Teemu Petäjäniemi

Software Testing

With Microsoft Test Manager

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

28 May 2016

Helsinki Metropolia University of Applied Sciences

Author Title Number of Pages Date	Teemu Petäjäniemi Software Testing with Microsoft Test Manager 27 pages 25 April 2016
Degree	Bachelors degree
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructors	Jaana Holvikivi, Principal Lecturer Janne Vesterinen, Software Engineering Consultant

This thesis aims at describing the software testing process as part of systems development. The main objective is to explain most common techniques and practices for software testing especially with Microsoft Test Manager and Microsoft Team Foundation Server. This thesis was made with Avanade Finland Oy and Seafarers' Pension Fund in Finland.

The case study in this thesis was completed in a project where a new public website was developed for the pension fund all the way from test planning to the end of the project and release of the product. Usually testing web services and especially informative websites does not include a considerable amount of functional testing and it usually consists only visual tests. Therefore, this part of a much larger project was perfect for briefly overviewing the software testing process.

As a result of this project the pension fund received a fully functional public website where customers can find the essential information about important services and other news.

Keywords	Software testing, Microsoft Test Manager, Team Foundation Server, Application lifecycle management

Tekijä Otsikko Sivumäärä Aika	Teemu Petäjäniemi Ohjelmistotestaus Microsoft Test Managerilla 27 Sivua 25.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaajat	Yliopettaja Jaana Holvikivi Onhjelmistotekniikkakonsultti Janne Vesterinen

Insinöörityön tavoitteena oli suunnitella ja toteuttaa testaus pienessä osassa asiakkaan, Merimieseläkekassan isompaa projektia. Tässä osassa projektia asiakkaalle toteutettiin uudet julkiset internet-sivut, joiden oli tarkoitus korvata käytettävyydeltään ja ulkoasultaan vanhentuneet sivut. Tämä osa projektia oli osana isompaa projektia, jossa asiakkaalle toimitettiin myös muita eläkealan järjestelmiä ja myös internet-sivut piti noudattaa yhtenäistä päivitettyä visuaalista ilmettä.

Projektin suunnitteluvaiheessa suunniteltiin visuaalinen ilme ja rajattiin projektin osa-alueet, joissa määriteltiin mitä toiminnallisuutta ja ominaisuuksia sivuilla tulisi olla. Tavoitteena oli toteuttaa visuaalisesti ja käytettävyydeltään hyvät ja helppokäyttöiset sivut, jotka noudattavat asiakkaan yleistä graafista ilmettä. Tuotteen täytyi olla myös helposti ylläpidettävissä asiakkaan itsensä toimesta.

Insinöörityön aikana toteutin tähän osaan projektista testin suunnittelun, joka suunniteltiin pian muun suunnitteluvaiheen päättymisen jälkeen. Julkisten sivujen testaamisen kanssa oli erityisen tärkeää testata sivuston ulkoasu tarkkaan. Toiminnallisen testaamisen osalta tärkeää oli testata kaikki mahdolliset toiminnalliset osat, kuten painikkeet ja sisällöntuotanto.

Itse testaaminen täytyi suorittaa kaikilla yleisimmillä selaimilla, mukaan lukien mobiiliselaimet. Testaaminen suoritettiin jokaisen osan valmistumisen jälkeen, millä pidettiin huoli siitä, että uudet osat projektissa eivät ole rikkoneet mitään aikaisempaa toteutusta.

Testauksen jälkeen saimme julkaistua asiakkaallemme toimivat ja nykyaikaista ilmettä edustavat sivut.

Projektin kokemukseen pohjautuen opin erityisesti regressiotestaamisen tärkeyden, että jokainen sivuston osa tulee testata uudelleen aina kun jotain uutta osaa ollaan liittämässä sivustolle. Lukuisia kertoja projektin aikana jokin uusi ominaisuus rikkoi aikaisempaa toteutusta. Myös eri selaimilla testaaminen on erittäin tärkeää, sillä monet osat kehityksessä toimivat eri tavalla eri selaimilla, varsinkin mobiiliselaimilla.

Keywords	Ohjelmistotestaus, Microsoft Test Manager, Team Foundation Server, testaus, projektihallinta

Contents

1 Introduction

In this thesis the world of software testing will be introduced to the reader. This thesis will go through common work phases and most common terms from planning tests all the way to driving them and reporting the results. In this thesis Microsoft Test Manager is used for testing.

I was working for Avanade Finland while doing my Final Year Project, in a project for Seafarers' pension fund (Merimieseläkekassa) and I was testing the website our project crew developed. The project itself covered multiple other systems, but this thesis will cover only the part of public website of Seafarers' pension fund.

Software testing is an important part of developing software. It is the first time the software is seen by other than the developer himself, and it will also be the last step before entering the user acceptance test. Testing should be done throughout the developing process multiple times. Nowadays companies start to realize the value of software testing. It can reduce the costs of the project and it also will make the company look better, when most of the errors are found by the developing team and not by the customer. Every single bug or broken feature will look bad for the customer and in the worst case scenario it might cost losing the project to another company.

Every time some block of code is changed or some new feature is implemented, software tests are essential to keep track of possible bugs and errors. If software development is done faster than testing, it might be hard for a developer to find what broke the system, and why something is not working. While testing is done both by the developer himself, and a dedicated software tester, this thesis will be mostly covering the job of the dedicated software tester. However, it will cover the basics for software testing done by developer.

2 Software testing fundamentals

Software testing is often the last time the software is reviewed before it is handed out to the customer. It is really important to find all the defects and bugs before the software is handed out. A tester's job is to ensure that the application is working and behaving as intended and report all possible problems back to the project and developers so they can be fixed before shipping the product to the customer or before the product is launched.

Testing should be part of the development throughout the entire project. At the beginning of a project testing should be planned as well as possible. All the platforms should be listed where the product has to be tested on, and also it should be well specified on the project plan what functionality is required and what the end product will look like. Usually in web applications the most common browsers are Internet Explorer, Google Chrome, Mozilla Firefox and Safari. If the customer has specified some other browsers in the project contract, they should be tested as well.

While testing might seem like a simple part of the process, there are multiple tasks a tester have to do. The tester needs to make sure all the functionality is working as intended, he is using the product like a customer or an end user might use it and test all the functionality of the product in case of problems. This part of the testing is called functional testing.

Usually the first thing a customer sees when they receive the product is the appearance, how it actually looks. The tester is the last one to see the product before it is shipped out, so it is their job to ensure the product looks like it was meant to look like and there are no visual problems with the product. This part is usually referred as visual testing.

When functionality and visual part are probably the most important parts of the process, tester also needs to make sure that the product safety and performance is working as promised. Nowadays digital security is really important, so it needs to be tested well and though out, so customer will not have any problems with security. Also performance is something nowadays a end-users are really interested in, how well it works and how fast they can use it. These are called security testing and performance testing.

Depending on a product, also stress testing might be necessary, when testers are creating virtually multiple users and testing how many users can use the product at the same time. Usually stress testing is big part of web-based services, where too many users are

an actual problem. Also services that are using servers and cloud can face performance problems under a lot of stress on the server.

The user interface and usability of the product is also important. Tester needs to think how client would think, and use the product as the user would use it. If the software has some improvement with user experience or interface, it will be reported back to the developer by the tester running the tests.

2.1 Application Lifecycle Management (ALM)

Application lifecycle management represents the whole process of the application development from making the requirements for the project all the way to delivery to the customer. There are multiple suites for application lifecycle management from various companies and platforms, but I will be using Microsoft Team Foundation Server as my Application Lifecycle Management suite. It is widely used and trusted suite and most of developers and IT-pros are familiar with it. It includes good tools for planning, developing, testing and reporting the project. It is easy to get reports on progress whenever customer requires an update of the project development. It is also easy for project management since tasks can easily be assigned from one developer to another. Also it has great tools for software testing included, such as Microsoft Test Manager and built in tools for unit testing.

There are also plugins for cross-platform development, such as GitHub [4] can be merged with Team Foundation Server, if developers are more familiar with GitHub and customer does not have a problem with it. Some customers prefer to have all their information in local servers, when GitHub is not so good project management tool since it works mostly online. Securitywise there have not been any major problems with GitHub, but especially working with sensitive data, such as social security numbers and customer data, offline services in the development phase is strongly suggested.

If the application lifecycle is well documented and managed, it is easy to report progress to the customer. With a project where customer is paying by working hours, it is really important to report all the used hours to the customer and then customer can see where all the time is used and how the project is progressing.

2.1.1 Governance

At the governance phase of application lifecycle management the project team and the customer will discuss about the goal of the project. What will be included and what kind of services the company provides to the customer. It is really important to report all of these requirements and get them accepted by the customer so there will be no misunderstandings between the developer company and the customer.

The goal of the project will be the first thing to document. What the customer needs and what tools will be used to produce the product for the customer. Project plan is a brief overview of the whole project and its components. Well documented project plan is the script of the project where everything else is based on.

Every single feature, layout and functionality will be documented on separate document to ensure that everyone included in the project will be informed and on the same page. Documentation for the functionality is called Use Case [2] and it describes how the functionality will be used and all the possible outcomes of the function and also possible error situations and how the application will handle possible errors. Well documented use cases can save up a lot of time, since sometimes testers are hired without a professional knowledge of the business model of the customer, so use cases are also testers script for planning the test cases. It should describe in detail how the software will be used.

The layouts of the project will determine how the product will look like once it is done. These are important to make before anything is developed, since it is easier to develop to a certain outlook instead of changing the whole appearance once all the functionality is done. Also in the middle of the project when the customer is reviewing the project, poorly made appearance in development face might effect on how pleased the customer is on the progress. When most users can see only the appearance of the software instead looking the functionality under the hood, it might give a bad first impression to the customer.

After everything is well documented and all the parties of the project are happy with the scope, everything will be broken into a pieces of tasks and the timeline of the project will be presented to the customer. The timeline of the project will be determined depending of the methodology used in the project. Different methodologies will be covered later.

There should always be extra time for all the phases. When customer receives the project before the deadline, it will give a positive image of the development progress.

All the tasks will be assigned to a person in the project, and they will all be put down to the Team Foundation Server, where all the tasks can be tracked and they are easily reported when the customer wants to get an update how the project is developing.

2.1.2 Development

When entering the development phase of the application lifecycle management, in an ideal project the developers are clear about what they are developing and when. At the first phase of the lifecycle management everything was well planned and well reported and tasks were made to the Team Foundation Server, where developers can pick a task and start working on it.

While developers are working on the application itself, software testers are planning how to test everything developers do. In well planned and executed project most of the test planning will be done even before the development starts, but usually there is changes to the project and functionality so test planning will continue while the application is getting its form. Every single use case and layout will have their own tests and they can be planned based on the use case and layout documents done on the development plan.

Usually developers are testing their own functionality while working on it, which is called unit testing, where every function and class is tested separately and it is usually done by the tester himself. However every time developer implements something new to the application, it might break something that is previously implemented in the project. That is why testing is done multiple times throughout the development process and one final time after one area of the application is finished.

Every time some task is developed by the developer, it will be handed out to the tester and tested for possible bugs and defects. If there are any problems with the part of the application, tester will initiate a new task and hand it back to the developer who tries to fix the problem.

In Team Foundation Server it is really easy to get different kinds of statistics from the project. The project manager is usually the one who is responsible for reporting about

the process to the customer, and it is easier to visualize than just explain how the project is moving forward.

After the development phase is over, the project will be reviewed once more by all the parties included in the process. Once software testers report all their test cases executed and passed, all the bugs are fixed and everything works as intended, it is time to hand the project out to the customer for the User Acceptance Testing (UAT) where customer tries to find all the rest of the bugs and problems with the product.

2.1.3 Maintenance

After the product is handed out to the customer it will enter the maintenance phase of the lifecycle management. When the customer receives the product to user acceptance testing customer starts to go through the product and decides if the product is what they were expecting and what they paid for.

Even though in a perfect project there are no more problems once the product is handed out to the customer, the project will have to be prepared for customer-reported defects from the customer. Usually the testing team will do their best to find all the bugs before the product is handed out to the customer, but sometimes something just slips by. When the customer finds a problem with the product, they will report the problem and development team will try to fix it as fast as possible. Microsoft Test Manager together with Team Foundation Server is a good tool for working with bugs. When customer reports a bug, the project testing team will try to validate and re-produce that bug. If the bug can be re-produced and it is valid, it will be reported as a new bug, and assigned to a developer who is working with that area of the project so they can start working to fix it. This time the development team can use their time with only valid bugs and time can be saved for next the phases of the project.

Also sometimes the project requirements might change after the product is handed out. Customer decides to add some functionality to the product or to change something already implemented. Then the project management and customer will discuss further how, when and how much does it will cost to implement, and the process will start all over.

2.2 Benefits of testing

Testing is an essential part of a software development process. If testing is well planned and executed following good practices and guidelines it can speed up the development process and lower the development costs. While developers are working on a sprint in agile development, testers can test and report all the bugs so developers can fix them as soon as possible, which will reduce the risk of bugs getting buried and more functionality worked on top of the faulty code.

Usually better quality sells better. That is why software testing is such a crucial part of the development process. It is important that someone who has not been part of the actual developing tests the software before shipping it out, since it is really easy to get used to some minor errors and problems when one works with the same software for a longer period of time. Then when someone outside the development process takes a look at the software, even the smallest mistakes will get caught before the software is shipped to the customer. Usually there are still some problems even when the customer uses the software, but the biggest and most embarrassing mistakes usually get caught in the testing phase.

Even after the software is shipped out, the testing team is usually an important part of the project. Right after a launch is usually the crucial time, since users of the software might not be so familiar with it, and there usually are reports of problems with the software. Here, testing team tries to validate and re-produce those errors and problems, and help the user to work with it. Often the problems cannot be re-produced by the testing team, and then the problem will be still reported, but it is not a high priority problem unless there is multiple reports with the same problem. However if the problem can be re-produced it will be reported as a bug, and transferred to a development team for a fix. This way team can save time and money from the customer, when developers can use their time and energy only with a valid problems, and one time user errors can be ignored.

3 Testing methodologies

3.1 Testing techniques

3.1.1 Manual testing

As the title describes, manual testing is testing which is run manually by testing team. Manual testing does not need any automation of test cases and every single test is run manually by a tester. The tester opens the software and tries how/ whether it works [1].

Even though there is a human running these tests, it is really important to prepare test cases and scripts well so tester can test the entire software and there will not be any parts untested. Most often it is suggested that the test planner and tester are two different persons, since while preparing the test cases it might start to be a routine, so some problems or bugs might be overseen by the person who planned the test case. Especially when testing is done by a separate person the test script needs to be in such detail, that tester can run the test case without any problems. Figure 1 illustrates Microsoft Test Manager test planning.

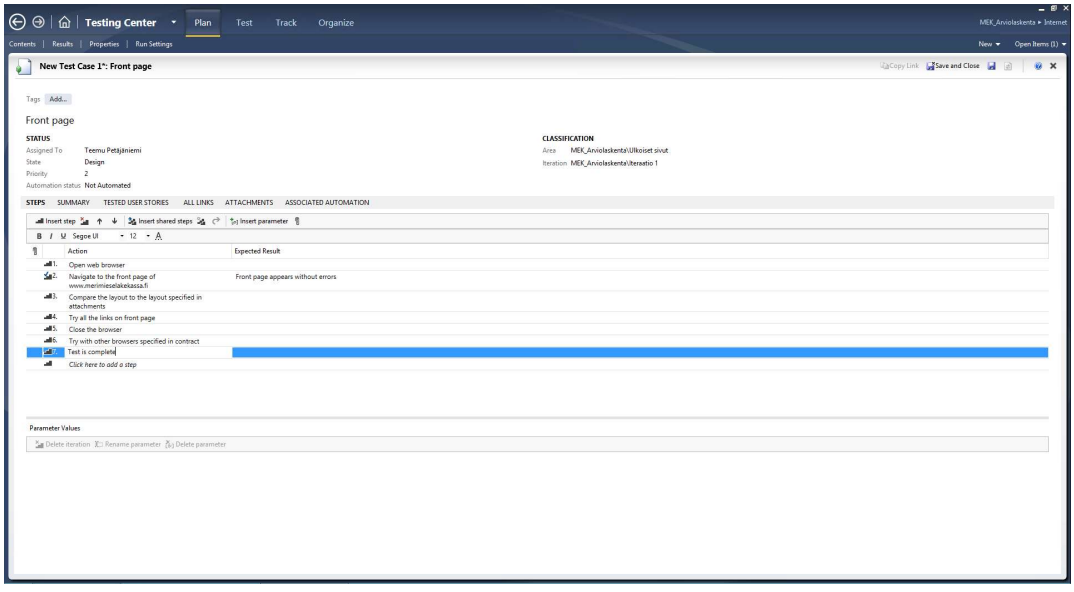

Figure 1 Planning a test case in Microsoft Test Manager 2013

Usually reporting is in a more detailed level in manual testing where there is no automation involved, but manual testing also costs more, since there is more people needed in the process. It also takes more time, but sometimes projects are using testers also for other parts of the process to save costs.

3.1.2 Automated testing

Automated testing is testing that is automated and programmed before the actual testing starts. These days automated testing is usually what the customer wants since usually it is more cost-effective in bigger projects. In smaller projects usually the cost difference is not the deciding factor, since writing automated tests takes a lot of time. Also like the actual software, there might be mistakes in programming the test, so programmed automated tests might not be the most reliable testing technique.

The Microsoft Test Manager allows testers to record the steps while they run test cases as shown in figure 2. This is probably the easiest way to make automated tests and usually this type of testing is called coded UI tests. It does not require any programming or scripting; tester just runs the regular test and the software records the actions users do on the software. In this way the user can come back to a test case and run desired steps of the case and continue from where it failed the last time. [1]

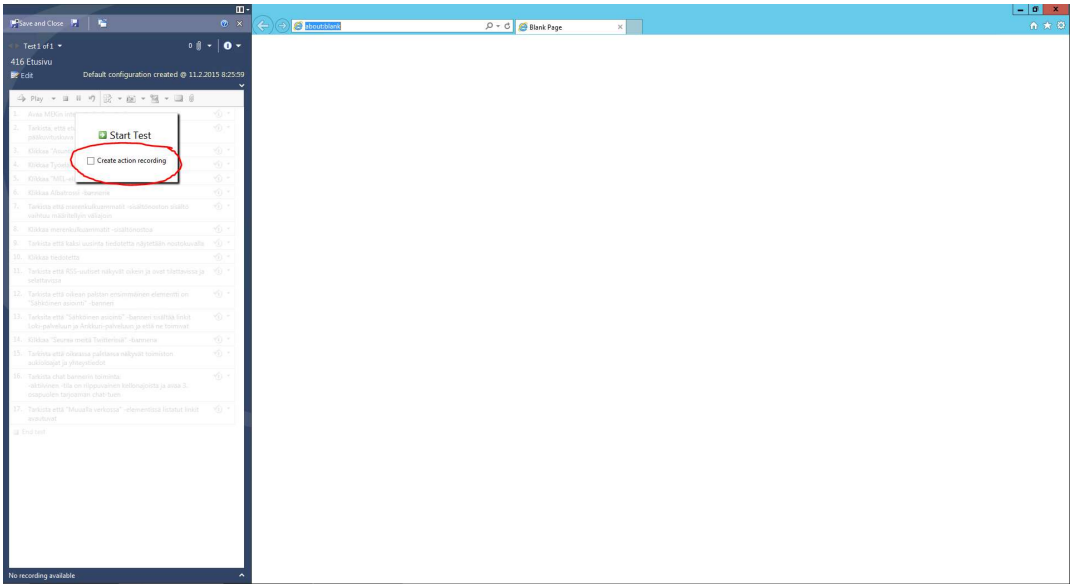

Figure 2 How to create action recording in Microsoft Test Manager 2013

Also for visual testing automated tests are not usually the best way to test. Even though there is software that can tell a difference between two pictures or location of a picture or difference in font, it is still not as reliable as a manual tester who can compare the software to a layout that was made before the development started.

3.1.3 Black box testing

Black box testing is testing that does not require any knowledge of how the software works under the hood. It is also called end-user testing, where tester is acting the part of the end user, and tries to use the software like the customer will be using it.

Usually when talking about software testing and usually people who are hired to be testers are mostly doing black box testing. Black box testing will test the software visually and functionally in a way how customer will see it. Black box testing usually starts after the whole development process is done and the software is ready to be tested before shipping it to the customer.

3.1.4 White box testing

White box testing is testing that requires the knowledge of how the product works internally. Usually white box testing is done by the developer himself, since it requires access to and knowledge of the source code of the product.

White box testing can be used for individual functions and parts of the program, that might not even be visible to the end user after the development is done. White box testing is usually used for background processes which are not possible to test individually while using the actual software.

Even though white box testing is usually done by a developer, depending on the project, there might also be testers who have technical knowledge of and skills for programming and software development. In this way the developer will not have to use their time and resources for testing and testing will still be done before the product is shipped out.

3.2 Testing levels

3.2.1 Unit testing

Unit testing is done by the developer. Unit testing is testing where developer tests the product part by part. One small piece at a time. While user uses the product and does something with it that looks fairly simple to the end user, it might trigger a series of a

functions under the hood, where multiple functions are called in a series where one function output is an input of the next one. In this series if one function or part of the code is not working as expected, the end result is not what the end user desired, and from a regular testing point of view it might be hard to identify what causes the problem. In unit testing every single function and part of the product is tested separately, where faulty input cannot affect the outcome of the function. [1]

3.2.2 Integration testing

After each block of the product is tested, the developer starts connecting these individual blocks together. Even though two separate functions work well by themselves, it does not mean they work well together. That is why developer needs to test their code every time they connect or integrate their functions and parts of the code.

Integration testing is also usually done by developer while he is working on the product. However sometimes there might be a dedicated person for both unit testing and integration testing who is working closely with the developer to ensure the most effective development process.

Both integration and unit testing is black box testing that requires high-level knowledge of how the code works.

4 Test planning

4.1 Waterfall model

Waterfall model of the project works much like waterfall. Work is done in one continuous flow, where next task is started directly after another. In waterfall model of development planning should be well done even before starting the project. After the project plan and use cases are made, developers start to work on the project and test planning can start. In waterfall testers work at the same time with developers, and in waterfall multiple working environments are important, so testers can work while developers are debugging without interrupting anything critical. [2]

Test planning in waterfall model is made mostly in the start of the project, right after general project plans and documentations such as use cases are made. After they are well documented test team starts to plan their tests. In waterfall model it is really important to remember to update test cases and plans, if the project encounters problems or for some reason needs to change the scope or the plan of the project. Regression testing is also extremely important when some part of the application changes after it is already tested. In regression testing tester simply tests already tested part of the product, and makes sure there is no new problems, if the product has changed on the way.

4.2 Agile model

In agile model the project is broken down in to a so called sprints, where a small fraction of bigger project is made and tested. Planning for agile model happens always in the start of each sprint, and once again, test planning happens right after the general plan is made. [2]

In agile model it is easier to test after a development process is done. After developers reports their sprint done, testers can start their job, and report all findings and test results before the next sprint will start. Sometimes developers need to fix some problems with previous sprint while working on the next one if development phase of next sprint is started before testing is finished from the last sprint.

4.3 Creating test cases

Test planning should be done clearly and in a way where anyone can run test cases. While creating a test cases all the steps should be specified clearly and in logical order. Each step should be planned and written so anyone who is a first time user of the application could run the test case and succeed finishing the required task. This is how test planner can be sure every single step is properly tested and documented.

In each step in Microsoft Test Manager there is also specification for expected result of the step. This should only be used when something is actually expected to happen. For example if user is expected to fill out a web for with user information such as name and address, each field should be specified in test steps, but each field should not have expected result. However possible “submit” -button should have expected result for example to send or erase the form when it is clicked. If a step has a specified expected result, Microsoft Test manager will not let user to skip this test and expects the step to either pass or fail. Skipping test steps might be useful if the test case is run while development process is still ongoing and tester is asked to test some specific part of the application.

When steps of the test are clear and there is enough of them, tests can be recorded with Microsoft Test Manager so in future possible re-testing or regression testing can be run almost automatically and Microsoft Test manager knows how to fill out fields and where to navigate in order to test the application just like an actual tester would test it and each field and part of the page is properly tested. Tester can choose which parts of the test case they want to be done automatically. Using the same example from web forms, tester can automate Microsoft Test Manager to fill out the form automatically, but then run the actual functionality by tester himself. So this type of automatization is not only for running test cases automatically, but also helps tester to run test cases faster and in a more efficient way when running the same test cases multiple times.

5 Testing

5.1 Managing tests

Microsoft Test Manager and Team Foundation Server allows one to manage test cases for better flow and organization inside the project [1]. Usually test team consist of multiple testers and test planners, and sometimes they are not in a same building so communication can be problematic. When all the test cases are stored in the same place and organized well, a team working in an off-site project can work well and the team can work without problems. Testing a whole system, all the use cases and all the functionality might be plenty of work for the testing team. Without organizing test plans well, it can be hard to test a whole system without areas that will be unnoticed.

How testing is organized varies between projects and teams, and usually it is decided before the project starts so the project and team can work in the most efficient way. Some teams break down the project to a timeline, when all tests are organized by the time, and others might be focusing more on the parts of the project, such as use cases, visual tests, stress testing or security testing. All these can be categorized in Microsoft Test Manager as test plans and then easily managed by the project management.

Regardless of whether the project testing is organized by timeline or test types, test plans can be furthermore categorized to test suites, where the test planner can organize test cases by part of the program or sprints and where one suite must be completed before the next one can start. Test case organization with Microsoft Test Manager is shown in figure 3.

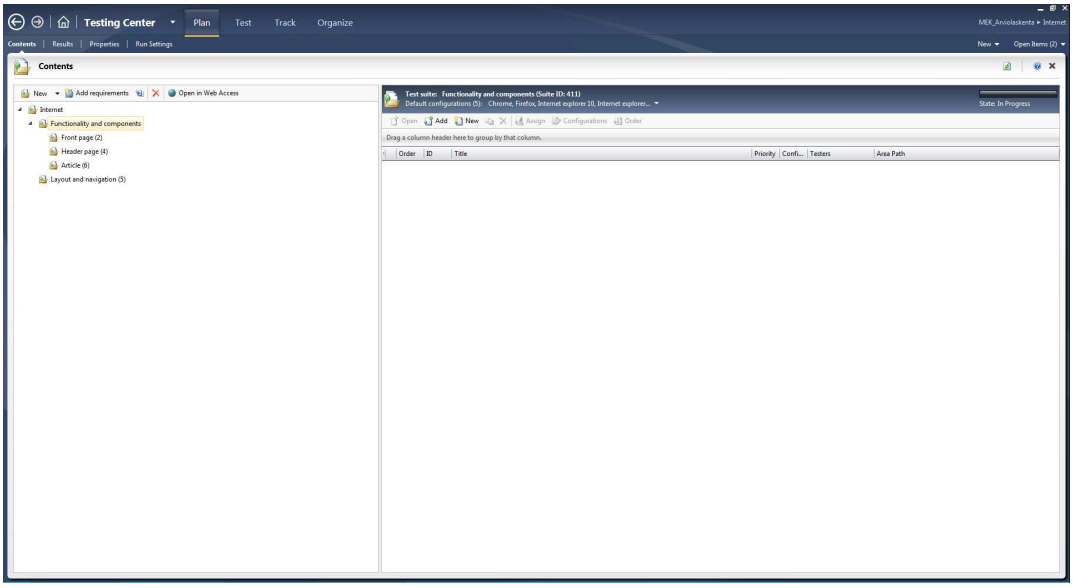

Figure 3 Test plan categories in test suites

5.2 Running tests

After tests are well planned and organized, we can start running the tests. Every time we choose to run the test, the test case needs to have some end result. After you start test case from Microsoft Test Manager, it will either pass or fail. That is why it is important to be sure you are ready to run the test before actually starting the testing process in Microsoft Test Manager. If the test case fails because it is aborted, it is not crucial mistake, but it will change the data on reporting services, and might look bad when reporting back to the customer in the middle of testing process. Multiple failed tests might give customer an impression that things are not running as smooth as expected.

However sometimes development process might take longer than planned, and some bigger test cases might need to be tested partially. Tester can start the test case, skip those steps that are not implemented yet and test the part that developers wish to be tested before continuing on the process. In this way, the test case will fail as whole, but important parts of the test can be tested and bugs reported.

While running test cases Microsoft Test Manager requires quite a lot of resources from the computer, so it is suggested not to run anything else heavy while running test cases. Also if tester is using action recording, it is important not to interrupt the recording by making any extra or external clicks while the recording is on. This will just make the

recorded test run heavier and make extra steps in future testing, if tester wants to use recorded steps while running tests.

While following the test steps each step should be tested unless it is specified separately by test planner that some steps can be skipped, for example in partial testing. If the test step has any data, such as names or other input data for forms, they should be used as they are specified. Figure 4 illustrates Microsoft Test Manager in testing mode.

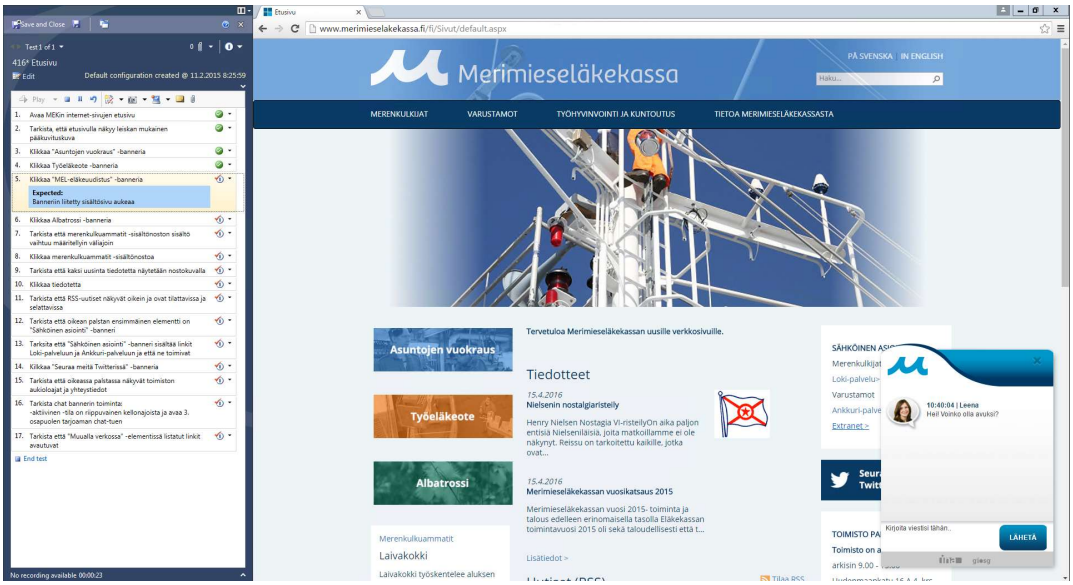

Figure 4 Running a test case

5.3 Reporting bugs

While running test cases tester might run into a bug. The product is not working as it was supposed to work, it might not look right or it might not work fast enough or give some strange output data or error messages. Everything unexpected should be reported back to developers so they could start working on it. Microsoft Test Manager gives good tools for reporting bugs to Team Foundation Server where developers will have as much data from the report as possible. Developers have access to test steps, test history and a large amount of other useful data. Figure 5 shows Microsoft Test Manager bug reporting –site.

When reporting a bug tester should describe everything there can be described from running the test. Also Microsoft Test Manager will allow tester to quickly and easily snap a screen capture and it will be attached to the bug automatically. Screen captures are

probably the most important information especially reporting visual problems with the product.

Other useful information for developers are simply what the tester did to cause such problem with the product. What did the tested do when that specific problem occurred and what kind of data and information was used when testing that specific part of the product. Rather have too much information than too little.

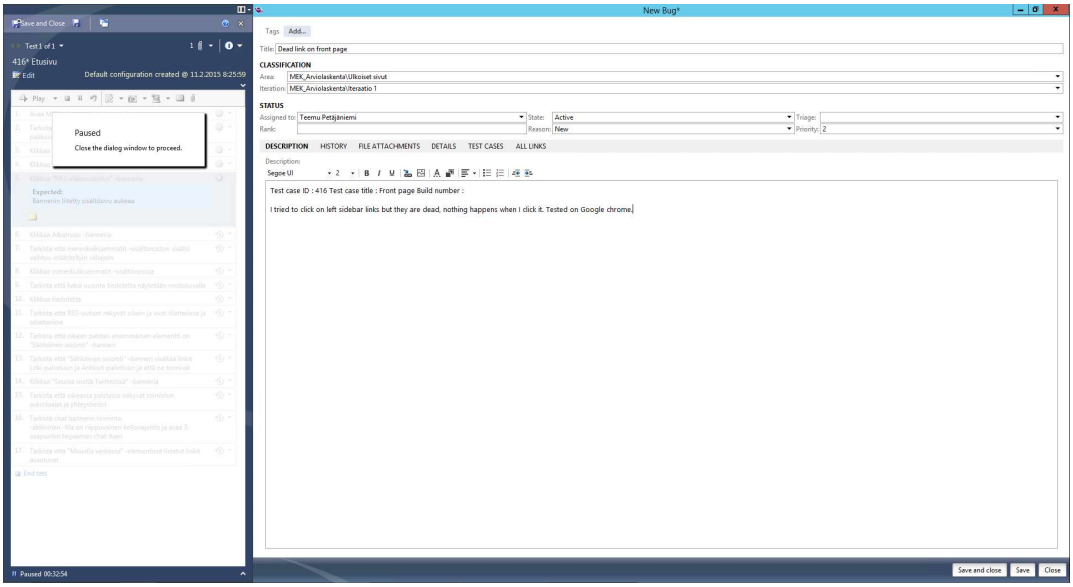

Figure 5 Reporting a bug in Microsoft Test Manager 2013

5.4 General reporting

While the project is under construction, customer might want a status report from the project team. Developers can produce their own report from Team Foundation Server where customer can see how many tasks are done and how much tasks are still undone. This will usually give really good impression to the customer how long it will still be until they get the finished product.

From a tester's perspective reporting can also be useful. Even though customer usually wants to know how many development tasks are done and how many still upcoming, testing report can draw a completely different picture. Even though development tasks are done and closed by developer, they will not show how many problems and bugs those parts of the product includes.

From Microsoft Test Manager tester can produce a clear and well specified report about the testing process, where all the passed tests and failed tests can be seen as a clear chart, so the customer will see how many problems the product has in total.

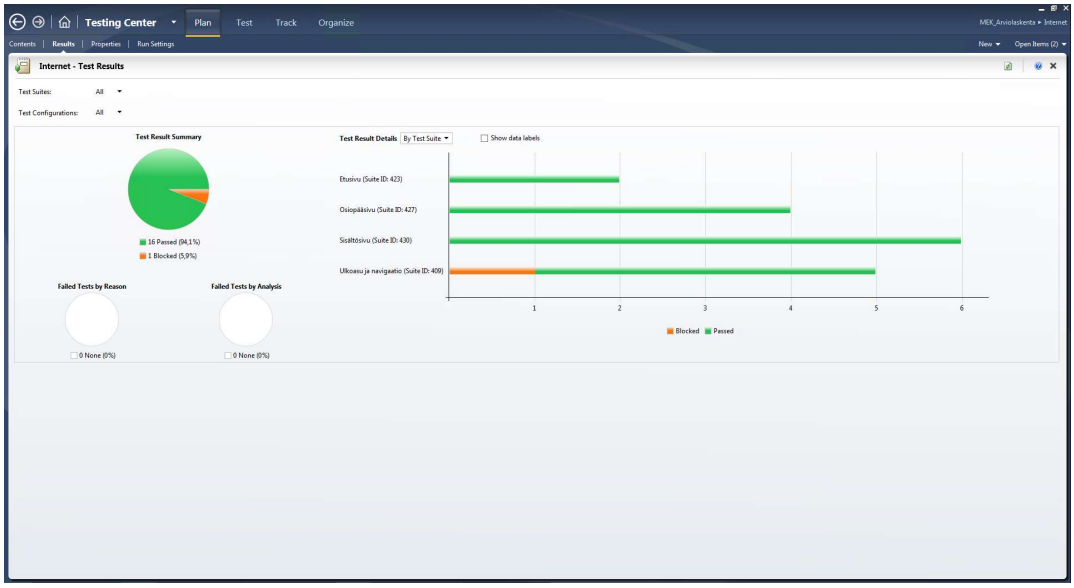

Figure 6 General statistics from test plan

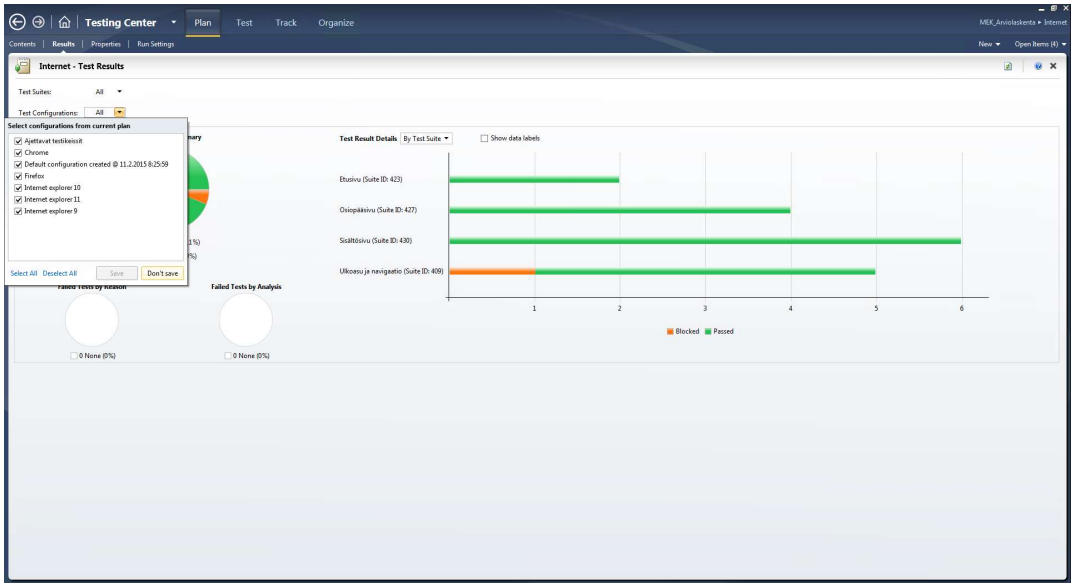

Figure 7 Reports can also be limited to a certain test configuration or suite

6 Case Seafarers' pension fund

Seafarers' pension fund (Finnish Merimieseläkekassa) is a pension fund for seafarers. It is controlled by Finnish pension law and it covers everyone who works at international seas. They also have for example rental apartments for their customers. Avanade Finland creates and updates various tools for Seafarers' pension fund in a larger project, from which this case is part of. Finnish pension law has been changed multiple times in the past decade, and pension funds needs to update their systems and tools. [5]

Testing team in whole concludes two testers and one test lead. Since the project contains multiple different tools and services, responsibility needed to be divided logically. Different parts of the project have been assigned to different testers, in order to keep testing and reporting as good as possible.

6.1 Introduction

This part of the project developed a new public website for Seafarers' pension fund to replace their current website. The new website should be modern, functional and easy to update and control for the customer.

Customer chose SharePoint to be the platform for the new website, because it has already included tools for managing the site. While it is true that the SharePoint has really well implemented controls for managing a website or even a collection of sites, it does not have the best tools for customizing the platform. For example, the style and look options of SharePoint is usually seen mostly on company internal networks and similar site collections. We had to make custom changes to the SharePoint platform to make the website more usable for the end user.

A public website is not usually made so much for sharing documents or other main functionality of the SharePoint platform, so the project team had to do multiple changes so the platform eventually started looking like a public website. Even though the team had to make multiple changes to the style of the site, SharePoint has really good and easy to use tools for other parts of the site. For example, page navigation and content management tools were already made, so the project team had to make just small adjustments to the platform in order to get it work as the customer wanted.

When I entered the project, test planning had just started. I got the website test planning and testing as my responsibilities as soon as I entered the project. My responsibility was to make the test planning and execute these tests. After testing was done, my job was also communicating with developers to get all the bugs fixed, and also I have been helping customers with the User Acceptance Tests (UAT).

6.2 Project planning

Even though the project might not seem like it would be complicated to do, it still has to be planned carefully. Project team and the customer has to be in agreement what the end product will look like and what kind of functionality the site contains. Especially when the customer already had an old website, new features and functionality needed to be specified well.

This kind of company usually does not have too much functionality on the website for the end user, but even so, functionality needed to be documented and specified. The most important planning phase was definitely the looks of the website. Really specific layouts contained all the parts and objects of the website, and it gave the customer a picture of what the end product will look like.

When looking at the site it does not look like a typical SharePoint -site which usually contains mostly shared documents and company internal information and communication. So it was clear at the beginning, that our developers would need to work hard to make the website look good so the customer would be satisfied.

Compared to the old website the new functionality mostly contained updates to the modern day. Nowadays companies often have an online chat for the customer service, and also different social networks are a new way to keep their customers posted on news and information. So links to all these services needed to be added to the website.

Also a responsive website was something the customer wanted. Nowadays end users are using more and more different kind of devices to access the services, such as tablet computers and mobile phones. Screen resolutions and sizes varies from side to side so the website should be usable with all of them. For laptops, tablets and desktop computers the site will look the same, but it will adjust the navigation and the information according to the screen size and resolution, while the mobile phone website will look completely

different so it is usable for also those customers who needs to access the site from their mobile phones.

While most of the managing tools were already in the SharePoint –platform, still some new tools needed to be made. Customer should be able to update custom information such as information regarding their rental apartments and information about the different positions in the field of business. While almost everyone who works at an international ship is their customer, they have a large variety of jobs in the field. Customer wanted to show the end user different kind of jobs other people were doing in the sea. These custom tools are called Web Parts in SharePoint platform.

6.3 Test planning

Even though testing in this project is not so technical testing, it does not have much functionality and website is more information based than functional, it still needs to be tested well. Especially when the site is built with custom layout on top of a SharePoint site, the appearance and features needs to be tested even more carefully than normally.

Test planning of the public website did not have any use cases, and test planning was made accordingly. Most of the test cases are for visual testing, but they have much more variations in browsers and screen resolutions than many other systems.

Project also had to make multiple custom Web Parts for SharePoint. Customer wanted these custom Web Parts to add better user experience and to make the site look better and to serve their customers better. These Web Parts are essential to test properly, so the user experience would be as good as it can be. Web Parts includes a box for the first page, where Pension Fund is presenting different professions in the field.

Content production is fully customer's responsibility. Even though content production tools are provided by Share Point platform, testers need to make sure it works as intended. Customer adds all the content to the site, but our responsibility was to make all the pages and their layout, so some sample data was required to add, and at the same time we tested the content production. The website includes for example site for contact information, which was produced by making a table Web Part which customer can modify.

Other functional testing includes everything on the website that has functionality on it, such as links and menus. All the links and menus needs to be tested on all the sites in the website. Even though some link works on the front page, it might be broken on other sites. Same thing goes for menus, even though they are basically same element, it might act differently on different parts of the website, www.merimieselakekassa.fi.

6.4 Testing

As mentioned earlier, responsibility of testing the website was only on one tester, so all the testing was made by one tester, and reported to our test lead while it progressed. Testing itself was fairly straight forward, since the planning was well made. Different areas of the website were reported as ready to test by developers in different phases, and we had to make a couple of changes to the test plan in order to be able to test some functionality without worrying about something that has not been made yet. Regression testing was also really important when working on the other parts of the website after some part was already tested and reported as functional.

6.4.1 Visual testing

When the custom layout started to be ready to test, I started working with developers in order to get the layout done. Tests were done multiple times during the process to make sure everything looked as planned.

Especially when the layout was completely custom made, it was really important to make sure it worked with the SharePoint as it was intended. Especially the mobile website caused some extra work from the developers, since menus and usability needed to be designed for a mobile devices and touch screens. Tester also needed to keep in mind that the mobile site was used with touch screen, so for example hover was not something that needed to be tested in the menus.

Testing with different browsers and screen resolutions was most important part of the visual testing. Since browsers are not always working the same way, we needed to make sure the site was designed to work on all the common browsers. Also people uses different screen resolutions, so responsible website is important to work as intended on different screen resolutions.

6.4.2 Functional testing

As I mentioned earlier this part of the project did not have any use cases, so the functional testing was somewhat easier. Functional testing included mainly testing if all the links and Web Parts were working as intended.

As I said in the test planning, all the links and menus needed testing on all the parts of the website they were visible. All the tests were ran in all the different locations of the website. Every single link and menu item was tested on every single page.

Custom Web Parts took little more testing than SharePoint included functionality, since they were custom-made. Sometimes the testing of these parts was a little challenging, when functionality was something the tester was unable to affect. For example, on the front page there is a Web Part that rotates information about different professions in the field. The content producer adds the information and picture of the profession, and chooses the dates this profession is visible. Testing needed to make sure it worked properly. Testing was done in two different phases, where in the first phase the tester added the content and published it, and the second phase was to make sure, on the selected date, that the content was changed.

The testing team also made sure the content editing and publishing was working well, even though it is included functionality on SharePoint. We had so much custom made for the website so testing team made sure nothing we made broke the functionality that was already included.

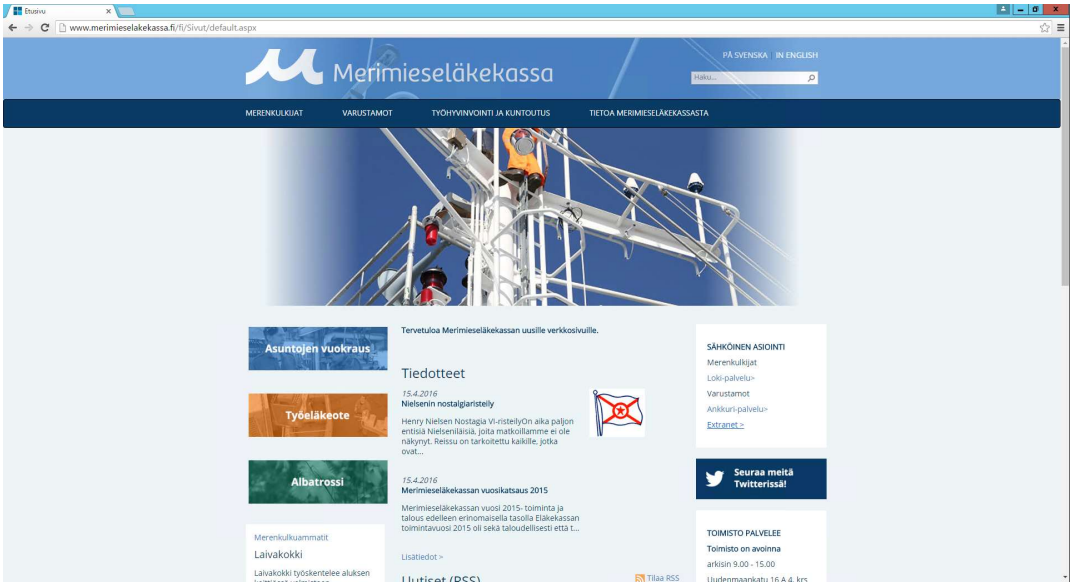

Figure 8 Finished product

7 Conclusions

In the project for a public website for Seafarers' pension fund software testing was an important part of the development process. It helped us to save time and money when problems were reported in time for developers, so they were able to fix them as soon as possible. The project team encountered some problems with custom style sheets for SharePoint, and these problems caused the project timeline to be postponed. Without the test team inside the project, these problems would not have been noticed before the product was shipped to the customer, so testing also helped Avande Finland to maintain a good customer relationship with Seafarers' pension fund. Excluding these style sheet problems in the project, it was carried out quickly and cost effectively.

As the case study shows, software testing is an important part of the development process. It usually helps to save time and money when problems are noticed and reported as quickly as possible. A good relationship with the client cannot be measured in money, and when the product is delivered without visible problems to the customer, it will help to keep up a good relationship with the client.

Even though software testing has always been part of the development process, good practises and official guidelines are a relatively new field of software development. Finding good resources and references might be hard for software testers. Software testing as an official part of the development process is growing rapidly, and the availability of software testing information and tools are getting better.

8 References

1. Professional Application Lifecycle Management with Visual Studio 2012 (Gousset, Keller, Woodward 2012)
2. Erik Downing, Thomas Dawkins. (2014). Software Testing Fundamentals. [online video]. 20 March. Available from: https://mva.microsoft.com/en-US/training-courses/software-testing-fundamentals-8305?l=mjfPZiYy_8604984382. [Accessed: 11 January 2016].
3. Anthony Borton, Steven Borg. (2013). Software Testing with Visual Studio 2012 Jump Start. [online video]. 20 June. Available from: https://mva.microsoft.com/en-US/training-courses/software-testing-with-visual-studio-2012-jump-start-8562?l=iYTwt110_9804984382. [Accessed: 10 January 2016].
4. Matthew Mitrik, Andy Lewis, and Martin Woodward. 2013. Getting Started with Git in Visual Studio and Team Foundation Service. [online] Available at: https://blogs.msdn.microsoft.com/visualstudioalm/2013/01/30/getting-started-with-git-in-visual-studio-and-team-foundation-service/. [Accessed 14 January 2016].
5. Changes to Kela benefits in 2015 - News archive for customers - kela.fi. 2016. Changes to Kela benefits in 2015 - News archive for customers - kela.fi. [online] Available at: http://www.kela.fi/web/en/news-archive/-/asset_publisher/lN08GY2nIrZo/content/changes-to-kela-benefits-in-2015. [Accessed 1 April 2016].