TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Marina Lavrentjeva

143688 IAPM

TESTING ON EXAMPLE OF 3D PRINTING SYSTEM

Master’s thesis

Supervisor: Jekaterina Tšukrejeva
Master of Science

Tallinn 2017

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Marina Lavrentjeva

143688 IAPM

TESTIMINE 3D PRINTIMISSÜSTEEMI NÄITEL

Magistritöö

Juhendaja: Jekaterina Tšukrejeva
Magistrikraad

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Marina Lavrentjeva

28.04.2017

Abstract

The current Master's thesis describes testing of 3D printing system. The purpose of this work is to find and use automation tool for 3d printing system, where requirements are not stable and change during project.

Firstly, project and application, for what automation tool is searched, is described generally and development process in it. Secondly described what testing is. Testing types, methods, levels and objectives is described in this theoretical part. In third place, manual testing is done and description of basic tests in it. Lastly automated testing with the analysis of basic tools is done with description of basic automated tests in it.

As a result, four automation tools are analysed and one automation tool is selected. This tool will be used in a project to automate testing.

This thesis is in English and is 55 pages long, including 5 chapters, 11 figures and 8 tables.

Annotatsioon

Testimine 3D printimissüsteemi näitel

Käesolev magistritöö kirjeldab 3D printimise süsteemi testimist. Käesoleva töö eesmärk on leida ja kasutada automatiseerimisvahend 3D printimise süsteemi jaoks, kus nõuded ei ole stabiilsed ja arendatavad tarkvarad muutuvad kiiresti.

Esiteks kirjeldatakse projekti ja süsteem, mille jaoks on vaja leida ja kasutada automaatika vahend arendamise protsessis. Teiseks kirjeldatakse testimise teooria. Testimise tüüp meetodid, tasemed ja eesmärgid kirjeldatakse käesolevas teoreetilises osas. Kolmandaks loodi ja kirjeldati käsitsi katsetamine ja kirjeldatakse põhilised testid. Lõpetuseks analüüsitakse testimise automatiseerimisvahendid ja automatiseeritud testimine on loodud. Selles osas kirjeldatakse ka põhilised testid.

Töö tulemusena analüüsitakse nelja erinevat automatiseerimisvahendit, kirjeldatakse nende omadusi ning valitakse välja ühe, mis vastab kõige enam seotud kriteeriumitele. Vahendit, mis välja valitakse, hakatakse kasutama kirjeldatud funktsionaalse rakenduse automatiseeritud testimiseks.

Lõputöö on kirjutatud inglise keeles ning sisaldab 55 leheküljel teksti, 5 peatükki, 11 joonist, 8 tabelit.

List of abbreviations and terms

DPI	Dots per inch
TUT	Tallinn University of Technology
SDLC	Software Development Life Cycle
GUI	Graphical User Interface
XSS	Cross-Site Scripting
UTF	Unified Functional Testing

Table of contents

List of figures

List of tables

1 Introduction

At the present time, IT companies are trying to develop a quality product and at the same time reduce the time required to the development. To simplify life for developers, testers appeared. Testers create tests and looking for errors in the generated product. Recently acquire a special urgency testing issues in information technology, as an activity that improves the quality of software products, due to the ever-increasing competition. [1]

This theme was chosen because, the relevance of testing currently underestimated by developers, who believe that they themselves can test the product under development or the manager may consider testing a waste of time and money.

However, in large-scale projects under testing in general it is very expensive and time-consuming activity. Reliable and efficient product development is impossible without testing. One of the obvious solutions to this situation is to automate the testing process.

At the moment, there are quite a few types of testing. The main ones will be discussed in this present work.

In this thesis work, there was a task to implement testing for «3dprinteros» system. The Company «3dprinteros» is developing a single system for printers and file management. System «3dprinteros» is a one easy-to-use platform for managing files, machines and users across your business. [2]

Initially, the problem was that the testing has not been implemented in the project. Which is why was a lot of mistakes in the project. They were not corrected, because a basic functional worked. Testing was created from scratch.

Since the Company is engaged in not only an ordinary site but also in software for printers, it was impossible to use only the automatic testing. The task of work is description of manual and automatic testing process of the «3dprinteros» system in programming language Java.

The object of the research is manual testing and GUI testing in Java programming language. As well will be discussed technologies for automated testing. Research of tasks and process of project testing is presented in this work.

2 Project description

Firstly, is necessary to tell about what this 3DPrinterOS system is which needs to be tested. Testing was not realized in this firm, which is why there were many errors in the project. It became a large problem that had to be solved. In the following chapters will be discussed about how testing was implemented manually and later automatically.

2.1 3DPrinterOS Client

3DPrinterOS Client part is the control software for 3D Printer's. Server part location is at URL: https://cloud.3dprinteros.com.

3DPrinterOS supports almost all printers based on Printrun and Makerbot Firmware.

2.1.1 Client Features:

- Port selection or baud rate not needed because we have auto-detection with auto-connection
- The connection and printing on multiple printers is possible on one computer at the same time
- It is also possible to work remotely with the printer. In particular, the preparation of a file for printing and printing itself
- Remote view of 3D printing thanks to the connected web camera (you can use more than one web camera but you will need minimum 2.0 GHz 2 cores CPU or more) [3]

2.2 3DPrinterOS Cloud

Cloud is a centralized file and printers management system. You can upload your 3D files or search for free models and save them in 3DPrinterOS account for future printing. Also, it is possible to change your stl model with applications such as Slicer or Magic Fix. As well you can specify the number of print to a file or printer. 3DPrinterOS provides fast connection with 3D printers, literally with one click. Cloud location is at URL: https://cloud.3dprinteros.com. [2]

2.2.1 Cloud Features:

Cloud webpage have a several licenses. In this work will be used only free license.

Free license:

- Upload and Print Gcodes
- Gcode and Toolpath Viewing
- Remote Monitoring
- Forum Support

Premium license:

- All Free Features
- Printer Sharing and Permissions
- File/Project Sharing and Permissions
- Print Queuing
- Live Platform Support

Educational license:

- All the professional features
- Advanced User and Machine Reports
- Export Report Data
- Single Sign On Support
- Workgroup Creation
- Design Tool Integration
- Live Platform Support

Enterprise license:

- All the educational features
- Customize AM Workflows

- Industrial Machine Support
- Custom APIs
- Private Cloud
- Success Manager
- Monthly Check-Ins
- 24/7 Phone and Platform Support

2.3 Development process

The Flexible method of development or Agile is used in this project. Agile – is an approach to product development that focuses on the use of iterations dynamically evolving requirements and maintenance of their implementation. The implementation happens as a result of permanent co-operation between group of developers, which includes both programmers and testers.

The hard phases of project development life cycle are usually used in the traditional testing. It all starts with the planning of release and definition of requirements, and ends with the hasty testing phase and delayed release version of the product. This approach, when testing takes place in the end, does not help to release a quality product. Testing carried out hastily, as writing code often takes more time than expected, but also because the development team gets to the error correction phase at the very end.

Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. Teams work through sprints to respond to unexpected changes in priorities during the iteration. This means that testers check every addition of code immediately after it was written. Iteration may be short - one week or longer - a month. The team assembles and testing a small piece of code, to make sure it is working correctly, and then proceeds to write a new one. Developers never break away from the testers, as their work is not complete until it is tested. [4]

3 Testing

Currently, the term "testing" means the search for errors in the project, but this is not true. Existing test methods do not allow us to find all defects and to establish that the test product is working properly. This means that all test methods formally checking product, which is being developed. To understand what testing is all about and to get acquainted with the principles of testing, it is necessary to understand what types and methods of testing exist and are applied.

3.1 Types of Testing

This part is a description of different testing types that can be used for software or webpage testing during SDLC. There are many signs, which are accepted to make types classification of testing. In this work types of testing will be considered only in the degree of automation of the product.

3.1.1 Manual Testing

Manual testing is testing with bare hands, when automation script’s not being used. Tester’s work is to identify any unexpected problem in software. For that he is working with program as an end-user. Tester checks the operability of all components of the program, emulating the actions of the user. Manual testing has different stages: user acceptance, unit, integration and system testing. Tester can use pre-prepared test plan, which marked the most important aspects of the program.

3.1.2 Automation Testing

Automation testing or Test Automation is the testing process with a test scripts and different software that tester can use to test the product. This process includes the automation of the manual process. The most common form of Automation is to test applications via a graphical user interface. Automation Testing is used to run the test cases that usually were used by hands quickly and repeatedly.

Automation testing is used to test applications with stress tests, load tests and performance tests. In comparison to manual testing, automation testing saves time and improves accuracy by increasing the test coverage.

Of course, everything is impossible to automate. For example, what should be automated, login or registration forms where user can make transactions and pages where many users can access the software at the same time. In addition, can be automatically tested, GUI elements, field validations, database connections etc. [5]

3.1.3 Semi-automated Testing

In this case, manual testing is combined with automated testing. For example, the tool creates a new account, and then created user actions are performed manually. In this work only manual and automated testing will be considered.

3.2 Testing methods

For software testing can be used many different methods. This chapter contains short description of some of the available methods.

3.2.1 Black-Box Testing

Testing method is called black-box testing when you test without having any knowledge of working processes inside the application. Tester don’t have any access to the source code of the application and regardless to the system architecture.

Usually, during the black-box test, the tester communicates with the user interface of the system, passing the input data and receiving the output data for later analysis. The tester does not know where the data is processed.

In this table are listed advantages and disadvantages of black-box testing (Table 1).

Table 1. Black-Box Testing Advantages Disadvantages

Advantages	**Disadvantages**
• Suitable and effective for large segments of code • Access to code is not required • User's perspective is separated from the	• Code coverage is limited, because we have only certain amount of test scenarios • Testing is inefficient, because the tester

developer's perspective • Application can be tested with big number of enough skilled testers without any information of realization or programming language	has limited information about application itself • Code coverage is blind, because tester can’t target particular code sections or mistake inclined zones • It is hard to write test cases

3.2.2 White-Box Testing

Testing the "white box" is performed in order to detect problems in the internal structure of the program. This requires the examiner to have a deep knowledge of the internal structure and, therefore, cannot be performed by the ordinary user. The common goal of such testing is to ensure that each step is checked according to the algorithm of the program.

In this table are listed advantages and disadvantages of white-box testing (Table 2).

Table 2. White-Box Testing Advantages Disadvantages

Advantages	**Disadvantages**
• The tester has access to the source code and can easily find more suitable data types for testing. Therefore, testing becomes more efficient • It helps in making the most out of the code • Extra lines of code can be taken away which can take in kept hidden problems • The maximum coverage is achieved, because the tester get needed knowledge about the code when he is writing test cases	• Testing cost is increased, because a skilled tester is expected to perform white-box testing • It is impossible to test everything in the code. Tester can’t look in every corner and find everything. Too many time is needed for that. Hidden errors may create problems • White-box testing is difficult to support, because it has need of special instruments like code analyzers and debugger

3.2.3 Grey-Box Testing

Grey-box testing is a combination of a white and a black box testing techniques together. It is a testing with low knowledge of the inside workings of an application. It is assumed, for example, access to the internal structure and operation software algorithms to write the most effective test cases, but the test itself is performed using black box techniques, from user's position. Tester in this method has access to documents, design and the database. Tester that have this knowledge can create better test cases and scenarios when he is creating a test plan.

In this table are listed advantages and disadvantages of grey-box testing (Table 3).

Table 3. Grey-Box Testing Advantages Disadvantages

Advantages	**Disadvantages**
• Grey-box tester does not depend on the program code. Grey-box tester uses functional specifications and interface definitions • Grey-box tester has limited information. Therefore, he can develop good test scenarios. For example, around data type handlings and communication protocols • The test is written from the user position, not the developer or tester	• We don't have access to source code, which is why we have limited ability for the test coverage • The tests can be unnecessary if they are repeated in test cases by tester many times • It's not necessary to test every possible input variable. It is unprofitable, since it would take too many times. That is why few program routes will be untested

3.3 Test Objectives

To properly test a project, we need to set goals that must be met during the testing of the project. This includes checking the functionality of the system, creating tests, possible automating and organizing tests. All this is described below in more detail's and will be performed during both manual and automatic testing.

3.3.1 Check the basic functionality of the system

The most important task of testing to check, that the necessary basic user functionality of the system is working properly. As if action that the user will perform every day, don't work correctly, or illogical, then it will affect the effectiveness of the user work in the future. From this comes the next task of testing – it is important to check up logic and comfort of user interface.

3.3.2 Create Tests

Testing of a product is always associated with the creation of tests. Tests must contain a minimum of code that, in future, they can be easily adjusted to a new version of the product. Mostly it concerns tests associated with a graphical interface, such as checking webpage items.

3.3.3 Automate testing

It is also important to automate the testing, if appropriate. Automation reduces testing time and simplifies the process. If for this purpose there is time and resources, it is needed to consider automation as important component of testing.

3.3.4 Organize testing

Organize testing is necessary so that not only detect, but also to prevent defects in the beginning. Create automated tests should be organized to use the same functions, rather than create new ones for each part of the tested product.

As a result, we can say that the testing process is quite complicated and at the same time, an easy task and it is should be treated seriously. In the following chapters, you will see the purposes of testing, how was done manual and automated testing and created tests examples.

4 Manual testing

Test absolutely everything is physically impossible. One reason is that the number of all possible input data combinations is too large, so it cannot be fully checked. Because of what is checked only the most frequently used input data and boundary values.

Another problem is the number of all possible sequences of code execution, as it is also too high, that it can be fully checked. The user interface is generally too complex to full testing. [8, p. 40]

4.1 Test Objectives

As a result of testing, we need to implement the following objectives:

- Implement testing so that the tested product was working with highest possible quality
- Implement testing so that the tested product met all declared requirements
- Carry out the most complete product testing in the shortest time

The time and project quality is the criteria for a good testing of the developed product. Manual testing will cover all the features from paragraph 2.

4.2 Planning of the testing process

Testing also implies a planned analysis and careful use of the tested product. This implies the need for a test plan or a testing strategy. The purpose of a test strategy is to clarify the major tasks and challenges of the tested project.

4.2.1 Creating a test plan

To formulate a test plan is possible by using the following points:

- Analyze the testing scope. Usually this is done by studying the documentation or a definition of requirements. In our case there was no documentation, so the work was limited by the developer requirements

- To analyze the testing approach by selecting the tests associated with each stage of development. Since the work was associated with only one version of the product, such analysis was not carried out. Tests were selected once
- Use of automated testing shall not exceed the time given to the testing project. In this project, there were no restrictions for the use of automated tests, so testing is not exceeded

4.2.2 Determining the testing scope

Test absolutely everything is physically impossible. Below are the reasons why the complete testing can never be done:

- The number of all possible input data combinations is too large for a complete testing
- The number of all possible code execution sequences is too large for a complete testing
- The user interface is generally too complicated for a complete testing [8, p. 40]

Since it is impossible to find all the bugs, what should be tested in the first place? If you spend too much time on testing, trying to find as many problems as possible, the development process will be too long and will tighten the term of product delivery.

However, you can miss serious errors, which in the future will be difficult to resolve if the project is not enough tested. That is why the test volume is determined on the experience of the tester and can change during the product testing.

4.2.3 Choice of testable parts

It is worth remembering that it is not necessary trying to find as many errors as possible, and should try to miss as little as possible. That is why it was necessary to decide what exactly should be tested:

- Firstly, it is important to test boundaries of input values, basically because most errors are appearing there. For example, if in the input line can only enter numbers in the range from 1 to 1000, it is worth trying to enter or 1001 or 0. If the length of the entered value must not exceed 10 characters, it is worth trying to enter 11 characters

- It is also necessary to test changes which just appeared in the developed product. For example, parts which changed because of the correction of the already found error. These tests are called «Regression test»
- It is necessary to test the parts where finding of defect is most credible
- It is necessary to focus on the modules of the site, which is often used by end users

4.2.4 Example the testing process

It is necessary to consider the example of a product testing process:

- New functionality has been created, for example, the new Slicer application was added
- Functionality of the entire site is checked, because the development could break down already working functions
- The created new functional is checked up
- Communication with the developer via Skype, if something is not clear in the work of a new Slicer
- If an error was found, issue is opened in https://youtrack.3dprinteros.com

4.3 Testing of 3D printing

Basically, 3D printing on a printer is tested manually, since it is a machine and automate machines a rather complex and expensive task. In the present case you can just automate launch of printing but it is already quite simple and fast process which is why it is not required.

For testing the different programs were used for the printing process, since the print quality might be better when using other software.

Such programs as: Cura, Slic3r, CraftWare, Kisslicer, Simplify3D, MakerWare, Makerbot Desktop. In different programs in different ways registered slicer. You need to compare and validate gcode delivered to the printer. It is necessary that at the use of the third-party software the printer printed identically well.

4.3.1 Problems when testing 3D printing

In this list, you can see the reasons why it is impossible to test printing automatically:

- We need to remove the finished model from the bed and clean the bed. If this is not done, then the model will be printed directly in the ready-made model, lying on the bed
- Check external factors such as whether there was movement of the bed during printing
- During printing, the model may come unstuck from the bed. This requires checking the heating temperature of the bed, if it is heated. Check is very easy. Need to send a print item and make sure that the first print layers don’t come unstuck
- Problems with the USB cable may stop printing halfway. If you have a printing on 24 hours, and 20 hours later you will accidentally touch a bad USB cable, printing will be interrupted, since there is no connection to the printer. To avoid this, you need to check all the connected cables to the printer before printing

A person must do these processes. Automate some of these processes in such small firm would be too expensive. For example, it is possible to automate the removal of the finished model from the table by using robot. In this case, the robot also will need maintenance of the average person.

The only testing that can be automated is when the printer operability is checked up. In the present case printing starts without plastic and checked whether the system and the printer is working. The system can write that the heating is, while the printer is actually cold. Again, such things only person is checking.

4.4 Testing of client

The client is the software used in 3D printing for launching and tracking of 3D printing. It must be tested for proper operation with the 3D printer. Testing process is divided into 3 categories. The following tests were used to test client:

4.4.1 Test client with a connected printer

In the first category, you need to test a client with a connected printer. For this you have to use these tests:

- Check CloudSync folder opening process (open a folder, where you can upload any of the files for testing)

- View logs button should open log files
- Wizard testing – check if any of the buttons are working well and responding to mouse clicks
- Report problem should be able to send messages (it is preferable to type different characters and use different languages)
- Rename printer name should open a window with rename option (it is preferable to type different symbols and characters in different languages)
- Button „Reset printer type“ should enable you to change the printer's type
- Button „Go to your account now“ should direct you to a cloud page
- Logout button should terminate the client, also system should re-ask a password and login username on a next launch
- Check if „Restart camera button“ is working properly
- During the printing process you need to press „Pause“ button and wait, after printing is finished – press „Resume“ (Printing should continue)
- During the second printing , you need to remove an USB cable, printing should continue
- In the Apps of printer there should be „Joystick and Console“, currently these functions are not supported in the Duplicator / Replicator printers
- You also need to check if printer is reacting minimum to 2 commands sent over a „Joystick and Console“

4.4.2 Test a detect feature of a client

Second category of testing is needed to test a detect feature of a client.

- While client is turned on, connect printers to the client. They should receive an idle status (ready)

4.4.3 Check a printing feature on all printers

Third category of testing is needed to check a printing feature on all printers

- You need to connect a printer and start printing on all printers

4.5 Basic Cloud testing

Cloud is a centralized file and printers management system. When you test Cloud you need to check all clickable buttons. In this chapter, everything is listed with tables with tests. In this table, you can see basic manual testing process when we use functionality of different Cloud web pages on different browsers. Example of basic testing (a table with categories) is provided below (Table 4):

Table 4. Basic Cloud Testing

	OK	NOK	Not supported	Unable to test
Tester name	**Tester 1**	**Tester 2**	**Tester 3**	**Tester 4**
Browser version	**Chrome Version: 50**	**Firefox Version: 46**	**Opera Version: 36**	**Chrome Version: 50**
My Profile				
Sign In	OK	OK	OK	OK
Sign Out	OK	OK	OK	OK
Profile Overview	OK	OK	NOK	OK
Share Live Videos	OK	NOK	OK	OK
Add Media	OK	NOK	NOK	OK
Add Post	NOK	NOK	NOK	NOK
Drag and Drop	OK	NOK	OK	OK
Map	OK	NOK	OK	OK
Upload				
Drag files here	OK	OK	OK	OK
Search				
Search button	OK	OK	OK	OK
Add file (Thingverse)	OK	OK	OK	OK
My Files				
Search button	OK	OK	OK	OK
Add files button	OK	OK	OK	OK

Quick view model	OK	OK	OK	OK
Edit name	OK	NOK	OK	OK
File Log	OK	OK	OK	OK
Slice button	OK	OK	OK	OK
Print button	Unable to test	Unable to test	Unable to test	Unable to test
Applications				
3D Viewer	OK	OK	OK	OK
Resizer	OK	OK	NOK	OK
Net fabb	NOK	NOK	NOK	OK
Slicer	OK	OK	OK	OK
STL Editor	OK	OK	OK	OK
Magic fix	NOK	NOK	NOK	NOK
Sculpteo	OK	OK	OK	OK
Kitization	OK	NOK	OK	OK
Leopoly	OK	OK	OK	OK
Hollowing	Unable to test	Unable to test	Unable to test	Unable to test
Simplification	OK	NOK	OK	OK
Obfuscator	OK	OK	OK	OK
Autodesk Mesh Repair	Unable to test	Unable to test	Unable to test	OK
Spark Slicer	OK	OK	OK	OK
Share file	OK	OK	OK	OK
Facebook	Unable to test	Unable to test	Unable to test	Unable to test
FB Sketchfab	Unable to test	Unable to test	Unable to test	Unable to test
Twitter Sketchfab	OK	Not Supported	OK	OK
Print through Virtual Factory	OK	Not Supported	OK	OK
Dashboard				

World statistics	OK	OK	OK	OK
Show top 100 button	OK	OK	OK	OK
My printing history	OK	OK	OK	OK
Printers				
Add Virtual Printer/CNC	OK	Unable to test	OK	OK
Make Offline printers Inactive	OK	Unable to test	OK	OK
Add Workgroup Printers	OK	Unable to test	OK	OK
INSTALL button	OK	OK	NOK	OK
Set location	OK	OK	OK	OK
Logs button	OK	OK	OK	OK
Live view	OK	OK	OK	OK
Share	OK	OK	OK	OK
Live view wall tab	OK	NOK	Unable to test	OK
Inactive Printers tab	OK	Unable to test	Unable to test	OK
Settings				
Camera mode	OK	OK	Unable to test	OK
Camera choose	OK	OK	Unable to test	OK
Reset camera module	OK	OK	Unable to test	OK
Make inactive	OK	OK	Unable to test	OK
Printer Applications				
Console	OK	Unable to test	Unable to test	OK
Joystick	OK	Unable to test	Unable to test	OK

Description of notations:

OK	Working perfectly
NOK	Working with problems

Not supported	Not supported in this browser
Unable to test	Tester don’t have camera, printer or in test webpage this functional is not done yet

To get this table, the Client and Cloud sites were fully tested. Several browsers were installed: Chrome, Firefox and Opera. For several days and a different number of hours, this table was filled.

Example Test Scripts for Cloud webpage is set out below:

Test Log In	
ID	**001**
Description	Test goal is to log in by test user without problems
Precondition	Web page is not opened. Nobody is logged in
Steps	1. Open webpage https://cloud.3dprinteros.com/sign/ 2. Type in first text box email: example@email.com 3. Type in second text box sample password 4. Click button Sign In 5. Wait for page to open
Post Condition	User is logged in with his username

Test Dashboard	
ID	**002**
Description	Test goal is to check Dashboard links and headers
Precondition	Test user is logged in
Steps	1. Open webpage https://cloud.3dprinteros.com/dashboard/ 2. Click link World statistics 3. Click link My printing history 4. Verify if header present with name Overview
Post Condition	Links are working and header is in place

Test Profile Overview	
ID	**003**
Description	Test goal is to check that in Profile Overview Bandwidth and Storage available is 10 GB

Precondition	Test user is logged in
Steps	1. Open webpage https://cloud.3dprinteros.com/ 2. Click link Profile Overview 3. Check header: Profile overview 4. Check License: FREE 5. Check Storage available: 10 GB 6. Check Bandwidth available: 10 GB 7. Check Profile Settings 8. Check Change Password 9. Check Sign out
Post Condition	Links and headers are in place

4.6 How much time is spent

It's time to calculate how long it took to manually test the Client and Cloud. Big amount of site functions was manually tested (Table 5).

In this diagram, you can see amount of time we need to test manually Client system (Figure 1):

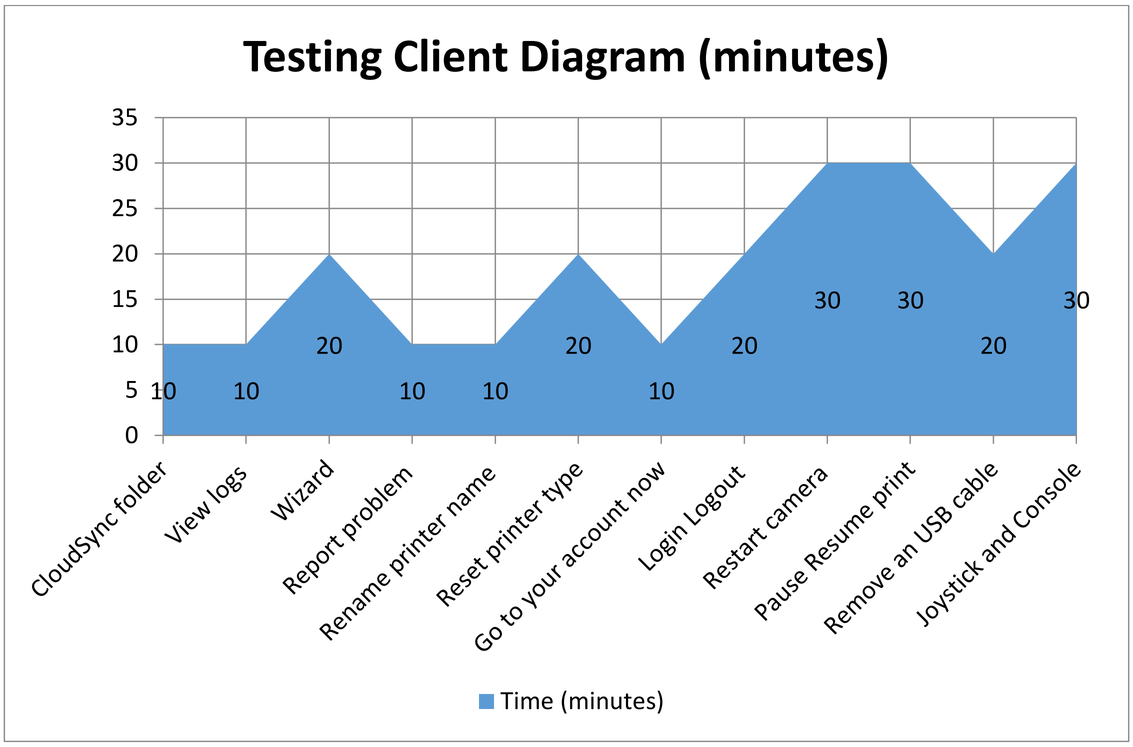


Figure 1. Testing Client Diagram

In this diagram, you can see amount of time we need to test manually Cloud system (Figure 2):

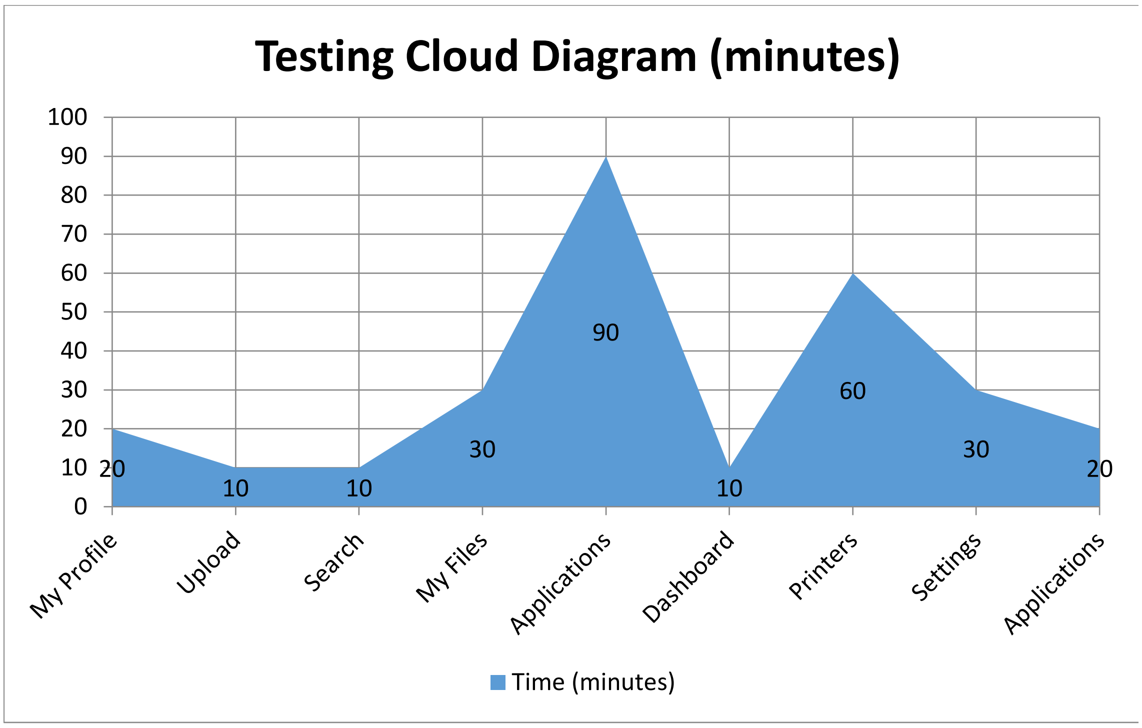


Figure 2. Testing Cloud Diagram

As a result, we can say that the client testing spent about 2-3 hours (Figure 1), and a complete testing system Cloud spent 3-5 hours (Figure 2). It's quite a lot of time and since the client launches the page in the browser, it can be automated with Selenium. And in the Cloud website can be automated verification of all links, buttons, and text captions.

In this system pretty much need to be tested, but you cannot automate everything. Many things can only be checked visually and manually.

For instance, in the Applications, you can check all of these programs that they are working, but when you work with them, you can see a lot of errors that are only visually determined as errors. For example, you can slice stl file with the STL Editor and get fully working gcode, but it will not look as it should.

5 Automated Testing

Most software products, produced today, are web applications and run in the browser. Testing efficiency of such applications differs in different companies and organizations. In the modern world, when many organizations are using Agile methodology in the process of software development, automation of testing often becomes necessary. By automation of testing means the use of tools in order to repeatedly perform repetitive tests to test the application. [9]

5.1 Why to automate?

With testing automation, as well as with many other highly directional IT - disciplines, associated a lot of misconceptions. Companies are now taking advantage of automation testing tools to increase their efficiency and productivity. Apart from several advantages and benefits of automation software testing there are certain disadvantages of these tools as well. The automated testing eliminating the need for manual labor of a tester and improving the testing time of an application. Basically, automation is used to simplify testing process and get rid of the routine repetition of everyday tests by tester. The more the tester checks the same part of the code, the less chance to notice the problem. It is necessary to list and describe the basic nuances of automation and provide an answer to the main question - when the automation should be applied. [10]

5.1.1 Test Automation Advantages:

- Repeatability – all written tests will always be executed monotonously, i.e. excluding the "human factor". The tester will not pass the test for negligence and do not mess up in the results
- Fast execution - an automated script does not need to consult the instructions and documentation. It saves a lot of execution time, because scripts execution is much faster than a human testing
- Confirmation of the known - automatic checks - is a great way to confirm that the application continues to function properly after the changes was made to it

We can easily identify new errors after changes in the code using automatic regression checks. The most important thing is to run automatic checks as often as possible

- Quick feedback – testers can give quick feedback about application problems as soon as application was changed. It is very good for the developer’s performance, since they need to fix everything that's broken before moving on to coding other things, that is why fast feedback is significant
- Frees up the testers time – when tests can be run regularly (can be automated), testers can work on other interesting parts of the program or on the parts that cannot be automated. It’s frees up the testers working time very efficiently

5.1.2 Test Automation Disadvantages:

- Repeatability – all written tests will always be carried out monotonously. It is both a disadvantage. Tester testing manually can pay attention to some details, and after spending a few additional operations find defects. Automatic test cannot do this
- Maintenance Time and Effort – it is especially important for regression testing - testing new versions. Each update of the interface or functionality of the software under test will require the completion of automation tests. Automation tests cannot adapt themselves to the new interface, and to continue their correct work, you have to change everything manually. The more the number of tests, and the more innovations in the software, the more time it will take to update. You have to spend the time on changes for the broken tests
- Not many bugs found - most of the errors are found randomly, since the input data for the test is most often changed during the experimental tests many times. On the other hand, the fact that automated tests go along the same path with the same input data reduces the probability of finding an error. But when we change the data in the tests it takes a lot of time, which we are trying to avoid
- The cost of the automation tool – in the case if you are using licensed software, its cost can be quite high. Free tool, usually are more modest with less functionality
- Minor errors – auto tests cannot detect small defects that do not harm the functionality of the code, but damage the visual interface and make it difficult for the end user work. Those auto tests can be succeeded when there can be

hidden problems that test will not see, because test do not have to look there. It was not written in test

In order to decide whether the automation is necessary you need to answer the question "Does the advantages of the automation outweigh its disadvantages?" - At least for some of the functionality of our product. If the disadvantages are not acceptable for this project, then you should not use the automation.

When making a decision it is worth remembering that alternative - a manual testing, with its own disadvantages. [11]

5.2 Automated testing tools

Automated testing has many advantages, mainly relating to the high-speed test execution and the ability to perform the same type of tests repeatedly.

It is necessary to define the types of testing that should be used:

- Functional Testing
- Performance Testing

1. It is necessary to determine which test cases is planned to automate, and which are not, and for what product components
2. You must define the technology that will be used in the project

5.2.1 Tool selection

For automated testing, it is essential to select a tool before we can use it. The requirements for test automation tools are listed below:

1. Free
2. Cross platform run
3. Big community
4. Easy to learn
5. The possibility of using continuous integration (need of Jenkins or TeamCity)
6. Ability to develop test code in a modern IDE (JetBrains IntelliJ IDEA)
7. Accuracy of emulating user actions in the browser

The tool we need must meet all these requirements. It is necessary to look more closely at some of the popular test automation tools to know if they support those requirements.

5.2.2 HP QuickTest Professional

HP Unified Functional Testing is the primary tool for automating functional testing of HP (HP Quick Test Professional). This tool can automate functional and regression tests by recording user actions while working with the application under test, and then executes the recorded actions to verify the functionality of the software.

The recorded actions saved as scripts. Scripts can be displayed in the tool such as VBScript (expert view) or as visual sequential steps with actions (keyword view). Each step can be edited and checkpoints can be added to it, which compare the expected result with the exact result. [12]

This tool is not suitable because of the following points:

1. It is not free
2. Cross platform run – only Windows
3. Ability to develop test code in a modern IDE – Visual Studio

5.2.3 IBM Rational Functional Tester

It is also a paid tool.

Rational Functional Tester provides testers automated testing tools that let you perform regression testing, functional testing, data-driven testing and user interface testing.

Using IBM Rational Functional Tester as a management tool, the testing process divided into three phases:

- Recording. The test script written "on the fly" as the user works with the application. You can also insert verification points to test the system's response and make test scripts dependent on the data to execute the same script with different sets of input data
- Improvement. Adding code that performs a variety of functions. Typical changes to test scenarios are conditional branching, refactoring, and exception handling
- Reproduction. Running scripts that emulate actions that the user of the application performed while writing the test. Discrepancies are recorded, and the

tester can conclude whether the application works well or regression testing has identified problems [13]

This tool is not suitable because of the following points:

1. It is not free
2. Cross platform run – recorder not supported on Linux

5.2.4 AutoIt

AutoIt v3 is a free scripting language designed to automate the Windows GUI and create general-purpose scripts. It uses a variety of simulation operations involving the use of keys, mouse movements and window / control elements to automate tasks in cases where the use of other languages is impossible or unreliable. AutoIt has a small size and does not require the presence of any third-party libraries in the system.

AutoIt was designed to be as autonomous as possible, not using external dll files or the registry, so that its use was safe on servers. Scripts can be compiled into stand-alone programs using Aut2Exe.

AutoIt uses syntax similar to BASIC syntax, which means that most people who have ever written programs in high-level languages will easily understand it. [14]

Simple test has been created (Figure 3):

```
Run("C:\... \chrome.exe --start https://cloud.3dprinteros.com", "")
Sleep(5000)
Send('sample@gmail.com')
Send('{TAB}')
Send('password')
Send('{ENTER}')
_ChromeShutdown()
```

Figure 3. AutoIt simple test

Steps:

- Run – Launch browser and open page https://cloud.3dprinteros.com
- Sleep(5000) – Timeout 5 second is needed because
- Send('sample@gmail.com') – Write in first text box on page email
- Send('{TAB}') – emulates pressing the TAB button on your keyboard

- Send('password') – Write in first text box on page ‘password’
- Send('{ENTER}') – emulates pressing the ENTER button on your keyboard

This tool is not suitable because of the following points:

- Cross platform run – only windows
- Can’t work in Intellij Idea

This automation tool is pretty simple to use and is free, but it was not selected for testing, because the browser Google Chrome is poorly supported. This tool can be well tested in Internet Explorer and Opera (old version 9.2), but it is not suitable for our system. This tool can work only on windows.

5.2.5 Selenium

Selenium is a free automation tool from the Company OpenQA.org. It’s a set of multiple tools, each of which assumes its own approach in test automation. In aggregate Selenium, tool set provides a rich set of capabilities specifically assembled for testing all types of Web applications. One of the key features of Selenium is the ability to run the same test in different browsers. It supports different operating systems: Mac OS, Microsoft Windows, and Linux.

«Selenium WebDriver» and «Selenium IDE» was used in this project, is necessary to tell more about them:

- Selenium 2 (Selenium WebDriver) - a software library to control the browser. The short name WebDriver is also often used. Sometimes it said that this is a "browser driver", but in fact, it is a whole family of drivers for different browsers, as well as a set of client libraries in different languages, allowing you to work with these drivers. WebDriver supports the following browsers and operating systems: Google Chrome; Internet Explorer; Firefox; Opera; HtmlUnit; Android; iOS 3+ to 3.2+ phones and tablet
- Selenium IDE is a plug-in to the Firefox browser that can record user actions, play them, and generate code for WebDriver or Selenium RC, in which the same actions are performed. In general, this is a "Selenium recorder". Testers who do not know how (or do not want to) program, use Selenium IDE as an independent product, without converting the recorded scripts into program code [9]

Selenium is suitable for all requirements. That is why Selenium has been selected. For this project, it was required as quick as possible to learn how to test and to write automated tests. «Selenium» is suitable for novice testers by its simplicity. It is important that this automation tool is free. It can be configured for almost any browser and even mobile phones. In addition, criterion was the fact that it supports Linux and Microsoft Windows.

5.3 Tools used for testing

In this project was used Selenium WebDriver, which was described above. To automate the project was used build Maven. Description from the official site: "Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.“ [15]

5.3.1 JUnit 4 Vs TestNG – Comparison

Also, was used Framework TestNG. Test Java Framework was the choice of: JUnit4 or TestNG. Because they both provide good performance and simple in the study. Below, you can see the difference between them. This table taken from website Mkyong [16] (Table 5):

Table 5. JUnit 4 Vs TestNG Comparison

Feature	**JUnit 4**	**TestNG**
test annotation	@Test	@Test
run before all tests in this suite have run	—	@BeforeSuite
run after all tests in this suite have run	—	@AfterSuite
run before the test	—	@BeforeTest
run after the test	—	@AfterTest
run before the first test method that belongs to any of these groups is invoked	—	@BeforeGroups
run after the last test method that belongs to any of these groups is invoked	—	@AfterGroups

run before the first test method in the current class is invoked	@BeforeClass	@BeforeClass
run after all the test methods in the current class have been run	@AfterClass	@AfterClass
run before each test method	@Before	@BeforeMethod
run after each test method	@After	@AfterMethod
ignore test	@ignore	@Test(enbale=false)
expected exception	@Test(expected = ArithmeticException.class)	@Test(expectedExceptions = ArithmeticException.class)
timeout	@Test(timeout = 1000)	@Test(timeout = 1000)

JUnit 4 is considered to be faster than TestNG, but in this table can be seen that in TestNG more functionality. The downside TestNG is that the TestSuite is necessary to create in the XML file. Criterion for choosing was the fact that that the system of priorities is supported in TestNG and @BeforeTest @AfterTest. “In JUnit 4, we have to declare “@BeforeClass” and “@AfterClass” method as static method. TestNG is more flexible in method declaration; it does not have these constraints.” [16]

5.4 Examples of automated tests

In this thesis work, in front of me there was a task to implement automated testing for «3dprinteros» system. The tests are designed for checking the basic functionality that is used frequently, such as authorization, all the buttons and headers.

5.4.1 3DPrinterOS Client Testing

Since the Client runs in the browser, all the functionality was tested with Selenium tests. There were tested (Figure 4):

- Test TestFullWizard (Appendix 1) – Check "Wizard" for all buttons and texts
- Log In page – on the login page was checked that all the buttons are working and the email validation is working correctly
- Log out – The webpage and software is closing correctly after exiting the system

- Test TestSetupCamera, TestRestartCamera: The camera has been tested only on the change of the radio buttons. Since the camera is working or not, can only be checked visually
- Button „Go to your account now" testing that new 3DPrinterOS page is opened with your account
- Test TestDetectNetworkPrinters verifies that the correct text is given, if the printer is connected
- Test TestEnableDisableCloudSync verify that Enable CloudSync button text changes correctly
- Test TestRestoreDefaultSettings – checks, that the new window is opened with the correct text
- Test TestSkipWizard – verifies that the Skip button brings us back to the main page
- Test TestViewLogs – verifies that the View Logs button opens a page with logs and a button

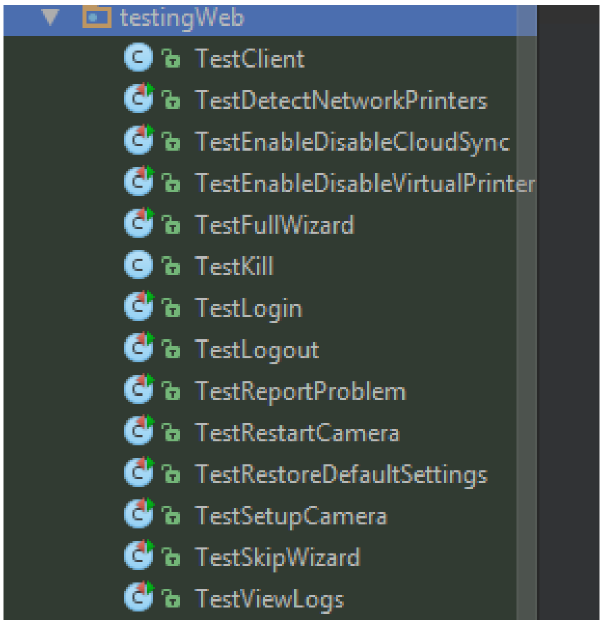

testingWeb
TestClient
TestDetectNetworkPrinters
TestEnableDisableCloudSync
TestEnableDisableVirtualPrinter
TestFullWizard
TestKill
TestLogin
TestLogout
TestReportProblem
TestRestartCamera
TestRestoreDefaultSettings
TestSetupCamera
TestSkipWizard
TestViewLogs

Figure 4. Testing Client

For the "Web Client" 12 tests were done. We have tests for all buttons and main functions of the client. Time to run these tests are spent very little, only 3 minutes. More time is spent on re-check all elements and rewrite test if something has changed.

5.4.2 3DPrinterOS Cloud Testing

All tests were originally created in the Selenium IDE and looked like this in the browser (Table 6, 7, 8):

Table 6. Example of the test SignIn

	testSignIn		
1	open	/sign	
2	type	id=signinUsername	sample@email.com
3	type	id=signupPassword	sample password
4	clickAndWait	name=signIn	
5	click	css=#uploadfiles_3dos > a.close	

Steps in table:

1. Open webpage https://cloud.3dprinteros.com/sign/
2. Type in first text box email: sample@email.com
3. Type in second text box sample password
4. Click button Sign In and wait for page to open
5. Click button Close in popup

This test checks the user's authorization by repeating his steps.

Table 7. Example of the test Dashboard

	TestDashboard		
1	open	/dashboard/	
2	clickAndWait	link=World statistic	
3	verifyElementPresent	link=World statistics	
4	verifyElementPresent	link=My printing history	
5	verifyText	css=h4	Overview

Steps in table:

1. Open webpage https://cloud.3dprinteros.com/dashboard/
2. Click menu link World statistic and wait for page to open
3. Verify if link World statistics present
4. Verify if link My printing history present
5. Verify if header text is Overview

This test verifies that the Dashboard menu item opens the desired page and it has all the necessary links, and their text is right.

Table 8. Example of the test ProfileOverview

	TestProfileOverview		
1	open	/myfiles/	
2	click	id=menuitem_account	
3	verifyText	css=div.head_label	Profile overview
4	verifyText	css=div.info_block > div	exact:License: FREE
5	waitForText	//div[@id='user_submenu']/div[2]/div[2]	Storage available: 10 GB
6	verifyText	//div[@id='user_submenu']/div[2]/div[2]	Storage available: 10 GB
7	verifyText	//div[@id='user_submenu']/div[2]/div[3]	Bandwidth available: 10 GB
8	verifyText	link=Profile Settings	Profile Settings
9	verifyText	link=Change Password	Change Password
10	verifyText	link=Sign out	Sign out

Steps in table:

1. Open webpage https://cloud.3dprinteros.com/myfiles/
2. Click link Profile Overview text
3. Check if header text is Profile overview
4. Check if License: FREE
5. Wait for text present Storage available: 10 GB

6. Verify if text in block with xpath //div[@id='user_submenu']/div[2]/div[2] is Storage available: 10 GB
7. Verify if text in block with xpath //div[@id='user_submenu']/div[2]/div[3] is Bandwidth available: 10 GB
8. Verify if link text Profile Settings is present
9. Verify if link text Password is present
10. Verify if link text Check Sign out is present

This test verifies that the menu item opens the Profile Overview page and it has all the necessary links, and their text is right.

Selenium WebDriver test examples

If we export from Selenium IDE testSignIn we will find this code (Figure 5):

```
@Test
public void testSignIn() throws Exception {
   driver.get(baseUrl + "/sign/");
   driver.findElement(By.id("signinUsername")).sendKeys(username);
   driver.findElement(By.id("signinPassword")).sendKeys(password);
   driver.findElement(By.name("signIn")).click();
   driver.findElement(By.cssSelector("#uploadfiles_3dos >
a.close")).click();
}
```

Figure 5. Selenium Test Sign In

In the code presented in Appendix 2 you can see how much code we've got, although the test is quite small.

As a result, with small changes, we have a code like this (Figure 6) and full version in (Appendix 3):

```
@Test
public void testSignIn () {
      webDriver.get(BaseUrl);

      getElement(By.id("signinUsername")).sendKeys(Username);

      getElement(By.id("signinPassword")).sendKeys(Password);

      getElement(By.name("signIn")).click();

      getElement(By.cssSelector("#uploadfiles_3dos > a.close")).click();

}
```

Figure 6. Selenium changed Test Sign In

If these 2 tests compare, we can see that a variety of methods was removed, repeated in each generated test. As a result, the test is simpler and occupies less space.

13 tests have been created covering verification of all links, buttons, and text captions on 3DPrinterOs Cloud webpage. Tests (Figure 7):

- TestActivePrinters, TestInactivePrinters – verify that the button "Printers" works and opens the desired page. All the buttons and headers on the page are right
- TestClientDownload – verify that the link to the client download file with the correct size
- TestDashboard – verify that the button "Dashboard" works and opens the desired page. All the buttons and headers on the page are right
- TestLiveViewWall – verify that the page exists and writes "No connected cameras"
- TestLogin – user authorization check
- TestMyFilesButtons – verify that the button "My Files" works and opens the desired page. All the buttons and headers on the page are right
- TestMyProject – verify that the button "My Files" works and opens the desired page. Popup „Add new files" opens and all the buttons are right
- TestProfileOverview – verify that the button "Profile Overview" works and opens the desired page. All text on the page is correct
- TestSearch – check that all links on "Search" page are working and desired page opens correctly
- TestSearchLinks – search validation is checked
- TestSettingsUserId – check that the user settings open and buttons on this page works correctly
- TestSignInOut – user authorization and verification check
- TestUploadFiles – verification of all buttons and headers on the "Upload Files" page

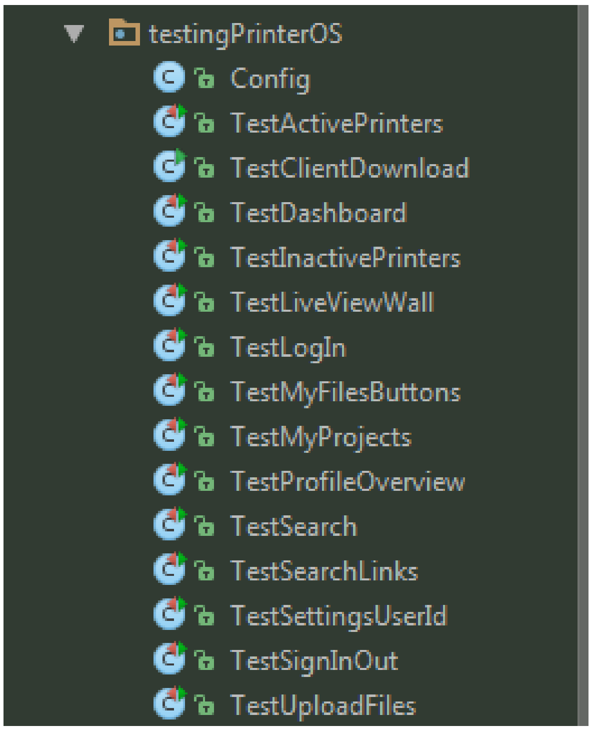

testingPrinterOS
Config
TestActivePrinters
TestClientDownload
TestDashboard
TestInactivePrinters
TestLiveViewWall
TestLogIn
TestMyFilesButtons
TestMyProjects
TestProfileOverview
TestSearch
TestSearchLinks
TestSettingsUserId
TestSignInOut
TestUploadFiles

Figure 7. 3DPrinterOS Tests

Thereby, we have tests for all buttons and main functions of the site and the client. Tests using mail, various files for changes or require printing is not automated, because they require visual inspection, to say exactly whether they are working properly.

5.5 Problems and bugs

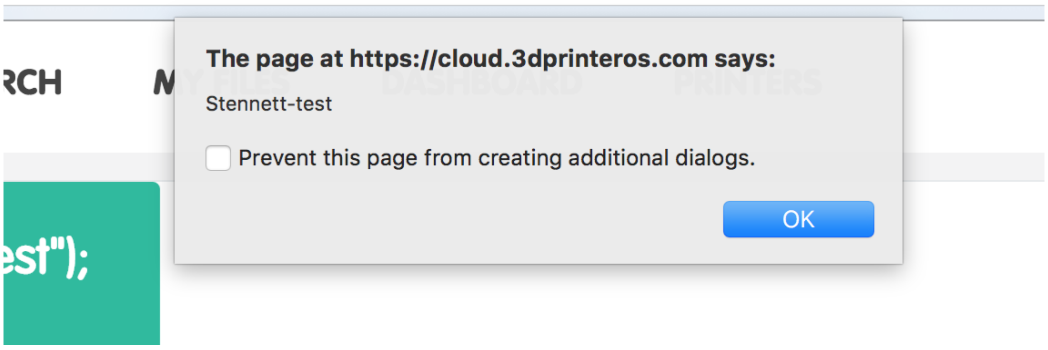
Sometimes tests can produce an error simply because they have broken down. Quite often we have to figure out why they broke. This happens because of the fact that the project is updated periodically which means functionality, css and buttons names changes too.

Quite a big problem, when as a result of testing is an issue which seems to be little and fix is not difficult, but none of the developers did not want to correct such minor problems particularly associated with the GUI. This is a communication problem with the developer.

We must not forget that name of elements on the page may change. When you create a test, you have to remember to refactor code for better maintenance in the future. And perhaps, the elements need to be renamed in the great number of tests. To avoid such problems Web Elements can be taken in a separate list and later change them there.

The great number of errors was found on XSS attacks. XSS is the abbreviation of the term decrypted as "cross-site scripting". In the actions of cross-site scripting and the main goal of XSS is the obtaining of user's cookies through the HTTP page of the HTML page.

Unlike SQL injection, this type of attack, on the one hand is safe for servers and is dangerous for site users. On the other hand, if you steal admin cookies, then you will get access to the administration panel. The hacker will have more chances to get to the database. [17] There were no checks on XSS injection in almost every text box. It was possible to save this code: <script>alert("test");</script> and receive a message on the screen (Figure 8).



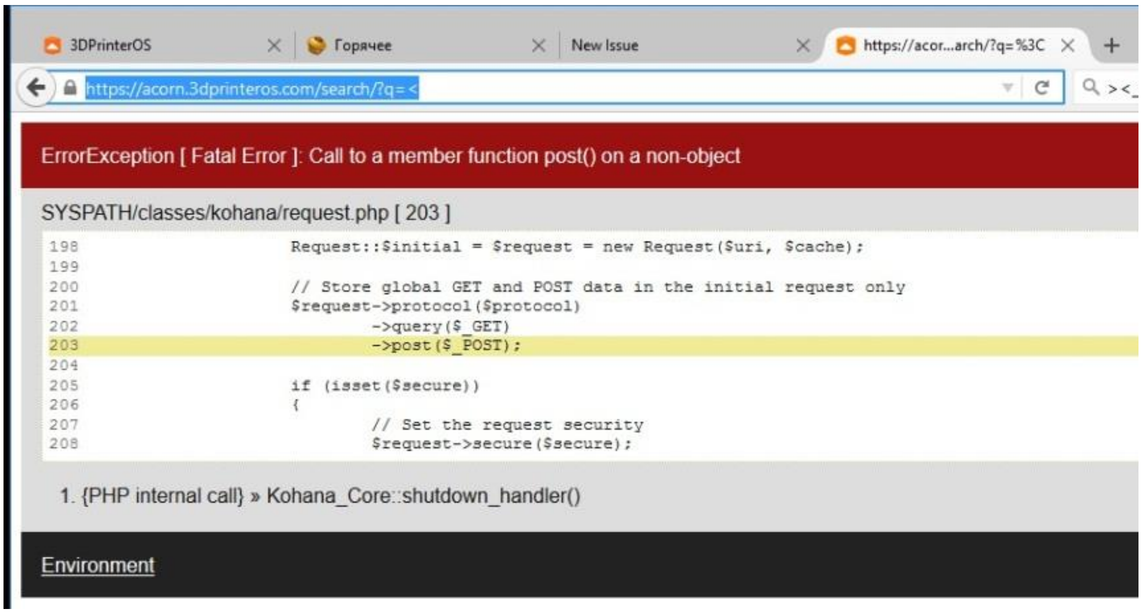
The page at https://cloud.3dprinteros.com says:

Stennett-test

☐ Prevent this page from creating additional dialogs.

Figure 8. XSS alert

Also during the “Search” testing was found error – Internal Server Error (Figure 9).



3DPrinterOS
Горячее
New Issue
https://acor...arch/?q=%3C
https://acorn.3dprinteros.com/search/?q=<

ErrorException [Fatal Error]: Call to a member function post() on a non-object

SYSPATH/classes/kohana/request.php [203]

```
		Request::$initial = $request = new Request($uri, $cache);

		// Store global GET and POST data in the initial request only
		$request->protocol($protocol)
			->query($_GET)
			->post($_POST);

		if (isset($secure))
		{
			// Set the request security
			$request->secure($secure);
```

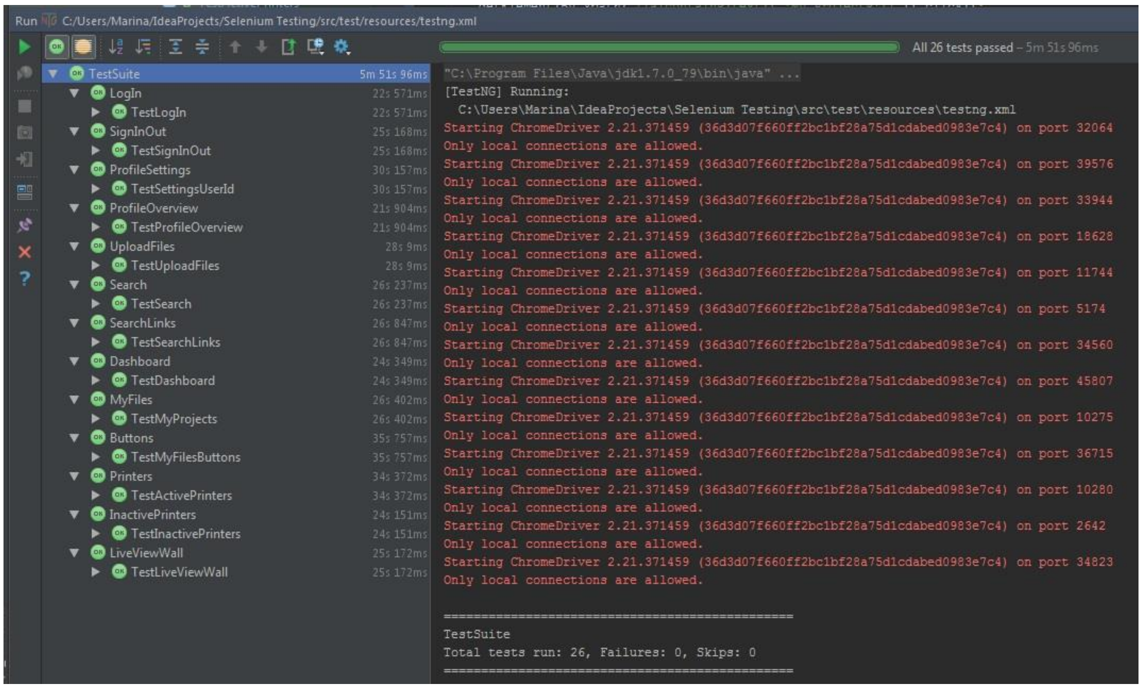
1. {PHP internal call} » Kohana_Core::shutdown_handler()

Figure 9. Internal Server Error

To find it, you had to write symbols «> <_ |?"» in a search textbox. After that, the site stopped working for a short period.

5.6 How much time is spent

As a result, we can say that on the complete Cloud system testing with automatic tests spent 6 minutes to run and pass all the tests (Figure 10) and approximately 20 - 30 minutes to change all tests. Changes made by developers in recent times were tested in about an hour.


Run C:/Users/Marina/IdeaProjects/Selenium Testing/src/test/resources/testng.xml
All 26 tests passed – 5m 51s 96ms

```
TestSuite                    5m 51s 96ms
  LogIn                      22s 571ms
    TestLogIn                22s 571ms
  SignInOut                  25s 168ms
    TestSignInOut            25s 168ms
  ProfileSettings            30s 157ms
    TestSettingsUserId       30s 157ms
  ProfileOverview            21s 904ms
    TestProfileOverview      21s 904ms
  UploadFiles                28s 9ms
    TestUploadFiles          28s 9ms
  Search                     26s 237ms
    TestSearch               26s 237ms
  SearchLinks                26s 847ms
    TestSearchLinks          26s 847ms
  Dashboard                  24s 349ms
    TestDashboard            24s 349ms
  MyFiles                    26s 402ms
    TestMyProjects           26s 402ms
  Buttons                    35s 757ms
    TestMyFilesButtons       35s 757ms
  Printers                   34s 372ms
    TestActivePrinters       34s 372ms
  InactivePrinters           24s 151ms
    TestInactivePrinters     24s 151ms
  LiveViewWall               25s 172ms
    TestLiveViewWall         25s 172ms
```

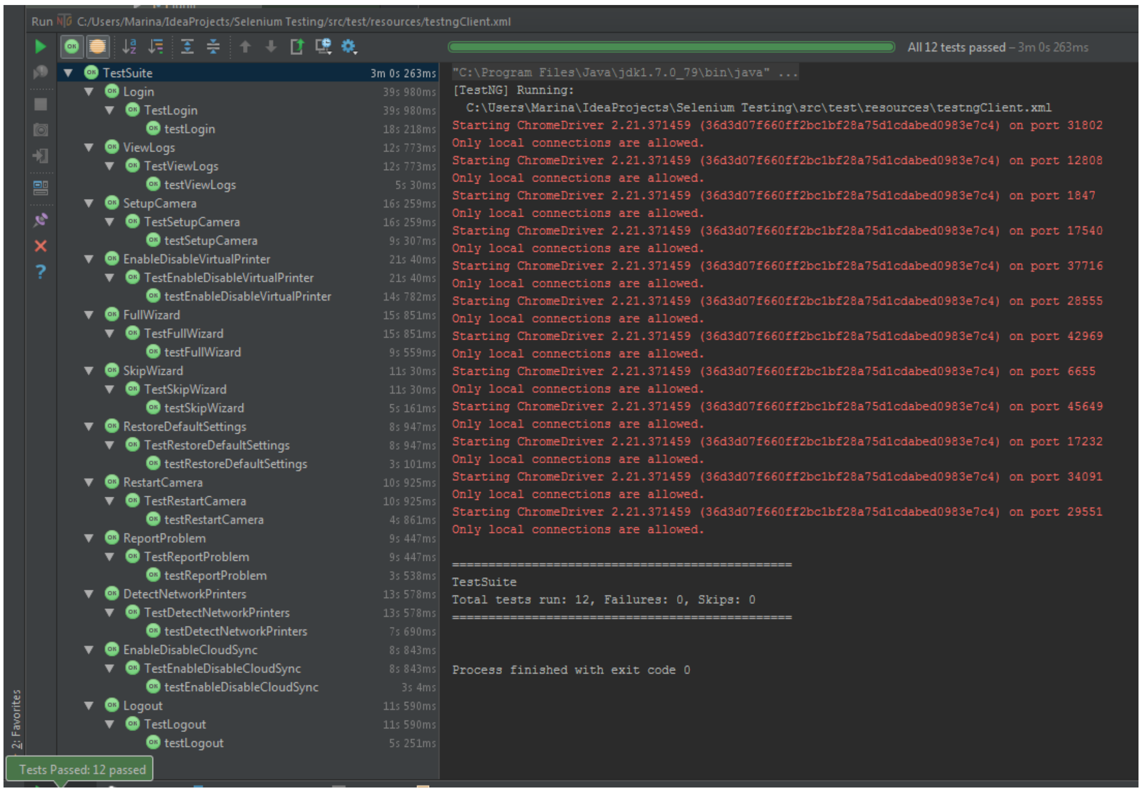
```
"C:\Program Files\Java\jdk1.7.0_79\bin\java" ...
[TestNG] Running:
  C:\Users\Marina\IdeaProjects\Selenium Testing\src\test\resources\testng.xml
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 32064
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 39576
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 33944
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 18628
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 11744
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 5174
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 34560
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 45807
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 10275
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 36715
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 10280
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 2642
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 34823
Only local connections are allowed.

===============================================
TestSuite
Total tests run: 26, Failures: 0, Skips: 0
===============================================
```

Figure 10. Cloud Auto Testing Time

Total tests for Cloud webpage were made 13, but 26 turned out, because for every test also the test of authorizing was started. This was done in order to be able parallelize the test launch in the future and reduce the time of their passage even more.

A total of 12 tests were done for the Client. As a result, we can say that on the complete Client testing with automatic tests spent 3 minutes to run and pass all the tests (Figure 11) and approximately 20 - 30 minutes to change all tests. Changes made by developers in recent times were tested in about half an hour.



```
Run C:/Users/Marina/IdeaProjects/Selenium Testing/src/test/resources/testngClient.xml
All 12 tests passed – 3m 0s 263ms

TestSuite                                  3m 0s 263ms
  Login                                    39s 980ms
    TestLogin                              39s 980ms
      testLogin                            18s 218ms
  ViewLogs                                 12s 773ms
    TestViewLogs                           12s 773ms
      testViewLogs                          5s 30ms
  SetupCamera                              16s 259ms
    TestSetupCamera                        16s 259ms
      testSetupCamera                       9s 307ms
  EnableDisableVirtualPrinter              21s 40ms
    TestEnableDisableVirtualPrinter        21s 40ms
      testEnableDisableVirtualPrinter      14s 782ms
  FullWizard                               15s 851ms
    TestFullWizard                         15s 851ms
      testFullWizard                        9s 559ms
  SkipWizard                               11s 30ms
    TestSkipWizard                         11s 30ms
      testSkipWizard                        5s 161ms
  RestoreDefaultSettings                    8s 947ms
    TestRestoreDefaultSettings              8s 947ms
      testRestoreDefaultSettings            3s 101ms
  RestartCamera                            10s 925ms
    TestRestartCamera                      10s 925ms
      testRestartCamera                     4s 861ms
  ReportProblem                             9s 447ms
    TestReportProblem                       9s 447ms
      testReportProblem                     3s 538ms
  DetectNetworkPrinters                    13s 578ms
    TestDetectNetworkPrinters              13s 578ms
      testDetectNetworkPrinters             7s 690ms
  EnableDisableCloudSync                    8s 843ms
    TestEnableDisableCloudSync              8s 843ms
      testEnableDisableCloudSync            3s 4ms
  Logout                                   11s 590ms
    TestLogout                             11s 590ms
      testLogout                            5s 251ms

Tests Passed: 12 passed
```

```
"C:\Program Files\Java\jdk1.7.0_79\bin\java" ...
[TestNG] Running:
  C:\Users\Marina\IdeaProjects\Selenium Testing\src\test\resources\testngClient.xml
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 31802
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 12808
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 1847
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 17540
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 37716
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 28555
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 42969
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 6655
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 45649
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 17232
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 34091
Only local connections are allowed.
Starting ChromeDriver 2.21.371459 (36d3d07f660ff2bc1bf28a75d1cdabed0983e7c4) on port 29551
Only local connections are allowed.

===============================================
TestSuite
Total tests run: 12, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

Figure 11. Client Auto Testing Time

We have tests for all buttons and main functions of the site and the client.

In summary, it is possible to say that testing went well. Many problems were found, both with an interface and with a base functionality. Time spent on basic functionality testing; both the client and the website has been reduced. Automatic tests helped simplify a testing process and get rid of the routine repetition of everyday tests by tester.

However, it is not worth to give up on the manual testing. Manual testing helps to identify the problems of logic. Automated testing helps to note problems in the recently running functional and correct them.

6 Summary

Manual and automatic testing should be used together. Because we will get the most completely tested and better product. Manual testing helps to identify the problems of logic. On the other hand, tester may not notice some small changes that will record test.

Testing is not only comparison of predefined test steps and comparison of results, but also visual control. Therefore, for more complete testing, human intelligence is also needed.

My goal was to test the project as much as possible and cover the product code with automatic tests. To do this, first was required to study the product under development, and to understand what kind of functionality will be used on a daily basis. Most of the buttons links and headers were checked using automated tests. All that is needed to be tested visually has been tested by hand. For example, the camera and printing were tested manually.

In this system pretty much need to be tested, but you cannot automate everything. Many things can only be checked visually and manually. A large number of tests have been created covering verification of all links, buttons, and text captions on 3DPrinterOs Cloud webpage and 3DPrinterOS Client. As a result, we can say that the client manual testing spent about 2-3 hours, and a complete testing system Cloud spent 3 - 5 hours.

On the complete Cloud system, testing with automatic tests spent 6 minutes to run and pass all the tests and approximately 20-30 minutes to change all tests. Changes made by developers in recent times were tested in about an hour. On the complete Client testing with automatic tests was spent 3 minutes to run and pass all the tests and approximately 20-30 minutes to change all tests. Changes made by developers in recent times were tested in about half an hour.

In this work was also discussed, what problems occur when you test 3D print.

When you print on 3D printer, you need to consider many factors, such as the heating temperature of the table, if it is heated, or the environment. During printing, the model may come unstuck from the bed, because of the fact that the first print layer is not glued.

It was considered, how to choose instruments for testing, the studied methodologies and types of tests are described.

In summary, it is possible to say that testing went well. Many problems were found, both with an interface and with a base functionality. Testing helped to simplify a testing process and get rid of the routine repetition of everyday tests by tester.

Time spent on basic functionality testing, both the client and the website reduced.

References

[1] H. L. Grippa, "Техносфера," 2006. [Online]. Available: http://tekhnosfera.com/avtomatizatsiya-testirovaniya-programmnyh-prilozheniy-metodom-klyuchevyh-sostoyaniy. [Accessed 18 March 2016].

[2] "3dprinteros," [Online]. Available: https://www.3dprinteros.com/how-it-works/. [Accessed 17 March 2016].

[3] "Github," [Online]. Available: https://github.com/3dprinteros/3dprinteros-client. [Accessed 04 March 2016].

[4] "Qa-helper," [Online]. Available: http://qa-helper.com/agile-scrum-kanban-xp/. [Accessed 08 April 2016].

[5] "Tutorialspoint," [Online]. Available: http://www.tutorialspoint.com/software_testing/software_testing_types.htm. [Accessed 20 March 2016].

[6] "Tutorialspoint," [Online]. Available: http://www.tutorialspoint.com/software_testing/software_testing_methods.htm. [Accessed 21 March 2016].

[7] "Softwaretestingfundamentals," [Online]. Available: http://softwaretestingfundamentals.com/software-testing-levels/. [Accessed 22 March 2016].

[8] J. F. H. Q. N. Cem Kaner, Testing Computer Software, Second Edition, Canada: Publishing by John Wiley & Sons, Inc, 1999.

[9] "Selenium," [Online]. Available: http://www.seleniumhq.org/docs/01_introducing_selenium.jsp#introducing-selenium. [Accessed 25 March 2016].

[10] "Testing excellence," [Online]. Available: http://www.testingexcellence.com/test-automation-advantages-and-disadvantages/. [Accessed 06 April 2016].

[11] "Protesting," [Online]. Available: http://www.protesting.ru/automation/functional/whytoauto.html. [Accessed 26 March 2016].

[12] "Automation consultants," [Online]. Available: http://www.automation-consultants.com/index.php/products/hp-products/hp-unified-functional-testing-quick-test-professional. [Accessed 09 April 2016].

[13] "Automation consultants," [Online]. Available: http://www.automation-consultants.com/index.php/products/ibm-products/rational-functional-tester. [Accessed 09 April 2016].

[14] "Autoitscript," [Online]. Available: https://www.autoitscript.com/site/autoit/. [Accessed 05 April 2016].

[15] "Maven," [Online]. Available: https://maven.apache.org. [Accessed 20 April 2016].

[16] "Mkyong," [Online]. Available: www.mkyong.com/unittest/junit-4-vs-testng-comparison. [Accessed 15 April 2016].

[17] "Acunetix," [Online]. Available: http://www.acunetix.com/websitesecurity/cross-site-scripting/. [Accessed 05 April 2016].

Appendix 1 – Test Wizard

```
public class TestFullWizard extends TestClient {
@Test
public void testFullWizard() throws Exception {
   webDriver.get(BASE_URL_CLIENT + "/");
   getElement(By.xpath("//input[@value='Start wizard']")).click();
   for (int second = 0; ; second++) {
      if (second >= 60) fail("timeout");
        try {
             if("Welcome!".equals(getElement(By.cssSelector("h3")).
             getText())) break;
            } catch (Exception e) {}Thread.sleep(1000);
       }
       assertTrue(getElement(By.cssSelector("center> div")).
             getText().contains("Welcome!"));
       getElement(By.xpath("//input[@value='Next']")).click();
       assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Connect your printer");
       assertTrue(getElement(By.cssSelector(
             "div>form> input.middle_button")).getAttribute("value").
             contains("virtual printer"));
       assertEquals(getElement(By.xpath("//div[@id='printers_info']/p/b")).
             getText(), "No printers detected");
       getElement(By.xpath("//input[@value='Next']")).click();
       assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Select your printer type");
       getElement(By.xpath("//input[@value='Next']")).click();
       assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Select live view mode");
       assertEquals(getElement(By.cssSelector("div.checkbox_div > p")).
             getText(), "Dual camera");
       assertEquals(getElement(By.xpath("//form[@id='switch_camera']/div/p[2]
       ")).getText(), "HD camera");
       assertEquals(getElement(By.xpath("//form[@id='switch_camera']/div/p[3]
       ")).getText(), "Multi camera");
         assertEquals(getElement(
             By.cssSelector("font[title=\"Current live view mode\"]")).
             getText(), "Disable camera");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Using the 3DPrinterOS Tray icon");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Congratulations!");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Renaming your printer");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Sharing access with Workgroups");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "CloudSync");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Restore default settings");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Reporting problems");
         getElement(By.xpath("//input[@value='Next']")).click();
         assertEquals(getElement(By.cssSelector("h3")).
             getText(), "Happy Printing!");
         getElement(By.xpath("//input[@value='Complete']")).click();
         assertEquals(getElement(By.xpath("//h3[2]")).
             getText(), "Printers connected to\ncloud.3dprinteros.com");
    }
}
```

Appendix 2 – Test SignIn

```
public class Test {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @BeforeClass(alwaysRun = true)
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "https://cloud.3dprinteros.com/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }
  @Test
  public void test() throws Exception {
    driver.get(baseUrl + "/");
    driver.findElement(By.id("signinUsername")).clear();
    driver.findElement(By.id("signinUsername")).sendKeys("sample@email.com");
    driver.findElement(By.id("signupPassword")).clear();
    driver.findElement(By.id("signupPassword")).sendKeys("sample password");
    driver.findElement(By.name("signIn")).click();
    driver.findElement(By.cssSelector("#uploadfiles_3dos >a.close")).click();
  }

  @AfterClass(alwaysRun = true)
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }
  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }
  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }
  private String closeAlertAndGetItsText() {
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
    } finally {
      acceptNextAlert = true;
    }
  }
}
```

Appendix 3 – Refactored Test SignIn

```
public class TestLogIn {
    public WebDriver webDriver;

    @BeforeTest
    public void beforeTest() {
        WebDriverFactory.startBrowser(true);
        webDriver = WebDriverFactory.getDriver();
    }
    @Test()
    public void signIn() {
        this.signInWithParameters(BASE_URL_WEB, USER_NAME, PASSWORD);
    }
    public void signInWithParameters(String BaseUrl,
       String Username, String  Password) {
        webDriver.get(BaseUrl);
        getElement(By.id("signinUsername")).sendKeys(Username);
        getElement(By.id("signinPassword")).sendKeys(Password);
        getElement(By.name("signIn")).click();
        getElement(By.cssSelector("#uploadfiles_3dos > a.close")).click();
    }
    @AfterClass
    public void afterClass() {
        WebDriverFactory.finishBrowser();
    }
    public boolean isElementPresent(By by) {
        try {
            getElement(by);
            return true;
        } catch (NoSuchElementException e) {
            return false;
        }
    }
    public WebElement getElement(By by){
        return webDriver.findElement(by);
    }
}
```