UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Computer science Curriculum

Rasmus Sõõru

Lab Package: Automated GUI Regression Testing

Bachelor's Thesis (6 ECTS)

Supervisor: Dietmar Pfahl

Tartu 2015

Lab Package:

Automated GUI Regression Testing

Abstract:

Graphical user interfaces (GUI) are very complex and difficult to test. Testers must take under consideration many factors. The lab package created in the scope of this bachelor thesis gives practical experience to the students regarding the creation of GUI test cases and GUI regression testing.

Keywords:

Sikuli, Graphical user interface testing, Automated regression testing, lab package

Laboripakett:

Automatiseeritud graafilise kasutajaliidese regressiooni testimine

Lühikokkuvõte:

Graafilised kasutajaliidesed on väga keerulised ja neid on raske testida. Testijad peavad arvestama paljude teguritega. Laboripakett loodud selle bakalaureuse töö raames annab praktilisi kogemusi automatiseeritud kasutajaliidese testide loomises ja kasutajaliidese regresiooni testimisel.

Võtmesõnad:

Sikuli, Graafilise kasutajaliidese testimine, Automatiseeritud regressiooni testimine, laboripakett

Table of Contents

1 Introduction

Software *testing* is described as a process performed to provide system stakeholders with information about quality of the product or service under test [1]. In a world where products are constantly evolving and where new technologies and ever more complex structures are being implemented, software testers face their major challenge: there is never enough time in testing. So what could be done to help testers stay in the time limits of the project plan and give them more confidence in testing the product? One possible answer lies in automated regression testing.

The automation of regression testing (including Graphical User Interface testing) can improve the software development process and benefit it in many ways [2,3]. First of all, tests can run repeatedly in perspective of long maintenance life of a product. Test code can be changed flexibly within the scope of the same project or enable testers to reuse portions of the test code in other projects. In comparison to manual test execution, which can be time-consuming and prone to errors, well designed automated tests would guarantee accuracy and improve testing efficiency [2].

The lab package developed in the scope of this thesis has been used and evaluated in the course MTAT.03.159 (Software Testing). Since the majority of students enrolled in this course have no experience and sufficient knowledge about the frameworks used for automated testing, we decided to create a lab package that offers an opportunity to develop both practical and theoretical aspects of this subject. The lab package consists of a set of practical exercises, conducted in three iterations, where each is designed to produce a certain output (report). Students are guided through the process focussing on the demonstration of the benefits of GUI testing with the help of the test framework built on Sikuli (http://www.sikuli.org/), starting from the installation of Sikuli and arriving to the submission of test reports. We strongly believe that this lab will be for the students not only a useful demonstration of the Sikuli framework in action but also create an interesting experience in automated testing, emphasizing its benefits.

2 Automated Testing

In software testing manual approaches are not always the best solution to take when planning the testing resources. Manual testing can be time consuming or inefficient when dealing with certain groups of defects, like, for example, minor graphical user interface defects, which are usually found during regression testing. This is the case when automated testing would be helpful. Automated testing is a process of checking the software product by another independent application, including the following steps [4]:

- **Test planning** - The process of estimating and definition of the testing activities that should be carried out within the specific project
- **Test design** - The phase of creating and writing the test
- **Test implementation (launching, initialization)** - The process to define the particular order the test cases should be launched in
- **Test execution** - The process of running the test script, by executing pre-designed test suits
- **Test evaluation (analysis, reporting)** - Verification and reporting of the test results, by comparing the actual and expected result, defined in specification

Although it can be cost-effective and require additional effort, it offers a good alternative for software projects that are planned to be maintained continuously [5]. The common approaches for automated testing are for example code-driven testing, used in Test Driven Development (TDD) [6], Application Programming Interface (API) driven testing [7] and Graphical User Interface (GUI) testing.

2.1 Graphical User Interface (GUI) Testing

However testing the functional side of the system is only one part of effort requiring process. When dealing with products that have a graphical appearance, or a Graphical User Interface, special type of testing should be applied. The purpose of GUI testing is to ensure that graphical interface of the product, system or application meets its

specifications. In that case attention should be directed to the artefacts like layouts, fonts, font sizes, contents, text formatting, labels, buttons, lists and icons [8].

Unfortunately, GUI testing requires a lot of programming effort and is time consuming no matter if it is done manually or automatically. Unlike command line interface testing, GUI has many more variations to test and it should verify functionality along with the user side behaviour: the number of combinations to test grows exceedingly with the complexity of the system. Another characteristic that turns into a problem in GUI testing is sequencing. A sequence of unique GUI events has to be performed before a concrete GUI element can be inspected. The longer the sequence is the more challenging the manual test performing becomes [9].

2.2 Regression Testing

Regression testing is a software testing method that reveals new software bugs in existing functional or non-functional areas of the program or a system after changes have been done to the software. Regression testing aims to ensure that new changes have not created new faults. Even after the smallest changes in the software code can produce unexpected and undesired behaviour in the software [10].

Regression testing is a natural part of software development and very cost-effective [11]. Before releasing a new software version, old test cases are checked against the new version to ensure the software behaves correctly. Modifying software in any way always produces a risk that the software behaves unexpectedly.

2.3 Sikuli

Sikuli is an open-source project developed by Sikuli Lab at the University of Colorado in Boulder [12]. Image recognition mechanisms allow Sikuli to identify different GUI components and enables automation of any action you can see or perform on the screen (Figure 1). The basis of the project is Sikuli script which represents a visual scripting API for Jython [13]. Based on image recognizing patterns, Sikuli enables to send keyboard and mouse events that are usually performed by a tester.

The core of Sikuli is a Java library that

i.) delivers above mentioned events to required locations (java.awt.Robot)

ii.) searches for given image patterns (performed by C++ engine). Java library is equipped with so called Jython layer, which contains a set of commands for end-users. The Python source file (.py), Sikuli executable script (.skl) and all the images used by Sikuli are held in the .sikuli (Sikuli Script) directory. So there is a possiblity to change the Python source file in case of any need with the help of simple editor.

In comparison to other tools that use image recognition, Sikuli is a flexible freeware tool with rather simple Java API. It has an extensive guide and is actively maintained. It also enables you to create your own custom framework based on Sikuli. According to [14] Sikuli its image recognition system provides an easy way to test Flash sites and manage images to direct tests.

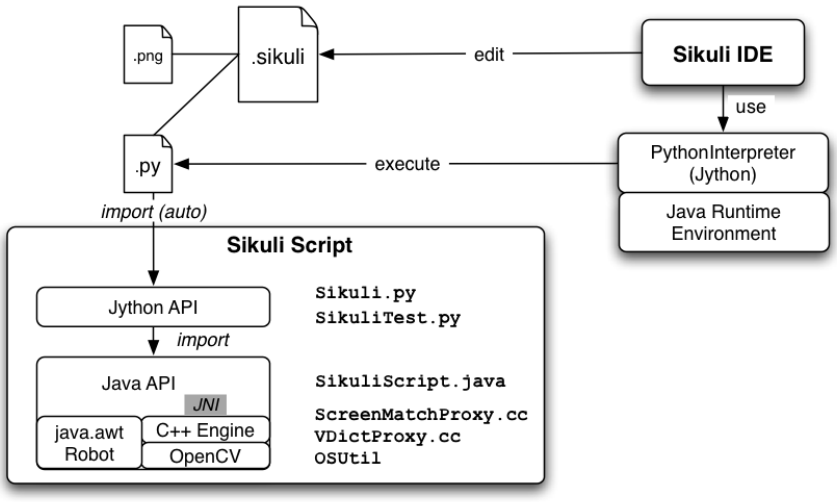


Figure 1 How Sikuli works (adapted from [11])

There are some other tools that use image recognition, such as for example:

- RoutineBot (http://www.routinebot.com/) - visual test automation software based on image patterns: allows manual script design as well as automatic script recording.
- Ranorex (http://www.ranorex.com/) - test automation tool for GUI objects recognition that supports dynamically built user interfaces.

3 Lab Description

The main principles the following Lab is built on are inherited from Standford's Laboratory Teaching Guidelines [15]. The Lab introduces the major concept of automated GUI testing, gives the guidelines of how to use Sikuli and provides the practical experiences in building an automated test using framework based on Sikuli java API.

The Lab simulates the automating of a very first program created by the company called “TabMaker”. In the first part of the Lab (*Iteration 1*) students are asked to create automated test flows that cover each requirement point from the specification document for the software *Version 1*. *Iteration 2* is directed onto discovering the new bugs based on the test cases created in *Iteration 1*. During the *Iteration 3* students have a chance to add a new test case and modify the old one. More details about the flow of the lab can be found in Chapter 3.3, named “Lab Workflow”.

3.1 Learning Goal

The higher-level goals for driving students through the Lab are to provide specific knowledge in the field of automated GUI testing and to introduce practical experience with Sikuli. This is achieved by pursuing the following more specific sub-goals (SGs):

SG-1: To provide students with theoretical knowledge through observation of the new concepts and definitions:

- The Lab is supported with theoretical materials, presented during the lectures of the Software Testing course (MTAT.03.159). The students receive necessary informational background regarding the major testing concepts and procedures before the Lab. On the course web page there are also available a “Course Readings" section, which is highly recommended for familiarization, and "Additional books on testing" for any special interests.

SG-2: To lead students through a step-by-step designed learning experience

- The work flow was designed as a step-by-step process of familiarization with the topic and its assets. Students research starts with understanding of more general concepts such as Regression testing and Automated GUI testing and then comes to introduce the details of how the Sikuli works. The instructions document contains information about every method, that the framework supported (some with examples how to invoke the methods).

SG-3: To supply students with required guidelines and equipment

- Students were provided with a detailed installation guide for Sikuli. The framework also included the demonstration of Sikuli capabilities through an example.

The lab also introduced some of the basic GUI Testing activities (GTAs) that should be part of the development of the test cases:

GTA-1: To demonstrate that every element of the GUI should be considered an object.

- Following the guidance of the Lab Package, students were asked to take screenshots and save them to the *Resources* folder, located in the project. Every picture saved in that folder can be called out multiple times (it's not necessary to take a new screenshot) by the unique name.

GTA-2: To demonstrate that reusability of the scripts can be beneficial

- Framework was built basing on the idea that some or all sections of the scripts can be executed by being called out from the other script. Like for example, the test class of the framework, which called out *SikuliSteps* object methods.

GTA-3: To explain that testing activity should provide coverage analysis

- Students had to decide, which result to append to the final report. Main idea of reporting was to raise awareness about actually giving back the results in a readable form.

GTA-4: To show that framework should be portable

- One of the java main principles is "Write once, run anywhere" (WORA), which means that the framework, built on java, is easily portable to multiple platforms.

GTA-5: To show that separation of the project artifacts can be essential

- The script simulating the user's actions and the screenshots for verification results of automated tests are located in different places in purposes to support separation. The principle of separation is used because we need to keep scripts independent from the changes of the environment and the verification data (GUI design) can change.

3.2 Materials Used In the Lab

The lab has 2 sets of materials. One batch was given to the students and the other one to the teacher assistants. The following bullet list distinguishes between those two:

- Materials for students
    - Lab - intructions.pdf
    - Lab – installation.pdf
    - Specification 1.pdf
    - Specification 2.pdf

    - Autotest for students.zip
    - Software1.jar
    - Software2.jar

- Materials for teacher's assistant's
    - Grading guideline.docx
    - Autotest for TA zip.

More detailed description and location are available in section 3.2.1 and 3.2.2, respectively.

Defects that are built into the example software are based on author's previous testing experience. The idea of these defects is that they are very easily automatable. Automating these test cases should be a stepping stone of how automated tests can help a tester who may not be so familiar with programming. There are 5 defects total that were created by the author - listed in section 3.2.1.

3.2.1 Materials for Students

Here are listed materials that the students used in the lab. Links to the actual locations from where the materials can be retrieved are listed in Appendix A.

- Lab - intructions.pdf

  Describes how the lab takes places, goes over definitions of Sikuli, Automated GUI testing.

- Lab – installation.pdf

  Describes how the install Eclipse IDE with Maven and how to enable Sikuli on the operating system.

- Specification 1.pdf

  Describes the requirements for software1.jar.
  There are 8 requirements in the specification for software *Version 1*.

- Specification 2.pdf

  Describes the requirements for software1.jar.
  There are 9 requirements in the specification for software *Version 2*.

- Autotest for students.zip

  Framework created in Java based on Sikuli java API that has all the necessary methods for this lab.

- Software1.jar

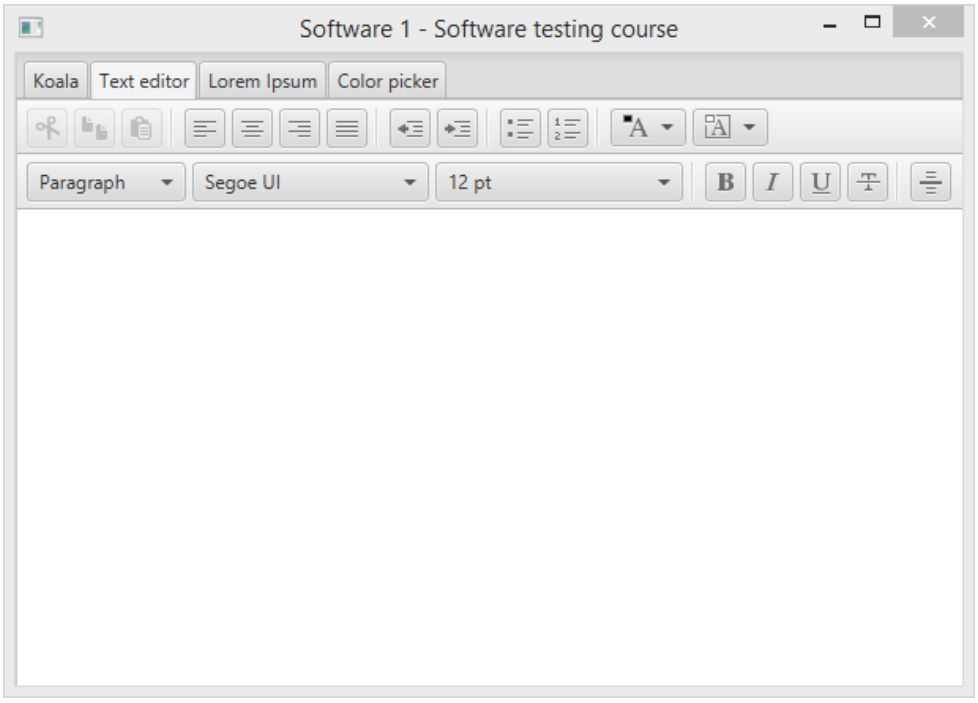


Figure 2 Software1.jar (*Version 1*)

Sample program written in javaFX is presented in Figure 2. No bugs have manually been introduced to *Version 1*. All tests (requirements for software 1 from Specification1.pdf) are expected to pass in *Iteration 1*.

- Software2.jar

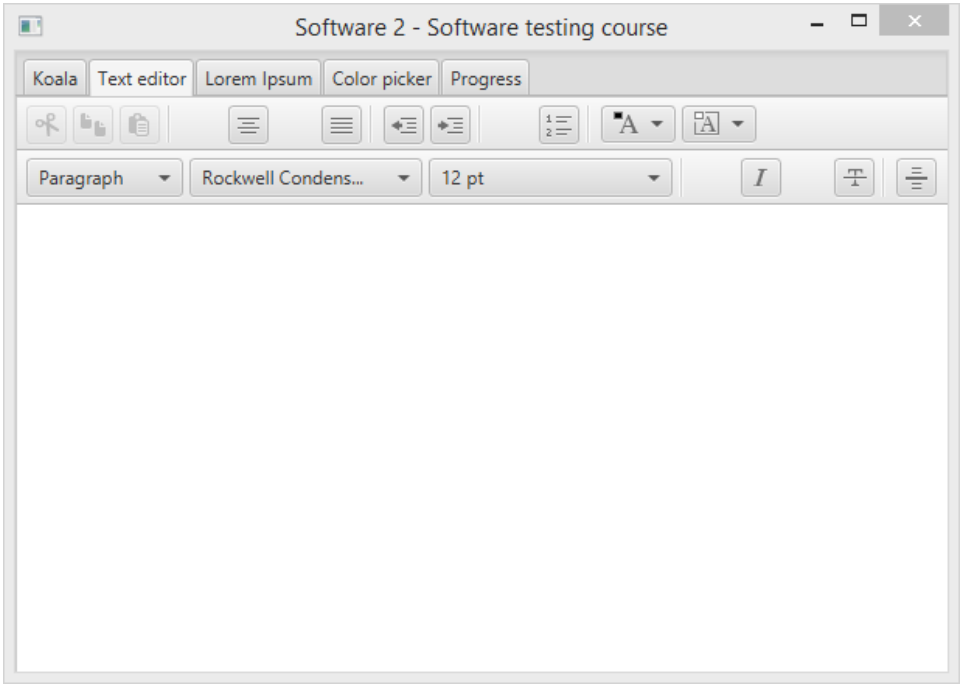


Figure 3 Software2.jar (*Version 2*)

Sample program written in javaFX is presented in Figure 3. *Version 2* introduces:

- 4 defects (bugs, that are findable via automation)
- 1 new specification
- 1 specification change

5 out of 9 test flows (requirements for the software 2 from Specification2.pdf) are expected to pass.

There are 5 defects built into the software2.jar:

- **Koala tab:**
    1) Koala must be exactly the same when returning to the tab – every time tab is visited, the picture is mirrored.
- **Text editor tab:**
    1) Typed text must remain in the same format when returning to the tab – when entering text and switching tabs, the text is put to *Italic font* and the number "1" is added.
    2) Selected text formatting settings must stay the same (Segeo UI) when returning to the tab – The first element is removed from the font list.
- **Lorem Ipsum tab:**
    1) Text length must stay the same when returning to the tab – every time tab is switched, a space (" ") is added to the end of the text.
- **Color picker tab:**
    1) Selected color code must stay the same when returning to the tab (Default: white) – color code is switched every time tab is opened.

Requirement changes/updates in software2.jar based on Specificaton 2:

- 1 new tab called **Progress**
  Loader that is in infinite state (going back and forth)
- HTMLEditor had some elements removed

3.2.2 Teaching Assistant Materials

Here are listed materials that the teaching assistants had available for use in the lab. Links to the actual locations from where the materials can be retrieved are listed in Appendix A.

- Grading guideline.docx
  Describes how the lab should be graded.

- Autotest for TA zip.
  Java project with full solution that is ran against *Version 1* and *Version 2*.

3.3 Lab Workflow

The lab was divided into 3 iterations. The first iteration (*Iteration 1*) represented automating the software for the very first time or in other words - establishing the base-line. This was the most important part in the lab. The test cases had to be automated properly, because they would be used in *both, Iteration 2* and *Iteration 3*. Students were given *Specification 1*, which contained 8 specification points to automate. The expected results (as shown below on Figure 3) for the *Report 1* were also predefined for the students to avoid covering too much or too little (since this lab is just for educational purposes)

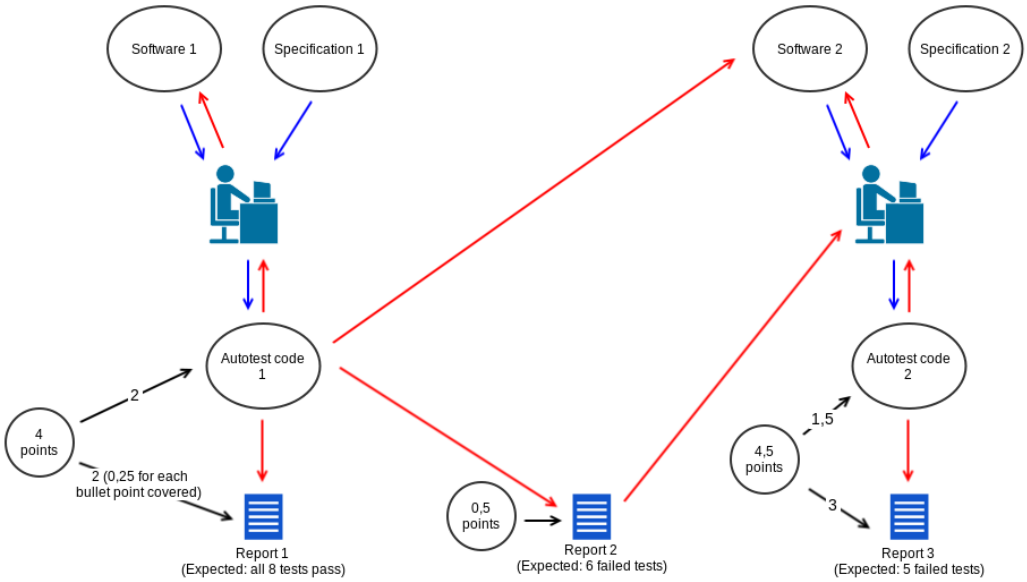


Figure 4 Lab workflow

Second iteration (*Iteration 2*) purpose is display the benefits of automated regression GUI tests. During the *Iteration 2* students are supposed to find all the defects introduced to the software. This simulates real world application on a smaller scale – if a new software version appears, then obviously application has changed or it would not be released. If the students automated the previous test cases in *Iteration 1* properly, then this

run would let them see all the mistakes (and one specification change) in the *Report 2*. **NB!** This run would not consider the new functionality or change of an old one.

In *Iteration 3*, the specification for *Software 2* has changed and this means that the autotest must be updated where necessary. Software version 2 introduces 6 defects. One of the defects is due to the specification change that the students must discover when comparing two specifications side-by-side. After the change is discovered, the autotest must be edited and the new requirement must be taken into account to pass the test successfully. Since the bug count is known, students rely on that to finish the assignment.

4 Feedback

Feedback about the lab was collected from the students, who participated in the lab. In total there were 88 participants in the lab. 86 of those students decided to answer the online questionnaire (1 student decided to skip answering to the questionnaire and 1 filled the questionnaire twice – invalid data has been removed from the graphs below). Students were asked to give the answers to 9 questions and give additional notes or remarks about the lab in the comment box of the questionnaire, which was volunteer. Before the lab was conducted, feedback from teaching assistants was gathered informally during meetings and Skype.

4.1 How the feedback was collected

The feedback was collected through a SurveyMonkey[16] online questionnaire. Students, who decided to answer to the online questionnaire were given 1 bonus point in the context of the Software Testing course.

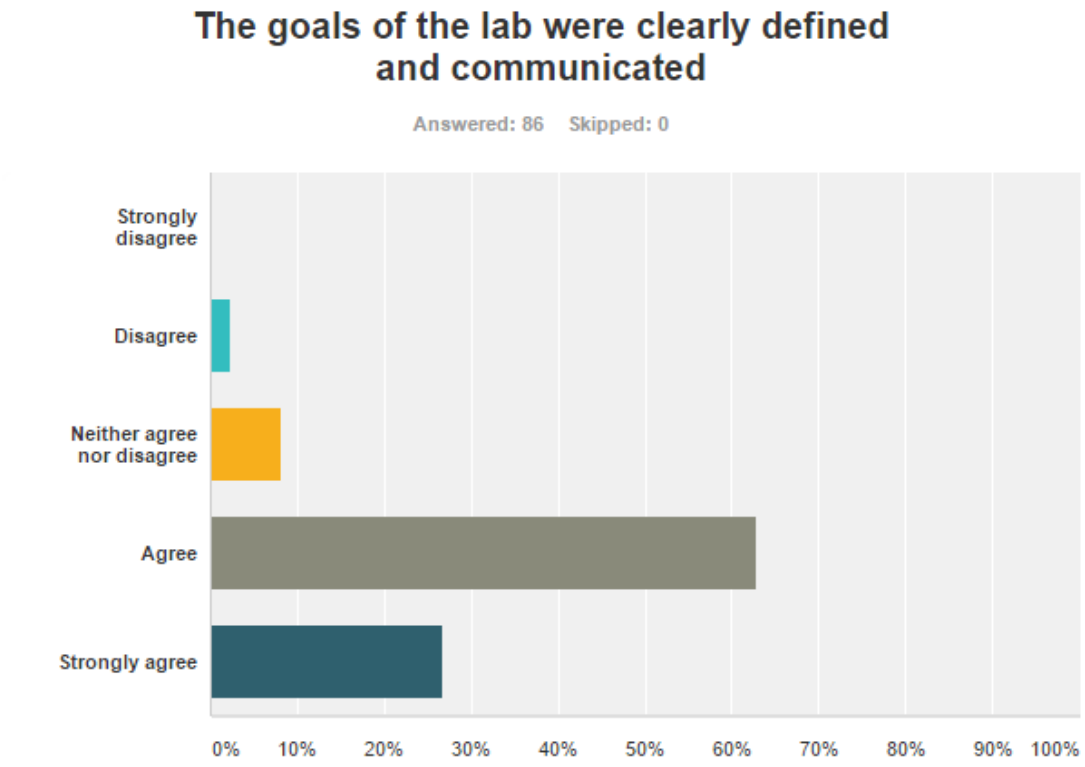
4.2 Evaluation Goals

Since the lab was not created by a professor, some kind of evaluation was needed. This was asked from the students, who this lab was intended to. The questions asked in the questionnaire should help to determine the value of the lab and if there is need for this kind of lab in the future.

4.3 Evaluation Results

For evaluation of the survey results we used a **Likert Scale** [17], psychometric scale which allows participants to express how strongly they are agree or disagree with one or another statement. There were five options students could choose to state their position: *Strongly agree, Agree, Neither agree nor disagree, Disagree, Strongly disagree*. Results were conducted into horizontal **bar charts**: each one represents the percentage division of students opinions.

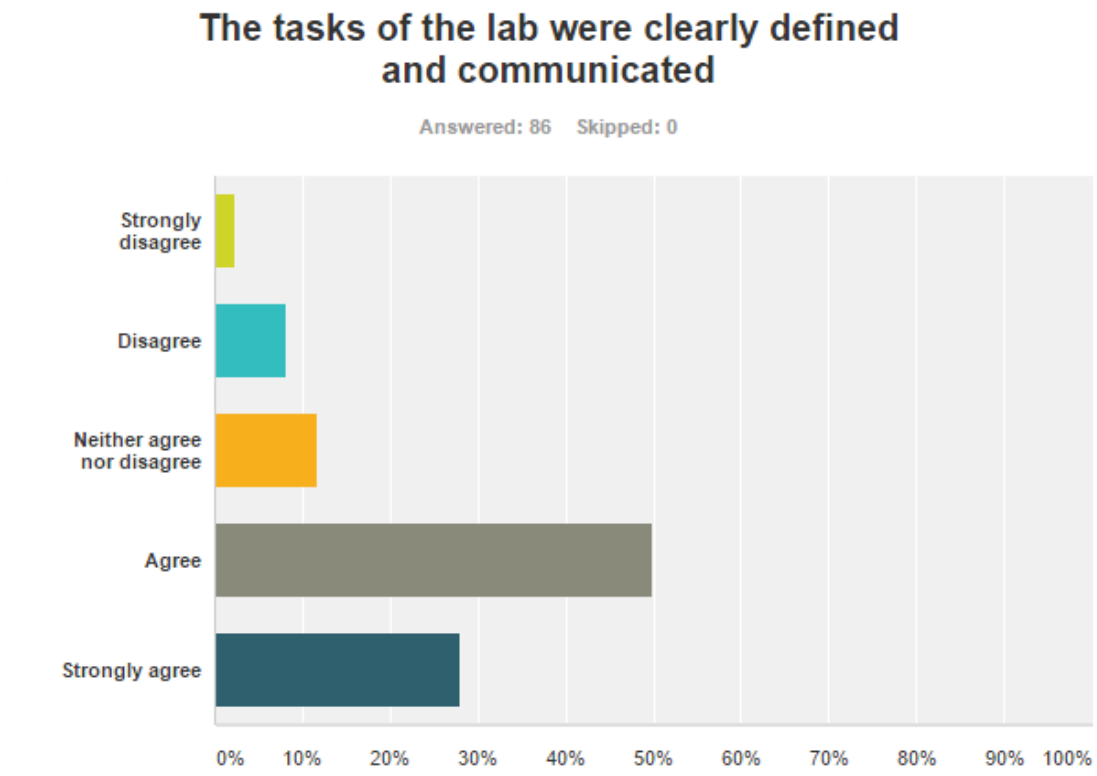
The questionnaire contained 9 different statements in affirmative form, besides students had a possibility to express an exclusive opinion or to share ideas in free form in the comment box. Analysis of the statements can be found below. Answers in the comment box were analysed by the author apart from the other results and handed (in a subjective manner) as an ideas for future improvements. Analysis of the results can be found in section 4.4.



Answer Choices	Responses	
Strongly disagree	0.00%	0
Disagree	2.33%	2
Neither agree nor disagree	8.14%	7
Agree	62.79%	54
Strongly agree	26.74%	23
Total		86

Graph 1. **The goals of the lab were clearly defined and communicated**

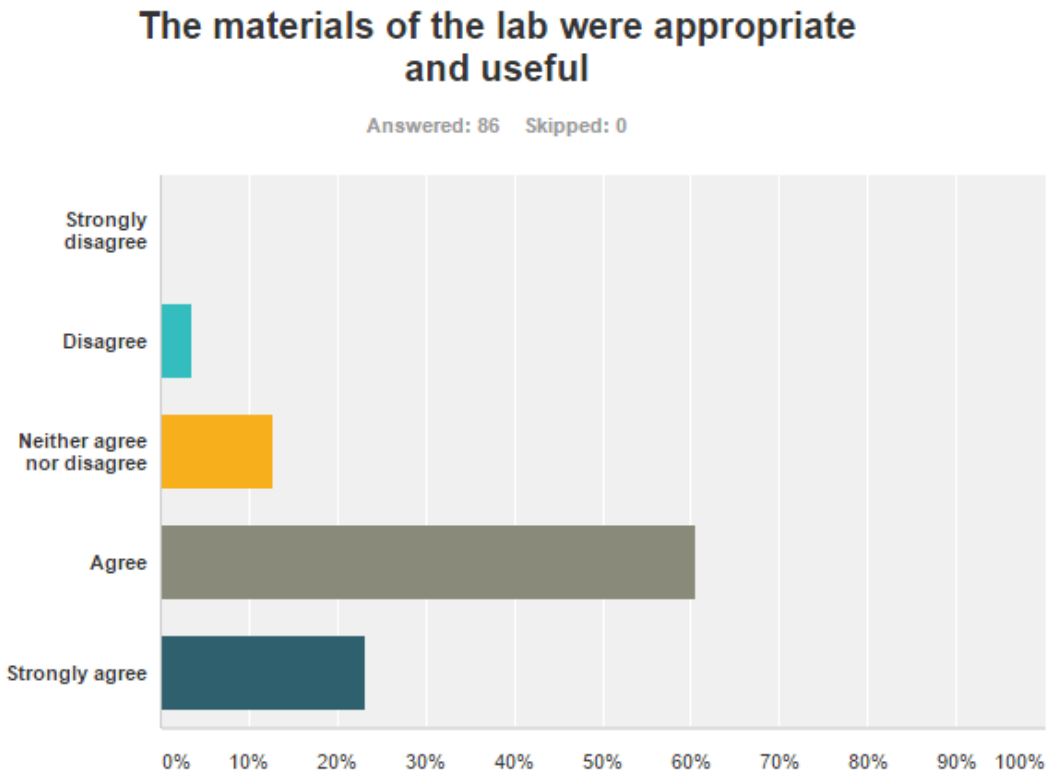
The first statement in the evaluation questionnaire was "*The goals of the lab were clearly defined and communicated*". In total 86 students expressed their opinion regarding this statement. The great majority, which is 89.53% of the students agreed or strongly agreed that the goals were clearly defined and communicated. 8.14% of the students decided to stay neutral, neither agreed nor disagreed, and only 2.33% of the students completely disagreed with this statement. It can be assumed that in general the goals of the lab were defined clearly and we succeeded in communicating those to the students.



Answer Choices	Responses	
Strongly disagree	2.33%	2
Disagree	8.14%	7
Neither agree nor disagree	11.63%	10
Agree	50.00%	43
Strongly agree	27.91%	24
Total		86

Graph 2. **The tasks of the lab were clearly defined and communicated**

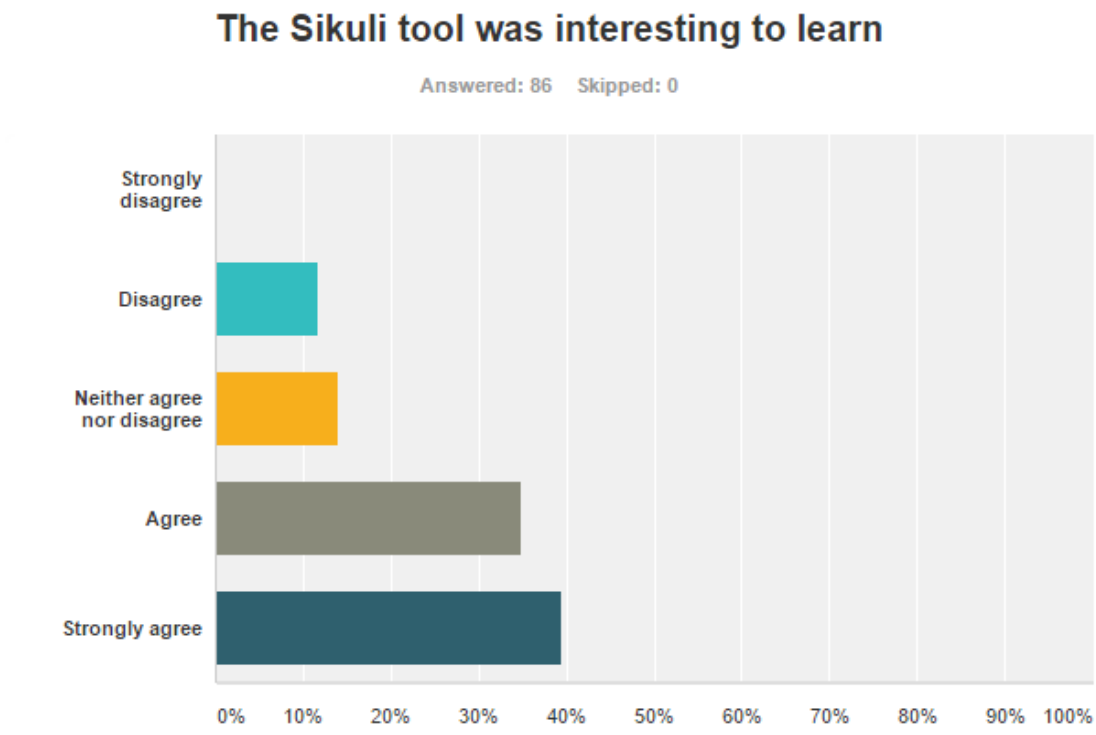
The next statement "*The tasks of the lab were clearly defined and communicated*" conducted answers from 86 students. The number of students who agreed with the statement (including both, Strongly Agree and Agree) is 77.91%, which is the superiority. However in comparison to previous statement, the higher percent of students chose to disagreed, which is 10.47%, and also 11.63% decided to abstain from stating the clear position. This means that there were some number of issues with comprehension of the way how tasks of the lab were defined, which can and should be improved.



Answer Choices	Responses	
Strongly disagree	0.00%	0
Disagree	3.49%	3
Neither agree nor disagree	12.79%	11
Agree	60.47%	52
Strongly agree	23.26%	20
Total		86

Graph 3. **The materials of the lab were appropriate and useful**

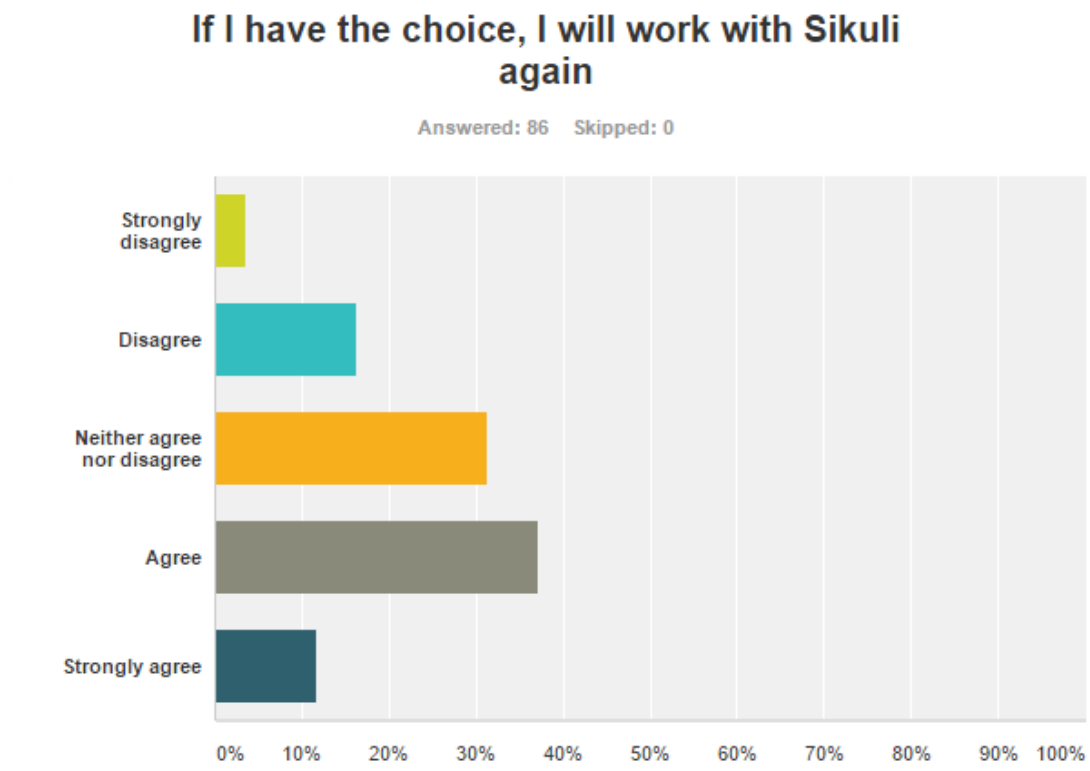
Total number of students who answered the statement "*The materials of the lab were appropriate and useful*" was 86. This number includes 83.73% of students who agreed with the statement, 12.79% of students chose the neutral position and 3.49% of those who decided that the materials of the lab were not useful. As the majority of students were positive about the statement it can be concluded that materials of the lab were useful and designed in appropriate way.



Answer Choices	Responses	
Strongly disagree	0.00%	0
Disagree	11.63%	10
Neither agree nor disagree	13.95%	12
Agree	34.88%	30
Strongly agree	39.53%	34
Total		86

Graph 4. **The Sikuli tool was interesting to learn**

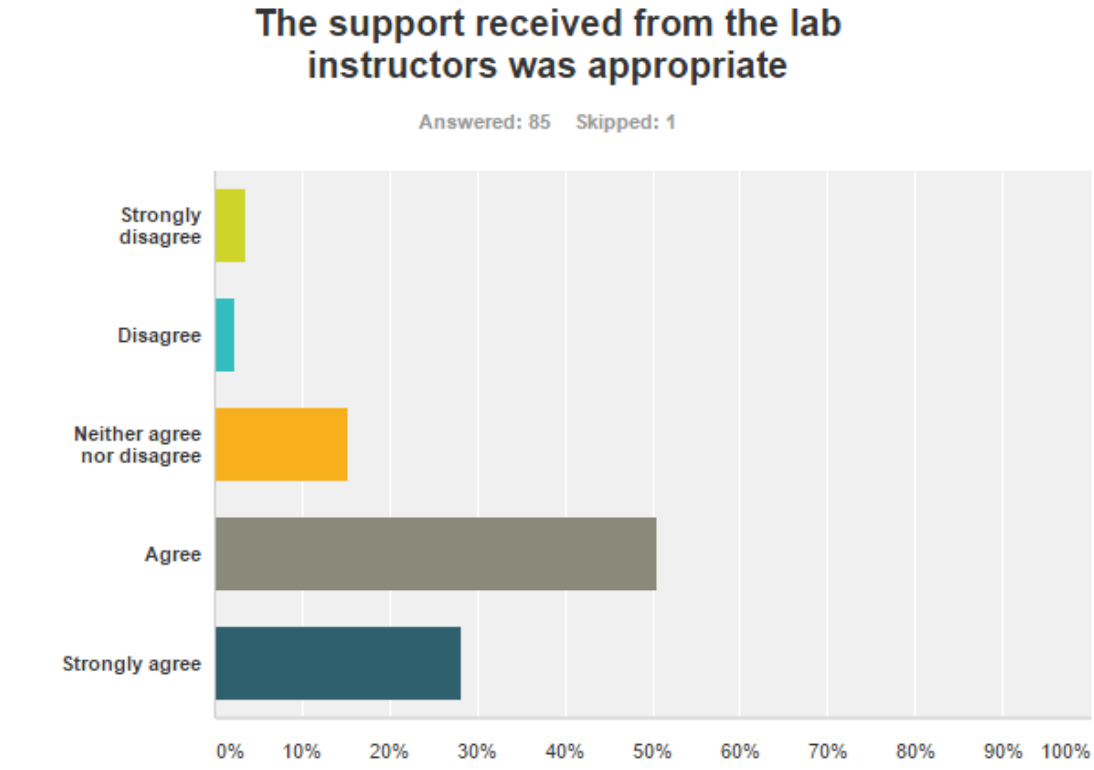
74.41% of the students agreed with the statement "The Sikuli tool was interesting to learn", this gives us the courage to state that learning Sikuli in frames of the Lab was fairly interesting. However 11.63% of the students on the contrary were not supportive about the idea of the statement. Some part of the students (13.95%) decided to keep neutral position.



Answer Choices	Responses	
Strongly disagree	3.49%	3
Disagree	16.28%	14
Neither agree nor disagree	31.40%	27
Agree	37.21%	32
Strongly agree	11.63%	10
Total		86

Graph 5. **If I have the choice, I will work with Sikuli again**

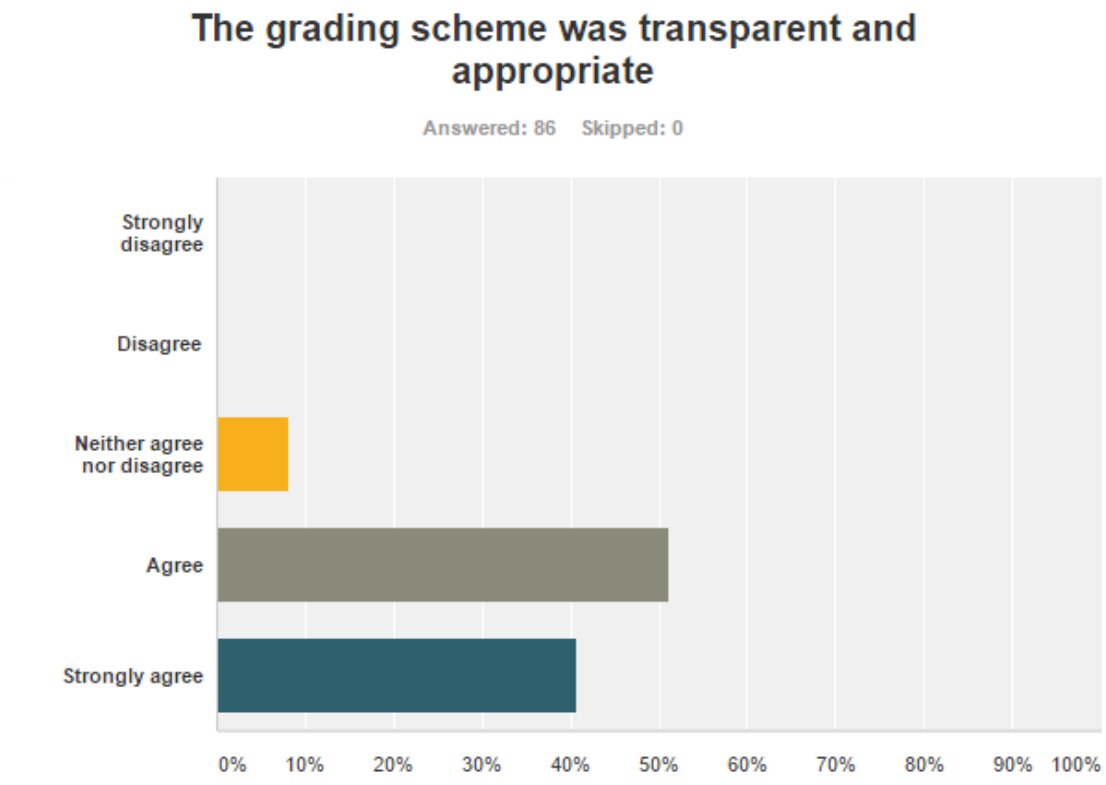
Another statement regarding assessment of the Sikuli framework is formulated like this "If I have the choice, I will work with Sikuli again". The analysis shows that 48.84% of 86 students who expressed their opinion about the statement, would agree to work with the framework again. However 31.40% of students didn't give any concrete answer and 19.77% were disagree and strongly disagree with the statement. But taking into account the subjective idea of the statement, we still believe that it is a good start for introducing the Sikuli for the course participants and we are glad that half of the students are ready to deal with the framework again.



Answer Choices	Responses	
Strongly disagree	3.53%	3
Disagree	2.35%	2
Neither agree nor disagree	15.29%	13
Agree	50.59%	43
Strongly agree	28.24%	24
Total		85

Graph 6. **The support received from the lab instructors was appropriate**

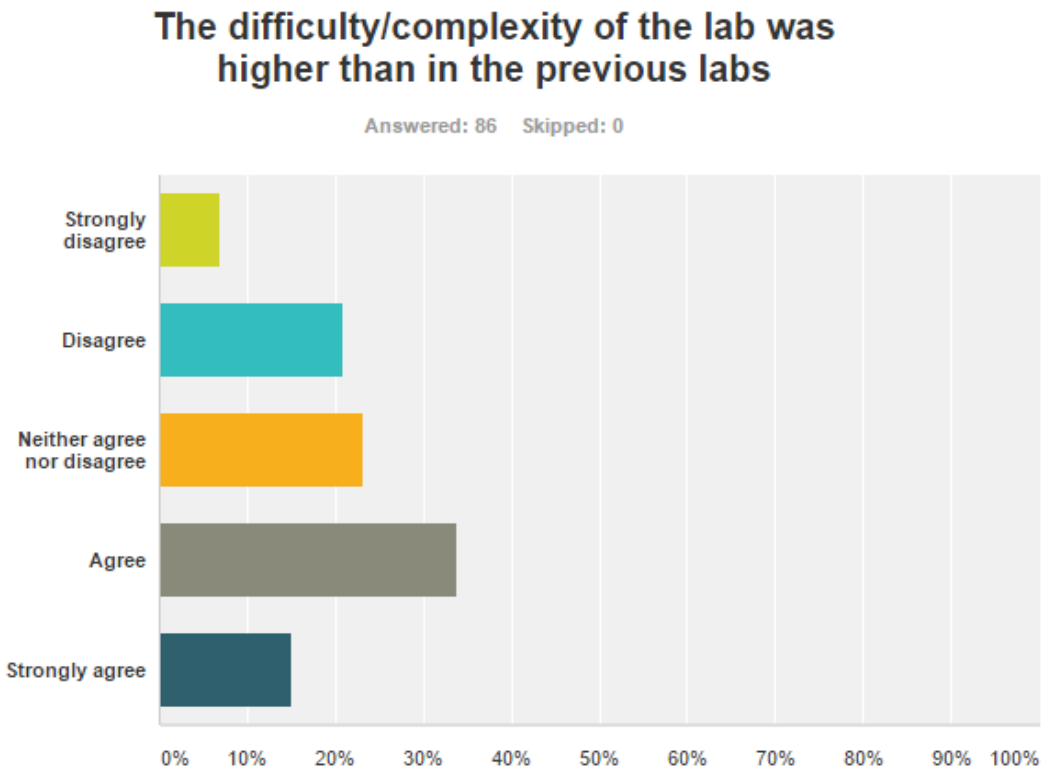
In comparison to other questions, 85 instead of 86 students gave answers to the following statement: "*The support received from the lab instructors was appropriate".* Percentage of the students who agreed or strongly agreed that the support received from the lab instructors was appropriate was 78.83%. 15.29% of the students neither agreed nor disagreed with the statement. 5.88% of the students disagreed or strongly disagreed with the statement. The overall satisfaction rate can be improved by simplifying the abstract idea of the lab and then doing a transfer knowledge to lab instructors.



Answer Choices	Responses	
Strongly disagree	0.00%	0
Disagree	0.00%	0
Neither agree nor disagree	8.14%	7
Agree	51.16%	44
Strongly agree	40.70%	35
Total		86

Graph 7. **The grading scheme was transparent and appropriate**

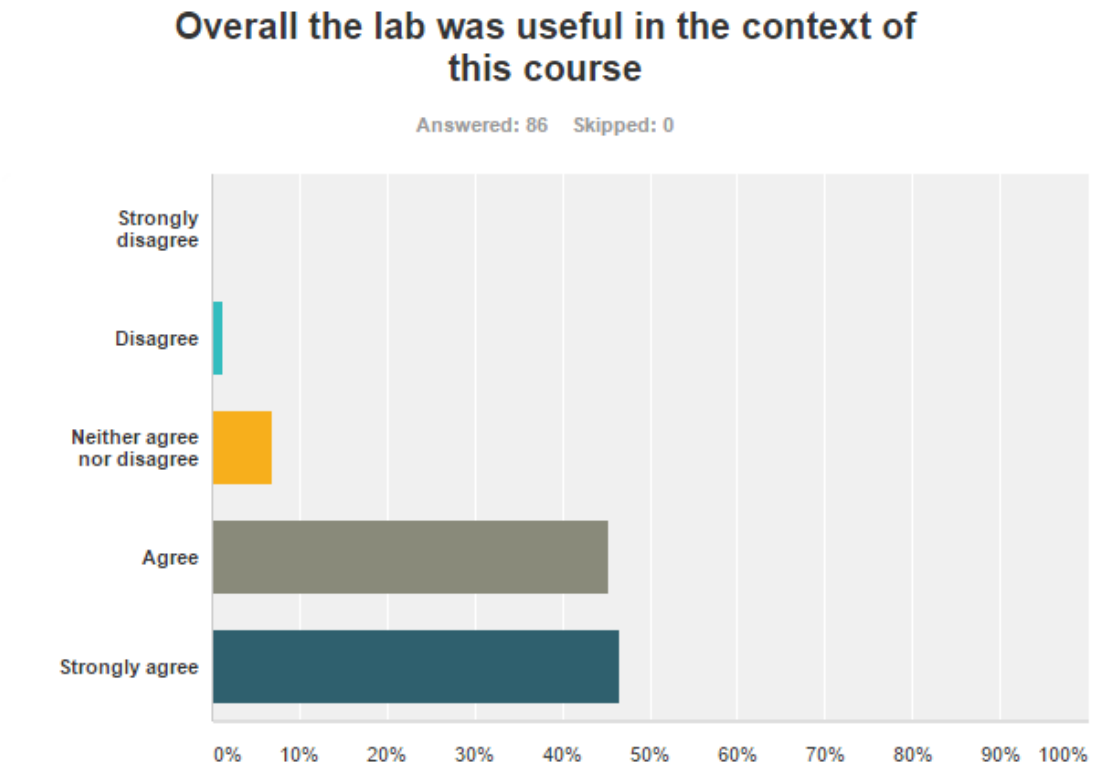
Total number of 86 students gave an answer to the following statement: "*The grading scheme was transparent and appropriate".* The percentage of students who agreed or strongly agreed with the statement was 91.86%. 8.14% of the students neither agreed nor disagreed with the statement. None of the students disagreed or strongly disagreed with the statement. This can be interpreted that the grading scheme is understandable to the students and doesn't necessary require improving.



Answer Choices	Responses	
Strongly disagree	6.98%	6
Disagree	20.93%	18
Neither agree nor disagree	23.26%	20
Agree	33.72%	29
Strongly agree	15.12%	13
Total		86

Graph 8. **The difficulty/complexity of the lab was higher than in the previous labs**

There were 86 students who gave an answer to the following statement: "*The difficulty/complexity of the lab was higher than in the previous labs".* 48.84% of the students agreed or strongly agreed that the difficulty or the complexity of the lab was higher than in the previous labs. 23.26% of the students neither agreed nor disagreed with the statement. 27.91% of the students disagreed or strongly disagreed with the statement. Simplifying instructions and installation guide in the future labs can help to lower the difficulty/complexity of the lab.



Answer Choices	Responses	
Strongly disagree	0.00%	0
Disagree	1.16%	1
Neither agree nor disagree	6.98%	6
Agree	45.35%	39
Strongly agree	46.51%	40
Total		86

Graph 9. **Overall the lab was useful in the context of this course**

There were 86 students who gave an answer to the final statement "*Overall the lab was useful in the context of this course".* From 86 answers, 91.86% of the students agreed or strongly agreed that this lab was useful in the context of this course. 6.98% of the students neither agreed nor disagreed with the statement and 1.16% of the students disagreed or strongly disagreed with the statement. As the majority of the students were positive about the statement it can be summarized that the lab was successful in the context of this course.

4.4 Ideas for improvement

In this chapter we conducted major ideas for the future improvements, basing on the comments students left in the comment box. The following list of ideas is the subjective opinion of the author how the lab package could be improved:

- To simplify the installation tutorial
- To emphasize the importance of automated testing as a separate paragraph
- To create a tutorial that covers installing Sikuli on Mac OS
- To make specification requirements more comprehensive for the students (documents Specification 1.pdf and Specification 2.pdf)
- To improve the framework by adding a way to set Sikuli similarity function globally
- To create the "Troubleshooting" section for problem solving (e.g. "Sikuli doesn't start")
- Introduce Selenium as a separate lab to demonstrate how to test web applications
- Refactor the skeleton code

5 Conclusion

We found that students are open-minded to practical lab. Overall the feedback seemed to have more positive tendency. Students gave valuable answers that can be used in the next year lab to improve the general satisfaction. Based on the feedback, it seems that the most important improvement topic is how to install the Sikuli rather than the lab package itself. We believe that if installation guide had been clearer, satisfaction rate in the lab would have also been higher. While the official operating system (OS) supported by the lab was Windows, many students use other OS on their computers. This created problems during the installation of Sikuli. Thus, there is a need to create a separate installation guide for other operation systems as well. We strongly believe that if the students had configured Sikuli with lower effort, they would have had more time and energy to focus on the main idea of the lab. This however was not the case with all students. Based on the feedback we can conclude that some of the students found the lab to be very useful, which is also supported by the last question asked from the students.

6 References

[1] J. Bentley E., Bank W., Software Testing Fundamentals—Concepts, Roles, and Terminology, Paper 141-30, pp. 1-2, 11, 2005

[2] Maurya V.N., Kumar Er. Rajender, Analytical Study on Manual vs. Automated Testing Using with Simplistic Cost Model, International Journal of Electronics and Electrical Engineering, Volume 2 Issue 1, pp. 1-4, 2015

[3] Sharma R.M., Quantitative Analysis of Automation and Manual Testing, International Journal of Engineering and Innovative Technology, Volume 4 Issue 1, pp.1-3, 2014

[4] Pocatilu P., Automated Software Testing Process, Economy Informatics, no. 1/2002, research paper, 2002

5) Zallar K., Practical experience in software testing, web resource (accessed 29.03.2015), http://www.methodsandtools.com/archive/archive.php?id=33

[6] Erdogmus H., Melnik G., Jeffries R., Test-Driven Development, Encyclopedia of Software Engineering DOI: 10.1081/E-ESE-120044180, 2011

[7] Quality Logic Inc., Web Services API testing, web resource (accessed 3.04.2015), https://www.qualitylogic.com/tuneup/uploads/docfiles/web-api-testing.pdf, 2013

[8] Blackburn M., Nauman A., Strategies for Web and GUI Testing, Software Productivity Consortium NFP, Inc. and TVEC Technologies, Inc., pp.1-2, 2004

[9] GUI Driven Design, The problem with GUI Test Automation, web resource (accessed 3.04.2015), https://guidriven.wordpress.com/2010/10/13/the-problem-with-gui-test-automation/

[10] Myers G., The Art of Software Testing, Wiley, ISBN 978-0-471-46912-4., pp.147-148, 2004

[11] Huston T., What is regression testing, web resource (accessed 7.04.2015), http://smartbear.com/all-resources/articles/what-is-regression-testing/

[12] Sikuli official web resource (accessed 8.02.2015), http://www.sikuli.org/

[13] Sikuli Doc Team, How Sikuli works, Sikuli X 1.0 documentation (http://doc.sikuli.org/devs/system-design.html), 2010

[14] Ravello Community, Review of 5 Modern Automation Test Tools for UI, web resource (accessed 27.04.2015), http://www.ravellosystems.com/blog/review-5-modern-test-tools-ui/

[15] Standford University., Laboratory Teaching Guidelines, web resource (accessed 28.04.2015) https://teachingcommons.stanford.edu/resources/teaching-resources/teaching-strategies/laboratory-teaching-guidelines

[16] SurveyMonkey online questionnaire, web resource (accessed 10.04.2015) https://www.surveymonkey.com/s/PPFCF9P

[17] McLeod, S. A, Likert Scale, web resource (accessed 5.04.2015) http://www.simplypsychology.org/likert-scale.html

7 Appendix A

All these files were accessed on 10th of May. Files are located at Institute of Computer Science homepage. Materials for the TA are located at dropbox folder for Software Testing course.

Lab - intructions.pdf –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06instruct.pdf

Lab – installation.pdf –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06install.pdf

Specification 1.pdf –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06spec1.pdf

Specification 2.pdf –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06spec2.pdf

Autotest for students.zip –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06autotest.zip

Software1.jar –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06software1.jar

Software2.jar –
https://courses.cs.ut.ee/2015/SWT2015/spring/uploads/Main/SWT2015-lab06software2.jar

Grading guideline.docx –
https://www.dropbox.com/s/w7vwbxxzvubrs59/Grading%20guideline.docx?dl=0

Autotest for TA.zip –
https://www.dropbox.com/s/442djxchq8vwhcr/Autotest%20for%20TA.rar?dl=0

License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Rasmus Sõõru** (date of birth: 24th of November 1990),

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: Automated GUI Regression 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, **26.05.2015**