Bachelor's thesis

Information and Communications Technology

2019

Markus Virtanen

# APPLYING TEST AUTOMATION IN USABILITY TESTING

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2019 | 31 pages, 5 pages in appendices

Markus Virtanen

# APPLYING TEST AUTOMATION IN USABILITY TESTING

The thesis studies the applicability of test automation in the context of usability testing. The first part is a brief introduction to testing in relation to a project and establishes the baseline from where to start any testing effort. The increased amount of testing required in modern software development cycles and especially the conflict between need for better usability and need for more test automation.

Usability is seen most often as a human driven effort, since it is difficult to truly pinpoint all aspects of usability without involving the people who would be using the product. In many cases the different understanding of the task at hand between the developer and end user ends in a creation of a tool that nobody wants to use. This becomes especially important in a modern market where any product has a virtually unlimited amount of competition and customers are quick to go for another product if the first one does not invest in usability.

Test automation is nowadays almost a pre-requisite for any software development. Because of the modern short development cycles that mean bringing smaller updates to software all the more often, manual testing is simply not fast enough. The modern solution is to use test automation for regression testing, but quite often testing ends up focusing only on functional testing. Test automation is simply not seen capable of simulating users adequately well, or repeatable usability testing is not seen as valuable.

For the purposes of comparison, a set of automated test cases was written to evaluate usability. The frame in which the work was conducted was based in Jakob Nielsen's research on usability heuristics, specifically the heuristic "consistency and standards" was chosen for the evaluation. Then a more traditional usability interview was conducted on four users using the same test object.

The results were compared and although not perfectly aligned, the test automation did seem to yield meaningful results in the context. The difference could be easily taken into account by updating the test automation script and thus with relatively little effort reduce the amount of user testing required.

KEYWORDS:

Testing, Test automation, Usability, Usability heuristics, Robot Framework

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikan insinööri

2019 | 31 sivua, 5 liitesivua

Markus Virtanen

# TESTIAUTOMAATION SOVELTAMINEN KÄYTETTÄVYYSTESTAUKSEEN

Tämä opinnäytetyö tutkii testiautomaation soveltuvuutta käytettävyystestauksessa. Aluksi työssä esitellään testausta yleisesti, liittäen testaus mittakaavaan projektiympäristössä ja antaen aloituspiste testaukseen. Modernissa sovelluskehitysympäristössä testauksen tarve on jatkuvasti kasvava, johtuen yhä nopeammista kehityssykleistä. Erityisesti se luo konfliktin paremman käytettävyyden ja testiautomaation välille.

Käytettävyys on yleisimmin nähty ihmisvetoisena osa-alueena, sillä kaikkien potentiaalisten käytettävyysongelmien löytäminen ilman oikeiden käyttäjien osallistamista on hankalaa. Useissa tapauksissa eri näkemys suoritettavasta tehtävästä työkalun kehittäjän ja käyttäjän välillä johtaa tilanteeseen, jossa käyttäjä ei koe hyötyvänsä työkalusta. Erityisen tärkeäksi käytettävyys on muuttunut nykyään, kun jokaisella sovelluksella on näennäisesti rajaton määrä kilpailua ja asiakkaat jättävät nopeasti palvelut, jotka eivät vastaa heidän näkemyksiään käytettävyydestä.

Testiautomaatio on nykyään lähestulkoon esivaatimus mille tahansa sovelluskehitykselle. Johtuen nopeista kehityssykleistä, tarkoittaen yhä useammin tehtäviä pieniä päivityksiä, manuaalinen testaus ei yksinkertaisesti ole tehtävissä tarpeeksi nopeasti. Moderni ratkaisu on käyttää testiautomaatiota regressiotestaukseen, mutta usein keskittyen vain toiminnallisiin osa-alueisiin sovelluksessa. Testiautomaatiota ei pidetä kykenevänä käyttäjien simuloimiseen tarpeeksi hyvin tai käytettävyystestien uudelleenkäytöllä ei nähdä olevan riittävästi arvoa.

Vertailun vuoksi luotiin prototyypinomainen kokoelma automaattisia testejä käytettävyyden arvioimiseksi. Teoriapohjana tähän käytettiin Jakob Nielsenin tutkimusta käytettävyyden heuristiikkojen osalta, erityisesti pohjaten ”johdonmukaisuus ja standardit” -heuristiikkaan. Lisäksi perinteisempi käytettävyystesti toteutettiin neljällä käyttäjällä testaten samaa testikohdetta.

Kokonaisuutena tulokset näyttivät onnistuneesti testiautomaation olevan toimiva tapa arvioida käytettävyyttä. Vaikka pieniä eroavaisuuksia tuloksissa esiintyikin, ne olivat selvästi testisuunnittelun puutteita. Testiautomaation erityisinä etuina perinteiseen testaukseen verrattuna näkyi uudelleenkäytettävyys ja verrattain pieni tarvittu työmäärä automaation luomiseksi.

ASIASANAT:

Testaus, Testiautomaatio, Käytettävyys, Käytettävyys heuristiikka, Robot Framework

# CONTENTS



# APPENDICES

# PICTURES

# LIST OF ABBREVIATIONS (OR) SYMBOLS

| Abbreviation | Explanation |
|---|---|
| Agile | Software development method aimed to produce valuable product faster and in shorter intervals (Agile Manifesto 2019) |
| ISO | International Organization for Standardization; Global organization that develops and publishes standards (ISO 2019) |
| ISTQB | International Software Testing Qualifications Board; Provides certifications for testing and quality assurance around the world (ISTQB 2019) |
| TMAP | Test Management Approach; A series of publications for testing and test management, developed by Sogeti Labs (TMAP 2019) |

# 1 INTRODUCTION

The need for testing has been on the rise in software development. Continued growth is really probable as well, since the complexity of IT solutions and software is increasing and the need for new interfacing between different products is ever more apparent. Additionally, the workload for quality assurance increases as new products and updates are required in ever shorter development cycles.

Test automation is an efficient way to increase the trust in testing and reduce the costs it creates. Because test automation requires a significant investment in the beginning, when new test environments are created and the scripts are being developed, it suits extremely well to cases that require a great amount of repetition. A good example would be regression testing, meaning testing of unwanted side-effects in the unchanged parts of system as a result of new updates.

In modern software development environments, usability testing is also increasingly needed. Especially in the mobile market there are plenty of different apps, created for the same customer need and the best way to ensure product success is to invest in accessibility and ease of use. To evaluate the usability of an IT solution, there are no absolute values available. A common method is to use the 10 usability heuristics as written by Jakob Nielsen in his book Usability Engineering written in 1993.

Because of the increasing need for usability testing and its resource intensiveness, it is only natural to aim to automate the usability evaluation. One of the clear potential advantages of that would be the elimination of human factor from the evaluations, to make test results comparable with each other and repeatable as required. This would naturally reduce the effects of organizational changes on usability tests.

However, the challenge with usability testing is that there is no standardised way of measuring it. The heuristics only give guidelines on what to consider. To automate any tests, strict criteria are required on pass or fail, as a machine cannot make judgement independently. This thesis work will present different methods of automating testing and evaluate the opportunities they offer if used in usability testing.

# 2 TESTING IN THEORY

Software development is still a heavily human driven effort despite all the advancements in artificial intelligence and machine learning. Human effort makes development, by its nature, prone to human error, which leads to a need for testing. Testing is usually conducted in order to detect those errors before they have a chance to affect the eventual user of the product.

It is notable though that any amount of testing does not detect all possible defects in a system. It is still important to do as testing most prominently gives confidence to all stakeholders that what is done is the right product and that the product is done right. This means that the product is designed to fulfil an existing need, the use case is established in product requirements and that it actually performs all functions as intended.

As the development process has evolved to provide faster production cycles, the old method of testing at the end of the product development has slowly become obsolete. The idea behind a modern process is to take testing into the development as early as possible. It is suggested that since product requirements are also defined by humans, it would be a good practice to include testers already when not a single line of code is actually written.

The most often used terminology around testing is based on the International Software Testing Qualifications Board glossary. The International Software Testing Qualifications Board is a non-profit organization that provides certifications for testers and have official glossaries in multiple languages (ISTQB 2019). There are other similar frameworks, for example Test Management Approach – TMAP, that are developed for more commercial purposes (TMap 2019). In many regards, these two approaches are not mutually exclusive, but when differences arise, this thesis will default to ISTQB vocabulary.

## 2.1 Different levels of testing

The way of thinking of testing as an ongoing process all the way through the software development is most often described in the “V model”, which presents the activities in development on one side and the activities in testing on the other. These activities are called different levels of testing. Reality is rarely as simplistic, but the model gives a good

tool for planning different tests fit for the situation. In the context of this thesis, the V model is used to highlight different levels of testing, as they are more pronounced than in most other software development lifecycle models.

Currently, the software development lifecycle models are mostly Agile, or iterative-incremental. The V model, being an extension to the old waterfall model, is not Agile. However, the growing trend seems to be that organizations are combining the approaches from Agile and DevOps with some elements from the Waterfall model to create an environment that best fits their own requirements (Capgemini et al. 2018). For this purpose, the V model fits well, as it is on its own quite simple while being similar to the familiar waterfall and allows for simultaneous processes.

When compared to the older waterfall model, the greatest difference is that with the Waterfall work is done on a single flow, whereas the V model works both sides simultaneously (Techspirited 2019). When transitioning from Waterfall to a more agile environment, the V model is a good starting point, since it focuses on synchronizing testing and development to a simultaneous process, without requiring a completely new workflow as shown in figure 1.

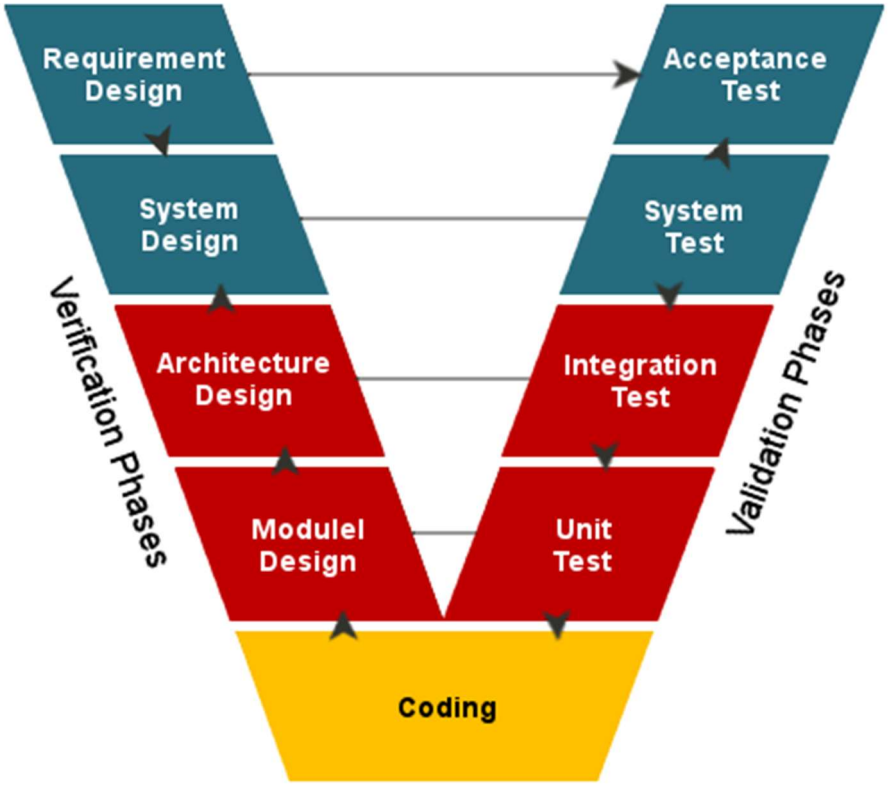


Figure 1 V-model combines development activities (on the left) with testing activities (on the right) (Professional QA 2019)

In addition to attaching a certain test level to the development activities, the V-model also quite naturally distinguishes between testing carried out by product vendor and the client. This separation usually means that tests conducted on the vendor side are done inside the organisation that provides the product, to verify the tests that are carried out by developers before handing the software to the client. These tests include unit tests, validating basic functionalities, and integration tests, performed on multiple functionalities, to give assurance that they work together as intended.

Tests performed by client are usually called different types of acceptance tests, such as system acceptance or user acceptance tests. These tests are in most cases performed by either the client or their own quality assurance department or by an external organisation that provides testing as a service to determine whether the product fulfils the product requirements set by the customer. It should be noted that the vendor and

client are not always from separate organisations, for example, when the same company creates their own software from the ground up to the end users.

## 2.2 Different types of testing

Test type refers to a group of activities that are intended for testing a system's quality characteristics (ISTQB 2019). In turn, a quality characteristic is a combined set of attributes that determines if the product is of high quality (ISO 2017). As an example, TMAP (Test Management Approach) describes 17 distinct characteristics, some with their own sub characteristics (TMap 2019)

Just as test levels differentiate testing activities in regards of the whole development process, a test type contains multiple actual tests for the specific characteristic. Usually test plans for a test level contain a description of the test object, including the requirements for different capabilities in the product, then a list of different quality characteristics that need to be covered and under them, the required coverage and test design technique that is supposed to be used to achieve that.

Test types and quality characteristics can be categorized into two separate groups, functional and non-functional testing. Probably the most prominent test type is functional testing – performing actions in a system to ensure that it works as intended. Functional testing is naturally most common in every test level for the simple reason that a product is rarely worth anything if the functional requirements are not fulfilled.

Non-functional test types, on the other hand, cover all the aspects that are not related to functional product requirements. These requirements cover quality characteristics such as security, performance, user-friendliness and regression, among others. Overall, non-functional tests are performed in every test level like functional tests, but in many cases, they can be included in functional tests as additional validations or checks. For example, when performing an end-to-end functional test, measuring performance can be done on the side.

For test automation, a natural target is usually something that is highly repetitive and requires large data sets, both of which are difficult to handle by humans. When it comes to test types, regression tests are typically a good candidate. They are usually tests that have been performed in previous test runs and their purpose is to ensure that no defects have been introduced to a part of the system where no changes have been made in an

update. Although regression testing is a growing need with shorter development cycles, it is not the focus point of this thesis work.

The test type that has been seeing even more increase in demand in the recent past is user-friendliness or usability because ensuring end-user satisfaction is seen as important as detecting software defects before go-live (Capgemini et al. 2018, 16). The usual method to meet an increased demand in testing is to implement test automation. However, as usability testing is seen as a purely human effort, it could be a potentially lucrative opportunity to take. This thesis attempts to discover whether this is the case. The difficulty in this is, however, that the topic relies heavily on the definition of user-friendly.

# 3 USABILITY

Usability is a relatively new field of study compared to some other aspects in software development. It is also gaining importance extremely fast for multiple reasons, most prominently because of increasing competition in the software industry as tools are more readily available to create software, web pages or similar, without years of training and education. Also, the products need to continually cater to broader audiences and customers, who might have vastly different levels of experience with similar software.

Usability is determined in ISO 9241-11 as “the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” Although not a definition that does not need any more explanation, it does tell that usability is distinct from user experience. Usability focuses on the context of a certain use case, whereas user experience takes in the whole experience of interacting with the product, regardless of context (ISO 2010).

Usability is also described in, already outdated ISO 9126 as “a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users“ (ISO 2001). The ISO 9126 has been already revisioned to conform with ISO 9241-11, but it clarifies that there is enough difference to show that there is not defined a singular consensus on what is usability as a whole. In the end though, this is not an issue in the context, as usability is intertwined with so many different aspects in the end-user experience that there might never be a fulfilling definition for it.

As usability as a term is difficult to define, there have been many attempts to divide it into smaller sub-categories. One way is to divide it into seven distinct areas, including effectiveness, efficiency, satisfaction, ease of understanding, ease of learning, attractiveness and robustness (Koomen et al. 2014, 317). Even though effectiveness and efficiency can be understood with little explanation, there are still “ease of learning” and “satisfaction” that should be clarified.

It is easy to understand the value in having products that provide customers with a satisfactory experience, but the difficulty lies in measuring that satisfaction. There have been many attempts at creating a system for numerically measuring different aspects of usability, but most of them have proven just as situational as others. This is why, even

today, commonly used usability measurements are based either on crowd testing by questionnaires or by conducting user interviews.

As with any testing that is done for the user interface, there is a danger that it is left to be done only when everything else works already. This means that the cost of testing goes up because the later defects are found, the heavier the required investment is for the fixes. In the worst case scenario, the whole product is designed in a way that is not appealing to the users at all and thus ends up dragging the whole product down. For this reason, it is suggested that usability testing should be started as early as possible, preferably at the stage when designs and product requirements are being defined.

## 1.1 Testing usability

Usability differs from other software quality characteristics in a sense that cannot really be evaluated in a bipolar manner. Whereas functionality can be measured in “works as intended” or “not working as intended”, usability almost always has more variety in the scale. As testing attempts to give insight to the quality of the product as objectively as possible, the nature of usability poses a problem.

Due to the difficulty in creating a clear distinction between passing and failing certain aspects, testing usability is in many cases either included in the test plan late into the project or excluded completely. As with all testing, this ultimately leads to a situation in which the proposed changes to a product are likely far too costly to implement. This is a direct result of the fact that, the later in the software production cycle a change is needed, the more different systems there are for that change to be addressed in.

To combat this reluctance to perform usability testing early, there are different standardized ways developed. These aim to give mostly similar results, but with a differing focus on certain aspects of usability. Because of this difference in focus, best results are usually achieved by a combination of different types of tests at different phases in the project. This way, the organization has the widest possible information available, also for decisions to not make the changes.

As there are too many different methods for testing usability to cover them all in the context of this thesis, the represented methods have been chosen to cover as wide a cross-section of different usability testing processes as possible before moving on to the heuristic evaluation, which is used for the demonstration.

Maybe the first things to come to mind when talking about usability testing, are all the different ways of using user-centred testing. User-centred testing or user testing, as the name implies, focuses on different ways of testing the product on actual users. This provides reliable feedback regarding usability because users have better knowledge of the task that the product should be fulfilling.

In user-centred testing, the usability expert is left with a responsibility to design the testing and analysing results. Interviews are one example of such testing. Interviews are usually conducted at a specific test site under a controlled environment. The user can be given tasks to perform and asked to explain their thoughts out loud. This is called the “think aloud” protocol (Nielsen, J. 2012). This way, testers gain as much organic and on the spot feedback as possible on which to base their later evaluation.

The drawback with having users testing the product, is that the feedback might not be precise enough. It may also be that using users to test the product is simply not feasible. This is when expert evaluations are helpful. Usability evaluations done by usability experts can, in most cases, be more specific and precise on an overall technical level, than feedback from the user, but at the same time requires more robust documentation on the product as the experts might not be familiar with the actual real-life use cases.

User testing and expert evaluation often similarly base their results on individual perception. Users base their knowledge on previous experiences with the task they perform and usability experts base their knowledge on learned patterns and established design practices. Both however, are difficult to measure, so in cases where numeric proof is needed, testers can use quantitative methods to establish whether certain usability criteria are fulfilled.

Quantitative studies can take on many forms. Questionnaires are often used with the aim of having a strict pre-set collection of metrics while gathering a large set of data on them (Budiu R. 2017). Such metrics could be the time taken to perform certain tasks or the amount of interactions needed to access specific functionalities.

Since purely quantitative data rarely give a complete picture on the experience users have, the quantitative methods are at their best when used over a long period of time and possibly over multiple iterations in the product development lifecycle. This of course helps testers to understand whether development has been going the right way and if there are certain aspects in usability that would need more attention.

If an organization has the expertise available, but no data to conduct quantitative studies and if user testing is for some reason not an option, then instead of going for more formal expert evaluation or inspection, heuristic evaluation could be used. Heuristic evaluation will be the base used in this thesis to evaluate whether usability evaluation can be automated.

## 1.2 Usability heuristics

Heuristic evaluation is a method of evaluating usability based on specific usability heuristics. Jakob Nielsen introduced heuristics as a usability testing approach in his book Usability Engineering in 1993. Heuristic evaluation combines a loose mix of expert evaluation and quantitative study by using a set of predetermined metrics but allowing evaluators to use their experience on determining good practices and potential issues with usability .

In his book, Nielsen establishes the existence of ten distinct heuristics. The heuristics describe certain functionalities and attributes that a piece of software needs to have for it to be regarded as usable. The heuristics seem to be quite simple and easy to work into any software, but many software solutions continue to lack them, even today.

The heuristics identified by Nielsen will be divided into three separate groups in this thesis to help make distinction between them and give insight on what properties might be good to measure with each of them.

The first group will be called interaction heuristics. It consists of characteristics that focus on the user's actions towards the system. This group can be best evaluated by the actions needed by the user and the difference between the designed "intended" use and the "actual" use that happens when users access the product without outside help. The four heuristics in the interaction group are called: (a) user control and freedom; (b) flexibility and efficiency of use; (c) recognition rather than recall; and (d) error prevention. In software, these aspects can be best considered by letting users access critical functionalities easily and limiting potential inputs in situations where failures might be caused by the user. For example, the option to change languages in a questionnaire form should always be visible, while free-form text boxes should be avoided where possible.

The second group contains the so called feedback heuristics. They focus on the system capabilities of giving clear and understandable feedback of current the system status to the user. This group focuses specifically on messages and instructions and includes the heuristics: (a) visibility of system status; (b) help users recognize, diagnose and recover from errors; and (c) help and documentation. For example, there's a big difference between giving user an error message with some debugger error code and a message saying that some incorrect entry caused a crash.

The last group of heuristics comprises: (a) match between system and real world; (b) consistency and standards; and (c) aesthetics and minimal design. This group is more focused on the actual presentation and user experience of the system. To get a high score in these aspects, systems should only give information as it is needed and use terminology that are both similar to other systems and relatable to the real world. An example would be to use an image of a gear or wrench as an icon for opening settings in an application or calling a pointer device a "mouse".

For the automation in this thesis, the "consistency and standards" heuristic will be used. Consistency and standards mean in this case the similarity of a system with others on the used platform, rather than strictly following written standards. (Nielsen 1994). This contains aspects such as using certain common words to describe objects and interactions and that the objects are available for the user when needed so that they are not changing location or visual appearance. One example could be the usage of a gear icon in corner of the screen as a way to access some settings and icon is always visible.

# 4 TEST AUTOMATION AND TOOLS

The answer to the increased requirement in quality assurance is test automation. It is the best fit for testing tasks that need high amounts of repetition or accuracy in data handling, both of which are usually difficult for human testers. There are drawbacks, for one, the initial investment for test automation is high because of the required infrastructure and expertise in designing and writing the test scripts. This means that the effort for test automation needs to be long-term and consistent to provide meaningful benefit for the investment.

Test automation, when used in professional environments, can be done in two distinct ways. The first one requires more knowledge of development and programming and is more useful in early phases of development, but makes for more reliable and faster tests. This is also called application programming interface (API) testing, which is done by creating automation scripts that send calls to the test object directly in code without ever interacting with the user interface. Additionally, tools that provide API testing functionality are most often aimed at developers rather than testers.

The second way is automation, where a script is written to imitate user actions on the screen. This is in many ways inferior to API testing, since actions in a user interface make for brittle tests. A minor change in the interface can prevent running the test successfully and loading unnecessary screen objects slows testing down. However, this way of test automation doesn’t require specific knowledge of the code base of the product and is thus doable by external testers. Additionally, it reveals possible issues in the user interface. For this thesis, test automation will be done using UI.

## 4.1 How testing is automated

The aspect that has the most impact on the form that the test automation is taking, is the choice of tooling. Depending on the project organization, for example, the available expertise and funding, the tooling might be anything from free open-source library for code editor to highly matured corporate tool with capabilities to record and play. The most important aspect is that the tool selection is based on correct criteria, as many tools still have quite big drawbacks when used for tasks they are not made for.

When talking about test automation tools, Selenium is usually the first one being mentioned. It is widely regarded as the most popular test automation tool, especially for web-based testing (Katalon et al. 2018). It has a lot going for it, being open source it is free to start using and experimenting and it has out of the box compatibility with almost all existing browsers. For a quick proof-of-concept type script creation, Selenium has an integrated development environment (IDE), which supports recording actions in the user interface and then playing them back.

Getting started with Selenium is however a lot easier than introducing it to larger organizations. Even with the current IDE capabilities, most users are pointing out that when scripting is needed, there is no easy way to go about it – the user will need to know one of the supported programming languages. An additional drawback with Selenium is that, being just a framework, it has no built-in capabilities for test execution automation. This means that when using it in larger organizations, setting up a continuous integration environment requires a notable infrastructure effort.

On the opposite side of the spectrum, when a more mature tooling is required, Unified Functional Tester (UFT) by Micro Focus, is a good example. The benefits of using UFT, is that in addition to testing UI, it supports API testing and is highly compatible with different software and frameworks that are used in corporate environments. Also, being part of a bigger product family, UFT has multiple different integrations available, ranging from business process testing to continuous integration.

The drawbacks of UFT include that UFT's license fees are too high for amateur use and that it only supports Visual Basic Script. UFT does however, provide enough documentation and helpful functionalities so that the user does not need specific knowledge of VBScript. However, the language is from 1996 and is no longer widely used with support for the language slowly being deprecated by Microsoft (Microsoft 2018). This comparison only goes to show that when starting test automation, choosing a correct tool for the job goes a long way.

For the automation in this thesis, a tool called Robot Framework will be used. Initially developed by Nokia and recently open sourced, this framework is in many ways similar to Selenium, but instead of using an actual programming language for test development, Robot Framework relies on keyword driven scripts. The tool itself is endorsed by a multitude of high-profile IT companies, which makes it relevant when talking about today's test automation effort (Robot Framework Foundation 2019)

## 4.2 Robot Framework

Robot Framework is an open source framework that can be used for test automation in a multitude of ways. Robot Framework most notably supports keyword driven test development, meaning that the syntax of the tests consists of keywords which act as function calls. Keywords can also be layered, so that instead of having to write a keyword for opening a browser and entering username and password, there can be a single keyword that handles the whole login flow.

This way of layering multiple keywords into more high-level functionalities, makes Robot Framework tests understandable and readable, even if the tester does not have expertise in programming. In a professional context, this makes reusing valid parts of scripts extremely easy, which in turn means that over time the added value from automation stacks up. In addition, there is a reduction in the time for test execution, as well as reduced time for test script creation.

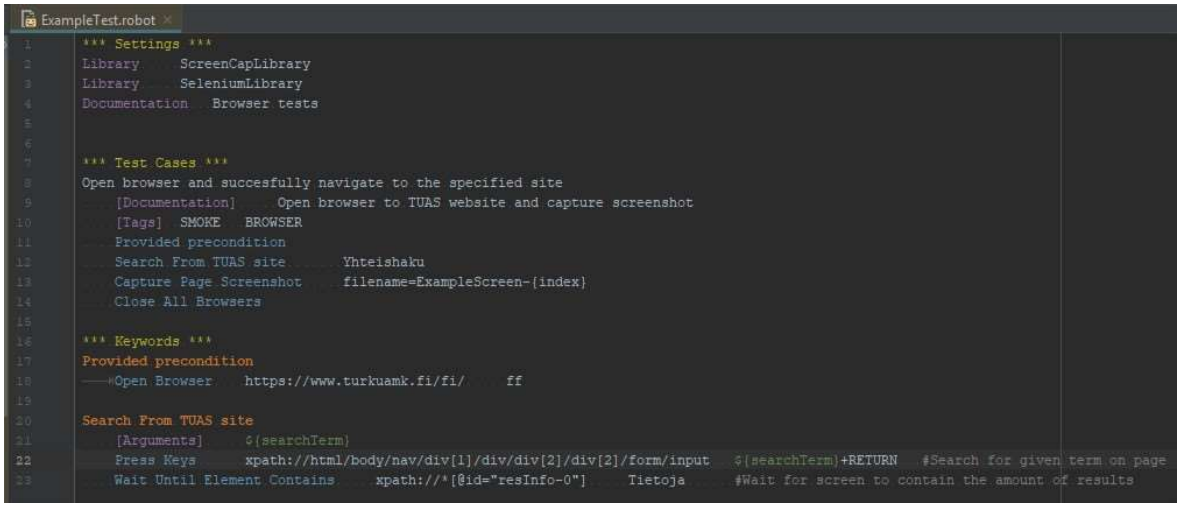


Figure 2 An example of a Robot test case that uses high-level keywords

Image 2 highlights the actual usage of Robot Framework. First, the user imports libraries that contain different keywords. There is a multitude of pre-made ones, but in case the user needs something specific, these can be developed from the ground up using Python or different Python interpreters if other languages are preferred. Additionally noteworthy, is that the keyword layering can use arguments to pass information and variables to lower level keywords, for example in this case the word “Yhteishaku” is passed to the “Search From TUAS site” keyword and used in performing that.

Depending on the libraries that are used, Robot Framework can be used either with the old Python 2.7 or the newer 3.x. If the old Python is used, there is a dedicated IDE called RIDE, Robot Integrated Development Environment, available. RIDE is a visual scripting environment, which makes it especially useful for new test automation developers, as it eliminates the need to manually format the script in editor. Unfortunately, many new functionalities would need to be sacrificed when using the older Python.

One last clear differentiator between Robot Framework and many other similar tools, is the automatic reporting function. Whenever Robot is used for running any test, it automatically creates a three-level test report, containing an HTML file for the raw output file for debugging, a log file for each test case that was run in the test suite and a clear report file that has an overview of all test cases together. Having sophisticated reporting built in, makes Robot Framework an extremely attractive tool since there's no need to spend time programming that from scratch.

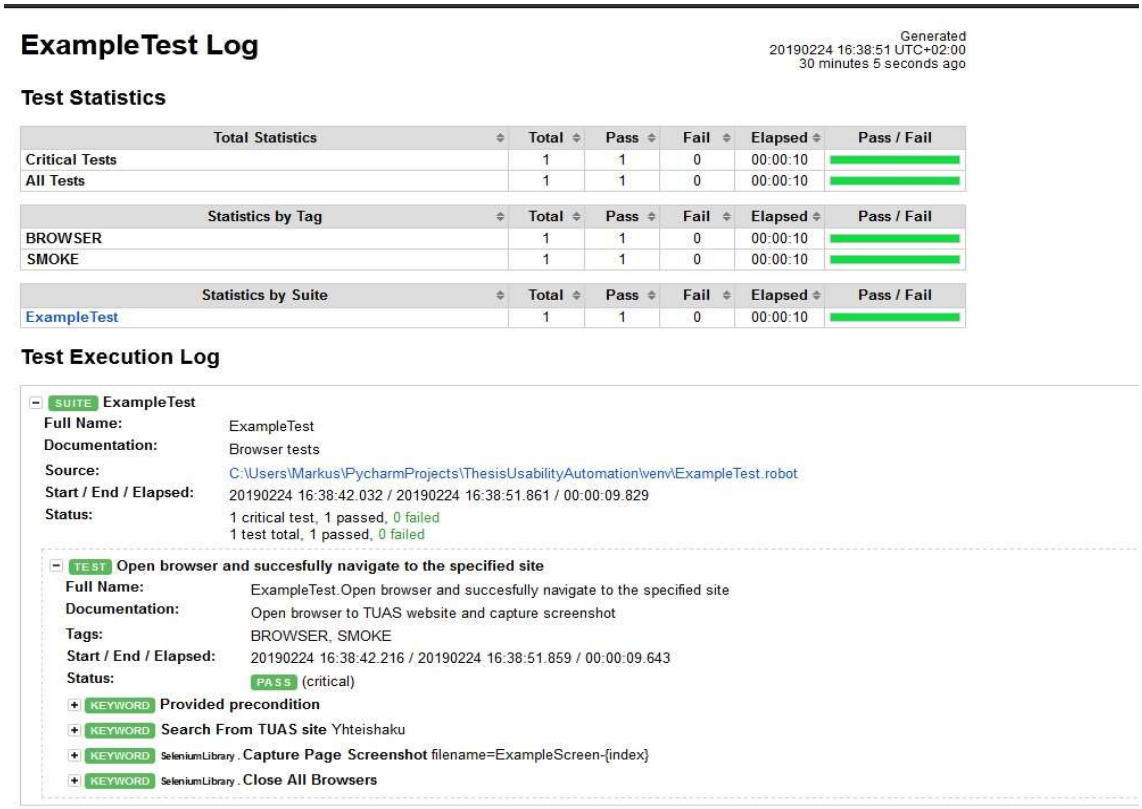


Figure 3 An example log of Robot Framework test execution – selecting a keyword provides more details

The test log is quite detailed since it contains a row for each keyword that is included, which makes the files potentially too large to be readable. This is why it is also highly modifiable for more experienced users. Some keywords can be prevented from being logged if that keyword is executed multiple times during the test or by adding screenshots to certain steps. For the context of this thesis, the basic report will be enough.

For test automation in professional environments, another important aspect is the ability to implement it into the continuous integration (CI) environment. Robot Framework has a command line interface and since the reports are XML based, it can be part of setup in Jenkins for example (Nokia Solutions, Robot Framework Foundation 2019). This however, is out of the scope of this thesis so that aspect will not be explored further here.

### 4.3 Other efforts in automating usability testing

Although usability testing has not been traditionally automated, it doesn't mean that usability testing does not have tools to support the efforts. A multitude of different tools are available, especially for web testing. Although these tools are rarely enough for professional environments, they do provide some idea on what aspects of usability are possibly good for automation. Though there are several tools, of varying level of usefulness, available than can reasonably be presented here, only a couple of examples will be explored.

One large group of products with a plethora of possible options is based in user tracking. Different tools offer different capabilities, but the common part with all of them is that these tools are implemented to already deployed and working websites and they follow different aspects of user interactions in the page. One creates heatmaps of where users are clicking most of the time on the web page and one has users to answer several questions when using the site to see how they like their experience. This creates results similar to a quantitative study, where the long-term comparison of large userbase can give useful data to see trends over time.

Especially useful for smaller organizations, are tools that can be used for testing different browsers and devices. These tools work in a way that users give a web address for their working service and the tool then attempts to access that with a selected set of devices and browsers. Because the feedback tends to be quite general and the tool doesn't access all possible functionalities, larger organizations may be better off creating a

compatibility test suite from the ground up. However, that can be, depending on the requirements, an extremely expensive undertaking when considering different mobile options, which makes it impossible without the ability to make the initial investment.

A large part of usability nowadays is also accessibility. Accessibility testing for web pages is important, especially when considering that many necessary services have been slowly moving towards only web. For example, testing against different types of color blindness can be done by using free tools. Although the approach is different, the end result is that the tool places a certain color layer over the page to simulate color blindness, after which the tester can compare the results to ensure all screen objects can still be identified from each other.

When talking about accessibility and testing for disabilities, there are various aspects ranging from visual to motoric disabilities and luckily there are tools available for testing pages for compliance with screen reader standards. For example, but issues like dyslexia are more difficult to test against. For this, there is something called Readability Test Tool by WebFX (WebFX 2019). It tests the readability of a web page against multiple indicators and then provides a score for the web page on how readable the content actually is. Of course, content and layout both need to be tested for readability, but in usability perspective it is an important issue.

# 2 PROOF OF CONCEPT

For this thesis, a usability test will be performed for the home page of Turku University of Applied Sciences in two ways. First, the usability test will be performed by running a Robot Framework script and then for comparison a more traditional approach will be applied, following as closely as possible usability testing guidelines presented by Nielsen.

The robot script will be designed and run first, focusing on the usability aspects that fall under the standards and consistency usability heuristic. To keep the effort focused, the functional aspects or other usability aspects will not be considered in this thesis. The main part of test scripts will be comparing locations and contents of certain persistent screen elements, such as language selection.

Then the traditional usability test will be conducted. The tasks in the test will be designed so that they would direct the focus to the same aspects as the automation script is testing. Additionally, the tests will be video-recorded so it will be easy to compare if the user was not finding the same link or button in two different screens.

## 4.4 The automation script

The Robot Framework script used is mostly built from pre-existing libraries. For the tests to be run for this thesis, there is no need for additional keyword development. The tests in this proof of concept will also be run with predetermined variables, like search terms and static checkpoints. This will make the results the same for every test run, while providing a good enough view to determine the potential of test automation in this context.

The test compares the location and contents of screen consistent screen elements. The test will return a failed step if the screen location of these elements or their text content change from the expected or they are completely unavailable. Text content is of course expected to change if language is changed. In addition, since the site is mainly for search of information, some information searching was to be tested by confirming that the search results yield the word that was searched for.

The tests were organized into three separate test cases, each testing for different parts and elements. All of them were built with a predetermined set of input data and every case assumes that they are starting from the main page and all of them end up at the main page after the execution of the steps. Each step includes one or more screenshots to the report for actual result comparison.

```
*** Settings ***
Library    ScreenCapLibrary
Library    SeleniumLibrary
Library    BuiltIn
Documentation    Browser tests

*** Variables ***
${elementXPos}
${elementYPos}


*** Test Cases ***
Quick navigation and language selection buttons consistency
    [Documentation]    Open browser to TUAS website and capture screenshot
    [Tags]    SMOKE    BROWSER

    Provided precondition
    FOR    ${item}    IN RANGE    1    4
        ${element}=    Run keyword if    ${item}==2    Set variable    xpath://*[@id="dropdown-list-toggler"]
        ...    ELSE    Set variable    xpath:/html/body/nav/div[1]/div/div[2]/div[1]/div[${item}]/a
        ${content}    ${elementXPos}    ${elementYPos}=    Get location and content of elements    ${element}
        Search From TUAS site    Yhteishaku
        Compare location of elements    ${element}    ${elementXPos}    ${elementYPos}
        Compare the contents of the elements    ${element}    ${content}
        Capture Page Screenshot    filename=ExampleScreen-{index}.png
        Click Element    xpath:/html/body/nav/div[1]/div/div[1]/a/img
    END

Navigation bar consistency
    [Documentation]    Go through the main navigation bar and see that the buttons stay where they should length should be
    ...    And content stays as it should be
    Go through all navigation bar links    #Click on link, then compare all links for difference in location or content from the main page
    Click Element    xpath:/html/body/nav/div[1]/div/div[1]/a/img    #Go back to the main page by clicking the Turku AMK logo

Search result relevancy
    [Documentation]    When searching for some information from the site, the returned results should be relevant
    ...    to the search term.
    FOR    ${searchTerm}    IN    Yhteishaku    Kampus    Tradenomi    Insinööri    Peliala    #Each word is a search criteria
        Search From TUAS site    ${searchTerm}    #Go to search bar and type correct word + submit
        Check that every result is relevant    ${searchTerm}    #Run through all results on page 1 and see if the results contain the search term
    END
    Close All Browsers
```

Figure 4 Robot framework test cases, the full script is available in Appendix 1

The first test case of the Robot Framework script focused on the navigational buttons on the screen. The currently considered ones are language selection, the quick links and the contact buttons. Their exact screen location is first captured, as is their text content, and then navigation will be done to a different page where that data is then compared to their current state. This test makes sure that the buttons are always available and that they can always be identified. This test case could be expanded by adding multiple devices or screen sizes for example.

After the quick navigation buttons are tested, the test moves to handle the navigation bar. The test once again first captures the contents and locations of all links on the screen, then it accesses the links one by one by clicking them. After each click, when

the page has loaded, the script again compares the location and contents of all links with the data from the main page. When expanding from this, the test could also access the links under the main navigation bar and other parts of the site as well.

The last test case is also a multipart test. It conducts a search from the site by using the search bar. The set of test terms are predetermined and are something that should be searched for quite often from the site. After the search term has been submitted and the search result page has loaded, the script will go through each result on the first page and check that the result contains the search term in some part of it. This test should then show us whether the search returns relevant results.

The tests were first conducted in the Finnish side of the site and then the test situation is reset, before the test automation attempts to access all the same functionalities in English. Of course all contents and search terms were translated for that, for example “Yhteishaku” would be “joint application”. One glaring omission on the site was the lack of ability to select Swedish as a language, so that was not tested at all.

## 4.5 Description of usability interview

In addition to other factors, to determine the value of the results given by the test automation, a more traditional usability test was conducted. The tests were conducted as usability interviews, following the traditional structure with predetermined tasks and using the think aloud -protocol. The tests were supervised locally and in addition to notes being taken, the screen was recorded to provide best additional material for later analysis.

The interview was conducted on four users. This should be enough to find most of the potential usability issues and minimize redundancy in the test results (Nielsen, 2000). This minimized the amount of time required for testing and thus made the comparison of effort to test automation most fair. Of course, when considering that the results were analyzed only in regards of consistency and standards -heuristic, the potential amount of issues were also significantly reduced, decreasing the overall need for test subjects.

With test automation, the test cases revolved around testing for contents and screen positions of different objects. This was supposed to simulate the consistency of those objects and pinpoint potential issues that users could have when trying to find those objects. To relate with that, the user tests were designed to follow a similar loop that had

them first navigate to a certain page with clear instructions, then for example use search to find some certain information with less instructions.

This way of structuring the tests gave a couple of advantages with regard to analyzing the results. First and maybe the more prominent one is that each tester performed some steps to get used to the system and gave the analyzer a comparison point, did the user spend more or less time finding objects without instructions compared to the instructed step. The clearer comparison point to the automation then was, whether the test subject managed to find the object from the current screen or did they end up visiting the main page first. In case of the latter, it might have indicated that the objects were not readily available in all screens.

The tasks for the subjects were handed as a printed paper with instructions stating that the test would follow speak aloud -protocol and that the subject should start by reading out loud the instructions, then read out loud the task and explain how they are approaching it and what are the actions they would be making. The tester could not give instructions and asking questions was deliberately avoided. The test was given 30 minutes time, with ten minutes reserved for the setup and introductions. Because the test subjects were native Finnish speakers, the tests were conducted in Finnish. An English version of the instructions and tasks is in appendix 2.

The test subjects were purposefully selected to fall into the target audience for the test object site. Demographically they were male and female between the ages of 19 and 25 with varying amount of technical knowledge in IT. Every test subject had interest in some of the education that TUAS would be able to offer.

# 5 RESULTS

Jumping straight to the actual results returned by the test automation, it can be seen that the report is clearly more precise than a human tester could provide. All objects in the screen are measured and checked by pixel measurement so even if the test was failed, it might be that human tester would never notice. The traditional user testing returned useful results as well, but as with any evaluation, the analyzing of results could not be done in absolutes.

## 5.1 Results of test automation script

Because no official documented requirements for the system were available, the test script was written so that the main page of the test object was used as a reference point. This meant that any deviation from the language respective main page would return failed test step. Also, irrelevant search results and missing objects would return failed test step.

The test automation proved to return pass for every test step in every test case. The elements that were tested for were consistent in location and content throughout the system, both the Finnish and English side of the system. The test execution for all iterations throughout the three test cases in two separate languages took, according to the Robot Framework test report, 40 seconds per language.

Most notably, the search functionality seemed to work even beyond expectations, returning relevant results for every configured search word regardless of language. It might be that the test was either designed to be too easy, but it was a positive sign that the keyword search was successful. Also, the test automation was successfully able to perform all the tests in different screen sizes and return pass from the test, which is further proof that the test automation is capable of reliably returning results regardless of system running the test.

## 5.2 Results of user interviews

The user interviews were a success in the sense that all test subjects were able to provide some insight to the usability the system. All test subjects managed to complete the given tasks during the time limit. Because the users were all familiar with similar web sites, for example other universities' web pages, the differences in execution were relatively small.

The consistency of the web page proved to be relatively well established for the test subjects as well. The navigation between different parts of the site was relatively quick, with no test subject showing any specific slowness between finding the location of next task in most parts. One exception was the task number 7 "Switch language to English", which was to determine whether the users were able to repeat the same task in English version of the site they just finished in the Finnish side.

Out of four subjects, three ended up taking more time with that task and two of those requested for help in either translating or reminding how they ended up to the previous site in the first place. Every test subject managed to navigate successfully on their own without additional help, but the confusion seems to be a case of the site not being similar in both languages.

A notable success for the test object was that whenever a search function was used, all four of the test subjects were able to locate and use the search bar regardless of which part of the site they were on. Additionally three out of four test subjects were able to provide answer for the question 8.a "What is the first result about" without using time to access the link, which tells that the snippet on the search results is at least in that case clear enough about the contents.

Performing the tests with four users took approximately two and a half hours, and analyzing the notes and video material doubled that so the effort for this would be approximately five hours.

## 5.3 The comparison

In the context of the thesis, it seems that the test automation and the traditional user interviews correlate. The results are formed from different material, but it is safe to say

that in regards of consistency and standards, the tests yielded similar results, proving that the test object site www.turkuamk.fi is consistent with itself and does function similarly to other web sites meant for the same purpose.

The successful tests from user interviews do point to the same direction as the test automation, that the test object site has clear navigation options and that when navigating the site, the navigation bar is stable and visible enough to be useful when finding out information about TUAS. The search bar can be found and used with relative ease regardless of where in the system the user currently. The results from the search are relevant.

The biggest potential usability issue was found when switching languages from Finnish to English and attempting to navigate there. This was also the point where the test automation fell short with its current design. Even though the site was consistent with itself when using in Finnish or in English, the user interviews pointed that the site looked somewhat different when switching to another language in middle of use and gave different information depending which language was used.

Because the test was conducted with only external knowledge of the system, the issue with different contents in different language can be seen either as an issue or something intended. Regardless, the user interview's value was shown in the sense that it gave some results that were not anticipated or planned for, whereas the test automation only gave expected results according to its design.

The amount of effort for each phase though, when comparing the two methods, is quite different for both. The work it took to design, setup and write the test automation script was approximately three to four hours by one person. After that, each execution of the test took approximately one minute and 20 seconds when executing for both languages all three test cases. The results can be duplicated as many times as wanted with any system capable of running the tool and the results require no further analysis in most cases.

When considering the user interviews, the actual design, setup and planning took approximately two hours. This includes setting up test machine with screen recorder, writing and designing tasks and setting up the actual test situation—traveling done by the test supervisor or test subjects is not accounted for. In addition, every test subject was interviewed for approximately 30 minutes and then analyzing the results took almost the same time, totaling one hour for each test subject. The test results are not

duplicatable in the sense that whenever a new test is required, new test subjects need to be found and new results again analyzed.

Comparing the feasibility of the two methods, the test automation prefers longer and more intensive development projects. Because the investment required is heavies in the beginning and is reduced over time, in addition to the growing returns when it comes to time saved on the test runs, the longer the project – the higher the returned value. Usability interviews on the other hand has more linear investment requirements, since it lacks the added value of reusability. Making more traditional usability testing more valuable when fewer test runs are necessary.

The results and effort in this thesis are not telling the whole truth though. The test automation in the form it was designed and written cannot take into account anything else but the test cases it was explicitly designed for, whereas further analysis of the user interviews might reveal all new usability aspects that were not expected at all. If more aspects were to be considered, the amount of test automation design and writing would of course multiply.

# 6 CONCLUSION

The aim of this thesis was to provide insight on whether test automation would be actually worth the effort in usability evaluation. The methods were based as closely as possible to current industry standards to make the results from test automation and the more traditional usability interviews comparable.

Overall both the test automation and the usability tests were successful when considered separately. Both tests yielded results that were in most respects as expected and gave insight into the consistency and standards of the test object. The first stand-out point from these tests is that test automation is clearly capable of handling usability aspects in addition to the usual automated test coverage.

The test automation yielded results for analyzing usability aspects of the web site in question. The similarities in the results between test automation and usability interviews were enough to imply that test automation is capable of reasonably well pinpointing potential usability issues when designed that in mind.

The test automation in this thesis was simplified and focused only on certain aspects but in a more professional setting it would not be too difficult to implement it into an already existing test set and expand into other usability areas as well. Also, this demonstrates that starting a test automation effort is not a massive undertaking, if the initial resources, including time and expertise, are provided.

Based on this and the experience gained from the thesis work, it is clear that when it comes to usability testing, the best method to cover everything, is to combine both, test automation covering earlier development phases as part of a full test automation suire and user interviews later in the process to detect issues that have not been expected in the documentation or design.

Going forward in time, the current progress in cognitive quality assurance will probably make the usability analysis of any product even more easily automatable, as AI takes more ground in test automation as well. This would make analyzing different contents in images or color schemes even easier than it is now. However near or far that is though, automating usability evaluation does not need to wait for it. Current test automation tools are perfectly capable of handling usability evaluation with reasonable effort.

# REFERENCES


Agile Manifesto. Referenced 27.1.2019. https://agilemanifesto.org/principles.html

Budiu R. 2017 Quantitative vs. Qualitative Usability Testing. Nielsen Norman Group 1.10.2017. Consulted 26.4.2019. Available at https://www.nngroup.com/articles/quant-vs-qual/

Capgemini; Sogeti; MicroFocus. 2018. World Quality Report 2018-2019 Tenth Edition. Electronically available at https://www.microfocus.com/media/analyst-paper/world_quality_report_analyst_report.pdf.

ISO. Referenced 26.4.2019 https://www.iso.org/home.html

ISO. 2001. ISO 9126-1:2001. Software engineering -- Product quality -- Part 1: Quality model.

ISO. 2010. ISO 9241-210:2010 Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems

ISO. 2017. ISO 24765:2017: Systems and software engineering – Vocabulary.

ISTQB. Referenced 26.4.2019. https://www.istqb.org/

ISTQB. 2015. ISTQB Glossary version 3.2. Electronically available at https://www.istqb.org/downloads/send/20-istqb-glossary/186-glossary-all-terms.html

Katalon, KMS Technology, ToolsQA. 2018. Challenges in Test Automation: Survey Key Results. Katalon Blog 4.2018. Consulted 17.2.2019. Available at https://www.katalon.com/resources-center/blog/infographic-challenges-test-automation/

Kooman, T.; Aalst L.; Vroon M.; Broekman B. 2014. TMap Next – For Result-Driven Testing. Amsterdam: Sogeti.

Microsoft documentation. 31.5.2018. Using VBScript. https://docs.microsoft.com/en-us/windows/desktop/lwef/using-vbscript

Nielsen J. 1993. Usability Engineering. California: Elsevier.

Nielsen J. 1994. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group 24.4.1994. Consulted 26.4.2019. Available at https://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen J. 2000. Why You Only Need to Test with 5 Users. Nielsen Norman Group 19.3.2000. Consulted 26.4.2019. Available at https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Nielsen J. 2012. Thinking Aloud: The #1 Usability Tool. Nielsen Norman Group 16.1.2012. Consulted 26.4.2019. Available at https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/

Nokia Networks, Robot Framework Foundation. Robot Framework User Guide Version 3.1. Consulted 24.2.2019. Available at http://robotframework.org/robotframework/3.1/RobotFrameworkUserGuide.html

Professional QA blog. 2019. V-Model. ProfessionalQA 19.2.2019. Consulted 19.3.2019. Available at http://www.professionalqa.com/v-model

Robot Framework Foundation. Home page. Referenced 17.2.2019. https://robotframework.org/

Techspirited blog. Waterfall Model Vs. V Model Referenced 27.1.2019. https://techspirited.com/waterfall-model-vs-v-model.

TMAP. Referenced 26.4.2019. http://tmap.net/

TMap. Quality characteristics. Consulted 27.1.2019. http://www.tmap.net/wiki/quality-characteristics

WebFX. Readability Test Tool Homepage. Referenced 24.2.2019. https://www.webfx.com/tools/read-able/

# Appendix 1 – Test automation script

```
*** Settings ***
Library     ScreenCapLibrary
Library     SeleniumLibrary
Library     BuiltIn
Documentation   Browser tests

*** Variables ***
${elementXPos}
${elementYPos}


*** Test Cases ***
Quick navigation and language selection buttons consistency
    [Documentation]     Open browser to TUAS website and capture
screenshot
    [Tags]  SMOKE   BROWSER

    Provided precondition
    FOR     ${item}     IN RANGE    1   4
        ${element}=     Run keyword if      ${item}==2      Set
variable    xpath://*[@id="dropdown-list-toggler"]
        ...             ELSE                                Set
variable
xpath:/html/body/nav/div[1]/div/div[2]/div[1]/div[${item}]/a
        ${content}  ${elementXPos}      ${elementYPos}=    Get
location and content of elements   ${element}
        Search From TUAS site       Yhteishaku
        Compare location of elements    ${element}
${elementXPos}      ${elementYPos}
        Compare the contents of the elements        ${element}
${content}
        Capture Page Screenshot     filename=ExampleScreen-
{index}.png
        Click Element
xpath:/html/body/nav/div[1]/div/div[1]/a/img
    END

Navigation bar consistency
    [Documentation]     Go through the main navigation bar and see
that the buttons stay where they should length should be
    ...                 And content stays as it should be
    Go through all navigation bar links     #Click on link, then
compare all links for difference in location or content from the
main page
    Click Element      xpath:/html/body/nav/div[1]/div/div[1]/a/img
#Go back to the main page by clicking the Turku AMK logo

Search result relevancy
    [Documentation]     When searching for some information from
the site, the returned results should be relevant
    ...                 to the search term.
```

```
    FOR     ${searchTerm}   IN      Yhteishaku      Kampus
Tradenomi       Insinööri       Peliala       #Each word is a
search criteria
        Search From TUAS site       ${searchTerm}       #Go to
search bar and type correct word + submit
        Check that every result is relevant     ${searchTerm}
#Run through all results on page 1 and see if the results contain
the search term
    END
    Close All Browsers


*** Keywords ***
Provided precondition
          Open Browser    https://www.turkuamk.fi/fi/     ff

Search From TUAS site
    [Arguments]     ${searchTerm}
    Press Keys
xpath://html/body/nav/div[1]/div/div[2]/div[2]/form/input
${searchTerm}+RETURN   #Search for given term on page
    Wait Until Element Contains     xpath://*[@id="resInfo-0"]
Tietoja      #Wait for screen to contain the amount of results

Get location and content of elements
    [Arguments]     ${element}
    ${elementYPos}=     Get Horizontal Position     ${element}
    ${elementXPos}=     Get Vertical Position       ${element}
    ${content}=         Get text        ${element}
    [Return]    ${content}      ${elementXPos}      ${elementYPos}


Compare location of elements
    [Arguments]     ${element}       ${elementXPos}
${elementYPos}
    ${newYPos}=      Get Horizontal Position     ${element}
    ${newXPos}=      Get Vertical Position       ${element}
    should be equal     ${newYPos}       ${elementYPos}
    should be equal     ${newXPos}       ${elementXPos}
    #"Ota yhteyttä nappi
    #/html/body/nav/div[1]/div/div[2]/div[1]/div[3]/a
    #/html/body/nav/div[1]/div/div[2]/div[1]/div[3]/a

Compare the contents of the elements
    [Arguments]     ${element}      ${expectedContent}
    Element text should be      ${element}      ${expectedContent}

Go through all navigation bar links
    FOR     ${item}     IN RANGE    1   5
        Click Element
xpath:/html/body/nav/div[2]/div/ul/li[${item}]/a
        ${link}=    Get text
xpath:/html/body/nav/div[2]/div/ul/li[${item}]/a
        #Location should contain     ${link}
        Capture Page Screenshot     filename=ExampleScreen-{index}-
${item}.png
```

```
        Contents and locations on navigation bar should remain
    END

Contents and locations on navigation bar should remain
    FOR     ${item}     IN RANGE    1   5
        ${content}  ${elementXPos}      ${elementYPos}=     Get
location and content of elements
xpath:/html/body/nav/div[2]/div/ul/li[${item}]/a
        Compare location of elements
xpath:/html/body/nav/div[2]/div/ul/li[${item}]/a
${elementXPos}      ${elementYPos}
    END

Check that every result is relevant
    [Arguments]     ${searchTerm}
    FOR     ${i}    IN RANGE        1   10
        ${resultElement}=      Set variable
xpath:/html/body/section[2]/div/div/div/div/div/div/div/div[5]/div[
2]/div/div/div[2]/div[${i}]/div[1]/div[1]/div/a
        ${resultElementContent}=       Set variable
xpath:/html/body/section[2]/div/div/div/div/div/div/div/div[5]/div[
2]/div/div/div[2]/div[${i}]/div[1]/table/tbody/tr/td[2]/div[2]
        #Element Should contain      ${resultElement}
${searchTerm}
        Run Keyword And Ignore Error     Element Should contain
${resultElementContent}     ${searchTerm}
    END
```

## Appendix 2 – Usability test

# Usability test

Turku University of Applied Sciences web page

Please read out loud the part introduction first. Then turn the paper around and before starting next task, read that task out loud.

1. Instructions

This test is conducted to determine the usability of the web page of Turku University of Applied Sciences. The results will be used as part of the thesis work that is aimed to determine whether test automation could be helpful when applied for usability testing. Your personal information or pictures will not be published anywhere and your results will not be stored with your personal information anywhere after this test.

The test is based on a set of tasks. The tasks should be finished in the order they are presented in this sheet. The test follows “think aloud” -protocol. Please speak out loud everything you do or experience when executing the tasks. Notes from the test will be collected based on Your remarks. Note that not everything is specifically instructed, in case something is unclear, attempt to complete the task on your own and explain your thought process. The test supervisor cannot provide additional information.

The test will take maximum of 30 minutes. Not all tasks need to be completed before that and there’s no need to hurry through the test. The test supervisor will stop the test when time limit is reached if necessary. In case all the tasks are completed before the time limit, the test will be concluded. Thank you for your participation.

The screen will be recorded during the test.

2. Tasks

1. Navigate to www.turkuamk.fi site
2. Click link “Turun AMK”
    a. How many students in Engineering
    b. What is “#innopeda”?
3. Navigate to “contact us”
    a. What are all different contact categories?
4. Search for “Yhteishaku”
    a. How many results were returned?
5. Find the schedule of “Yhteishaku” and enter the site
    a. Where does one apply for education in TUAS?
    b. When does the application period end?
6. Navigate to “Työelämäpalvelut”/”Alumnitoiminta”
    a. When is the next Alumni event going to be?
7. Change the site language to English
    a. Navigate back to the site that was previously open
    b. Can you find where is the next Alumni event going to be?
    c. Change the language back to Finnish
8. Search for “Peliala”
    a. What is the first result about?
9. Where can I find information about course selection in open studies?
    a. Navigate there
10. Open again “Contact Us”
    a. What navigation path did you use?

Thank you for participating in the test!