedback from students and lab assistants. The students had to write feedback for the lab in the following lab, in the following week. It was decided that way, because previous year, 2018, had showed, that if the students would have opportunity to give the voluntary feedback, only few would do it [7].

From 97 students, 59 answered the feedback questionnaire.

5.1 Feedback Collection

Feedback was collected from students in the following lab using a quantitative questionnaire on a paper. The questionnaire was based on a scale from “strongly-disagree” to “strongly-agree”. The statements were:

1. The goals of the lab were clearly defined and communicated
2. The tasks of the lab were clearly defined and communicated
3. The instructions of the lab were appropriate and helpful
4. The tools used in the lab were appropriate and useful
5. Compared to the previous labs, the homework assignment was more difficult
6. Overall, what I learned in the lab is relevant for working in the software industry
7. Overall, the lab was interesting and inspiring

In addition, a free form text field was provided for qualitative feedback.

5.2 Analysis

The feedback from students was mostly positive. However, it was different in qualitative and quantitative forms – it was more positive in the quantitative form and more neutral or negative in the qualitative form. This happened mostly because, the students with positive experience did not write any additional comments, while students with negative experience expressed their opinion in the comments section more often.

5.2.1 Qualitative Feedback

Qualitative feedback was neutral or negative, with few exceptions. Most of the students explained their thoughts about Testar[1] and did not comment the lab materials.

On the positive side, students told that they liked the idea of the lab and the virtual machine, although some of them pointed out it being quite slow. As some of the students had doubts of the usefulness of Testar[1], others told they liked it and think it is useful. One student told that the potential of Testar[1] would have been seen better, if the SUT for the homework task had been more complex. On a neutral or negative side, it was pointed out that Testar[1] needs improvements regarding to user experience, and that it needs more online documentation with examples and tutorials.

Some of the students pointed out, that they could not get started because of some error. Some of them told that they had problems with initial configuration and some pointed out they had difficulties getting the filters working. In some cases, students raised issues about the subjects, which were covered in the lecture or in the lab. Due to that, it seemed like the writer of the feedback did not participate in the lab nor in the lecture, otherwise they would not have had issues they were writing about. For example, one student told that he/she could not get anything running within 10 hours. This means, it is important that the students participate in the lab, although all the materials, which are not meant for lab assistants private use, such as grading and homework solution, are freely accessible in the course webpage[2].

5.2.2 Quantitative Feedback

Results of the quantitative part are summarized in Table 3 and illustrated in Figure 5. Based on these results, the lab materials proved to be useful and the lab was a success. 86.4% of the students found the goals of the lab clearly defined and communicated. 81.4% found the tasks clearly defined and communicated. 83.9% found the instructions to be appropriate and helpful. The choice of the tool, Testar[1], was approved by 66.1% of the students and 17.0% did not have clear opinion on this matter. Although, 50.8% of the students found the homework task equal or more difficult, than the other homework in the course[2], the most popular opinion was "disagree" to the Q5, seen in Table 3. 52.5% of the students learned something relevant for working in the software industry, 32.2% did not have clear opinion, and 15.3% thought they did not learn anything useful. In conclusion, 52.6% of the students found the lab interesting and inspiring, 30.5% had neutral opinion, and 16.9% found the lab not interesting nor inspiring.

Table 3. Quantitative questionnaire feedback.

	Strongly disagree	Disagree	So-so	Agree	Strongly-Agree
Q1: The goals of the lab were clearly defined and communicated	0% 0	6.78% 4	6.78% 4	37.29% 22	49.15% 29
Q2: The tasks of the lab were clearly defined and communicated	3.39% 2	5.08% 3	10.17% 6	28.81% 17	52.54% 31
Q3: The instructions of the lab were appropriate and helpful	3.57% 2	3.57% 2	8.93% 5	41.07% 23	42.86% 24
Q4: The tools used in the lab were appropriate and useful	6.78% 4	10.17% 6	16.95% 10	33.90% 20	32.20% 19
Q5: Compared to the previous labs, the homework assignment was more difficult	11.86% 7	37.29% 22	27.12% 16	16.95% 10	6.78% 4
Q6: Overall, what I learned in the lab is relevant for working in the software industry	3.39% 2	11.86% 7	32.20% 19	42.37% 25	10.17% 6
Q7: Overall, the lab was interesting and inspiring	8.47% 5	8.47% 5	30.51% 18	35.60% 21	16.95% 10

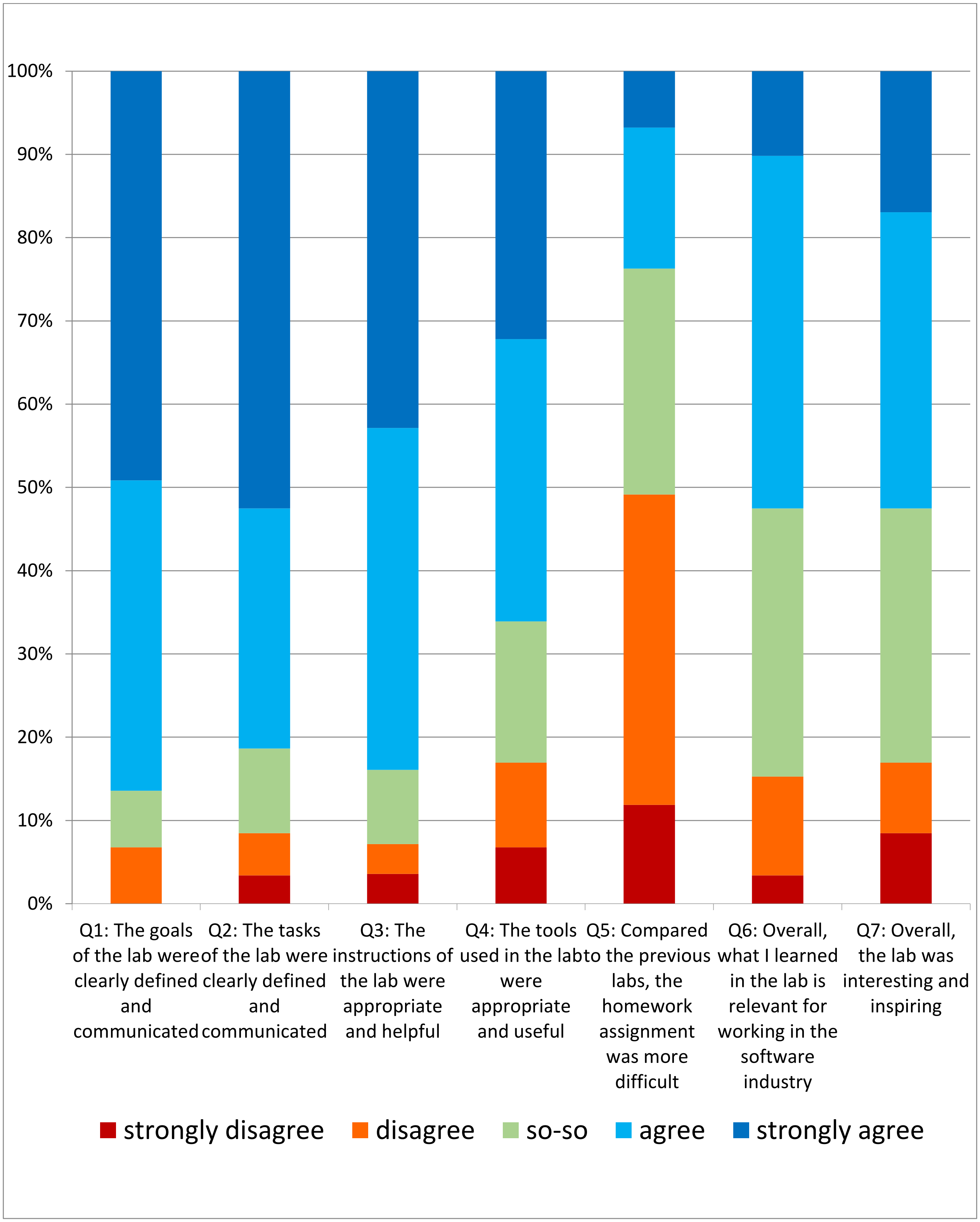

Figure 5. Quantitative questionnaire feedback.

5.3 Future Improvements

Improvements can be made for the lab. First, an introduction exercise with regular expressions[8] could be used in the introduction part of the lab materials as a warm up for the upcoming tasks. This would reduce the issues, where students think that filters do not work, but instead there is an issue with miswritten regular expressions[8]. Next, the provided virtual machine, could contain an already configured Testar[1], with previously found failures. This would be an example in addition to the clean Testar[1], which the students would use for the lab and homework task. It could be done so that the students would have a working and configured example, which would help them in the initial configuration of the lab task, but would be different enough so that copy-paste would not work. This is something that can be done to minimize the problems of not having enough examples of how to use the tool.

In order to give better understanding of the usefulness of "scriptless" testing and Testar[1], a more complex homework task could be implemented. This task could have fewer failures, but they would be more difficult to find. One thing that can increase the satisfaction of the lab, is the speed of which Testar[1] executes tests. To increase the speed, Testar[1] could be used outside of the virtual machine. It will require additional information in the lab materials of how to use Testar[1] outside the virtual machine and how to do it safely. This is because on some cases Testar[1] can execute actions elsewhere than on the SUT.

6. Conclusion

The purpose of this thesis was to create lab materials for "Software Testing" (LTAT.05.006) course[2], taught at the University of Tartu. In the scope of this thesis, lab materials for automated GUI testing were created and used in 2018/2019 spring semester. This way of GUI testing, with automatically generated scripts, had not been taught before in this course[2].

After the execution of the lab, feedback was collected from the students to analyze the usefulness of the materials created and to find out the shortcomings. Although, the feedback was mostly positive, some negativity was received as well. Based on the feedback, future improvements for the created materials were made. All in all, the materials proved to be useful.

References

[1] University of Tartu, "Õppeaine Üldandmed," [Online]. Available: https://www.is.ut.ee/rwservlet?oa_aine_info.rdf+1228876+HTML+0+text/html. [Accessed 31 03 2019].

[2] T. E. Vos, N. Condori-Fernandez, P. M. Kruse, J. Wegner, "TESTAR: Tool support for test automation at the user interface level" *International Journal of Information System Modeling and Design*, 2015.

[3] Testomat Project, "2 - How can we automate GUI testing?," [Online]. Available: https://youtu.be/KiIgyoPpcTE. [Accessed 31 03 2019].

[4] The TESTAR team, "Automated Testing at the GUI level Hands-on do it yourself session," 2018 [Online]. Available: https://testar.org/images/Hands_on_TESTAR_Training_Manual_2018nov27.pdf. [Accessed 31 03 2019].

[5] U. Rueda, T. E. J. Vos, F. Almenar, M. Oreto, A. E. Alcazar, "TESTAR - from academic protoype towards an industry-ready tool for automated testing at the User Interface level" 2015 [Online]. Available: https://biblioteca.sistedes.es/wp-content/uploads/2015/08/4_paper_711.pdf. [Accessed 31 03 2019].

[6] Testomat Project, "3 - What does TESTAR test Out Of The Box?," [Online]. Available: https://youtu.be/8BRE7r09P9M. [Accessed 31 03 2019].

[7] H. H. Raju, "Lab Package – Debugging", University of Tartu *Institute of Computer Science*, 2018. Available: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=61635&year=2018

Appendix

I. Lab materials

1. Free Access Materials

1.1 “Lab Instructions”, pdf file - https://courses.cs.ut.ee/LTAT.05.006/2019_spring/uploads/Main/SWT2019-lab08-20190314.pdf

1.2 Slides, pptx file - https://courses.cs.ut.ee/2019/SWT2019/spring/uploads/Main/SWT2019_Lab_8_slides_05.04.19.pptx

1.3 Virtual machine (includes Testar and SUTs), ova file - https://owncloud.ut.ee/owncloud/index.php/s/9ZyrR2tkHSSdiYK

1.4 Testar with SUTs (is also included in the virtual machine), zip file - https://courses.cs.ut.ee/2019/SWT2019/spring/uploads/Main/SWT2019-lab08-testar.zip

2. Restricted Access Materials

2.1 “Lab Instructions for TA”, pdf file

2.1.1 Solution for lab task

2.1.2 Solution for homework task

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

II. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kert Prink**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Lab Package: Automated GUI Testing,

   supervised by Dietmar Pfahl.
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kert Prink

08/05/2019