UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Aleks Kožajev

Lab Package: Visual GUI Testing

Bachelor's Thesis (9 ETCS)

Supervisor: Dietmar Pfahl

Tartu 2020

Lab Package: Visual GUI Testing

Abstract:

The goal of this thesis is to create a lab package about visual GUI testing for the course Software Testing (LTAT.05.006) at the University of Tartu. The thesis introduces the materials produced for this lab, analyses the feedback gathered from students who participated in the lab in the spring semester of 2020 and, marks possible areas of improvements based on such feedback.

Keywords:

Software testing, lab package, Sikuli, GUI

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

Praktikumimaterjal: Automaattestimine kasutades CI/CD-süsteemi

Lühikokkuvõte:

Antud bakalaureusetöö eesmärgiks on praktikumimaterjalide loomine visuaalse graafilise liidese testimise kohta aine "Tarkvara testimine (LTAT.05.006)" jaoks, mida loetakse Tartu Ülikoolis. Töös kirjeldatakse loodud materjale, analüüsitakse tudengite, kes osalesid praktikumis 2020 aasta kevadsemestril, tagasisidet ja tähistatakse võimalike arenduskohti.

Võtmesõnad:

Tarkvara testimine, praktikumimaterjal, Sikuli, GUI

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Table of Contents

1 Introduction

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect-free. The course “Software Testing” LTAT.05.006 consists of 11 labs where each lab focuses on one specific topic of software testing. The labs are intended to give an introduction to a topic and teach students practical skills about a given testing practice.

Until now, the topic of Visual GUI Testing wasn’t covered in this course. However, one lab that included visual GUI testing practices took place in 2015. The previous lab’s main objective was to introduce students to regression testing while testing the GUI with Sikuli. The goal of this thesis is to create a Visual GUI Testing lab package that focuses solely on visual GUI testing and helps teach the concepts of visual GUI testing.

Visual GUI testing is a technique that uses image recognition to interact with what is shown on the screen. In the process of VGT, the tester creates a script with valid instructions.

This thesis consists of five main sections. The first section, “Background Information and Existing Materials”, focuses on the background information of the topic and makes the distinction between this work and the previous lab that used Sikuli. The next section, “Lab Design”, describes the materials and software under test that were used in the lab and for homework tasks. The third section, “Lab Execution”, summarizes the execution of the lab. The fourth section, “Feedback”, gives a brief overview of students' feedback and makes suggestions for future improvements of the lab. The final section, “Summary and Conclusions”, concludes the thesis.

2 Background Information and Existing Materials

2.1 LTAT.05.006 "Software Testing"

"Software Testing" (LTAT.05.006) is a 6 ETCS course at the University of Tartu. The course is a part of the Software Development (24 ECTS) specialty module in the bachelor's degree program in the Informatics curriculum. It can also be taken by students from other curriculums, who have passed the compulsory subject for this course (LTAT.05.003 "Software Engineering") [1]. "Software Testing", according to the course outline [2], 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. In the 2019/2020 spring semester, the course consisted of 14 lectures and 11 labs. The topics of the labs were as follows [3]:

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

This thesis will focus on the design and execution of the eighth lab: Visual GUI Testing.

2.2 Automated Testing

Emil Borjesson and Robert Feldt explain in their work [4] that software companies that only work with manual test practices cannot keep up with the industry's demands for faster developed high-quality software. "A market with demands for saving time and money poses challenges for everyone involved in software creation" - state Borjesson and Feldt [4]. Since the software is prone to change, the problem grows even deeper, as it requires spending valuable time on extensive manual testing. Today's software is also usually heavily dependent on the Graphical User Interface (GUI), which in Borjesson's and Feldt's opinion [4], makes manual testing not only time-consuming but complex.

Borjesson's and Feldt's opinion can be proven with the example based on Joseph Bosas' article [5]. Consider, for example, updating web page software. Testing even a basic website with a low count of GUI elements requires the tester to click and test the effect of many buttons and probably a couple of dropdown items. To ensure that everything works as intended on a page,

extensive time of manual testing is required, and it is still unlikely that the system is tested the same way as it was during previous changes.

In his article [5], Bosas points out that one of the ways to ensure the consistency of testing after changes in the software is automated testing. With automated testing, testers can write scripts that cover all the needed GUI elements which ensure that tests stay consistent even after changes in the software. Bosas' article [5] suggests that such consistency helps detect new bugs and also check for the possibility of the old ones being returned. Automated testing also helps companies save time and money. Tests that would take hours to complete manually can be done in a matter of a few minutes. Because tests are automated, there is no need for extensive test reports, which saves money and manpower. Another powerful benefit of using automated tests is cutting down mistakes. Humans are prone to making errors, states Bosas in his article [5]. Considering the fact that the tests have to be run after every change in the software, it is fair to assume that mistakes will be made. Automated testing practices minimize the chance of a human mistake during continuous testing, concludes Bosas [5].

2.3 Visual GUI Testing

This subchapter focuses on the origin of visual GUI testing and explains why it was created. This chapter is based on Emil Alegroth's, Robert Feldt's and Lisa Ryrholm's article [6]. Alegroth, Feldt and Ryrholm establish that GUI-intensive systems require high-level tests. In today's software development industry, test automation techniques approach testing from a low-level abstraction. To battle this problem, a considerable body of work has been devoted to high-level test automation, resulting in such techniques as Record and Replay.

A tool that uses this technique, records the coordinates of GUI-components, with which the tester manually interacts, explain Alegroth, Feldt and Ryrholm [6]. Then, the recording of the test can be played again to simulate user's interaction with the system. Alegroth, Feldt and Ryrholm conclude in their work [6] that the aforementioned technique provides a much better solution than manual testing because the test needs to be done manually only once. Later tests are repeated automatically, which saves money and manpower, they say. Alegroth, Feldt and Ryrholm also found downsides to Record and Play technique in their work [6]. In their words, there are major flaws to this technique, which raise the costs of maintaining the tests and make this technique non-applicable in the software industry. Techniques, such as Record and Replay, are typically sensitive to GUI layout, which means that it is either problematic or nearly impossible to run the tests consistently on different computers, with different screens, etc, find the authors.

The aforementioned problem is solved by Visual Graphical User Interface Testing (VGT), find Alegroth, Feldt and Ryrholm [6]. Visual GUI testing is a technique that uses image recognition to interact with what is shown on the screen, explain the authors of the article. In the process of VGT, the tester creates a script with valid instructions. In their article Alegroth, Feldt and Ryrholm [6] explain how exactly VGT works: "GUI interaction, that was previously done by

specifying the exact coordinates of the pixels of the element, are now done by taking a screenshot of GUI element in question". With VGT there is no need for the tester to worry about GUI elements being in slightly different coordinates because of the screen or machine changes. Alegroth, Feldt and Ryrholm [6] explain the technology behind VGT further: "VGT tools interact with the system's bitmap level, e.g. computer's screen and look for matches of these screenshots with what the system currently shows on the screen." The power of image recognition can be used not only for interaction with the GUI elements but also for comparing tested visual output to the expected output. This makes VGT not only more efficient and reliable compared to Record and Play, but also more practical.

2.4 Sikuli

Sikuli is an open-source VGT tool developed by MIT researchers. In their work Jin-lei Sun, Shi-wen Zhang, Song Huang, Zhan-wei Hui [7] describe Sikuli as a tool that identifies and interacts with elements of GUI through the principle of image matching. By their words, Sikuli consists of Jython-based API, which gives the tester an opportunity to write readable tests scripts, and the integrated development environment Sikuli IDE, which is a fully functioning IDE for creating visual GUI tests.

Based on Sikuli's official documentation [8], Sikuli can interact with anything the user sees on the screen of the computer running Windows, Mac or some Linux/Unix. Practical uses of Sikuli are for example:

- Sikuli can be used to automate Flash Objects / Flash Websites. Sikuli provides extensive support for this task.
- It can be useful to automate Window based applications.
- It provides a simple API. i.e. all methods can be accessed using screen class object.
- It can be easily integrated with Selenium and other test automation tools.
- Sikuli can be used for the automation of desktop applications.

Sikuli supports the usage of various functions to make visual GUI testing possible and provides testers with simple IDE for comfortable test-writing.

2.5 Previous Sikuli Lab

In 2015, Rasmus Sõõru created an "Automated GUI Regression Testing" lab package for the "Software Testing" course. Sõõru defended a thesis on this topic. In his work [9], Sõõru focused on the regression testing, however, the tool used for testing was Sikuli. Some visual GUI testing aspects can be found in his work, however, in his lab, Sikuli API is used which took a significant amount of time from the lab to set up the project. Regression testing techniques

were also had to be discussed, hence the practical tasks using Sikuli were left simplistic and easy to understand.

The author of the "Visual GUI Testing" technique decided to create a new lab that used Sikuli and focus solely on the visual GUI testing part. It was decided to use Sikuli IDE which works straight out of the box instead of Sikuli API that needs to be set up properly. This greatly decreased installation and preparation time from the lab, which allowed the author to explore visual GUI testing techniques and concepts, as well as Sikuli's functionality, in greater depths.

In Sõõru's lab [9] software under test was an application with different tabs that all represented different functionality. Author of the "Visual GUI Testing" lab decided to adopt this strategy and created 2 different applications with 5 tabs each: one application for the lab and one application for the homework tasks.

3 Lab Design

The following chapter provides an overview of the structure of the lab and the materials provided for the students and lab assistants.

3.1 Lab Schedule

The amount of hours expected to be spent on each lab is approximately 8 academic hours (360 min) – 2 hours (90 min) for the lab itself and 6 hours (270 min) for the homework task. Lab schedule is as follows:

- The introduction to visual GUI testing and Sikuli tool to be used in the session should take approximately 15 minutes.
- The 6 provided lab tasks should take approximately 50 minutes. Each solution will be explained by the TA which means that completion of the tasks does not depend on students' experience.
- The introduction of the homework assignment and further questions takes approximately 25 minutes.

Homework assignments could be solved alone or in pairs if both students are from the same lab group.

3.2 Lab Materials

Lab materials created for this lab can be divided into two groups: lab materials for students and lab materials for teacher assistants. This chapter provides an overview of the aforementioned materials and lab and homework tasks.

3.2.1 Student Materials

The following lab materials are provided for the students:

- Lab Instructions (student version) - a PDF documentation file containing everything required for the successful lab completion - introduction to visual GUI testing, Sikuli IDE installation instructions, explanation of the usage of helperClass.sikuli and practical part
- Specification document for the application used in the lab session
- Specification document for the application used for the homework tasks
- Guide for Sikuli - compressed version of Sikuli's official documentation
- helperClass.sikuli - component provided by the author of the lab to make test writing and execution simpler for the students
- tests.sikuli - skeleton code of the tests' file

- Lab.jar - an application that is used as software under test in the lab
- Homework.jar - an application that is used as software under test in the homework tasks

Links to student materials can be found in Appendix I.

3.2.2 TA Materials

All the student materials have been also provided to teacher assistants. The most important materials for teacher assistants in the student materials list are:

- Specification document for the application used in the lab session
- Specification document for the application used for the homework tasks
- Lab.jar
- Homework.jar

Materials mentioned are crucial for the evaluation of students' homework tasks and grading. Author of the "Visual GUI Testing" lab has also provided restricted access materials that can only be accessed by teacher assistants:

- Lab Instructions (TA version) - lab instructions document with comments for the teacher assistants that help better teach and understand the lab
- lab_solutions.sikuli - solutions examples for the lab tasks
- hw_solutions.sikuli - solutions examples for the homework tasks that are used as the base for grading

Links to teacher assistant materials can be found in Appendix I.

3.2.3 Lab Application

Lab.jar is a simple Java application that is used in the lab as software under test. Lab.jar was created with JavaFX and Maven. Lab.jar contains basic GUI elements and is almost entirely stripped of functionality. The simplicity of Lab.jar is intentional - the author of the "Visual GUI Testing" lab used as little functionality as possible to shift the focus of testing from the traditional functional testing to the visual GUI testing. Lab.jar was created to allow students to dive into the world of visual GUI testing with ease, hence, the GUI elements in the application used for the lab session are basic GUI elements. Lab.jar consists of 5 tabs. The tabs have no meaningful connection - each tab is a separate application. Many different applications in one allowed the author to create tasks that explored more visual GUI concepts without the need for writing a long convoluted test for each task. The descriptions of the aforementioned tabs are as follows:

- Buttons - a pane with 3 differently colored buttons. Pressing each button display some text (see Figure 1)
- Editor - expandable pane that includes a text editor (see Figure 2)
- Copiable - a pane with 5 words. Each word represents a text block that should be copiable (see Figure 3)

- Folder - imitation of macOS desktop which consists of one folder and 4 “apps” (see Figure 4)
- Resizer - simulation of different screen sizes (see Figure 5)

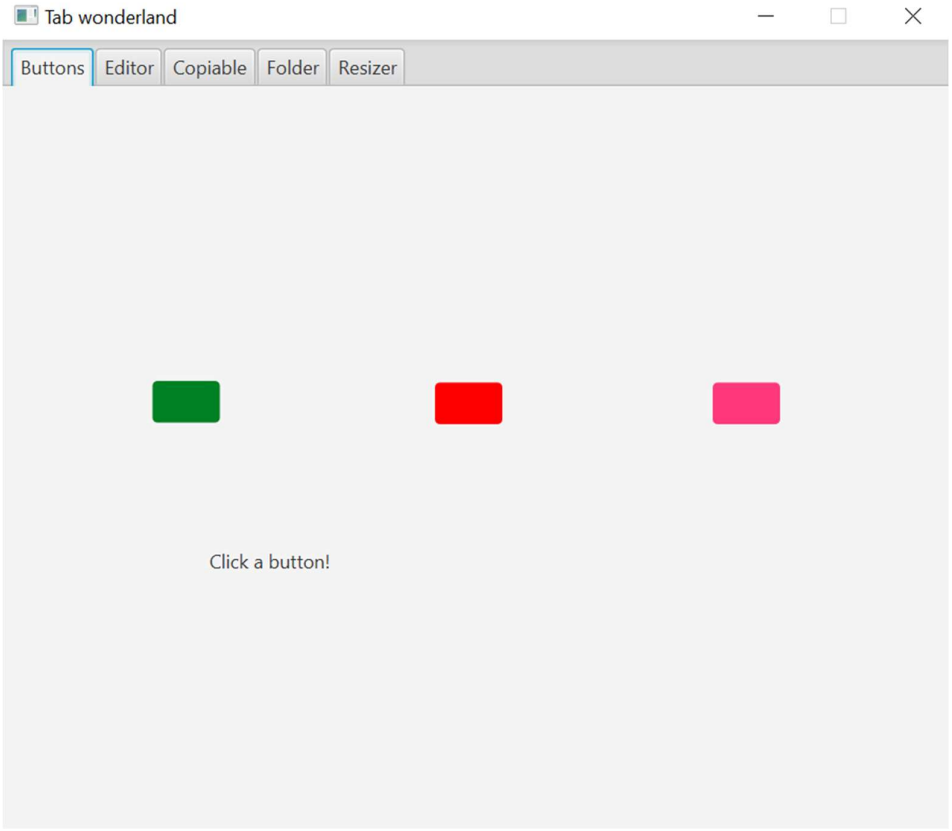


Figure 1. Lab application: "Buttons" tab

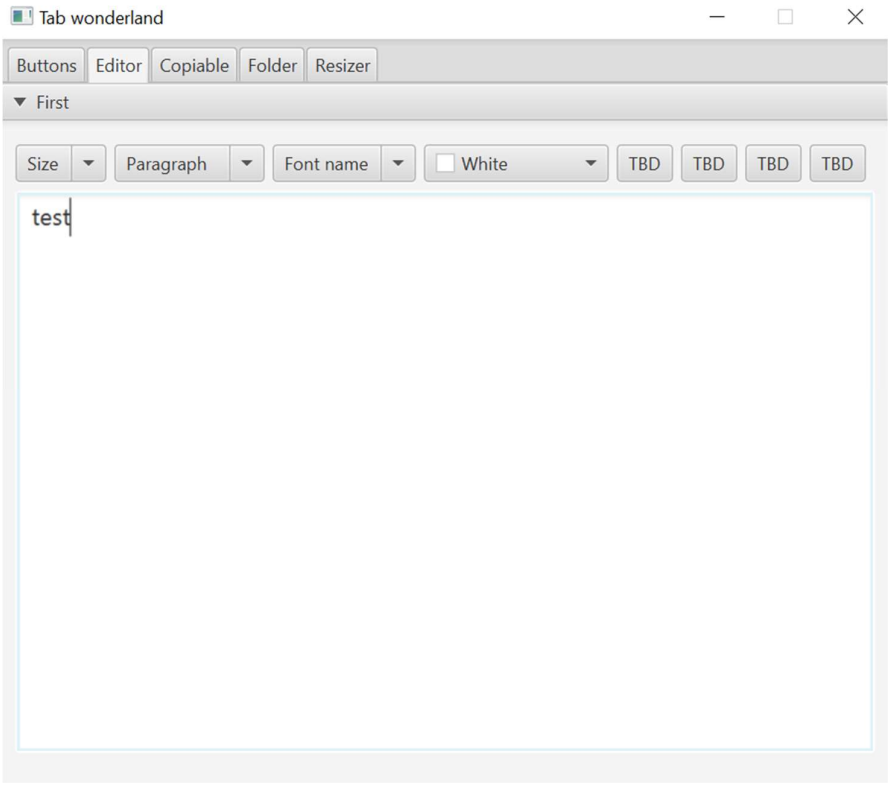


Figure 2. Lab application: "Editor" tab

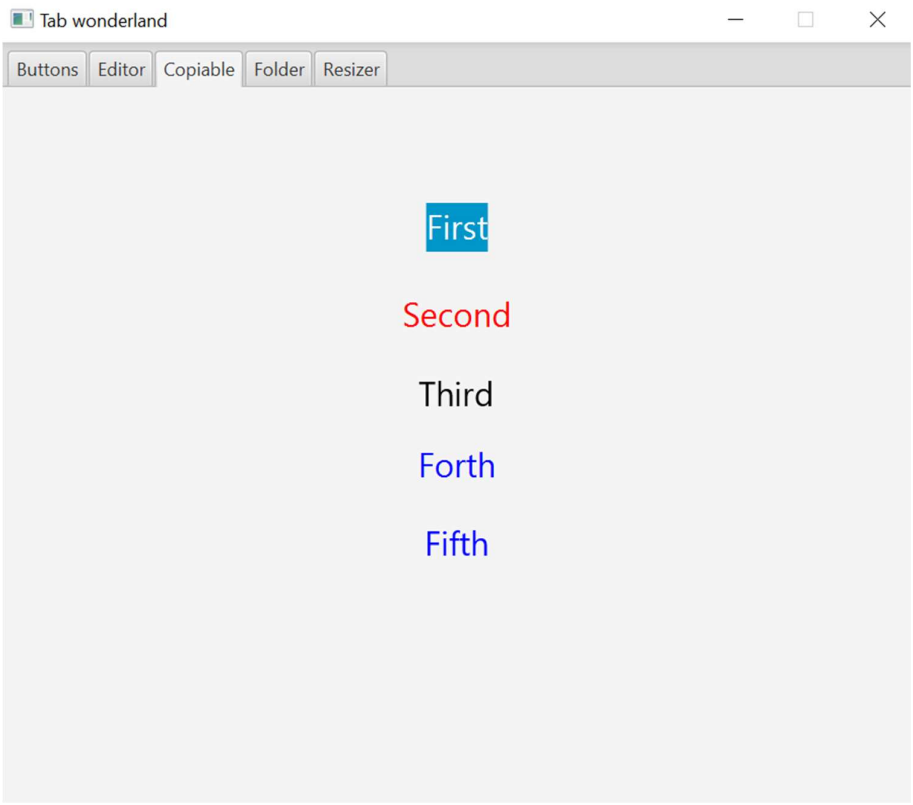


Figure 3. Lab application: "Copiable" tab

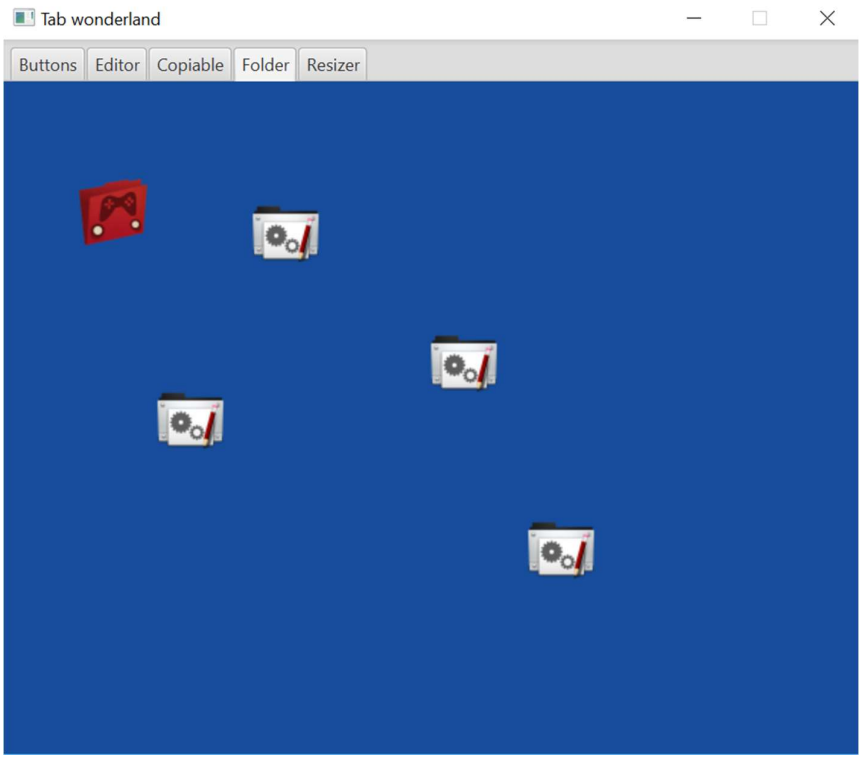


Figure 4. Lab application: "Folder" tab

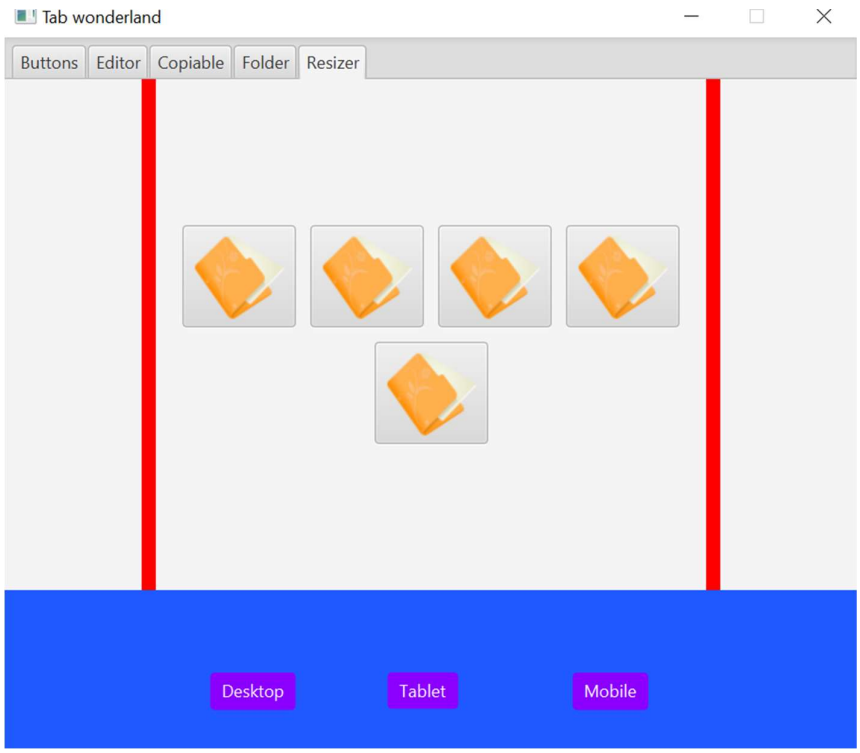


Figure 5. Lab application: "Resizer" tab

The choices of various GUI elements and the meaning of each tab will be explained in the "Tasks" subchapter alongside with the lab tasks.

3.2.4 Homework Application

Homework.jar is a similar application to Lab.jar. It was also created using JavaFX and Maven. Homework.jar also has 5 different tabs, which are meant to represent different applications. The difference between Lab.jar and Homework.jar is that while Lab.jar consists of the most basic GUI elements and is not meant to have applications that can represent useful applications, Homework.jar's tabs include applications that are closer to real-world applications. The tabs of Homework.jar consist of:

- Folders - improvement of "Folder" tab: instead of one folder has 3 folders with different max capacities (see Figure 6)
- Calculator - a calculator that evaluates the expression after each press of the button (see Figure 7)
- Pin-code - imitation of smartphone's lock-screen (see Figure 8)
- Converter - Google Translate style Unicode converter (see Figure 9)
- CatFlower - simple game with cats and flowers (see Figure 10)

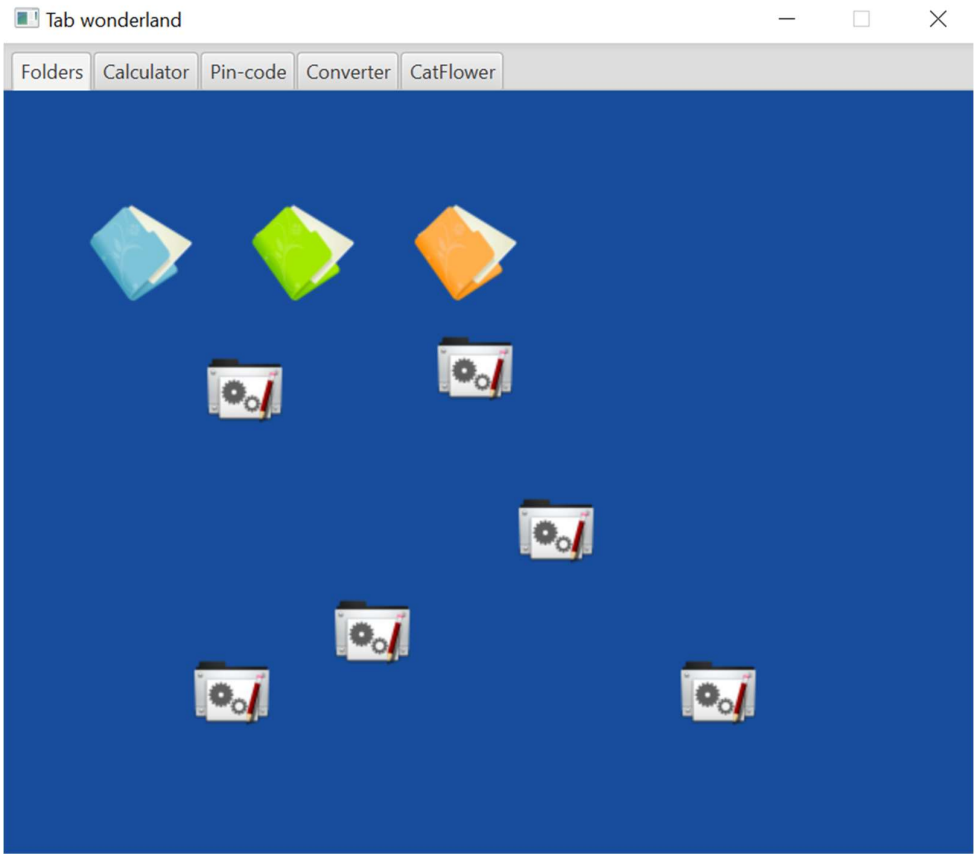


Figure 6. Homework application: "Folders" tab

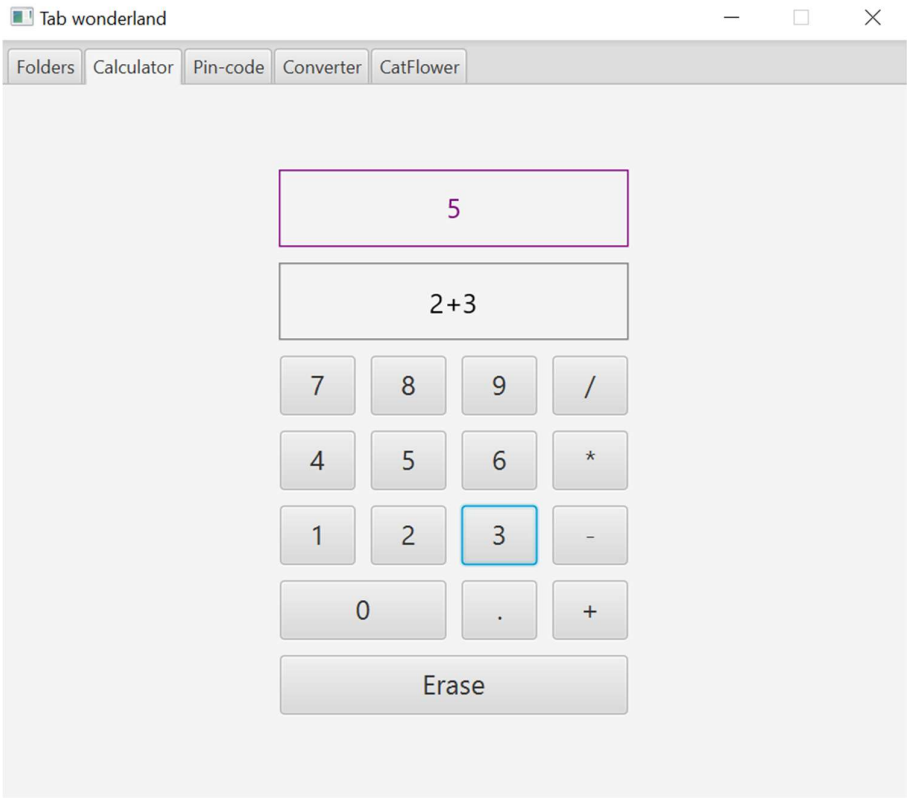


Figure 7. Homework application: "Calculator" tab

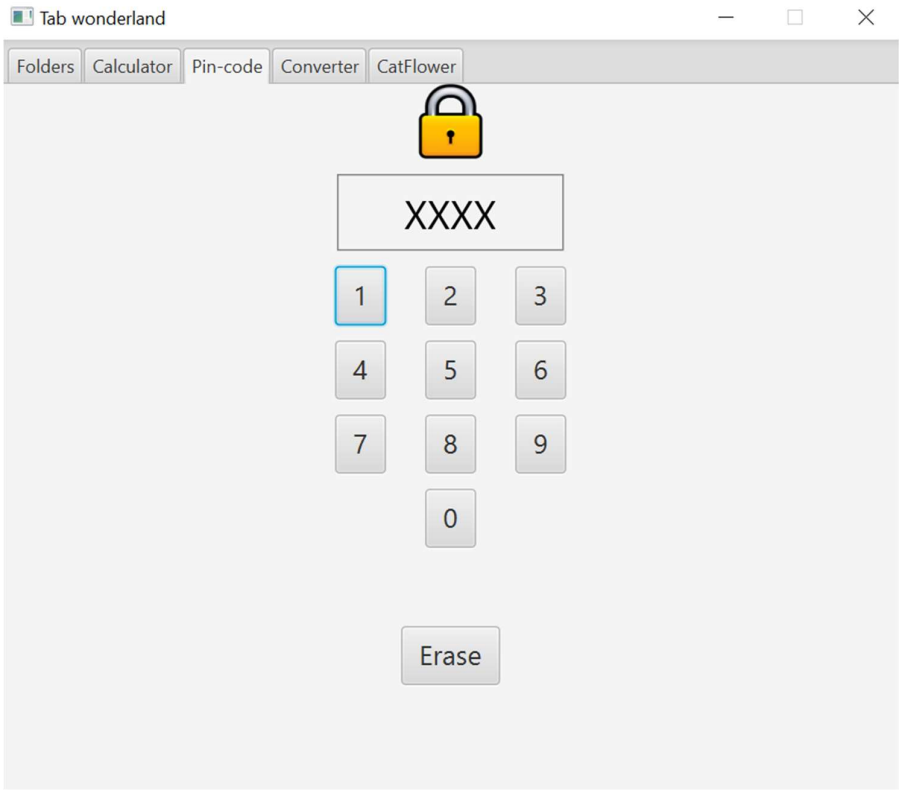


Figure 8. Homework application: "Pin-code" tab

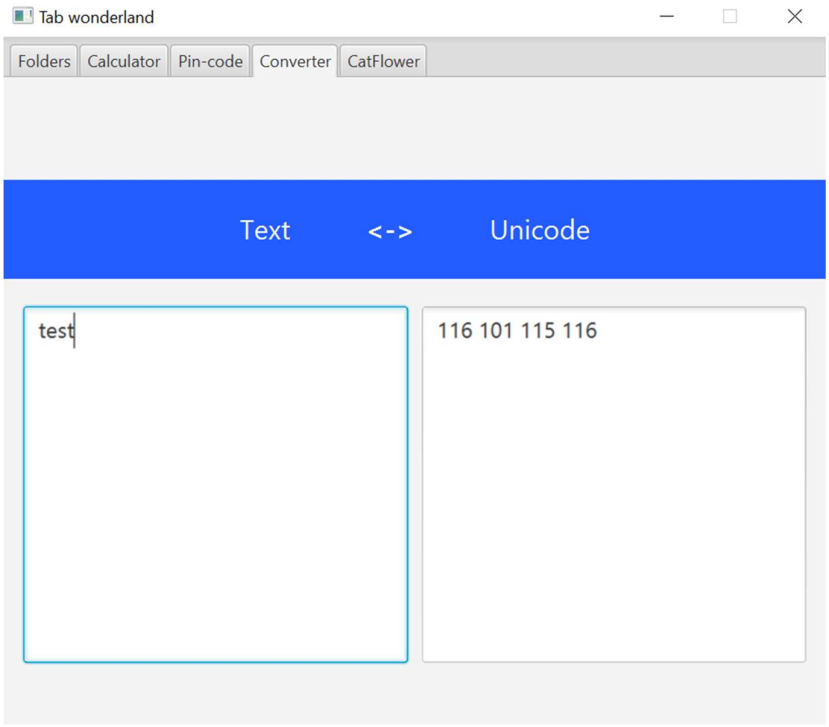


Figure 9. Homework application: "Converter" tab

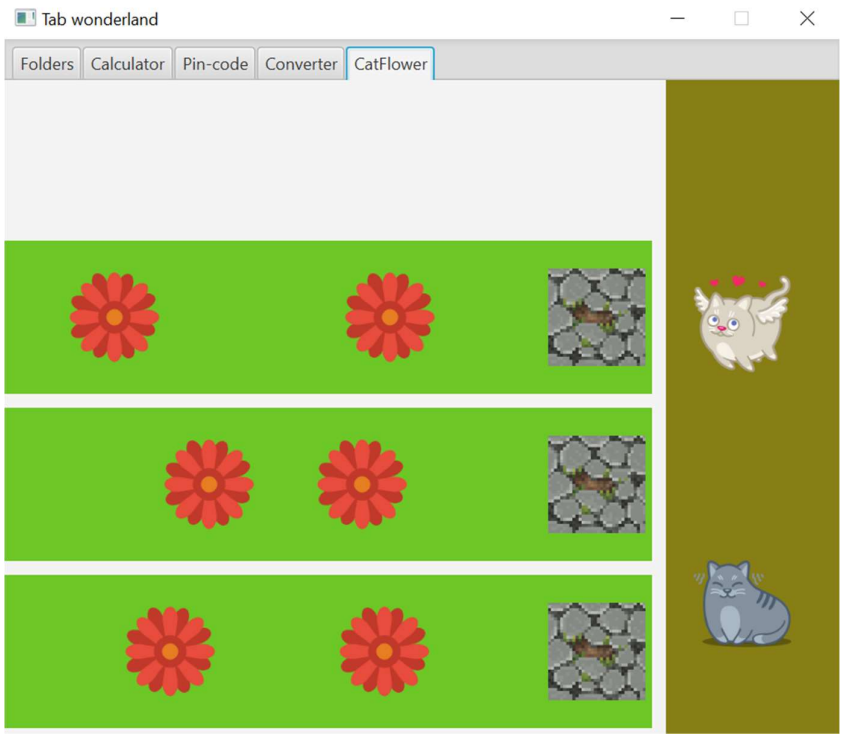


Figure 10. Homework application: "CatFlower" tab

For confidentiality reasons the choices of the GUI elements and the meaning behind each tab will not be discussed in this thesis but are explained in the materials for teachers.

3.3 Tasks

For this lab, both lab and homework tasks were created. Lab tasks focus on different aspects of visual GUI testing - from introduction to the topic to the testing itself. Students who participated in the lab should have a setup environment, correct project structure, and skeleton code of the tests, hence the homework tasks focus solely on the testing part of visual GUI testing.

3.3.1 Lab Session Tasks

The purpose of the lab session tasks is as follows:

- Let students become familiar with Sikuli tool
- Introduce techniques, possibilities, and challenges of visual GUI testing and apply them on the Lab application
- A brief overview of the homework assignment

The lab should start with the introduction of visual GUI testing by the lab supervisor. After that, the lab supervisor shows the installation process of the Sikuli IDE and guides students through possible problems.

When the students have properly set up the Sikuli IDE, the lab supervisor proceeds with the "preparation" and "project setup" parts of the lab instructions. During the preparation part, the students have to download Lab application, helperClass.sikuli, and tests.sikuli. The lab supervisor makes sure that students have downloaded everything in the right place. The project setup part focuses on the explanation of helperClass.sikuli's functionality and its usage, as well as unittest setup. Sikuli allows its users to write regular code, however, for the purpose of writing structural tests that can be easily understood by both students and teacher assistants, python's unittest was chosen as the testing framework.

Afterward, the lab supervisor starts with the practical part of the lab session. The lab supervisor shows the lab application and explains its functionality. After that, the lab supervisor goes through the specification file of the lab application. The lab supervisor explains that the specification covers each tab and that the students have to test the lab application against each bullet point in the specification. The specification file for the lab application has 6 bullet points to test in total. The first test is designed to be understandable for first-time Sikuli users. With each next test, the difficulty gradually increases, and new small visual GUI testing concepts and Sikuli's functionality are introduced.

The first task, "test_buttons_text", introduces some of Sikuli's basic functionality. While showing how to create a test for the first task, the lab supervisor explains how to create screenshots, how to store them in variables, how to find and collect text on the screen, and how to use regions. Region is one of Sikuli's core concepts. It allows the tester to choose the part of the screen that needs to be tested in order to exclude all unwanted GUI elements. During this task, students learn how to create a region on their own, how to use Sikuli's functions only inside the created region, and how to use a region that is already defined in helperClass.

The second task, “test_buttons_visible”, focuses on another key feature of Sikuli: similarity. The human eye and computer see the screen differently. What may appear as entirely different for humans, such as red and pink color, may be interpreted as similar by the computer. The colors of the buttons in the “Buttons” tab were designed to show this aspect of visual GUI testing. By default, the similarity setting, which ranges from 0 to 1, is given a value of 0.7 by Sikuli. With this similarity setting Sikuli matches the red and pink button when only red one is searched for. This task teaches students how to properly use the similarity setting.

“test_editor” emphasizes that not everything that happens on the machine can actually be seen. In this task, students have to retrieve text from the editor and make sure that it stayed unchanged. The text appears to be the same, however, a space is added to it. This is counter-intuitive to the message of visual GUI testing, however, the author of the lab found this example important, because it helps the students better understand the limitations and workarounds of VGT.

Forth task, “test_copiable”, combines 2 text collection techniques.

In the next task, “test_folder”, students have to figure out to imitate dragging and work with a number of identical GUI elements.

The final test, “test_resizer”, expands on the concept of region creation. In this task, students have to create each region by calculating it from the positions of different GUI elements.

By this point, students should have practical knowledge of visual GUI testing tool Sikuli and should have learned various visual GUI testing techniques and concepts. Lab supervisor briefly shows the homework application, explains how to create a report for the tests and how the homework will be graded.

3.3.2 Homework Tasks

The purpose of the homework tasks is as follows:

- Cover 18 tests in the specification of homework application using Sikuli IDE
- Learn how to create a report using helperClasse’s functionality

Homework tasks expect students to use the concepts and functionality learned in the lab to write 18 tests for 5 different tabs. Each test can be written in many different ways, therefore the students are encouraged to read “Sikuli Guide” or turn to the official documentation which will provide a better overview of Sikuli’s functionality.

The reporting part of the task expects students to tailor their tests to run correctly more often. While students write the tests for each bullet point in the specification, they tend to run each test individually. This approach, however, may lead to the tests being unreliable and not always giving the correct output. While reporting, the students have to combine all 18 tests into one place and run them all at the same time.

4 Lab Execution

"Visual GUI Testing" lab was the 8th lab in the "Software Testing" course and took place on the 7th and 8th of April. Students registered to the course were divided into 5 groups. Each group participated in the lab with one lab supervisor. The author of the lab was present during each lab and helped with potential problems and questions.

This lab was different compared to the labs from previous years because of the COVID-19 situation. The lab took place online using BigBlueButton. During all lab sessions, lab supervisors started with the introduction to visual GUI testing and the installation of Sikuli. Some students had problems with installation because they did not follow the instructions properly, however, the issues were quickly resolved.

After the students were done with Sikuli's installation and prepared their environments, lab supervisors described the lab application and proceeded to start with lab session tasks. Some lab supervisors took the strategy of the author of the lab: completing all the tasks during the lab, some lab supervisors decided to give more in-depth explanations to the first tasks and did not have time to complete all the tasks. The second strategy turned out to be also viable because lab supervisors who focused only on the first tasks, explained how to write the tests in different ways and showed the concepts that could have been learned in later tasks.

During the practical part, some issues were encountered both by students and lab supervisors. One of the issues was confusion with the Java version. The lab application runs on Java version 11 and students and lab supervisors downloaded that version. However, Sikuli opened the application with the default Java version, which many of the students had as Java 8 at the time. This issue was quickly resolved and instruction on fixing the issue was given. Other small issues or confusions were also resolved during the lab.

The last 15 minutes of the lab was left for the homework introduction and questions. On average, lab sessions lasted about 90 minutes.

5 Feedback

The following sections include an overview of students' feedback, analysis of said feedback, and possible future improvements of the lab proposed by the author.

5.1 Feedback Collection

More than 100 students participated in the lab, 29 of whom took the Google Forms feedback. The students were given 7 statements, as shown in Figure 11, with answers being a numerical range from 0 to 5 (from "strongly disagree" to "strongly agree") and one optional free-form answer question about any comments or suggestions about the lab. The statements with a range answer are as follows:

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 in the software testing industry
7. Overall, the lab was interesting and inspiring

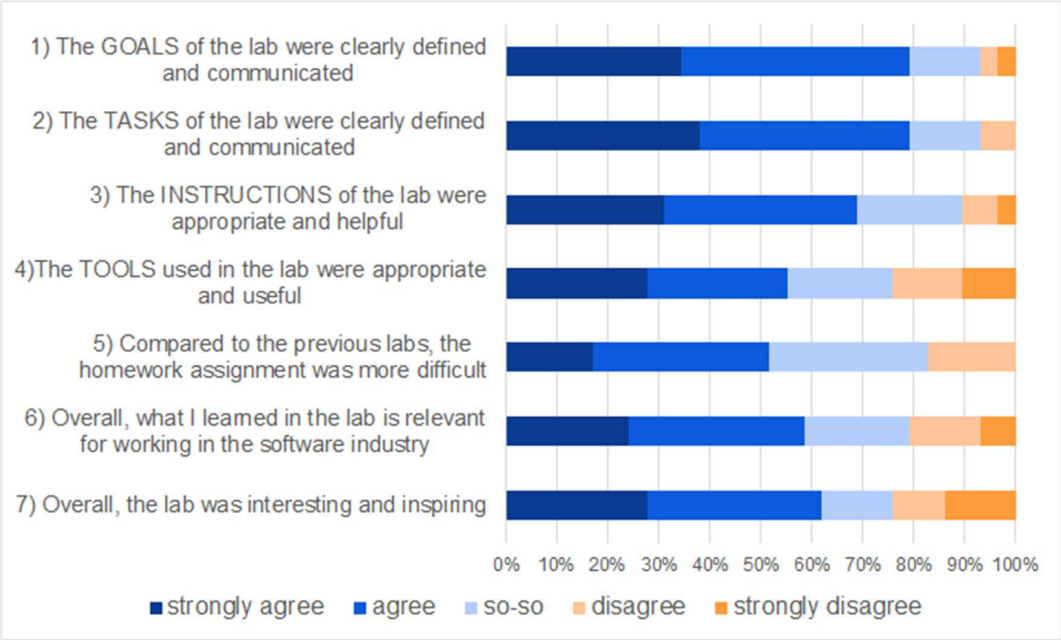


Figure 11. Feedback questions and results

The following subchapter focuses on the analysis of the collected feedback.

5.2 Analysis

Answers for the first question show that the goals of the lab were clearly defined and communicated. More than 75% of the students were positive regarding this question.

Similarly to the goals of the lab, the tasks of the lab were found to be clearly defined and communicated. As shown in the second question answers, none of the students expressed strong disagreement regarding this question and more than 75% of the students agreed or strongly agreed.

Students were mostly positive regarding the instructions of the lab, however, roughly 20% of the respondents answered “so-so”, which means that lab instructions may have some confusing parts or may not answer all the possible questions of the students. It is very important for the lab instructions to be as appropriate and helpful as possible.

The statement “The tools used in the lab session were appropriate and useful” was met with the second least positive feedback by the students. Roughly 55% of the students answered “strongly agree” or “agree” to this statement.

The statement “Compared to the previous labs, the homework assignment was more difficult” was met with neutrality/agreement by the students. The author of the lab intended to make the tasks of this lab more difficult compared to the previous labs because the “Visual GUI Testing” lab took place late in the course.

The sixth and seventh question was amongst the lowest positively answered questions amongst the students. Overall, students found visual GUI testing practices to be relevant in the industry, however, only around 55% of the respondents were positive and roughly 20% answered “so-so” regarding this question. Author of the lab agrees that visual GUI testing is not one of the industry's most popular practices, however, the author found those practices viable for the course because they teach orthodox methods and out of the box thinking that most popular testing practices may not provide.

The statement “Overall, the lab was interesting and inspiring” had mixed answers. More than 60% of the students found that the lab was interesting and inspiring, however, more than 20% of the students reacted with strong disagreement with this question. Judging by the comments, it seems that students who encountered problems with the tool, found the lab not interesting, however, those who had no problems with the tool enjoyed the lab.

In the “additional feedback” section some students expressed their joy over simplicity of the installation and setting up Sikuli IDE, while others expressed their disregard with Sikuli’s errors. Fortunately, all the problems discovered by students were solved either by themselves or with the help of the author and lab supervisors.

5.3 Future Improvements

Students' feedback and comments from lab supervisors were taken into account to create a list of possible future improvements for the lab. The list of future improvements contains N elements:

1. Improve Lab Instructions file
2. Look into other visual GUI testing tools
3. Tailor the tasks better to ensure that students don't get many errors

The author of the "Visual GUI Testing" lab intends to improve the lab based on point 1.

Other possible improvements may or may not be executed by the staff of the "Software Testing" course depending on the staff's decision.

6 Summary and Conclusions

The aim of this was to create a lab package containing separate lab instruction for students and for teacher assistants, guide for Sikuli, applications to be tested and specification files for those applications. The lab package was created for the Software Testing (LTAT.05.006) course at the University of Tartu and given to 121 students registered to the course in the 2020 spring semester.

The lab execution went differently for every lab supervisor, but overall no critical problems were found in the lab design. After the lab, 29 students provided feedback, based on which a list of possible future improvements was made. All things considered, the created lab materials met course requirements and can be used in the future for the Software Testing course.

References

[1] “ÕIS Course Description (2020),” [Online]. Available: https://ois2.ut.ee/#/courses/LTAT.05.006/version/74d709257c8fdbd1980eb7f2bd3c815a/details. (12.04.2020)

[2] “Course Outline (2020),” [Online]. Available: https://www.is.ut.ee/rwservlet?oa_aine_info.rdf+1228876+HTML+0+text/html. (12.04.2020)

[3] “Course Labs (2020),” [Online]. Available: https://courses.cs.ut.ee/2020/SWT2020/spring/Main/LabsPracticeSessions. (12.04.2020)

[4] Borjesson, Emil, Feldt, Robert, Automated System Testing Using Visual GUI Testing Tools: A Comparative Study in Industry. 2012 IEEE Conference in Montreal, QC, Canada https://ieeexplore.ieee.org/document/6200127

[5] Bosas, Joseph, Automated Testing Importance and Impact. 2018 IEEE Conference in National Harbor, MD, USA https://ieeexplore.ieee.org/document/8532522

[6] Alegroth, Emil, Feldt, Robert, Ryrholm, Lisa, Visual GUI testing in practice: challenges, problems and limitations. 2015 EMPIRICAL SOFTWARE ENGINEERING https://link.springer.com/article/10.1007/s10664-013-9293-5

[7] Sun, Jin-lei, Zhang, Shi-wen, Huang, Song, Hui, Zhan-wei, Design and Application of a Sikuli Based Capture-Replay Tool. 2018 IEEE Conference in Lisbon, Portugal https://ieeexplore.ieee.org/document/8431949

[8] Sikuli Doc Team, Sikuli X 1.0 documentation. 2012 http://doc.sikuli.org/genindex.html (14.03.2020)

[9] Sõõru, Rasmus, Automated GUI Regression Testing. 2015 https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=46461&year=2015 (23.04.2020)

Appendix

I. Lab Materials

Lab Materials for Students

- Lab Instructions "Visual GUI Testing" for students, PDF file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/SWT2020-lab08-2020v01.pdf
- Lab Application Specification, PDF file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/lab08-LabSpec.pdf
- Homework Application Specification, PDF file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/lab08-HomeworkSpec_v2.pdf
- helperClass.sikuli component, ZIP file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/lab08-helperClass.sikuli_v2.zip
- tests.sikuli skeleton code, ZIP file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/lab08-tests.sikuli_v2.zip
- Sikuli Guide, PDF file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/SikuliXGuide.pdf

Lab Materials for Lab Supervisors

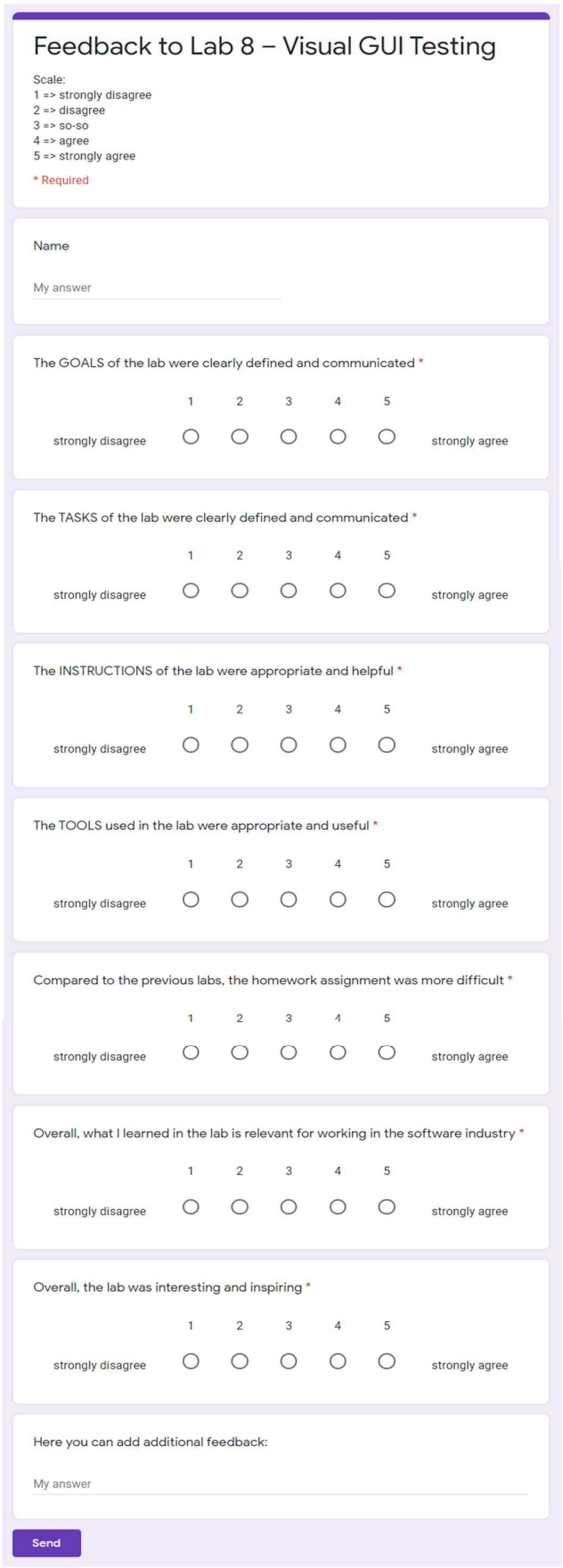
- Lab Instructions "Visual GUI Testing" for lab supervisors, PDF file
- Lab tasks solution examples, Sikuli code
- Homework tasks solution examples, Sikuli code

For confidentiality reasons lab supervisor materials are not made available in the thesis but will be made available on request.

Software Under Test Applications

- Lab Application, JAR file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/lab08-Lab.jar
- Homework Application, JAR file https://courses.cs.ut.ee/2020/SWT2020/spring/uploads/Main/lab08-Homework.jar

II. Feedback Questionnaire



Feedback to Lab 8 – Visual GUI Testing

Scale:
1 => strongly disagree
2 => disagree
3 => so-so
4 => agree
5 => strongly agree

* Required

Name

My answer

The GOALS of the lab were clearly defined and communicated *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

The TASKS of the lab were clearly defined and communicated *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

The INSTRUCTIONS of the lab were appropriate and helpful *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

The TOOLS used in the lab were appropriate and useful *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

Compared to the previous labs, the homework assignment was more difficult *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

Overall, what I learned in the lab is relevant for working in the software industry *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

Overall, the lab was interesting and inspiring *

	1	2	3	4	5	
strongly disagree	○	○	○	○	○	strongly agree

Here you can add additional feedback:

My answer

Send

III. License

Non-exclusive license to reproduce thesis and make thesis public

I, Aleks Kožajev,

1. herewith grant the University of Tartu a free permit (non-exclusive license) 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: Visual 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 license 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 license does not infringe on other persons' intellectual property rights or rights arising from the personal data protection legislation.

Aleks Kožajev

27/04/2020