A comparison between UFT and LeanFT

LUND UNIVERSITY
Campus Helsingborg

LTH School of Engineering at Campus Helsingborg
Department of Computer Science

Bachelor thesis:
Viktoria Andersson
Zohal Turial

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden

Abstract

This thesis was an investigation to survey how the new automated testing tool LeanFT compared to the older tool UFT. The thesis was an exploration into the tool LeanFT, examining how well LeanFT performed with GUI testing, what features the tool offered and how usable it was for a tester of software.

The purpose of this thesis work was to come to a resolution on whether the LeanFT tool was a more suitable alternative than UFT and to determine if LeanFT should be used on its own or in conjunction with UFT. To come to a conclusion on these matters, the thesis workers created tests in both UFT and LeanFT, tested the LeanFT connection with the version control tool Git and the continuous integration tool Jenkins. To make a fair comparison between these tools. the tests cases for the tests created in LeanFT and UFT, were identical.

The results of the investigation regarding the two tools led to the conclusion that LeanFT can be used instead of UFT. LeanFT is quite a stable tool and no specific bugs were found during the investigation. The results of the thesis workers' investigation regarding the ability of connecting LeanFT projects to Jenkins was that the connection was possible and did not take a very long time to set up. Connecting LeanFT projects to Git also worked well and they did not find any major issues during the process.

The thesis is also written as a guide for people starting to work with LeanFT, it contains sections on how to set up a LeanFT project and on how to create GUI tests in LeanFT.

Keywords: UFT, LeanFT, GUI testing, Continuous Integration, BDD

Sammanfattning

Detta examensarbete är en undersökning för att se hur det nyare automatiserade testverktyget LeanFT jämför sig med det äldre verktyget UFT. Arbetet är en undersökning av verktyget LeanFT, det undersöker hur väl LeanFT fungerar med GUI-testning, vilka funktioner verktyget har och hur användbart det är för en testare av mjukvara.

Målet med detta examensarbete var att undersöka om det är möjligt för IKEA IT att använda det nya verktyget i sina projekt när de testar ny programvara. Examensarbetet kommer också att undersöka mer allmänt om LeanFT är ett bra verktyg att använda istället eller i samband med UFT. Det undersöker också vissa funktioner som LeanFT har t.ex att ansluta till versions verktyget Git och det kontinuerliga integrations verktyget Jenkins.

Resultaten av undersökningen av de två verktygen var att LeanFT kan användas istället för UFT åtminstone i GUI-tester, det är ett ganska stabilt verktyg och inga specifika buggar hittades vid undersökning av LeanFT. Resultaten av elevernas undersökning av möjligheten att ansluta LeanFT-projekt till Jenkins var att kopplingen var möjlig och tog inte mycket tid att inrätta. Deras undersökning av möjligheten att ansluta LeanFT-projekt till Git fungerade också bra och de hittade inga större problem under processen.

Arbetet är också skrivet som en guide för personer som börjar arbeta med LeanFT för första gången. Det innehåller avsnitt om hur man skapar ett LeanFT-projekt och hur man skapar GUI-test i LeanFT.

Nyckelord: UFT, LeanFT, GUI testing, Continuous Integration, BDD

List of contents

Foreword

We would like to thank Tomas Bylander for mentoring us during this entire process and Pontus Nelderup for giving us the opportunity to work with this thesis work at IKEA IT TestCenter.

We would also like to thank Emil Långberg and Aakanksha Dave for helping us with technical issues with UFT and ALM, Fredrik Hjort for giving us a lecture on ALM and all other personnel from IKEA IT met during the thesis work, for their assistance during the course of completing our thesis work.

We would also like to thank our supervisor, Christian Nyberg for always being easily accessible and our examiner Christin Lindholm.

Chapter 1: Introduction

1.1 Background

IKEA Group is responsible for global support and business development of the IKEA Retail organisation.

Test Center under Shared Services within IKEA Group, support IKEA Business Solutions with development and delivery of relevant test services. IKEA IT´s Delivered services are, among others, service virtualization, test data management, test automation, performance test and test tools in order to work with continuous improvements within the test and quality area.

The Test Automation Service delivers automated functional tests to Business Solution teams, projects and programs. The purpose of test automation is to reduce the time and resources required for testing, thereby speeding up the execution of tests, and increasing the overall coverage of testing (e.g. regression tests, smoke tests). The components that can be utilized within the service include feasibility studies and development and delivery of automated scripts.

Universal Functional Testing(UFT) is a program licensed by Hewlett-Packard and is a functional automated testing software. IKEA has a lot of legacy code/tests in the program UFT for different projects, and a lot of competence when it comes to UFT. One of the main challenges of UFT is the difficulty in adapting to a more agile development process, which is becoming the standard in most software development projects. This means that the testing process is no longer considered as a separate phase but instead as an integrated part of the entire development process. To achieve this software teams need to be able to test continuously and use a good version control system for the tests, which would facilitate the agile development process. This is, as of now, a struggle with UFT. At present, UFT requires a Visual Basic script implementation. The current lack of use of Visual Basic script has led to difficulty in filling staff positions. This has also led to IKEA IT needing to devote time to teach VBscript to their new staff before they can start to work with UFT.

LeanFT is Hewlett-Packard's improvement of UFT, where agile development is intended to be more easily applicable. LeanFT is built specifically for continuous integration and continuous testing and also uses more modern programming languages for its implementations, such as Java and C#. This would in turn eliminate the problem with recruitment at IKEA IT.

UFT is not solely used by IKEA, but is a common testing tool for many companies. As previously stated, agile development, including agile testing is becoming more common in the IT world. The hindrance of this by the UFT program is making companies look for other alternatives, LeanFT being an example.

This thesis work can therefore be useful for these companies, in search of more agile friendly test automated tools.

1.2 Purpose and Goal

1.2.1 Purpose

The purpose of this thesis is to investigate the benefits, adaptability and flexibility of LeanFT and also conclude if IKEA should shift focus from UFT towards LeanFT as a go-to test automation tool. This report will be a great help for both IKEA IT and other companies using UFT, on making a decision whether to focus on LeanFT over UFT. The report is also intended to be used as a manual for the IKEA IT TestCenter staff, when and if they choose to work with LeanFT.

1.2.2 Goal

The outcome from the thesis should be a clear indication on whether test automation at IKEA should shift focus towards LeanFT. To emphasize that LeanFT can be used as the new test automation tool an example of a test program using LeanFT shall be implemented written in C# or Java. It will be implemented as a test case with generic functions where logic is incorporated in element interaction and by doing that, displaying the strength (and possible weaknesses) of using LeanFT.

1.3 Problem Definition

The problem definitions are prioritized by rating them on a scale of 1 to 10, 1 is the highest priority and 7 the lowest.

1. LeanFT maturity level – is the usability of LeanFT comparable to UFT? Any major bugs which hinder the use of the LeanFT tool?
2. Public community for LeanFT - is there a public community for LeanFT where one can get assistance if needed?
3. What is/is not possible with LeanFT when comparing with UFT?
4. LeanFT – standalone – will there be any lost functionality if deciding to leave UFT, can LeanFT be used without UFT?
5. Is it possible to create BDD tests using LeanFT? - as shift left is the overall approach for development at IKEA, a BDD way of thinking around test development is of course something that everybody wants but can it be done with LeanFT?
6. Can one incorporate a better versioning system (e.g. Git) with LeanFT?
7. LeanFT – use together with Jenkins and create a packaged deal as deliverable,

distributable to projects together with ALM connection and shared libraries for utility functionality, is it possible?

1.4 Motivation for choice of thesis work

The purpose of the thesis seemed valuable for IKEA and for other companies working with test automation. UFT is not solely used by IKEA, but is a common testing tool for many companies. This thesis work can be used by not only IKEA IT but for all other companies sitting in the same situation as IKEA IT, deciding on whether to switch focus to LeanFT as the main test automation tool.

1.5 Limitations

This thesis will not explore these problems.

- Conversion of VBScript to C#
- LeanFT - investigate if IKEA IT can and should continue with the current way of working (with UFT) with LeanFT?

Chapter 2: Technical Background

This chapter contains descriptions of the different tools used in the completion of the thesis work, such as programs, programming languages and other technical tools. The chapter provides either a basic description of the tool or an in-depth description on how to do the specific setup used by the thesis workers.

2.1 UFT

UFT(Universal Functional Testing) is a software program used for creating and executing functional tests on software. It is licensed by Hewlett-Packard(HP) for use mostly by companies developing software.

UFT is used to automate testing and help developers and testers to collaborate. UFT makes it possible to manage multiple tests simultaneously which can result in a faster and easier testing process for larger companies [1].

UFT can be used for both API testing and GUI testing. where the functionality and behaviour of the application/website needs to be tested. It is important to analyse the application that is meant to be tested, determine what actions that should be tested and what resources are required for the test. Based on the demands of the test, different resources are required. For an example a GUI test usually requires resources that include object repositories, which is a container that can contain test objects, and function libraries, where functions needed for the tests can be added.

The creation of a GUI test in UFT consists of several processes:

1- Creating object repositories
2 - Adding steps to the test, which represents the desired user actions in the application
3 - Creating functions in a *function library* to use in the test
4 - Creating checkpoints to validate specific objects in the application
5 - Parameterizing the values of test object to see how the applications reacts to different input values
6 - Running and analyzing the test

Create GUI test

- To create a GUI test, click on the Add button down arrow in the UFT toolbar.
- Select New Test
- Select GUI test, as the test type and name the test, see figure 2.1..
- Click save

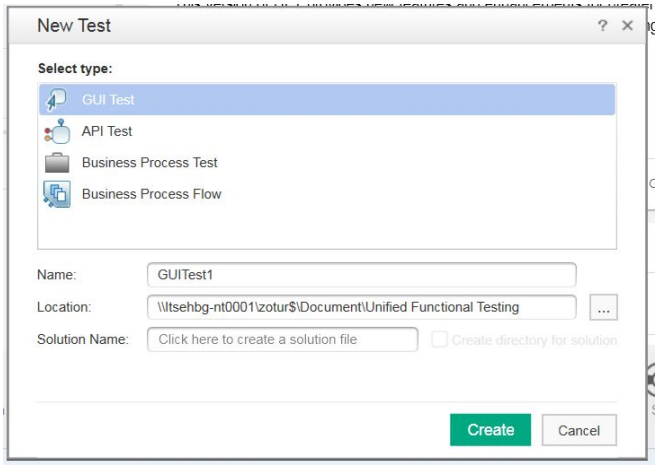

Figure 2.1: UFT GUI test

Before creating the steps in the test, the user is required to create actions, which are meant to be used in the various steps. Each GUI test consists of calls to actions. Actions help with dividing the test in logical sections which in turn helps the user to design more efficient, modular and understandable tests. When a new test is created, an empty action is created by UFT.. However, when a test requires more actions, this then has to be created by the user. There are several ways of adding actions to the test. An action can be a call to a new action, a call to a copy of an action or a call to an existing action.

Create new actions

- Click on the test in the Solution Explorer, see figure 2.2
- Right-click anywhere on the canvas and select Call to new action

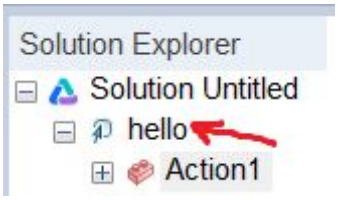

Figure 2.2: UFT test

2.1.1: Object repositories

After creating the structure of the test, there is a need of creating an object repository. The objects of the application, learned by UFT, should then be added to the object repository. UFT uses these test objects to recognize the objects in the application and create test steps based on these objects. When UFT has learned an object in the application, it adds a corresponding test object to the object. The object

repository can therefore be seen as a storehouse for the test objects. When running a test, UFT will look in the object repository, belonging to the test, for the objects contained in the test steps.

When an object is added to an object repository, UFT does the following:

- Identifies the UFT test object class that represents the learned object in the application and creates an appropriate test object.
- Reads the current value of the object's properties in the application and stores the list of identification properties and values with the test object.
- Chooses a unique name for the test object

Create object repository

- Click on Resources in the UFT toolbar and select Object Repository Manager
- To create a new object repository, click on the new button, see Figure 2.3

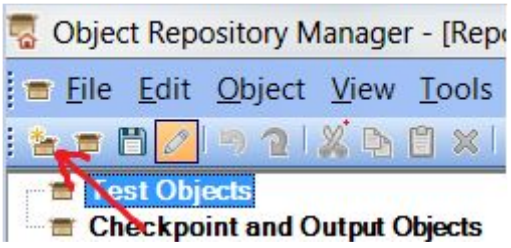

Figure 2.3: UFT Object Repository

The objects can be added in the repository in three different ways. The different ways are by using either the spy feature or the Navigation and learn feature. To use the Object Spy feature, click on the Object Spy button 

With the spy feature you can elect specific objects by hovering over the objects on the webpage.

- Drag the Object Spy dialog box to the side to be able to see the objects on the page
- Click on the pointing hand button 
- Hover on the object that should be added to the repository

The elected object can then be added to the object repository by clicking on the Add to Object repository button 

The other option is to use the Navigate and learn process. In this method UFT is instructed to learn all of the object of the pages that the necessary test objects exist. After the objects are added in the objects repository, the unnecessary objects can be deleted.

To add object with the Navigate and Learn feature the following steps should be performed:

- Click on Resources>Object Repository Manager
- In the new object repository, click on Object>Navigate and Learn
- In the Navigate and Learn toolbar, click on the Define Object Filter button 
- In the Define Object Filter dialog box, select all object types and click OK
- Click on the application title bar to bring it into focus as the page you want UFT to learn.
- Click Learn

An alternative to using Object Repositories is to use descriptive programming, where the test objects are coded by the tester.

2.1.2: Adding steps to a test

When creating steps to add in the test, the test objects need to be used in the test steps and UFT has to be instructed on what actions to perform on the objects. UFT can then translate the test objects methods into actions and UFT will be able to replay the actions performed on the application. The steps can be added in different ways, for an example with the record function, that records the actions performed by the user on the application/website, or adding steps by using the toolbox pane.

The recording feature is used by clicking on the record button.  UFT will then open up the application, or a new browser navigating to the website that the test is meant to be performed on. Thereafter the user can choose what steps to add to the test by executing them on the website. When the recording is stopped, all actions executed by the user will be added to the action page in UFT, in form of code. See figure 2.4

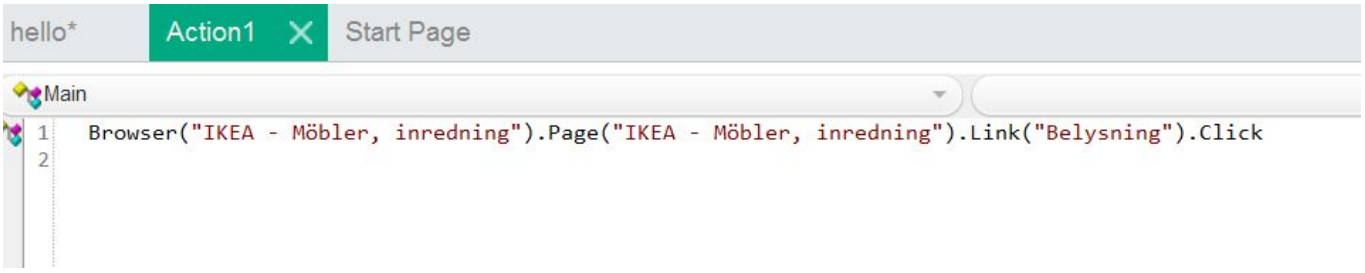

Figure 2.4: Code provided by the record feature

2.1.3: Function Libraries

Creating function libraries are optional. Functions can be used if there is a need of a particular task to be performed in the test. An example of a function can be a task that checks the date format of a flight booking website. Creating functions are also beneficial if the task needs to be performed repeatedly. By placing this function in a function library, that works as a container of functions, the functions can be easily accessed from here. After creating the function library and adding the relevant tasks for the test, the function library must be associated with the test.

Creating checkpoints to validate specific objects in the application

A checkpoint verifies that the expected information is displayed in the application by searching for specific values of a page, object or a test string.

Parameterizing the values of test object to see how the applications reacts to different input values

To test how the application performs with different values, the values of the test object can be parametrized. This step is optional when creating tests.

2.1.4 Running Test

After the steps has been added, the test is ready to be executed. In the application, it is possible to see UFT perform each line in the code, see figure 2.5.:

Figure 2.5: UFT execution[25]

When the test run is completed, the results are shown in a UFT report, in a separate tab. An UFT report can look like the report in Figure 2.6.

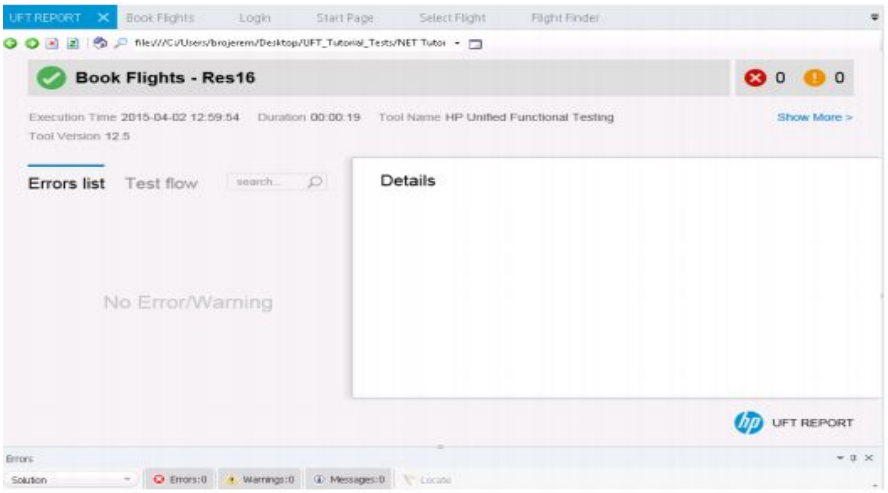

Figure 2.6: Running UFT tests[25]

For the complete UFT guide see[25]

2.2 Visual Basic

Visual Basic is a programming language developed by Microsoft and first released in 1991. Visual Basic is developed from the programming language BASIC which is an older which was a general purpose high level programming language developed in the 1960's. There are several versions of Visual Basic, it started with version 1.0 in 1991 and the last version of 6.0.

Visual Basic was supported by Microsoft until 2008 when they stopped supporting the runtime and development environments for all releases of Visual Basic, 1.0-6.0.

Visual Basic was developed to be a good tool to use when developing GUI's and was intended to be easy to use by newer programmers. It has developed into a language used connecting to databases and in some web programming, it has therefore been used with other Microsoft development tools in many ways, see [22] for more information on Visual Basic.

2.3 LeanFT

LeanFT, or UFTPro as it is also called, is a newer testing software licensed by HP. LeanFT is HPs improvement for UFT where agile development is intended to be more easily applicable for example, by enabling testers to use continuous integration tools, such as Jenkins[4]. LeanFT is built specifically for continuous integration and continuous testing. LeanFT also uses more modern and popular programming languages for its implementations, such as Java and C#.

LeanFT works across multiple IDEs such as Visual Studio and Eclipse. Which helps testers and developers to collaborate more while using the same tools for both development and testing. To aid the tester in the construction of tests it has an Application Model which is like an Object Repository in UFT.

2.3.1 Object Repositories/Application Model

The equivalent of the UFT object repository in LeanFT is the Application Model[5]. LeanFT can also be used without an Application Model but Application Models help with larger projects.

The Application Model contains a number of test objects which are relevant to the application that is being tested, such as links, buttons and other objects. These objects are located in the class Application Model. This class can be created inside the LeanFT project, or in a separate project which can then be linked to any test project.

There are different ways to add objects to an Application Model, by filling in the properties of the object in the Application Model Editor which exists inside Visual Studio when it is configured for LeanFT, see figure 2.7.

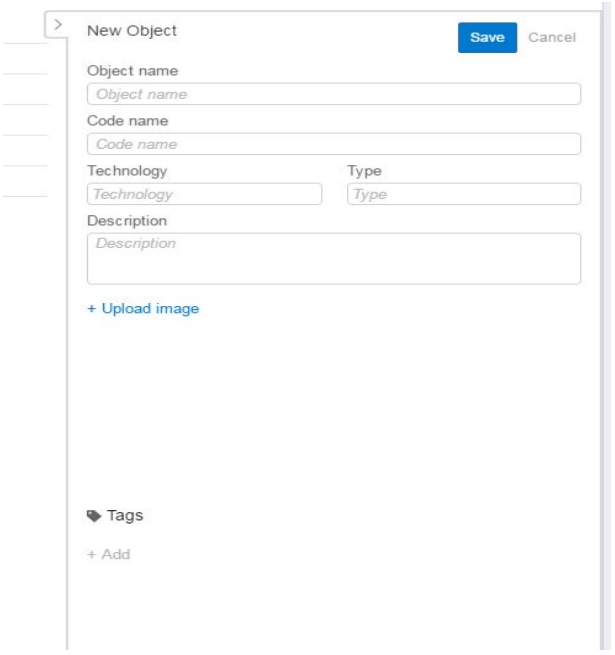

Figure 2.7: Application Model Editor

Objects can also be added by using the OIC(Object Identification Center). This is a tool which exists

inside Visual Studio when it is configured for LeanFT. Figure 2.8 shows how the OIC looks when the IKEA webpage object has been selected and added.

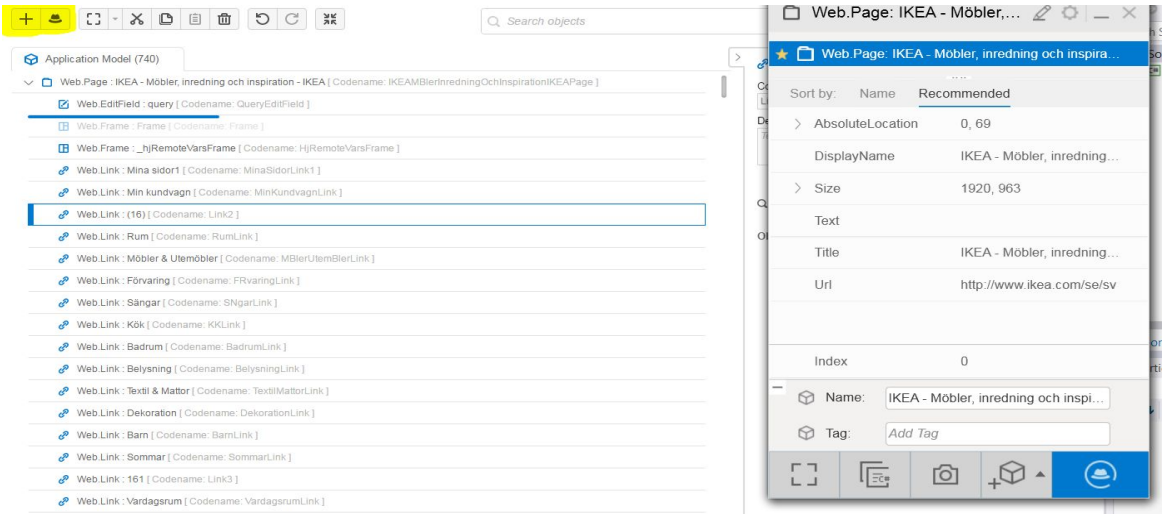

Figure 2.8: OIC in Application Model 1

The OIC captures objects on a specific page and then adds it with an inherent hierarchy to the Application Model. It captures the current details of a test object when you "spy" on the object with the OIC. The OIC can either be used to capture specific objects on a page or all the children objects to a parent page, for example all links and other objects on a specific web page. These objects are then added to the Application Model. See figure 2.9 for an example of an Application Model with parent and child objects.

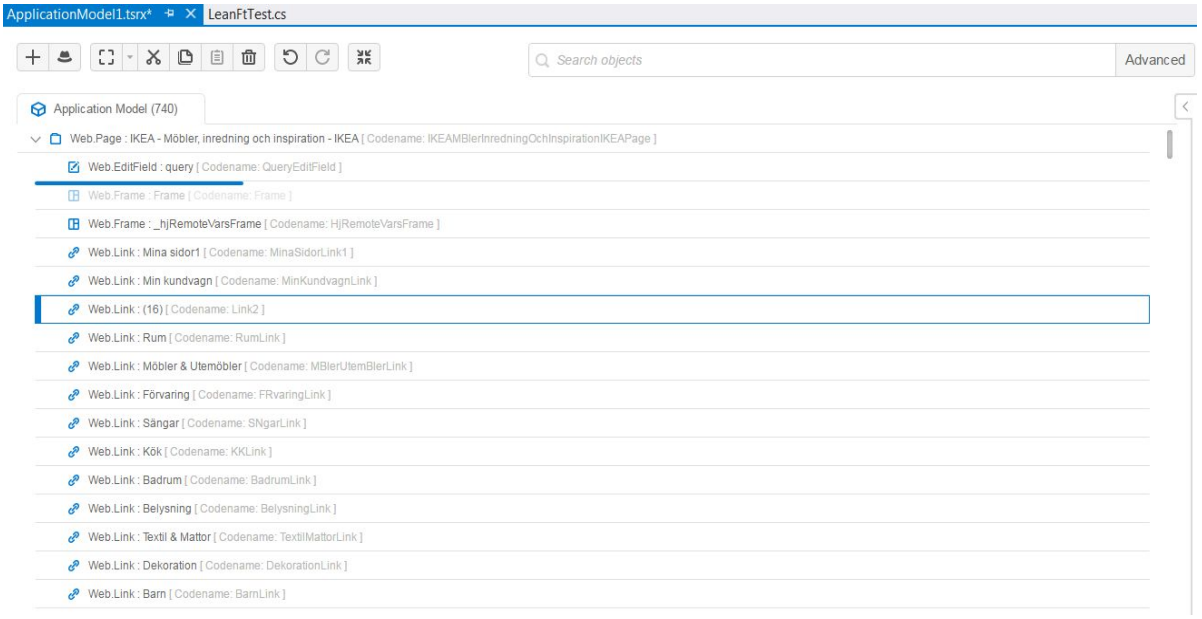

Figure 2.9: Application Model objects 1

2.3.2 Step by step guide to set up for GUI test

To create a GUI test in LeanFT like creating a GUI test in UFT consists of several steps.

1. Start Visual Studio
2. Create new leanFT NUnit project
3. Set Up NUnit in Visual Studio environment
4. Create Object Repository/Application Model
5. Create test steps for the test

Before starting the guide to creating a LeanFT test make sure that there is a LeanFT with a valid license installed on the computer.

Step 1: To start using LeanFT with Visual Studio, first open the Visual Studio program. When it has opened the program the Visual Studio start page comes up, which can be seen in figure 2.20 in chapter 2.6.

Step 2: To create a LeanFT project, go to new project and choose Installed->Visual C#->Test->LeanFT NUnit 3 Project, as seen in figure 2.10. Then choose a name for the project and a new LeanFT project will have been created.

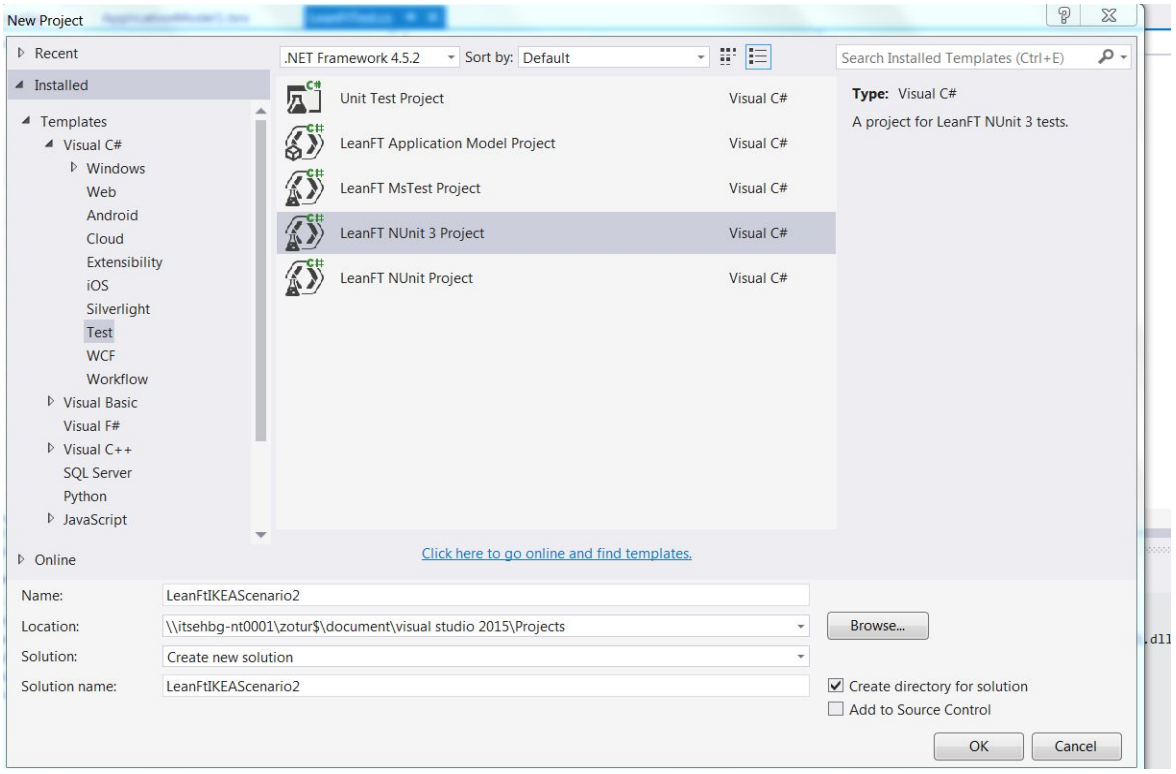

Figure 2.10: Create LeanFT project

When the new LeanFT project in Visual Studio has been created there will be a page which contains a template for a NUnit test. After the template has come up, set up the testing environment.

Step 3: Inside Visual Studio go to the menu bar and press on Tools->Extensions and Updates, see figure 2.11 for an image depicting this. This is done to run the test as a NUnit test and to be able to use some of NUnits functions it must be installed in the IDE.

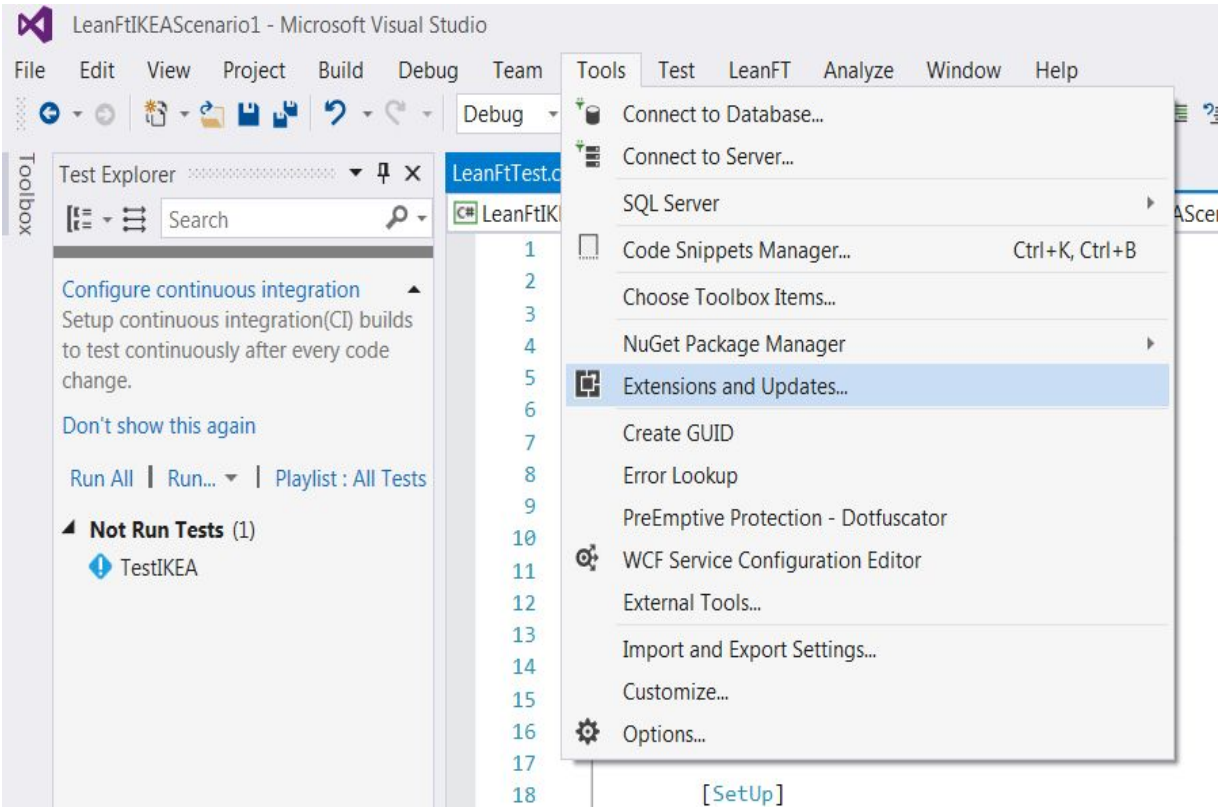

Figure 2.11: Extension and updates

Step 4: After step 3 is finished, the Extensions and Updates window opens up. First click Online on the left and then in the search bar in the top right write in "NUnit". When it has finished searching for NUnit many options will appear, choose "NUnit 3 Test Adapter" and then click "Download" as seen in figure 2.12. The version of NUnit test Adapter chosen must be the same as the Template chosen when creating the project.

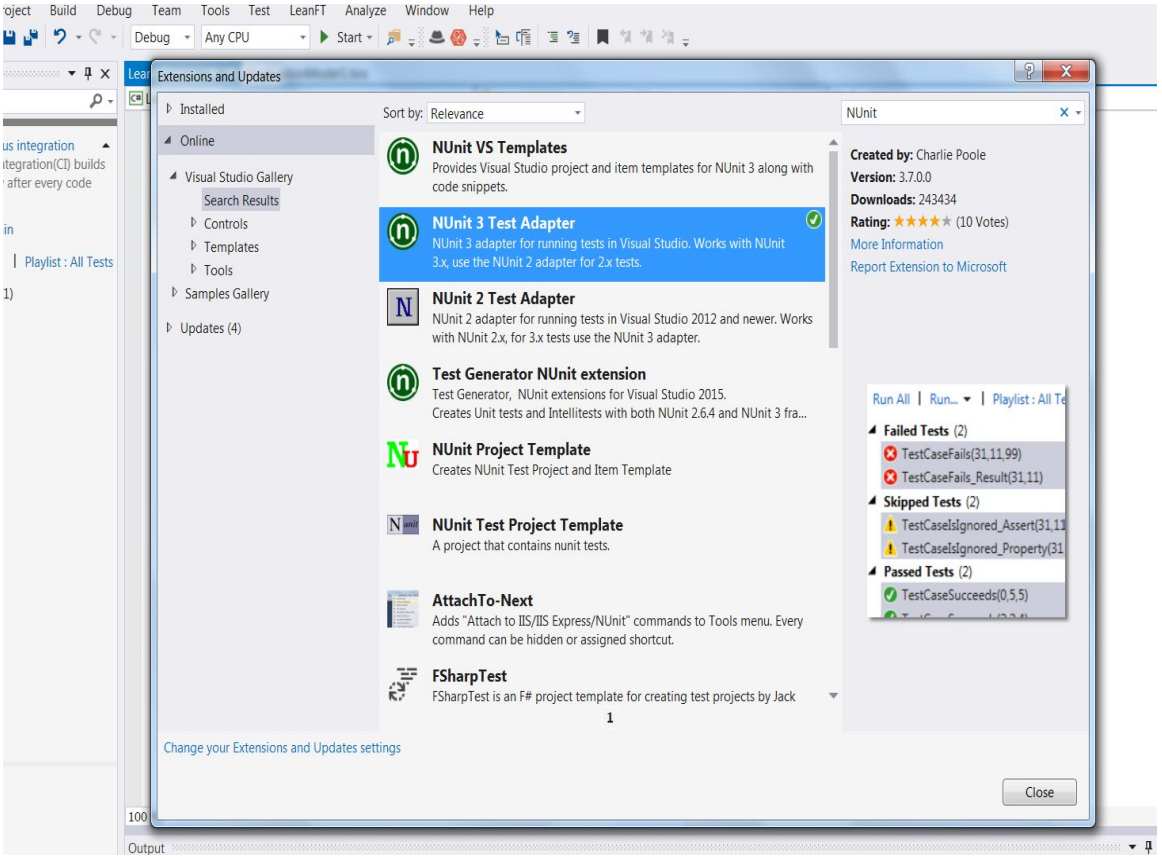

Figure 2.12 NUnit 1

Step 5: Then to further set up the environment after pressing "Close" on the Extensions and Updates window go to the Menu bar again and press Tools-> NuGet Package Manager -> Manage NuGet Packages for Solution, as seen in figure 2.13.

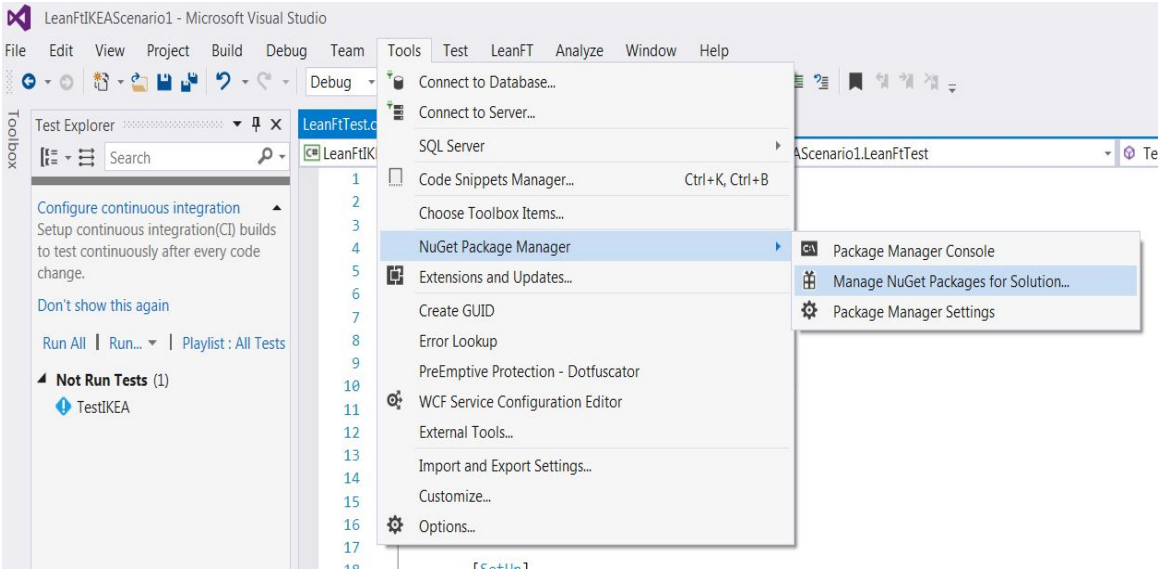

Figure 2.13: Open NuGet Package manager

Step 6: Then the window for managing NuGet packages opens up. First see to it it that "Browse" is highlighted on the left then search in the search bar for "NUnit". Then choose "NUnit by Charles Poole" as seen in figure 2.14. When it has been chosen click in the box for the current project then choose "Install".

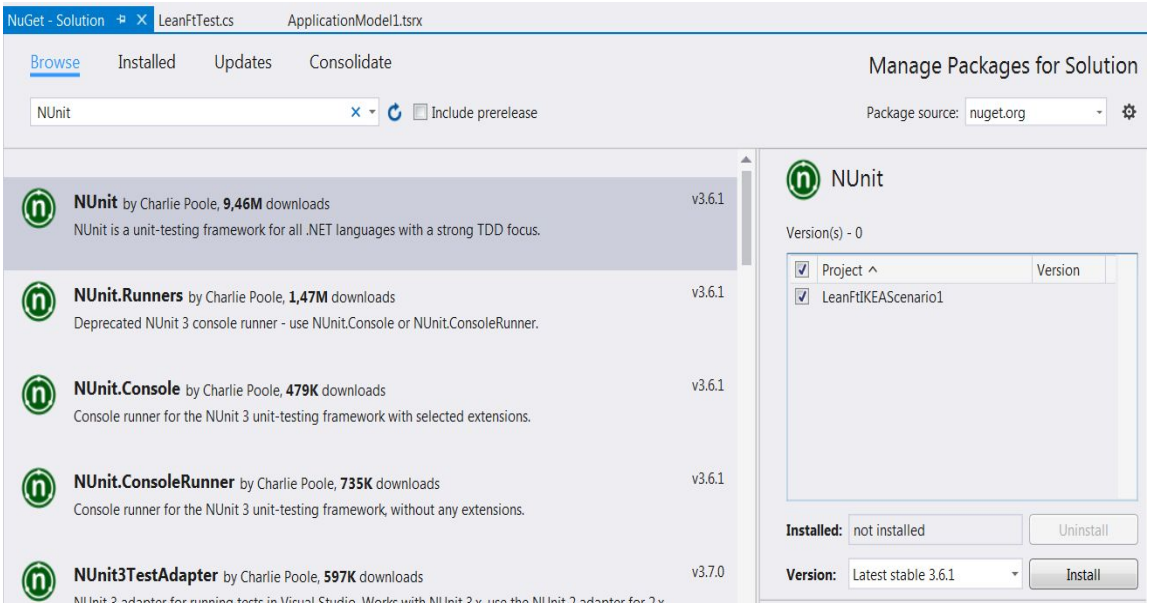

Figure 2.14: Install NUnit package

Then the set up for the project is complete. The next step is on how to set up to use an Application Model with the project, this step can be optional there are ways to get objects into the project without using an Application Model, you can get by just using the OIC for generating codes to objects but if there are many objects in an application and you use those objects often in the tests then it is easier to create an Application Model.

Step 7: To create an Application Model in the project go to the "Solution Explorer". It should be in a box on the right side of the open Visual Studio window, see figure 2.15.

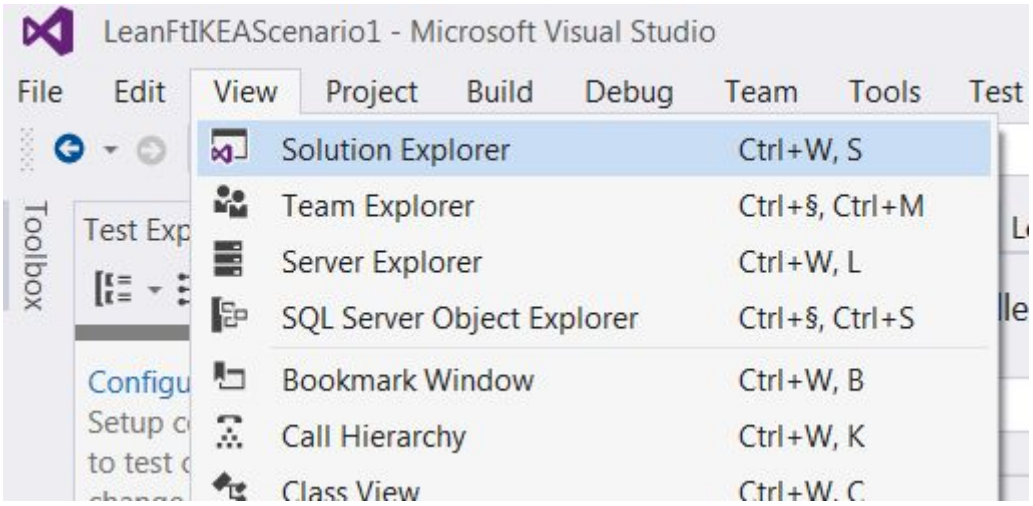

Figure 2.15: Open Solution Manager

Then in the Solution Explorer you will see the project and the different files which it contains. Right click on the project name and then choose Add LeanFT->Application Model as shown in figure 2.16.

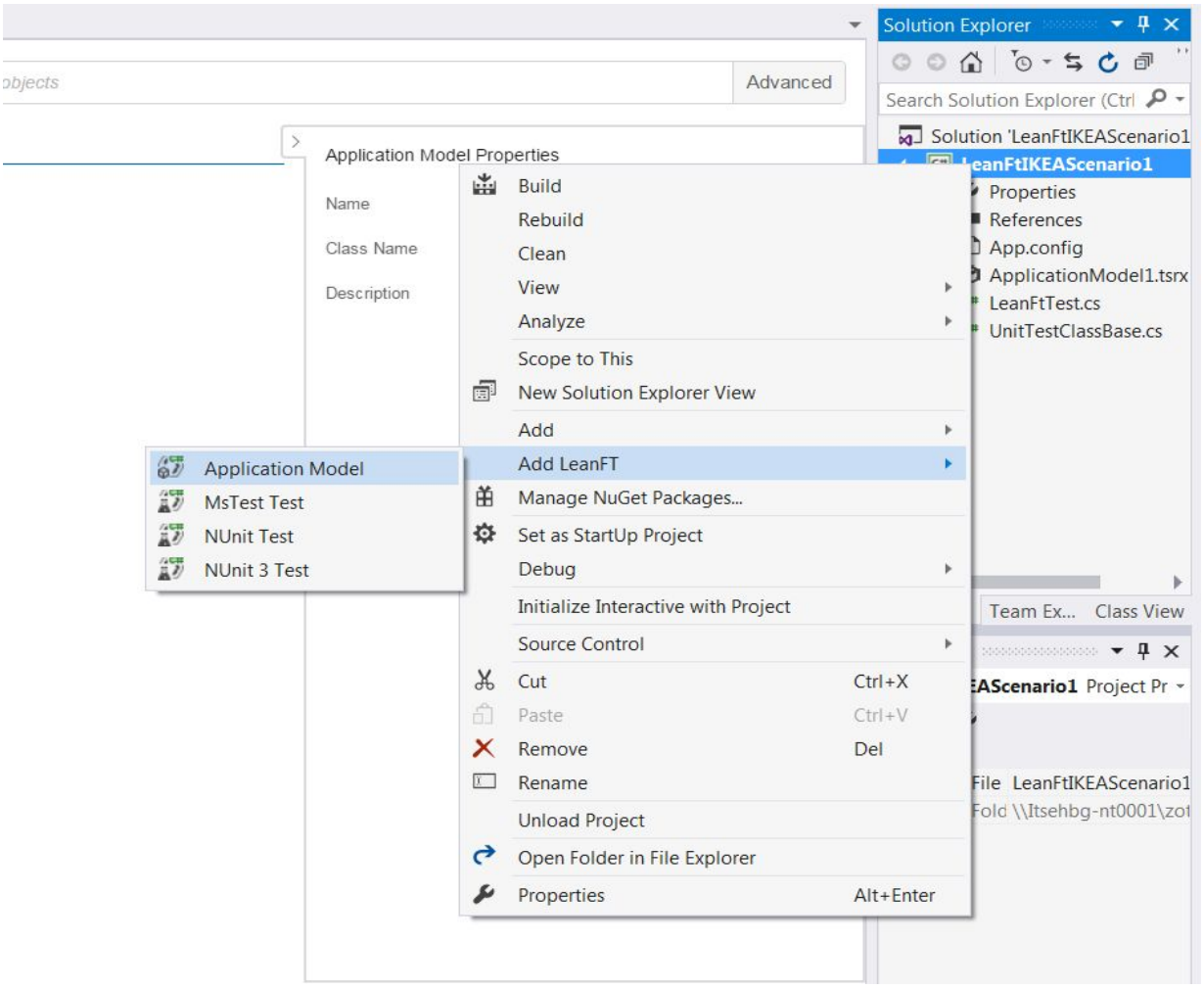

Figure 2.16: Add Application Model

This finishes the guide to set up LeanFT to be able to start making tests.

2.3.3 Adding steps to a test

To run a test in LeanFT the test project must contain test steps which LeanFT then must follow when it runs through the test. These steps determine what LeanFT should do with the tested application. If it should press a button, go to a specific webpage and so on.

To add steps to a test in LeanFT when using NUnit template you add code, either manually or from the Application Model or OIC. The OIC and the Application Model have functions to generate code that can be pasted in the project.

All code is added to the same place in the code, see figure 2.17. It is all added in the .cs file which is where all the code is located and where the test later is built from.

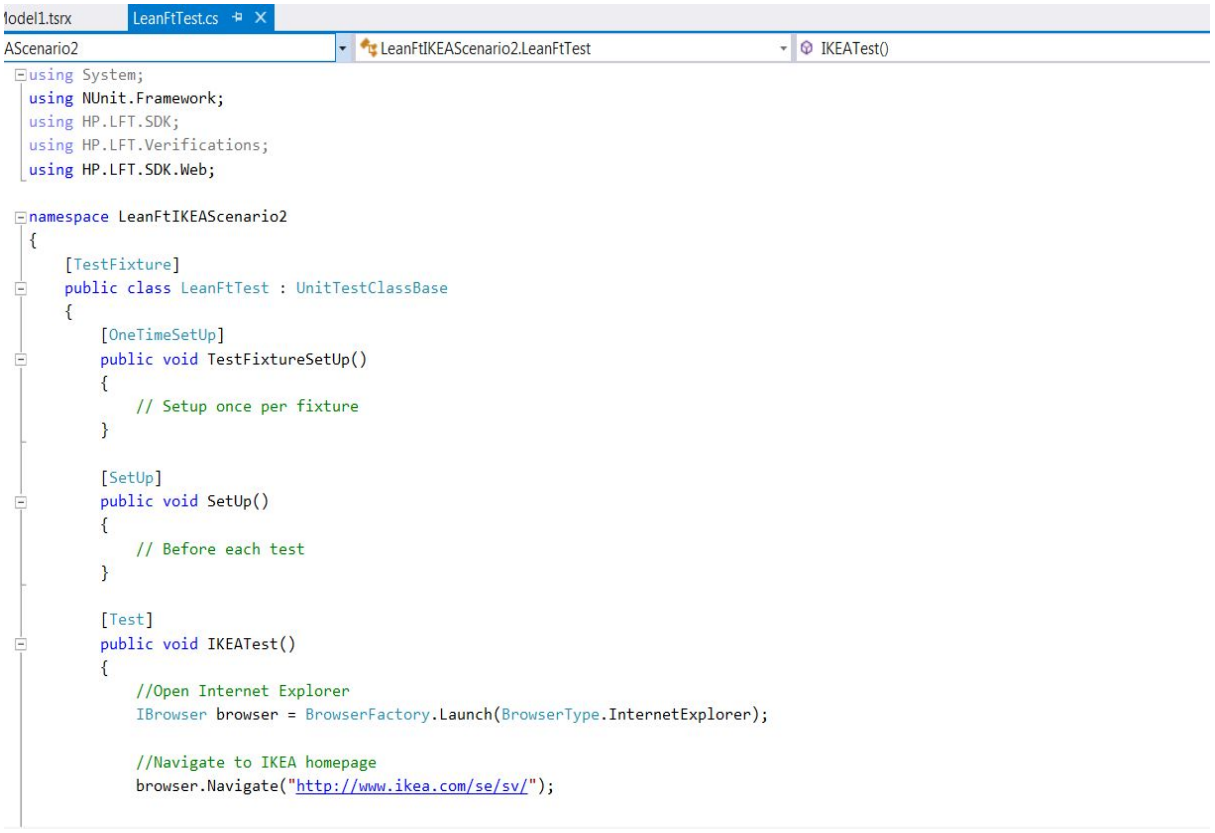

Figure 2.17: LeanFT cs file

All steps which are part of a test gets added inside the [test] public void MyTest{ *this is where the test steps are*}. This is for LeanFT to know that the steps inside are for a test and not for setting up a test or something else.

2.3.4 Analysing test run results

LeanFT generates a report for the user after each run through of tests, this report gets generated even when a test run fails. The report contains useful information for the tester such as:

- If the test went through as a fail or a pass
- All test Flow steps, with icons indicating passed/fail/warning

- The time it took to run through the test
- Which version of the test tool used
- The exact time the test run was started
- A list of all the errors located during the run

To see the report go to the menu in Visual Studio click LeanFT->See latest test run report.

In LeanFT you can also add a setting to add pictures, called snapshots, to the report. It then takes a picture of every test step in the test flow and adds it to the appropriate place in the report. See figure 2.18 for an example of a report with snapshots.

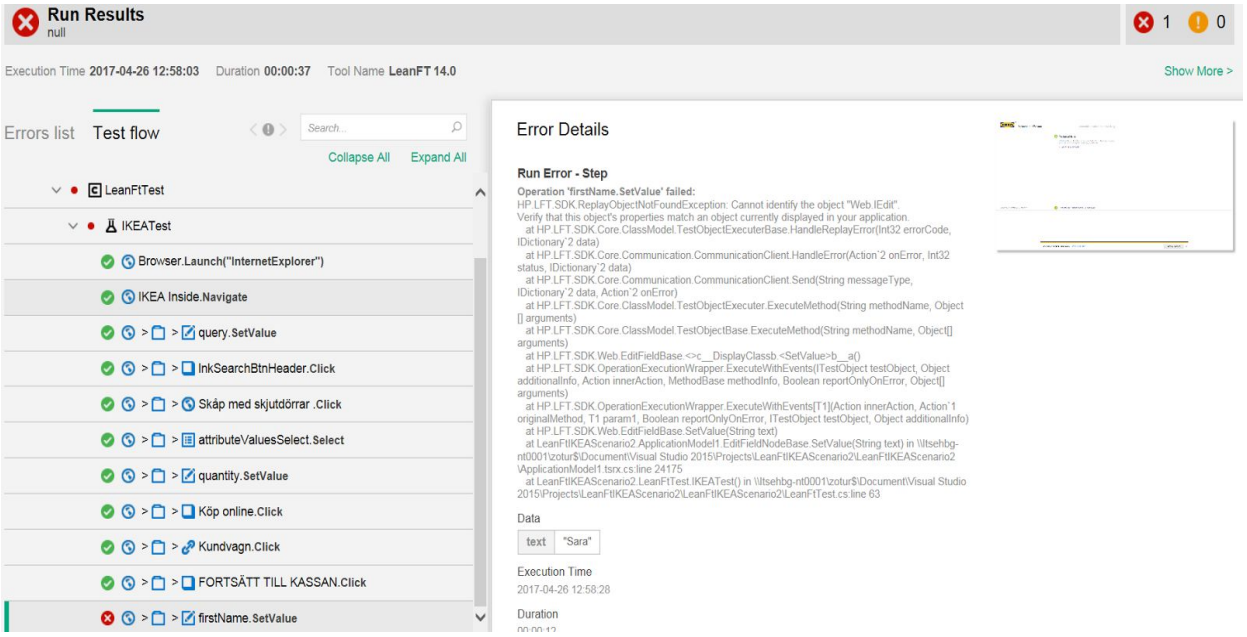

Figure 2.18: LeanFT Run Results

2.4 ALM

ALM stands for Application Lifecycle Management and is a web application which provides project planning and is used to help support project leaders with the task of managing projects. This is done by giving an overview of the project and more specifically the test schedule. In the application a user can see the status and plan different projects, in these project a user can plan tests, put in requirements and then connect these requirements to specific tests and project cycles which have a specific start and end date[6].

A project goes through several cycles before it can be released. The common cycles are:

1. Planning, release details, determining the number of cycles needed for the project.
2. Drafting a requirement specification
3. Creating test plans and test cases
4. Executing tests
5. Tracking and handling defects [6].

2.5 Git

Git is an open source version control system that is used by many projects. Version control enables project members to upload versions of the currently worked on development project. It is also possible to recall earlier versions during the development process.

Git's version control system is built so a user can commit changes to a project and if the change is wrong or just not what they wanted in the end the user can go back to a previous commit and reverse the changes that they made.

In Git the user or users can have different "branches" for their commits. One user can have their own branch of the project where all their commits are stored and when the project manager feels those changes can come into the "main/master" branch then they can "merge" the two or more branches into the project. As is shown in figure 2.19 there can be many branches off of the master branch at the same time as there are branches branching out from "lower" branches.

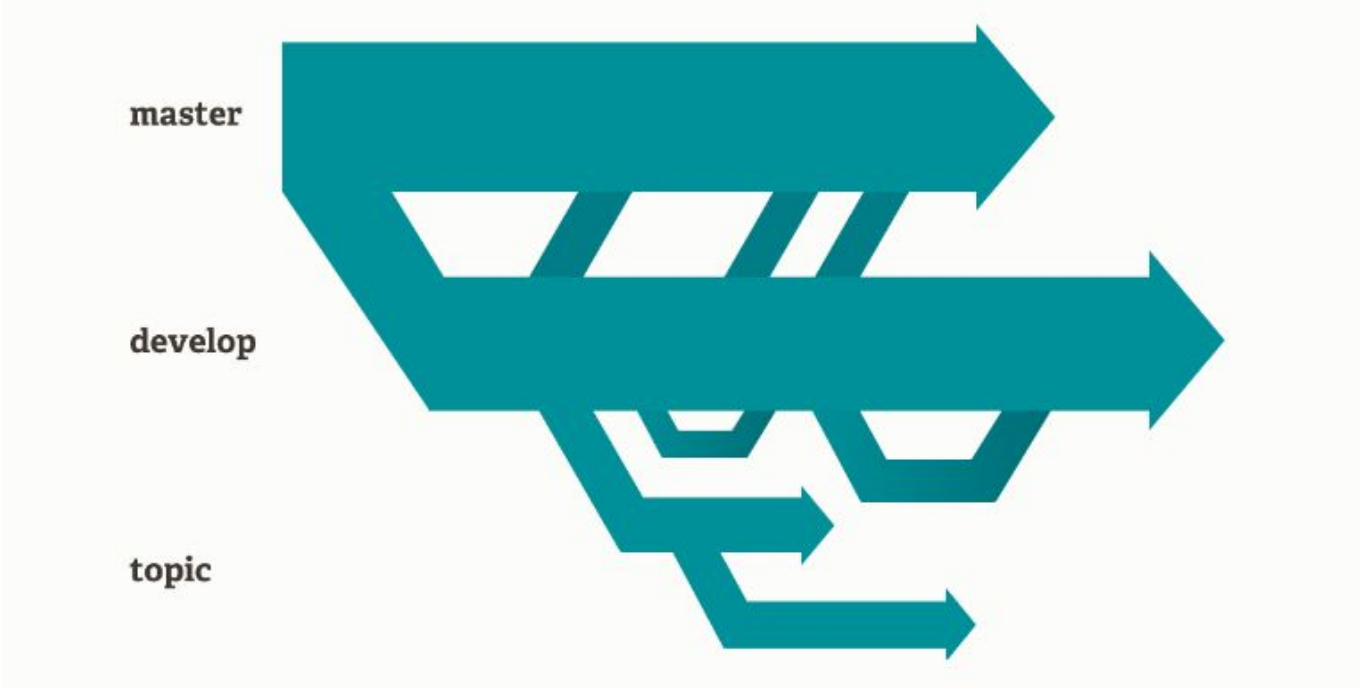

Figure 2.19: Git repository structure 1

A user of Git can download a remote repository to their computer, where they can create local branches and then upload to the repository those changes they want to have in the final project. At the same time someone who looks at the new branch can see what changes were made and when those changes were committed to the repository even if those changes were made to their local repository before being uploaded to the remote repository.

To manage a remote repository a user can either have their own Git server where they store all their repositories or use an online service such as Bitbucket or GitHub. These provide an online service to store a user or users, repositories and have links to connect these to the local repositories[7].

2.6 Visual Studio

Visual Studio is an integrated development environment(IDE) that can be used when writing programs in C#, C++, Android etc. Visual Studio uses advanced filtering to navigate to the error/problem, which reduces the time needed for problem solving.

Visual Studio was developed by Microsoft for use to develop Microsoft Windows programs in 1997. It

was used for developing in C++, Java and some other languages. It has been modified and released in many different editions since then but the most recent release as of writing this thesis was the Visual Studio 2017 release.

In Visual Studio many different type of projects can be created in several different programming languages, but Visual Studio is mainly for C# and .NET programming.

When Visual Studio is opened the Start page is shown as in figure 2.20. From here a new project/file/other can be created or an existing project can be opened. Visual Studios also offers the option to receive news about updates to the IDE. For more information on Visual Studio see [8].

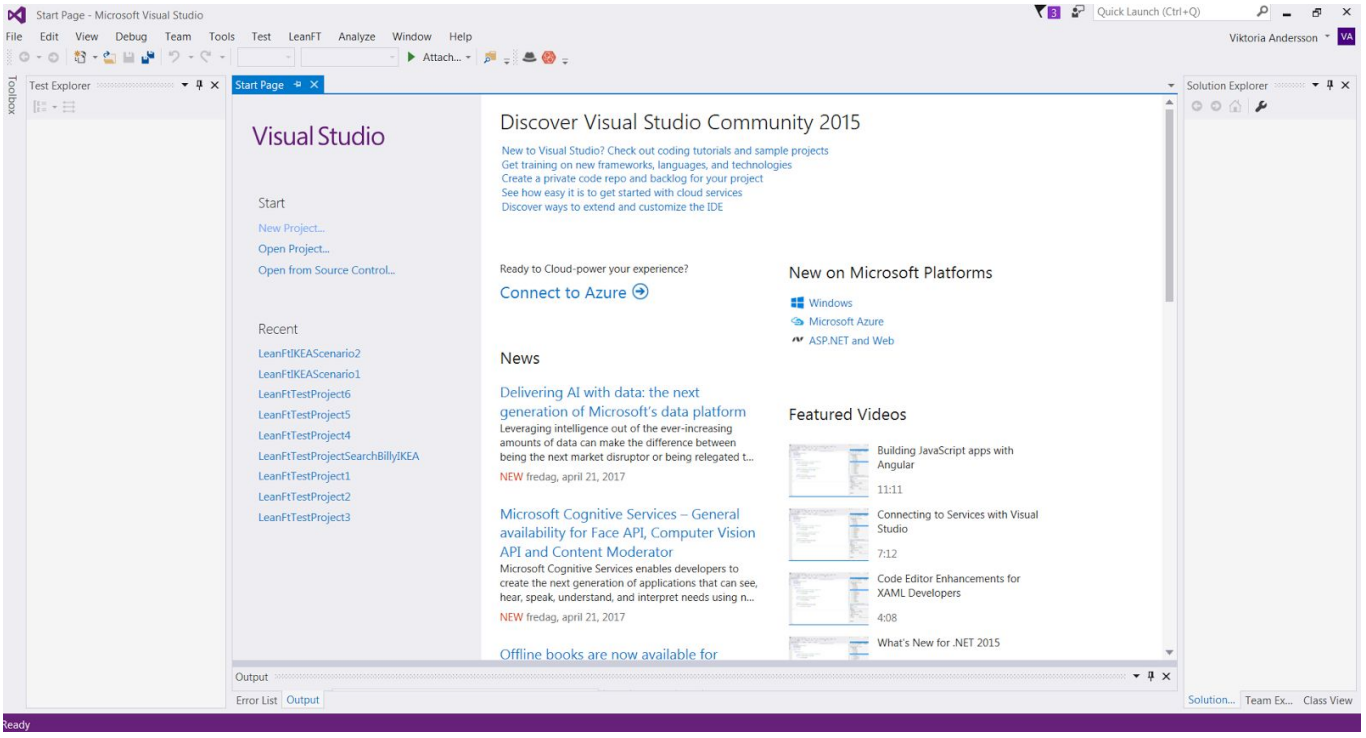

Figure 2.20: VIsual Studio start page

2.7 BDD

BDD stands for behaviour-development driven and is a software development method. The application that is being developed or the test that is being written, is designed and specified in a way that describes the behaviour of the application to future users, which explains why BDD is also known as Specification by Example. The project of an application usually begins with stakeholders offering requirements and

examples of the desired behaviour of the application. The developers then work on developing a product that meets the stakeholders' requirements. To test that the products do in fact meet these requirements, validation tests are performed. The results of these tests should assure the stakeholders that the application achieves their desired business objectives for the software. All of these steps requires documentation throughout the project process, and ensure that the documents are accurate and up to date.

Normally, most of the stakeholders come from a limited programming background which in turn limits their involvement in the development process. One of the goals of BDD is to have business stakeholder or possible end users with a little programming knowledge, contribute and have them included in the development process by giving input and feedback. This expanded feedback loop may be a reason for development teams to use BDD in continuous integration and continuous delivery environments.

BDD describes the acceptance criterion in test scenarios, which can be read as small stories. The behaviour of test scenarios/stories are the acceptance criterion for the product. If the system satisfies the criterion, it is behaving as desired and if does not meet the criterion, it is not behaving correctly. A test scenario has the following form:

Given some initial context
When an event occurs, a user performs an operation
Then ensure some outcomes

An example of a test scenario can be found in Appendix C. For more information on BDD see [12,13,14]

2.7.1 SpecFlow

The SpecFlow program is used to create Behaviour driven development tests. According to the Specflow website[10], Specflow is a tool that enables .NET developments teams to "define, manage and execute automated acceptance tests as business readable specifications." The goal is to reduce the gap between developers and domain expert(or other personnel with limited developing skills). The acceptance tests in SpecFlow follows the BDD paradigm, by defining their specification with examples. The specification serves as a documentation of the program/test. SpecFlow can be integrated with Visual Studios and in turn it can then be combined with LeanFT projects[10].

2.7.2 Connecting SpecFLow with Visual Studios and LeanFT

- Step 1: Installing the SpecFlow extension in Visual Studios
- Step 2: Creating a new LeanFT project
- Step 3: Install SpecFlow for the new project

These steps are described more thoroughly in section 3.2.4.

2.8 C#

C# is a programming language developed by the company Microsoft. It became available to the public in 2002 and is a part of a large framework, also developed by Microsoft, called .NET.

The language was originally developed by Microsoft specifically for running and developing the .NET framework. It is similar in syntax as the programming language Java which is a modern and well used programming language.

It is essentially platform dependent. To run a C# program then the .NET framework must be installed on the system which is used, if the .NET framework is installed then the program can be run without the need to change it.

C# has many tools for creating and maintaining graphical user interfaces(GUI), which is why it is often used to create graphical heavy programs. Programs with buttons, windows etc. used to interact with a user.

What makes C# a more modern programming language than for example Visual Basic is that it works well for object oriented programming. C# is built for object oriented programming where objects are created in the form of classes in C#, and then work with them together with other objects. For an example on how C# code may look see figure 2.21. For more information on C# see [9].

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Figure 2.21: C# code example

2.9 GUI Testing

GUI, or Graphical User Interface is the interface that is seen by the user and which the user can interact with. The graphical interface is the link between the user and the application and ergo it must be ensured that the GUI is functioning correctly. An example of a GUI is IKEA's webpage[29]. The user can interact with the IKEA webpage by entering search words in the searchfield, clicking on items, placing items in their virtual cart and so on.

To make certain that the GUI of one´s application is functioning properly it is required to perform GUI tests. A well-rounded GUI test normally checks the following:

- The size, position, width, of the GUI elements, such as button, edit fields etc
- Acceptance of input in the edit fields
- The functionality of the GUI
- The correctness of the displayed error messages
- The positioning of the GUI elements in different screen resolutions
- The readability of the font used in the application
- The alignment of the texts
- The clarity and alignment of images
- That sections on the screen has a clear demarcation

GUI testing can be performed manually and by automation. There are several test automation tools available, UFT and LeanFT being two of these. For more information on GUI testing see [11,15].

2.10 Jenkins

Continuous Integration is becoming a common practice in the software industry, which requires developers to integrate their code frequently. The code can then be reviewed and tested and in turn allow project teams to detect problems early. This would mean less back-tracking to detect the origin of errors.

Software teams often face challenges when software is being developed at multiple sites and platforms. A solution for these challenges can be to make the Continuous Integration process faster and more efficient. This can be done by automating the build and testing process. Jenkins is a Continuous Integration tool that reduces the work of a developer by helping in automating the testing process[16].

Chapter 3 : Method

This section gives a detailed description of the project method used during the process and lists the resources needed to complete the thesis work. This chapter was divided in the different phases of the thesis work. The goals of the phases would have to be met before moving onto the next phase. The different phases of the thesis work were:

- Information gathering
- Creation of tests
- Analysis
- Connecting LeanFT with Git and Jenkins

3.1 Resources

In order to successfully complete the thesis work there was a need for specific resources used during the research into UFT and LeanFT, these were:

- IKEA on premise access
- configured computer (full access to IKEA network at Sockerbruket)
- license for latest release version of UFT (14.00) and LeanFT (should be a part of the license)
- Visual Studio 2015
- ALM access
- from-time-to-time access to UFT experts, e.g. Emil Långberg and people with .NET/BDD development experience e.g. Tomas Bylander.

3.2 Methodology

3.2.1 Information gathering

This was the starting phase of this thesis. The goal of this phase was to find facts about the two test automation tools that were meant to be compared and also to get an understanding with these two tools. In the initial stages of the thesis work there were some issues faced with the internet access on the IKEA IT computers and with the licenses needed for working with UFT and LeanFT. These issues prohibited the thesis workers from starting to explore the UFT and LeanFT tool until the issues were fixed, which led to a slower start. To prevent any major time loss, the thesis workers decided to work from home with

their personal computers to gather information.

The thesis workers were also handed an UFT tutorial/guide by their mentor at IKEA IT which helped the them to understand the existing features in UFT. Later on, when they had been granted access to the tools the thesis workers began creating small test projects in the UFT program and to learn more about the structure of the program, they also used online tutorials during this research see [2,3].The thesis workers were also granted access to the IKEA IT ALM workspace and could therefore use the test scripts written by the IKEA IT staff as tutorials. The tutorials helped to give a better understanding on how test scripts were written at the IKEA IT Testcenter.

For the LeanFT research it was quickly discovered that the main information source would come from HP´s own help pages, guides and tutorials. A possible reason for this was the fact that LeanFT was, at the time. a relatively new tool on the market. Both UFT and LeanFT are closed-source tools which normally means that the online community, outside of the owner company, is constrained.

3.2.1.1 Interviews

The way chosen to investigate LeanFT compared to UFT by the authors was to do specific automated tests in UFT and then try to recreate these in LeanFT and compare certain factors in the results. To gather knowledge of the most important features of UFT. three interviews were conducted. Before starting with the interviews the thesis workers began to study different interview methods, see [23]. Then the thesis workers went on with deciding on the questions that needed they wanted answers to. These questions were then written down and used during the interview as a guideline/checklist.

The interviews started with the interviewee explaining their role in the company, their work assignments using UFT and their technical background. Depending on these factors, different questions were asked during the interviews. The fact that every interviewee was asked different questions was one of the main reasons why there were no group interviews held. The full list of questions asked during the different interviews can be found in Appendix A.

The main questions asked to all of the interviewees were the following:

- What are the main motivation for wanting to shift focus to another test automation tool such as LeanFT?
- What are the strengths and limitations of UFT?
- What functionality must be present in LeanFT?

Interviewee X originally comes from a non-technical background. This gave an insight for the experience of conducting tests in UFT for a person with minimal programming experience. Since IKEA IT was not planning on excluding non-programmers from the test team in the near future, it was important for the

thesis workers to interview employees with different programming knowledge.

Both interviewee Y and Z were experienced UFT users. Interviewee Y worked in the UFT support team at IKEA IT. The results of the interviews, which can be read about in section 3.2.3, helped with the decision on what features to look into when creating tests in UFT and LeanFT.

3.2.1.2 Gathering information on BDD

Before implementing the BDD tests, the thesis workers had to gain knowledge of the purpose of creating BDD tests and how it would benefit the IKEA IT team. This step consisted of searching the web for relevant information during a week's time.

It was recommended by IKEA to use the SpecFlow program for BDD testing and therefore this also required research. The majority of the help was found on HP's website, where they had a section on how to integrate LeanFT with SpecFlow. The HP website also provided simple tutorials on how to get started with BDD testing with LeanFT.

The BDD research was started after the completion of creating and analysing the UFT and LeanFT tests. The thesis workers and their mentor at IKEA IT came to an agreement that the BDD matter had a lower priority and was therefore not worked on before the higher prioritized matters were finished.

When the thesis workers were more familiar with the concept of BDD, they proceeded with the process of creating the tests.

3.2.1.3 Examining the Public Community

To start examining the extent of the public online community availability for LeanFT the thesis workers started by exploring the web for sources. They searched for informative tutorials for starting to work with LeanFT and looked for good forums for asking LeanFT related questions.

The thesis workers searched first for general knowledge sources for gathering information on LeanFT and to see if those sources might be a good help and support to future users of LeanFT.

Then the thesis workers searched for more specific problem solutions to see if a person using LeanFT and encountering a problem would be able to find support and help for the specific problem online.

After terminating this phase, the thesis workers proceeded with the creation of tests. The gathering information phase was the groundwork that needed to be completed before the tests could be created. Based on this information the tests that would help examine these points, were developed .

3.2.2 Creation of tests phase

After analyzing the interviews and having discussions with their liaison at IKEA, the thesis workers decided to create two GUI tests using both UFT and LeanFT. The interviews gave a good basis for determining what aspects to look for when comparing the two tools.

The goal of the creation of tests phase was to use the information gathered in the research phase to create tests and detect the strengths and weaknesses of UFT and LeanFT.

For the purpose of time efficiency and for the relevancy of this thesis work, the thesis workers decided to perform GUI tests on IKEA's website. The thesis workers decided on two test cases to test in UFT and LeanFT.

The decision to create two scenarios, though similar, instead of just one was to test more functionality, which would result in more data for the comparison between UFT and LeanFT. The test cases were created based on discussions between the thesis workers on the different functions that existed on the website, such as search fuction, buy online function and so on. The thesis workers chose to incorporate a few of these functions in the tests to best be able to show the range of UFT and LeanFT when conducting GUI tests.

After the test cases were constructed the thesis workers started to create the four tests, two in UFT and two in LeanFT.

Test A consisted of the following steps:

- Navigating to IKEA´s home page
- Navigating to the section "Sängar"
- Navigating to the section "Lakan"
- Choosing item "Dvala lakan Beige"
- Clicking on the button marked "köp online"
- Clicking on the cart to see the chosen items
- Continuing to checkout
- Adding the following data at checkout:
    - Firstname - Sara
    - Lastname - Berg
    - Address - Sjögatan 1
    - Zip code - 252 25
    - City - Helsingborg
    - Email address - sara_berg@hotmail.com

    - Mobile number - 0722569887

Test B consisted of the following steps:

- Navigating to the IKEA homepage
- Entering “GALANT” in the search field
- Choosing item “GALANT skåp med skjutdörrar”
- Changing the color field to grey
- Changing to the quantity field to 2
- Continuing to the shopping cart
- Continuing to the checkout page
- Entering the following data:
    - Firstname - Sara
    - Lastname - Berg
    - Address - Sjögatan 1
    - Zip code - 252 25
    - City - Helsingborg
    - Email address - sara_berg@hotmail.com
    - Phone number - 0722569887

The steps for test A would mainly test the OIC feature and the ability of descriptive programming with LeanFT. Test B was intended to test using an Application Model. This was decided on after the interviews were conducted, see section 3.2.1.1. The results of the tests can be found in section 5.1.

3.2.2.1 Creation of BDD tests

This step was started with searching for demos on how to integrate SpecFlow with Visual Studio and LeanFT. HP´s website provided a step-by-step guide on how to combine the SpecFlow program with LeanFT and Visual Studio, see [27]

When the integration was completed the thesis workers began the process of creating the BDD tests. The test scenarios were identical to the tests that were created in UFT and LeanFT and this meant that the code written in LeanFT could be utilized in creating the BDD tests.

Test A was translated into a BDD Scenario in the following way:

Scenario: Buy an item on the ikea website by using the search field
Given I am on the IKEA homepage
And I navigate to the field "*SÄNGAR*"

And I choose the field "*LAKAN*"
And I choose the "*DVALA lakan beige*" item
When I press the buy online button
And I go to the shopping cart site
And I continue to the checkout
And I fill in my personal information
And I continue to the delivery information site
Then I shall come to the delivery information site

Test B was translated into a BDD Scenario in the following way:

Scenario: Buy the "GALANT skåp med skjutdörrar" item from the IKEA website
Given I am on the IKEA homepage
And I search for "*Skrivbord*" in the searchfield
And I have chosen the "*GALANT skåp med skjutdörrar*" item
And I choose the colour grey
And I select the quantity *2*
When I press buy online
And I go to the shopping cart
And I continue to the checkout
And I fill in my personal information
And I press the continue to delivery information button
Then I should come to the delivery information webpage

Both scenario one and two could have been divided in several scenarios, where every step was it's own scenario. However, the thesis workers decided to gather the steps in one single scenario due to the simplicity of the test and the various steps. The outcome of the steps are straightforward and therefore the outcome of the entire scenario is the result of the final step in the test. Gathering the steps in one scenario also gives a better understanding of the entire test structure, which is one the main purposes of BDD testing. It also made it easier to compare the BDD Scenarios with Test A and Test B in LeanFT and UFT.

The reasons for why these test cases were chosen is discussed in section 4.1.

To create the Scenarios in LeanFT these steps were followed:

Step 1 was to install the SpecFlow extension on Visual Studio. This was done by clicking on Tools> Extensions and Updates, see figure 2.15 for example, Specflow was searched for under the Online tab. SpecFlow for Visual Studio 15 was then installed from the search results.

Step 2: In this step a new NUnit project was created in the same manner as in the other LeanFT tests, see section 4.1. Thereafter, the *DLL* was installed by installing SpecFlow for the newly created project. This was done by clicking on Tools> NuGet Package Manager> Manage NuGet Package Solutions see figure 2.15. Then, under the Browse tab, SpecFlow was searched for. The first item in the result page was

installed.

Step 3: To add a *Feature* to the project, the project was right-clicked. Then Add> Component was selected. The SpecFlow feature was selected, which generated a feature file with default content.

Step 4: In this step, the default content of the file generated in step 3 was deleted and replaced with the IKEA test Scenario.

Step 5: When the test scenario was completed, it was time to implement it using LeanFT. This was done by generating a class file for the step definitions in the following manner:

The thesis workers right-clicked inside of the newly created feature file and then **Generate Step Definitions** was selected. Then **Generate** was clicked. This step generated a cs file, which is the file type used in Visual Studio to store C# code. The test methods were implemented using LeanFT SDK.
Since the new BDD test that was created was a recreation of test A in LeanFT, the code written for that test was reused. The method for the implementation is written in Section 4.3.

Both of the BDD tests were created in the same manner. The code for both of the BDD tests can be found in Appendix D.

3.2.3 Analysis phase

The goal of the analysis phase was to analyse the results from the research and creation of test phases to be able to compare LeanFT and UFT with each other.

After the tests were created and executed the thesis workers started the analysis phase. The results of the tests and the ease of creating the tests in UFT and LeanFT were considered when analyzing the two tools.

Interviewee X helped the thesis workers realise the importance of a user friendly testing tool. It is desirable that new users with or without much programming knowledge can get started with creating tests fairly easy. The interviewee also mentioned which UFT features that simplified the creation of GUI tests, features such as recording user actions, object repositories, Object Spy, Navigate and Learn. To examine if the usability of LeanFT was comparable to UFT, the thesis workers tested out similar features in LeanFT and then analysed the results given by these features.

Interviewee Y, who was an experienced UFT user, informed the thesis workers about the issues faced with UFT and the motivation factors for using LeanFT. These factors were based on UFT´s stability issues and hindering of object oriented programming in UFT by not being able to use modern programming languages.

The last interview was given by interviewee Z, who explained the way of working with UFT at IKEA IT. It was mentioned that object repositories were helpful for new users, however code that is dependant on object repositories can lead to issues. An example of this is when a group of team members are working on the same tests and thus share the same object repositories. When an object is deleted by one user with that object having code dependant on it, all tests that are dependant on the deleted test object will no longer pass. Therefore, the staff at IKEA IT prefers to not use object repository dependant code and instead use *descriptive programming*.

To evaluate whether the maturity level of LeanFT is comparable to UFT and to determine if the desired featured were existent in LeanFT, the thesis workers created a list with functions that were meant to be analysed, see table 3.1. The list of features were decided on after analyzing the results of the interviews

Table 3.1: Features wanted for LeanFT

Feature in UFT
Object repository
Object spy
Drag-and-drop feature with object repository
Descriptive programming
Function library
Connecting project to ALM

3.2.4 Connecting with Jenkins, ALM and Git

Since, the connection with Jenkins, ALM and Git were all considered valuable by the IKEA IT team, the next step for the thesis workers was to try to achieve this.

Connecting with Git

When the thesis workers had finished the test projects in LeanFT and therefore had some code base, they decided to work on connecting one of their LeanFT projects to the versioning system Git. To do this they first researched if there were tutorials for connecting a LeanFT project to Git , which they did not find.

Then the thesis workers searched for tutorials in how to connect Visual Studio to Git through the free online Git solution Bitbucket, which both thesis workers already were familiar with. The thesis workers followed the directions that were provided by a video tutorial when connecting the bitbucket repository to

Visual Studio, see [18] for video.

Connecting with Jenkins

Since, the thesis workers did not have access to a Jenkins server they were required to install Jenkins on the host computers. Jenkins was installed from the Jenkins website and after the installation was completed, the user accounts were created to start exploring Jenkins.

The thesis workers had no prior experience with Jenkins so before connecting LeanFT with Jenkins they had to gather information about the tool and search for tutorials and guides on the internet. The information that was searched for were mainly the purpose of Jenkins and the features offered.

For connecting the tool, the step-by-step guide on HP's webpage was of great help, see [28].

3.3 Communication

The thesis workers held several meetings per week amongst themselves, in form of both Skype and physical meetings. Other than these meetings, daily communication occurred via mail or text messaging. During these meetings, the upcoming work sessions were planned out, a to-do list for the work session/week was drawn up and the thesis workers had discussions about issues and updates to their previous lists. The work sessions were held both from home and at the IKEA IT building. To decide on the location for the upcoming work session the thesis workers considered both their schedules and the work plan. They chose to work from home on occasions where research was being done or when writing on the thesis. This was usually done while engaging in a Skype meeting.

Communication with IKEA IT

The communication with IKEA IT consisted of physical meetings and emailing on a regular basis. The thesis workers also received contact information of several employees at IKEA IT TestCenter, whom they could contact when they were in need of help. These employees had expertise in the relevant fields of the project and were therefore at times better suited for the different issues that arose than the mentor.

The thesis workers were also able to get an one hour lecture of the ALM program in the beginning stages of the project, by an expert at the company. The working sessions were held at the IKEA IT TestCenter building several days of the week, which helped the thesis workers to get help more quickly when they could not solve the problem internally.

3.4 Criticism of the sources

Most of the references in this thesis are links to web pages. It can sometimes be difficult to determine the

reliability of a source of reference and therefore the thesis workers used caution when choosing their references. The majority of the chosen references were the developer's own web pages or web pages that offered tools for the mentioned software development principle. These links consists of the following references: [1, 4, 5, 6, 7, 8, 10, 17, 19, 20, 21, 24, 26, 27, 28]. Considering that it´s in these companies interest to deliver as reliable information as possible for possible and current consumers, the thesis workers deemed that the technical information given by these sources were trustworthy.

The other links used in references in this thesis work is believed reliable through trial and error. the thesis workers tested the different tutorials and tools mentioned in the references:[2, 3, 11, 12, 13, 18].

The references: [14, 15], are links to sites moderated and run by whatis.com which according to their website info: "for the last decade site content has been researched and written by editorial director Margaret Rouse and content editor Ivy Wigmore with the assistance of contributors and technical experts from over 60 countries." they have also according to their site been awarded by The New York Times and many other reputable newspapers for their site. Because of this the thesis workers believe these links to be reliable with their information.

There are some books and articles also used as sources [9, 16, 22, 23, 25] and the thesis workers believe these are credible because they are published by credible publishers, such as Studentlitteratur and Microsoft publishing.

Chapter 4: Execution of tests and information gathering

This chapter provides an in-depth description of the work done in the different phases of the thesis work, see section 3.1. The reasonings for decisions made during the creation of tests are also included in this chapter.

4.1 Creation and execution of tests

To be able to compare the programs, UFT and LeanFT, the thesis workers decided to create GUI tests in both of the programs and then compare the results.

After discussions with IKEA the thesis workers chose not to create API tests, for this assignment and only focus on creating two GUI tests that would be relevant for the IKEA IT TestCenter team. The decision for not creating API tests was based on the fact that API testing is a more complex way of testing and considering that the student had no prior experience with UFT or LeanFT, the creation of these tests would have needed more time and resources.

With the information the thesis workers gathered from the interviews in the research phase, see section 3.2.1.1, they created a list of features in UFT to test for in LeanFT, see figure 3.1.

To be able to test these features they created the two tests described in chapter 4.1.1-4.1.4.

4.1.1 Test A: UFT

Step 1: To create test A in UFT the thesis workers started with creating a new UFT GUI test. To see how to create a GUI test in UFT see Chapter 2.1 UFT.

Step 2: This step consisted of creating an object repository and adding the needed test objects to the repository. After the objects were added to the object repository, the repository was saved to a desired location. The “Navigate and Learn” feature was used for adding object in the repository, see figure 4.1. The initial method of adding test objects was by the spy feature but this method was quickly discarded because of the poor detection accuracy of the specific test objects. Therefore, the switch to the “Navigate and Learn” feature seemed like a better choice. To get a better knowledge about this feature and see the instructions, see section 2.1 UFT.

Step 3: To be able use the object repository in the previous step, the repository needed to be associated with the action that involved the test objects.

Step 4: After the object repository was associated with the test action, the test steps could be added. This was done by the record feature. For a relatively small test, the record feature was the best option due to its simplicity. To read more about the record feature and for more specific instructions see section 2.1.

Step 5: At this point the test is ready to be compiled. To run the test the run button was clicked on.

4.1.2 Test A: LeanFT

To create test A in LeanFT the thesis workers started with creating a new LeanFT NUnit project in Visual Studio as described in chapter 2.3.2.

Step 1: The thesis workers started the test by adding to the Template using HP.LFT.SDK.Web, at the top of the file with the test code. This was to be able to the use web functionality from the LFT.SDK.Web library, such as BrowserFactory and BrowserType. Which was used to be able to start up a specific browser and use that browser later in the test steps.

Step 2: The thesis workers added in the [test] Test{} function the first step to their test, opening the Internet Explorer browser. Which was added by adding an object of the type IBrowser to the test. Then using that object to navigate to http://www.ikea.com/se/sv as seen in figure 4.1.

```
[Test]
public void TestIKEA()
{
    //Open Internet Explorer
    IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);

    //Navigate to IKEA homepage
    browser.Navigate("http://www.ikea.com/se/sv/");
```

Figure 4.1: Code to Navigate the page

Step 3: The thesis workers then used the OIC to identify all the specific objects needed for the test, starting with the link for "Sängar". See figure 4.2 for an example of the OIC identification or "spying" as it is called. This took quite some time because it was sometimes difficult to find the specific object when "spying" on the specific page, the object wasn't always where you thought it would be.

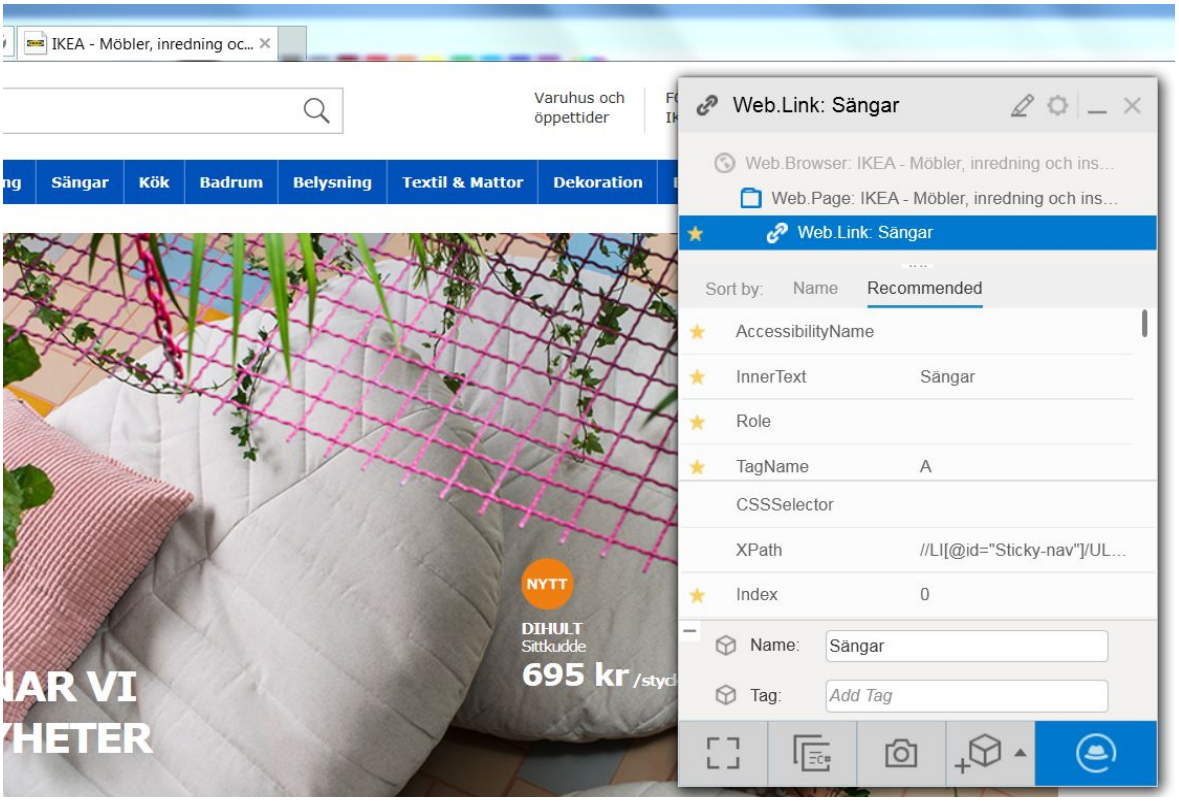

Figure 4.2: Using OIC to identify object

Step 4: Every time the thesis workers identified the right object with the OIC they pressed the button in the OIC which generates C# code to the clipboard, see figure 4.3, which they then pasted into the test project and then added what was meant to be done with it, if it was meant to be clicked, or written in and so on.

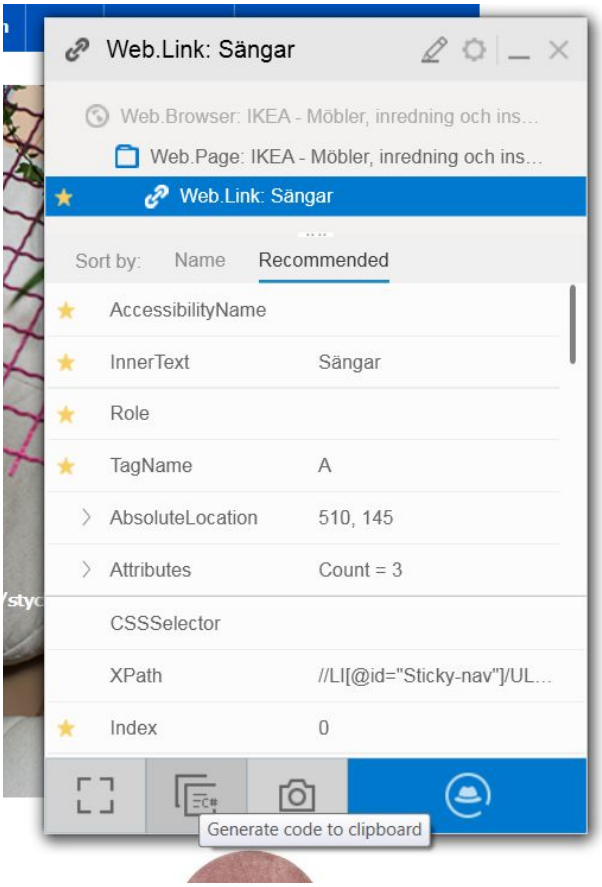

Figure 4.3: Generate code to clipboard

To see all the code for the test see Appendix B: LeanFT Test A.

4.1.3 Test B: UFT

To create the test for test B in UFT the thesis workers did almost all the same steps as in Test A in UFT, they just changed some of the objects.

4.1.4 Test B: LeanFT

To create test B in LeanFT the thesis workers started the same as with test A. They followed the steps in chapter 2.3.2. Step one and two were the same as for test A in section 4.1.2.

Step 3: Instead of spying on a specific page and adding a specific object to the test using the OIC as in step 3 of test A, the thesis workers created an Application Model for the project and added all the objects on the specific pages instead. See step 7 in section 2.3.2 for how to add an Application Model to your project.

They did this because they sometimes had some problems to find specific objects using only the OIC to identify them.

Step 4: They created an object of the type ApplicationModel 1, which was the name of their Application Model, in the test code and then used this object to call the specific test objects for the test steps, see figure 4.4 for example.

```
//Search for "Galant"
//Create new ApplicationModel where our links are
ApplicationModel1 appModel = new ApplicationModel1(browser);

appModel.IKEAMBlerInredningOchInspirationIKEAPage.QueryEditField.SetValue("Galant");
appModel.IKEAMBlerInredningOchInspirationIKEAPage.LnkSearchBtnHeaderButton.Click();

//Click "Galant"
appModel.SKresultatIKEAPage.SkPMedSkjutdRrarWebElement.Click();
```

Figure 4.4: Code for Application Model

With the Application Model they called all the different objects needed for the test and added the different functionalities to them as they did to the objects captured by the OIC in test A, such as if the object should be clicked or text added to an editfield.

4.2 Creation of BDD tests:

The prerequisites for creating BDD tests is to enable and install SpecFlow in Visual Studio. This step is described thoroughly in section 2.7.2. The BDD test that was created was identical to the tests that were made in LeanFT and UFT. The test was translated into a BDD test scenario and the code linked to the steps was then taken from the tests in LeanFT. The reason for not creating an entirely new test, was so there could be a fair evaluation of the effort needed to create BDD tests with LeanFT.

The test was created similarly as in the guide found on HP´s help center website, see [27]. It was not obvious how the test should be translated to a BDD test scenario, which was the main challenge with the creation of the BDD tests. For the test scenarios to be as self-evident as possible for those with no

programming knowledge, the thesis workers decided to have every step in the test as one line in the scenario.

As seen before in Appendix B the code in the regular LeanFT test for navigating to the IKEA homepage was written as seen in Example 4.1.

```
IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);
//Navigate to IKEA homepage
browser.Navigate("http://www.ikea.com/se/sv/");
```

Example 4.1

Since this was considered as a prerequisite for the test, the first line in the test scenario describes precisely this step:

Given I am on the IKEA homepage

The first legitimate step in Test A was to enter “GALANT” in the search field. This step was translated as a line in the test scenario 1 in the following manner:

And I navigate to the field "*SÄNGAR*"

This line states that the test is supposed to perform the action of navigating to the field “SÄNGAR” on the IKEA homepage. The following steps were handled in the same way.

The entire code and test scenarios for BDD test 1 and 2 can be found in Appendix D.

4.3 Public community help LeanFT

To analyse if there is a big support for LeanFT online, for users of LeanFT to be able to find help when they have issues, the thesis workers decided to analyse this when they started to use LeanFT.

When they started with LeanFT they searched for what they believed to be common phrases people tend to use when searching for answers to problems, such as:

- “Set Up LeanFT”
- “LeanFT with Visual Studio”
- “LeanFT tutorial”

And other similar phrases for learning about the program.

After they had researched if there was support for general setup and knowledge of LeanFT they started to research to see if there were online help and support for more specific issues such as:

- "Integration LeanFT and ALM"
- "LeanFT NUnit test"
- "LeanFT BDD"

They found most of the support on HP's own help and support pages. They found much support on HP's LeanFT help centre and on their UFT Community pages see [19,20].

HP's help centre had much help for general setup and how to get started with LeanFT, both step by step tutorials for API and GUI test and setups for integrations with for example ALM. The UFT Community pages had it so that a user could ask questions relating to UFT and therefore also LeanFT(UFT Pro) and get answers from either users or HP developers of UFT/LeanFT.

4.4 Connecting a LeanFT project to Git

Before the thesis workers started with their analysis of the problem defined in section 1.3 Problem definition; if one can use a versioning system such as Git with LeanFT, they were already quite familiar with how Git worked with the free hosting system Bitbucket, created by Atlassian. They already knew what Git was used for and how a user uses it for version control.

They started their analysis by watching a tutorial on how to integrate Bitbucket with Visual Studio [18]. The tutorial went through the steps to connect the project with the online remote repository located at Bitbucket.

To connect a LeanFT project with a remote repository the thesis workers went through theses steps:

Step 1: First create a new repository in a Git agent, either Bitbucket, GitHub or other such as an own Git server. The thesis workers chose Bitbucket because they were already familiar with it.

Step 2: In the LeanFT project go to File->Add to Source Control, see figure 4.5, this step can also be made when creating the project.

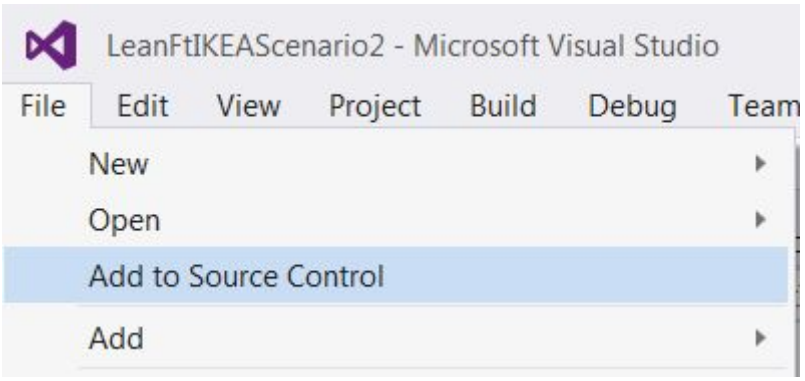

Figure 4.5: Add to source control

Step 3: Open the Team Explorer which is located by the Solution Explorer on the left side of the IDE, if it is not there click View->Team Explorer.

Step 4: In the Team Explorer click Sync, after that there will be a button to Publish to remote repository where the project can be published to the remote repository, see figure 4.6.

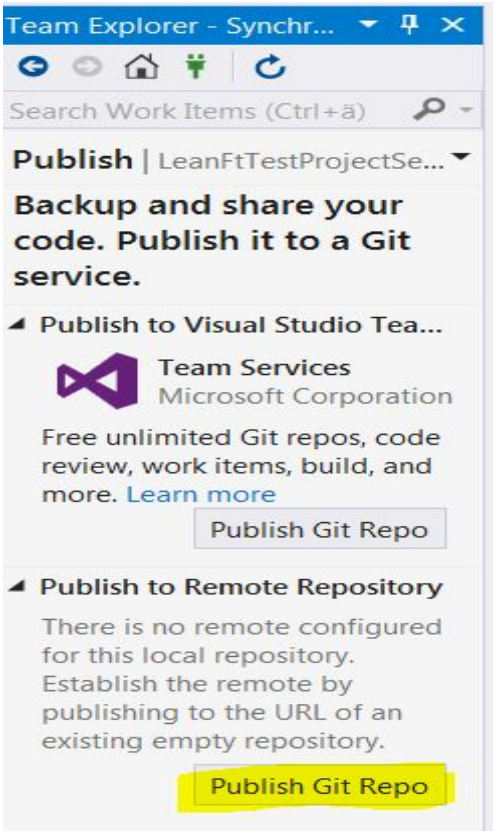

Figure 4.6: Publish to Git repository

Step 5: An editfield will come up after the click, inside the field write in the address for the repository to use for the project. The repository must be empty to publish the project to it.

Then the project will be published to the remote repository chosen for the LeanFT project. The thesis workers also tried to clone a LeanFT project to Visual Studio with the help of the Team Explorer as when publishing the project to the repository. To do this they followed these steps:

Step 1: Go to the Team Explorer in Visual Studio. Click on Manage Connections, see figure 4.7.

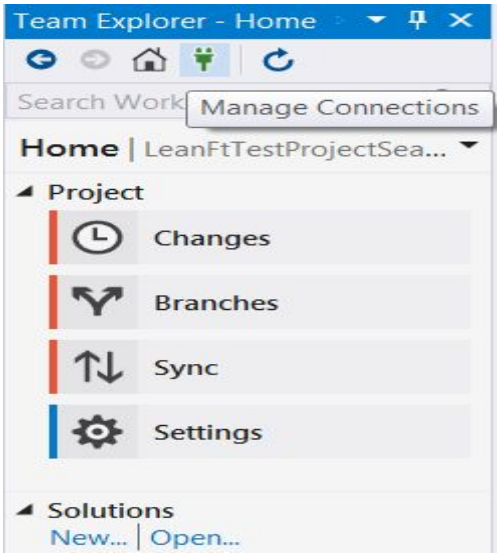

Figure 4.7: Manage Connections

Step 2: Under the header "Local Repositories" click on the button "Clone". An editfield will appear where the link for the repository with a LeanFT project is located, then click "Clone". This clones the remote repository to the local computer.

4.7 Connecting LeanFT with Jenkins

The prerequisites for connecting LeanFT with Jenkins are:

- An installed Jenkins server
- An installed LeanFT runtime engine
- An installed unit testing framework, such as NUnit or JUnit

After these prerequisites were met the following steps were executed.

Step 1: The tests were build and run locally in visual studios before connecting with Jenkins.

Step 2: A new job was created in Jenkins by clicking on New Item and choosing Freestyle project

Step 3: The job was configured to run the LeanFT tests by adding a build step to run the tests using the NUnit that was installed with the NuGet package manager, in Visual Studios, see figure 2.14. To configure the job. Add build step > Execute windows batch command was chosen. A new command textbox then appeared. The command that was typed in the textbox consisted of two parts see figure 4.8. The first part pointed to the NUnit runner file, see example 4.2, and the second part contained the path to

the DLL file of the test, see example 3.2.

“C: \Users\viand21\ IKEATestScenario1\ packages \NUnit.ConsoleRunner.3.6.1 \tools\nunit3-console.exe”

Example 4.2: Command line for the NUnit runner

“C: \Users\viand21\ IKEATestScenario1\bin \Debug \IKEATestScenario1.dll”

Figure 4.3: Command line for the DLL file

After these steps were completed, the tests was able to run from Jenkins.

Chapter 5: Results

This chapter contains the results from the different analysis the thesis workers did to be able to answer the problems defined in section 1.3: Problem definition.

5.1 Comparison between UFT and LeanFT

When comparing UFT and LeanFT, there were a number of parameters that were focused on. These parameters were:

- performance
- stability
- error messages
- usability.

Performance:
The performance of the two tools were mostly focused on the speed factor. The same tests were executed in both UFT and LeanFT on the same computer to ensure that the conditions were as similar as possible for the two tools. In the speed aspect the results for UFT and LeanFT did not differ much. A factor to this can be the work that has been dedicated to improving the stability of UFT by the IKEA IT staff.

Stability:
The stability of the UFT tools was poor in comparison to the LeanFT tool. UFT would often crash or freeze up whenever a simple task was performed. For an instance, whenever the save button was pressed, to save the code written, UFT would freeze for several seconds, sometimes even up to a minute. The UFT program needed to be restarted every time it crashed thus this and the frequent freezing of the program, led to frustration building up and therefore more time was spent on creating tests in UFT.

Another problem faced with the UFT tool was the inconsistent results given when executing the tests. The result would at times differ when there had not been any changes done to the program. For an example: the test would at times fail or take a different path even though no modification had been done to the code or the settings. This problem was often fixed by restarting the program.

The usage of the LeanFT tool was less troublesome in this sense. The program was more stable and would very rarely crash or freeze, thus it took less time to create the tests in LeanFT. The results of the test were consistent throughout the process, when there had not been done any changes to the test program.

Error messages:

In UFT error messages would not appear before running the test. This means that the tester can not detect miniscule coding errors, such as forgetting a parenthesis, etc. before running the test. As expected, this led to time-wasting when creating the tests. The error messages was not accompanied by possible solutions for the errors, by UFT.

When executing the test, UFT highlights the code line that is currently being executed. The student found this feature to be helpful in detecting where the error is located.

Unlike UFT LeanFT did not offer the live step by step execution of the code, but this was not of great importance because the error line was given in the result console tab. LeanFT did provide with possible solutions for errors and detected errors immediately after it was written.

Usability:

Even though UFT offers many user friendly features, the usability was often negatively affected by the lack of accuracy of these features. One apparent problem was the difficulty in using the spy feature for the thesis workers. When using the object spy feature to add test objects to the repository the thesis workers faced a great deal of difficulty in detecting the desired objects. The objects would be detected in areas outside of its actual position. The Object Spy dialog box did not offer enough information of the object before the object was selected, which made it near impossible to know if the right object was hovered over. This added to the difficulty in using the Object Spy feature.

Even Though the Object Identification Center(OIC) was not fault free, there were positive factors that simplified the process. One of these being that the OIC offered enough information about the test objects before selecting it, thus making it easier to know what object was detected.

The record feature in UFT, which was considered helpful for introducing new users into the test automation world by the interviewees, also showed weaknesses. It would on occasion skip some user actions or generate the code for the steps in the wrong order. Although LeanFT does not offer a record feature, the guides and tutorials available for LeanFT were found helpful for getting started with LeanFT test automation.

5.2 BDD tests

The full implementation of the BDD tests can be found in Appendix D. There was no significant change in speed or stability compared to the regular tests in LeanFT, when running the BDD tests. The guides founded on SpecFlow´s and HP's website were informative enough for a new user to start creating BDD tests.

When the thesis workers gained enough familiarity with the LeanFT tool to create their own tests, there were not any major struggles to create BDD tests. The creation of BDD tests only required an extra step, compared to creating regular LeanFT tests. The extra step was the making of the test scenario(feature file), which did require some time and effort to design. More about the design decisions for the test scenario can be read about in section 4.1.

The BDD tests can be run from the *feature file*, which means that non-programmers can execute tests without reading any code.

5.3 Public Community Help for LeanFT

HP's own help page offers support and help with LeanFT, but there is not much help existent outside of HP's own resources as there is with UFT. For UFT there are many tutorials, walkthroughs and other help for users using UFT outside HPs own webpages. When searching for solutions on specific problems encountered with UFT or tutorial on specific test methods, the results pages are not all linked back to HP.

This is of course because LeanFT is still quite new and there is not many full time users yet. Another possible reason can be the fact that LeanFT users are mostly companies and not many private users, usually when there is a lot of online support it is from private users using their free time to help other private users. Companies usually get support from the company providing the product.

HP's own help centre for LeanFT also had help on other subjects than just setup, they had help for integration between ALM, Selenium and other programs a user might like to use together with LeanFT. They also provide with references for UFT users where the equivalent UFT features in LeanFT are listed.

HP also has a community for UFT practitioners where users can ask questions and get help from other users. Experts from HP can also help by answering questions written in the community page. This community also accepts questions on LeanFT but at the moment the community seems to be mainly for UFT related problems. See [17] for the online community.

The links to pages which were not HP's own tutorials and help for LeanFT usage, were not as helpful or as detailed as those found at HP's help centre for LeanFT, see [18]. HP offers step by step guides for setup and multiple examples of how to build test steps for GUI tests and API tests in LeanFT.

5.4 Connecting LeanFT project to Git

The execution for connecting an LeanFT project or Git conducted by the thesis workers in section 4.4, gave the results that it is possible to connect a LeanFT project, with Application Models and NUnit

settings to a remote Git repository.

With Visual Studio it was quite simple to connect the tests to an online repository with Git and Bitbucket.

5.5 Connecting LeanFT with Jenkins

The issues faced when connecting LeanFT were few and miniscule. The first problem was determining what path to type in the textbox. Since, the NUnit runner was installed with the NuGet Package Manager, the path to this laid in the test project directory.

The other issue was realising that the test projects were not existent on the local disk, which hindered the Jenkins tool from accessing the test. This was solved by re-locating the test projects to the local disk. Besides these issues, the process of connecting LeanFT tests with Jenkins was uncomplicated.

Chapter 6: Conclusion

This chapter consists of a reflection of the ethical aspects of the thesis work and the answers to the problems defined in chapter 1: Introduction. The subheadings 6.3-6.8 are the problem definitions of the thesis work which can be found in chapter 1: Introduction.

6.1 Future development

During the thesis work the thesis workers were only able to test GUI testing with LeanFT and they were only able to create smaller tests, because of this the future development of the investigation work should be to create more complex tests and to test API testing with LeanFT. Because the thesis workers only had their own two computers to test on it was hard to examine if the results of the thesis work would be consistent when a large company like IKEA IT run their tests projects in LeanFT.

LeanFT is a very new tool and is therefore still under development by HP, therefore it is likely that the program will have added functionality and features in the future.

Page Object is a design pattern which can be beneficial in test automation by improving test maintenance and reduce code duplicity. The Page Object pattern follows the object oriented programming principle, where every page object is seen as an object class. The page objects serves as an interface to pages in the application or webpage. It would be interesting to examine if the Page Object pattern can be used when creating LeanFT tests, and if it would benefit IKEA IT TestCenter.

6.2 LeanFT maturity level

This section answers what conclusion the thesis workers came to on the question: *LeanFT maturity level – is the usability of LeanFT comparable to UFT? Any major bugs which hinder the use of the LeanFT tool?*.

The thesis workers strived to test out most of the features that LeanFT offers, in order to come to a valid conclusion. However, it was still important to acknowledge the limits of this research. An example of this, is the fact that the thesis workers did not work with API testing, which is an important part of the IKEA IT test process. Due to this, it was not possible to determine whether the LeanFT tool is comparable to UFT in this area.

Another example is the relatively simplicity of the tests that were created. The tests were testing the IKEA website, which has already been tested and approved as a well functioning webpage. The thesis workers were not able to change the source code to intentionally generate errors for testing purposes. The

results may have varied if the tools were testing faulty application.

During the process of using UFT and LeanFT for creating tests, there were several points noted on the stability of these testing tools. As mentioned in Chapter 5.1, UFT suffered from stability issues. The program would often freeze or crash during execution or creation of tests. This was also mentioned by one of the interviewees, as a common issue with UFT. There were also issues with UFT giving inconsistent results, which was often easily fixed by restarting the program but this can nevertheless lead to increased frustration for users of the program.

LeanFT had less stability issues compared to UFT and was therefore an easier tool to use. Though the thesis workers did not try test with too much complexity, due to time constraints, they came to the conclusion that LeanFT does seem to have quite a good level of maturity, they did not notice any major bugs which impeded their testing process. Because they were unable to try big projects they can not say that there aren't any bugs at all or that there won't be issues when big projects are tested, but from their tests and results they believe that LeanFT has a good maturity level.

6.3 Public community for LeanFT

From the research conducted by the thesis workers, as outlined in section 3.2.1.3, the thesis workers were able to analyse as outlined in section 4.3 if there exists, as of the writing of the thesis, an online public community for LeanFT. This section outlines what conclusion the thesis workers came to on the question: *Public community for LeanFT - is there a public community for LeanFT where one can get assistance if needed?*

With the results, as outlined in section 5.3, the thesis workers came to the conclusion that there exists quite a bit of online support for LeanFT. The support that exists is mostly on HP's own help pages, such as their help centre and their community for UFT practitioners.

A problem with the community help page is that to be able to contribute answers or questions to the forum you have to be a registered user and to become a registered user you must be connected to a company, which limits an open online community support. This is a difference for other online support sources for other programs which are more for private users and are mostly open for all, such as stackoverflow.com where pretty much anyone can ask questions and answer questions. This is a small problem because most users of UFT and LeanFT are within companies and work with a company license for LeanFT and UFT.

The thesis workers' conclusion for the problem of "Is there a public community for LeanFT" is that yes, there is, it is mostly on the developer's own web pages but it does exist and have a lot of helpful tutorials. There might be a larger community in the future when there are more users of LeanFT who would like to

share in their acquired knowledge, but for now it is adequate for getting started with LeanFT.

6.4 What is/is not possible with LeanFT when comparing with UFT

One of the problem definitions of this thesis, as defined in section 1.3: Problem definition was: *What is/is not possible with LeanFT when comparing with UFT?* . The thesis workers researched this during the research, analysis and the creation of tests phases.

From the information gathered in the research phase and analysis phase, see chapter 3.2, and the results from the GUI tests the thesis workers made in the creation of tests phase, see chapter 4.1 and 5.1, the thesis workers came up with a list of things from UFT and their equivalence in LeanFT see table 6.1.

Table 6.1: Table with LeanFT and UFT features

Feature in UFT	**Equivalence in LeanFT**
Object repository	Application Model
Object spy	Object Identification Center
Drag-and-drop feature with object repository	Application Model, export to clipboard
Descriptive programming	OIC code to clipboard
Function library	Class with functions as methods imported to project
Connecting project to ALM	

In table 6.1 the left side of the table is the same as table 3.1, the right side are the features that are similar, or the same, that exist in LeanFT.

In LeanFT it is also possible to connect the project with both the continuous integration tool Jenkins and the version control tool Git, according to the developer's website [21] it should also be possible to connect to other version control systems.

It should also be possible, according to the developers [4] to run multiple tests simultaneously while testing web applications which is explained here [24]. The thesis workers tested this, which can be seen in Appendix E where they did Test A in both Chrome and Internet Explorer at the same time and it works

very well.

When a tester is not using the Application Model to identify their objects and are therefore only using the OIC to identify single objects and then adding the code to their projects, as seen in section 4.1.2 and in Appendix A: Test A LeanFT, it is easy to change for example the browser to test in by just changing one line of code, there is no need to change anything in the objects because they are not connected to any specific repository. This makes it easier to do parallel tests of the same functionality in web pages on different browsers. LeanFT works with many different browsers when doing web based GUI tests, such as Internet Explorer, Chrome, Safari and many others.

There are some things which LeanFT can't do at the moment, these are:

- Visual API Testing / Web Services Testing
- Business Process Testing (BPT) Integration

The thesis workers were not able to, because of time constraints and complexity, to test these functions and are therefore unable to say if they work or not. According to the developers', HP, own webpage explaining what LeanFT can and can't do these functions are things it can't do, see [21]. This information comes directly from the developer it is therefore deemed credible.

The conclusion for this question the thesis workers made were that there are many things a tester can do in LeanFT and that a tester doing GUI tests should be able to do everything they did in UFT in LeanFT.

6.5 LeanFT – standalone

This section outlines the conclusion the thesis workers came to on the question, as defined in section 1.3: Problem definition, : *LeanFT – standalone – will there be any lost functionality if deciding to leave UFT, can LeanFT be used without UFT?*. The thesis workers researched this while creating their tests, see section 3.2.2 for how they came up with the tests and see section 4.1 for the execution of the tests. To read their analysis of comparing LeanFT and UFT see section 5.1.

The conclusion the thesis workers came to after doing their research and testing LeanFT with their own test cases, was that they believe that when performing GUI tests it will be possible to use LeanFT as a standalone program for automated tests. The thesis workers were only able to test some simple GUI tests due to time constraints on their thesis work and can therefore not say that there won't be *any* problems when going from performing tests in UFT to performing tests in LeanFT, but from their work they can say that it *should* be possible to do it.

6.6 Is it possible to create BDD tests using LeanFT

One of the questions asked in the beginning of this thesis work, as defined in section 1.3: Problem Definition, was: *Is it possible to create BDD tests using LeanFT? - as shift left is the overall approach for development at IKEA, a BDD way of thinking around test development is of course something that everybody wants but can it be done with LeanFT ?.*

.

Projects often involve a large group of people with some of them coming from a non-technical background. These people can be stakeholders of the projects, business analysts and so forth. With the help of BDD tests IT teams can include these members in the developing process. As seen in section 5.2, the tests can be executed from the feature file, which is the test scenario file. This means that it is not required to examine any code to be able to apprehend the purpose of the test or method, that is being tested. Stakeholders can therefore be invited to give feedback and input during the development process, despite them not being able to understand the implementation code.

Based on the analysis done on the process of creating behaviour driven tests, in Specflow, the thesis workers have come to a conclusion on this matter. The conclusion is that creating BDD tests using LeanFT is not more complex than it is to create regular tests using LeanFT. However, it is still important to mention yet again that this evaluation is only based on GUI testing. Other forms testing might require more effort when attempting to create BDD tests.

6.7 Can one incorporate a better versioning system (e.g. Git) with LeanFT

One of the questions asked for this thesis work was: *Can one incorporate a better versioning system (e.g. Git) with LeanFT?*. From the analysis conducted in chapter 4.4 and the results written in chapter 5.4 the thesis workers came to the conclusion that it is possible to connect LeanFT with Git. They thought that it was quite simple to do with the tools provided by Visual Studio in integrating with Git repositories.

The thesis workers only tested Git with the online tool Bitbucket and not with an own Git server so they can therefore not be completely certain that it will work as well as their tests, but they believe that it should work as well as their connection with Git and Bitbucket or other online sources.

With this the thesis workers believe that it should be possible for IKEA test centre and other companies to use Git as a versioning system while working on LeanFT projects. This could help the companies to have a good control on which version of a project or file they are supposed to use during their testing.

6.8 Using LeanFT together Jenkins and ALM

The last question defined in section 1.3: Problem Definition was: *LeanFT – use together with Jenkins and create a packaged deal as deliverable, distributable to projects together with ALM connection and*

shared libraries for utility functionality, is it possible?.

IKEA IT are planning on moving in a direction where projects can practise Continuous Integration. Jenkins helps developer teams with continuous integration, by offering teams to schedule builds and help keep track the complete process of automating tests. Testers can also upload tests which other testers can access without needing the Visual Studio program. This would result in an economic advantage, since there would not be any need for additional licenses.

As written in Section 5.5, the process of connecting and running LeanFT test with Jenkins was simple and even though the thesis workers were not granted access to the IKEA IT Jenkins server, the steps for connecting with a Jenkins server, which is not existent on the host computer, would be the same.

6.8 Reflection over ethical aspects

The main aspect the thesis workers reflected on during their thesis work was the value of testing software. If a software is well tested it can protect against privacy invasion brought on by for example compromised databases filled with personal information. If the database connection and security is tested well enough through for example using continuous testing early on in a software development project, it can help find problems early on, problems that could compromise data.

The thesis workers found no ethical dilemma regarding the testing tools. UFT and LeanFT do not have any ways of deliberately infringing on personal data or gathering confidential information when being used.

Chapter 7: Terminology

Shared object repositories - Contains test objects that represent the objects in one´s testing application
Function library - Contains custom functions that can be used in the developed test
DLL file - a file containing functions of a program which can be shared with different programs.
Descriptive programming - a description of an object, which describes where on a page it is, its unique name and other factors, used in UFT when programming without an object repository
Feature file - A file which is used in BDD which contains one or several scenarios.

Chapter 8: References

[1]: General UFT information: https://saas.hpe.com/en-us/software/uft (2017-04-06)

[2]:UFT tutorial recording: https://www.tutorialspoint.com/qtp/qtp_record_and_play_back.htm (2017-04-22)

[3]: UFT GUI tutorial from HP: http://technodivine.com/downloads/Help-and-Guides/02-QTP/05-UFT-Documentation-12.02/04-UFT_GUI_Tutorial_Web.pdf (2017-04-26)

[4]: General LeanFT information: https://saas.hpe.com/en-us/software/leanft (2017-04-06)

[5]: LeanFT Application Model information: http://leanft-help.saas.hpe.com/en/14.00/HelpCenter/Content/HowTo/TestObjects_AppModel.htm#hp-minitoc-item-0(2017-04-25)

[6]: General ALM information: https://saas.hpe.com/en-us/software/application-lifecycle-management (2017-04-06)

[7]: General Git information: https://git-scm.com/about (2017-04-06)

[8]: General Visual Studio information: https://www.visualstudio.com/vs/ide/ (2017-04-06)

[9]: General C# information: Skansholm, Jan. 2008. *Skarp Programmering med C#*.Poland: Studentlitteratur.

[10]:General SpecFlow information: http://specflow.org/documentation/FAQ/(2017-05-01)

[11]: Complete GUI Testing Complete Guide http://www.guru99.com/gui-testing.html (2017-05-08)

[12]: An introduction to BDD: https://dannorth.net/introducing-bdd/ (2017-05-03)

[13]: Explaining Test Scenarios BDD: http://www.agiletestingframework.com/atf/testing/behavior-driven-development-bdd/ (2017-05-03)

[14]: The Purpose of BDD: http://searchsoftwarequality.techtarget.com/definition/Behavior-driven-development-BDD (2017-05-03)

[15]: What is GUI Testing:

http://whatis.techtarget.com/definition/GUI-testing-graphical-user-interface-testing (2017-05-08)

[16]: Jenkins, a Continuous Integration tool: ACI (automated Continuous Integration) using Jenkins:Key for successful embedded Software development` 2015 *2015 2Nd International Conference On Recent Advances In Engineering & Computational Sciences (RAESC), Recent Advances In Engineering & Computational Sciences (RAESC), 2015 2Nd International Conference On,* p 1, IEEE Xplore Digital Library, EBSCO*host,* viewed 13 May 2017

[17]: Git repository structure image: https://git-scm.com/images/about/branches@2x.png (2017-04-06)

[18]: Video connecting Visual Studio to Bitbucket: Trey50Daniel. (2016). *Using Bitbucket with Visual Studio*. [Online Video]. 23 June 2016. Available from: https://www.youtube.com/watch?v=p0YSX5kDfP8. [Accessed: 4 May 2017].

[19]: LeanFT help center:
http://leanft-help.saas.hpe.com/en/14.00/HelpCenter/Content/Resources/_TopNav/_TopNav_Home.htm (2017-04-27)

[20]: UFT community help:
https://community.hpe.com/t5/Unified-Functional-Testing/bd-p/sws-Fun_TEST_SF#.WQsTtuXyiUk (201-05-04)

[21]: Comparison UFT and LeanFT: https://saas.hpe.com/en-us/software/uft-leanft-comparison (2017-05-08)

[22]: Visual Basic: Halvorson, Michael. 2013. *Microsoft Visual Basic 2013, Step by Step*. USA: Microsoft Press.

[23]: Interview methods: Magnusson, E, & Marecek, J 2015, Doing Interview-Based Qualitative Research : A Learner's Guide, n.p.: Cambridge : Cambridge University Press, 2015., Library catalogue (Lovisa), EBSCOhost, viewed 13 May 2017

[24]: Running paralell tests in LeanFT:
http://leanft-help.saas.hpe.com/en/14.00/HelpCenter/Content/HowTo/RunLeanFT-Parallel.htm (2017-05-12)

[25]: UFT Guide:
http://technodivine.com/downloads/Help-and-Guides/02-QTP/05-UFT-Documentation-12.02/04-UFT_GUI_Tutorial_Web.pdf (2017-03-02)

[26]: Page Object Pattern:
http://www.seleniumhq.org/docs/06_test_design_considerations.jsp#page-object-design-pattern
(2017-05-14)

[27]: Integrating SpecFlow with LeanFT:
http://leanft-help.saas.hpe.com/en/14.00/HelpCenter/Content/HowTo/Cucumber.htm
(2017-05-01)

[28]: LeanFT connecting with Jenkins tutorial:
http://leanft-help.saas.hpe.com/en/14.00/HelpCenter/Content/HowTo/CI_Tools.htm (2017-05-14)

[29]: IKEA homepage:
http://www.ikea.com/se/sv/ (2017-05-17)

Appendix A: Questionnaires

This appendix contains the questions the thesis workers used during their interviews.

Questionnaire 1

How are you currently using UFT?
What are you using UFT for?
Is UFT limiting you in any way?
Is it important that LeanFT is as graphical as UFT?
What functionality in UFT is important for you to have in LeanFT?
What are the strengths of UFT?
Do you think it will be more difficult for non programmers with testing in LeanFT?
What is being tested at IKEA IT testcenter?

Questionnaire 2

What functionality is important for you in UFT?
What is your main motivation for wanting to use LeanFT?
Is it possible to integrate UFT with Git?
Mostly code scripts used?
What is most important for Leanft to work?
How is your vision for the future at testcenter(non programmers)?
Do you feel that UFT limits you in any way?
What are your issues with VBScript?
How is the UFT support?
What functionality must exist in LeanFT?

Appendix B: Test code Test A and B

This appendix contains the code for the tests conducted in LeanFT and UFT.

LeanFT Test A

```
using System;
using NUnit.Framework;
using HP.LFT.SDK;
using HP.LFT.Verifications;
using HP.LFT.SDK.Web;


namespace LeanFtTestProject4
{
	[TestFixture]
	 public class LeanFtTest : UnitTestClassBase
	{
		[OneTimeSetUp]
		public void TestFixtureSetUp()
		{
		// Setup once per fixture
		}
		[SetUp]
		public void SetUp()
		{
		// Before each test
		}
		[Test]
		public void Test()
		{

				IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);
				browser.Navigate("http://www.ikea.com/se/sv/");

				browser.Describe<ILink>(new LinkDescription
				{
					Role = string.Empty,
					AccessibilityName = string.Empty,
					TagName = @"A",
					InnerText = @"Sängar",
					Index = 0
				}).Click();

				browser.Describe<ILink>(new LinkDescription
				{
					Role = string.Empty,
```

```
            AccessibilityName = string.Empty,
            TagName = @"A",
            InnerText = @"Lakan",
            Index = 0
}).Click();

browser.Describe<IImage>(new ImageDescription
{
            Alt = @"DVALA lakan, beige Längd: 260 cm Bredd: 150 cm",
            Type = HP.LFT.SDK.Web.ImageType.Link,
            TagName = @"IMG"
}).Click();

browser.Describe<ILink>(new LinkDescription
{
            Role = string.Empty,
            AccessibilityName = string.Empty,
            TagName = @"A",
            InnerText = string.Empty,
            Index = 4
}).Click();

browser.Describe<ILink>(new LinkDescription
{
            Role = string.Empty,
            AccessibilityName = string.Empty,
            TagName = @"A",
            InnerText = string.Empty,
            Index = 2
}).Click();

browser.Describe<IButton>(new ButtonDescription
{
            ButtonType = @"button",
            Role = string.Empty,
            AccessibilityName = string.Empty,
            TagName = @"INPUT",
            Name = @"FORTSÄTT TILL KASSAN",
            Index = 1
}).Click();

browser.Describe<IEditField>(new EditFieldDescription
{
            Type = @"text",
            TagName = @"INPUT",
            Name = @"firstName"
}).SetValue("Sara");
```

```
browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"lastName"
}).SetValue("Berg");
browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"address1"
}).SetValue("Sjögatan 1");

browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"zipCode"
}).SetValue("252 25");
browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"city"
}).SetValue("Helsingborg");
browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"email1"
}).SetValue("sara_berg@hotmail.com");
browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"email1retype"
}).SetValue("sara_berg@hotmail.com");
browser.Describe<IEditField>(new EditFieldDescription
{
        Type = @"text",
        TagName = @"INPUT",
        Name = @"phone1"
}).SetValue("0722569887");
browser.Describe<IButton>(new ButtonDescription
{
        ButtonType = @"button",
        TagName = @"INPUT",
        Name = @"Spara och fortsätt till leveransinformation"
```

```
                }).Click();
                }


        [TearDown]
        public void TearDown()
        {

        // Clean up after each test
        }

        [OneTimeTearDown]
        public void TestFixtureTearDown()
        {

        // Clean up once per fixture
                }
        }
}
```

UFT Test A

```
Browser("IKEA - Möbler, inredning").Page("IKEA - Möbler, inredning").Link("Sängar").Click
Browser("IKEA - Möbler, inredning").Page("IKEA - Möbler, inredning").Link("Lakan").Click
Browser("IKEA - Möbler, inredning").Page("Lakan hjälper dig sova").WebElement("DVALA").Click
Browser("IKEA - Möbler, inredning").Page("DVALA Lakan - 150x260").Link("jsButton_buyOnline_lnk").Click
Browser("IKEA - Möbler, inredning").Page("DVALA Lakan - 150x260").Link("Kundvagn").Click
Browser("IKEA - Möbler, inredning").Page("Kundvagn - IKEA").WebButton("FORTSÄTT TILL KASSAN").Click
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("firstName").Set"Sara"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("lastName").Set"Berg"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("address1").Set"Sjögatan 1"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("zipCode").Set"252 25"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("phone2").Set"072261671"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("city").Set"Helsingborg"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("email1").Set"sara.berg@hotmail.com"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("email1retype").Set"sara.berg@hotmail.com"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA").WebButton("Spara och fortsätt till").Click
```

LeanFT Test B

```
using System;
using NUnit.Framework;
using HP.LFT.SDK;
using HP.LFT.Verifications;
using HP.LFT.SDK.Web;
namespace LeanFtIKEAScenario2
{
	[TestFixture]
	public class LeanFtTest : UnitTestClassBase
	{
		[OneTimeSetUp]
		public void TestFixtureSetUp()
		{
			// Setup once per fixture
		}

		[SetUp]
		public void SetUp()
		{
			// Before each test
		}

		[Test]
		public void IKEATest()
		{
			//Open Internet Explorer
			IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);
			//Navigate to IKEA homepage
			browser.Navigate("http://www.ikea.com/se/sv/");

			//Create new ApplicationModel where our links are
			ApplicationModel1 appModel = new ApplicationModel1(browser);
			//Search for "Skrivbord"
			appModel.IKEAMBlerInredningOchInspirationIKEAPage.QueryEditField.SetValue("Skrivbord");
			appModel.IKEAMBlerInredningOchInspirationIKEAPage.LnkSearchBtnHeaderButton.Click();
			//Click "Galant"
			appModel.SKresultatIKEAPage.SkPMedSkjutdRrarWebElement.Click();
			//Click grå
			appModel.GALANTSkPMedSkjutdRrarVitIKEAPage.AttributeValuesSelectListBox.Select("grå");
			//Add 2 to cart
			appModel.GALANTSkPMedSkjutdRrarGrIKEAPage.QuantityEditField.SetValue("2");
			//Click "KöpOnline"
			appModel.GALANTSkPMedSkjutdRrarGrIKEAPage.KPOnlineButton.Click();
			//Click "Kundvagn"
			appModel.GALANTSkPMedSkjutdRrarGrIKEAPage.KundvagnLink.Click();
```

```
                        //Click "Fortsätt till kassan"
                        appModel.KundvagnIKEAPage.FORTSTTTILLKASSANButton.Click();
                        //Fill in the form for Kundvagn
                        appModel.AdressIKEAPage.FirstNameEditField.SetValue("Sara");
                        appModel.AdressIKEAPage.LastNameEditField.SetValue("Berg");
                        appModel.AdressIKEAPage.Address1EditField.SetValue("Sjögatan 1");
                        appModel.AdressIKEAPage.ZipCodeEditField.SetValue("252 25");
                        appModel.AdressIKEAPage.CityEditField.SetValue("Helsingborg");
                        appModel.AdressIKEAPage.Email1EditField.SetValue("sara_berg@hotmail.com");
                        appModel.AdressIKEAPage.Email1retypeEditField.SetValue("sara_berghotmail.com");
                        appModel.AdressIKEAPage.Mobilephone2EditField.SetValue("0722569887");
                        appModel.AdressIKEAPage.SparaOchFortsTtTillLeveransinformationButton.Click();

                }
                [TearDown]
                public void TearDown()
                {
                        // Clean up after each test
                }
                [OneTimeTearDown]
                public void TestFixtureTearDown()
                {
                        // Clean up once per fixture
                }
        }
}
```

UFT Test B

Browser("IKEA - Möbler, inredning").Page("IKEA - Möbler, inredning").WebEdit("query").Set "Skrivbord"
Browser("IKEA - Möbler, inredning").Page("IKEA - Möbler, inredning").WebEdit("query").Submit
Browser("IKEA - Möbler, inredning").Page("Sökresultat - IKEA").Link("GALANTSkåp med skjutdörrar").Click
Browser("IKEA - Möbler, inredning").Page("GALANT Skåp med skjutdörrar").WebEdit("quantity").Set "2"
Browser("IKEA - Möbler, inredning").Page("GALANT Skåp med skjutdörrar").Link("jsButton_buyOnline_lnk").Click
Browser("IKEA - Möbler, inredning").Page("GALANT Skåp med skjutdörrar").Link("Kundvagn").Click
Browser("IKEA - Möbler, inredning").Page("Kundvagn - IKEA").WebButton("FORTSÄTT TILL KASSAN").Click
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("city").Set"Helsingborg"

```
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("email1").Set"sara.berg@hotmail.com"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("email1retype").Set"sara.berg@hotmail.com"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("firstName").Set"Sara"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("lastName").Set"Berg"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("zipCode").Set"252 25"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA_2").WebEdit("phone2").Set"0722569887"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA").WebEdit("address1").Set "Sjögatan1"
Browser("IKEA - Möbler, inredning").Page("Adress - IKEA").WebButton("Spara och fortsätt till").Click
```

Appendix C: BDD Test Scenario Examples

Test scenario: Subway ticket machine

Title: Customer withdraws subway ticket
As a customer
I want to withdraw a subway ticket from the machine
so that i don't have to wait in line at the ticket office

This story card can be translated into a test scenario that tests that the machine behaves correctly

Scenario: Given the machine has tickets
And the customer provides trip details
And the customer provides an acceptable form of payment
When the customer requests the ticket
Then ensure the ticket is dispensed
And ensure the payment is debited

The entire example can be found [16].

Appendix D: BDD Scenario 1 and 2

This appendix contains the two BDD test scenarios conducted in the thesis.

BDD Test 1 feature file/Test scenario

Scenario: Buy an item on the ikea website by using the search field
Given I am on the IKEA homepage
And I navigate to the field "SÄNGAR"
And I choose the field "LAKAN"
And I choose the "DVALA lakan beige" item
When I press the buy online button
And I go to the shopping cart site
And I continue to the checkout
And I fill in my personal information
And I continue to the delivery information site
Then I shall come to the delivery information site where i can see the items chosen

BDD Test 1:

```
using System;
using TechTalk.SpecFlow;
using HP.LFT.SDK.Web;
using HP.LFT.SDK;
using HP.LFT.Verifications;
using NUnit.Framework;



namespace BDD1
{
   [Binding]
   public class SpecFlowFeature1Steps
   {
      IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);
      ApplicationModel1 app1;
      int quantity;
      [Given(@"I am on the IKEA homepage")]
```

```
public void GivenIAmOnTheIKEAHomepage()
{
   browser.Navigate("http://www.ikea.com/se/sv/");
   app1 = new ApplicationModel1(browser);
}

[Given(@"I navigate to the field ""(.*)""")]
public void GivenIHaveNavigatedToTheField(string p0)
{
   browser.Describe<ILink>(new LinkDescription
   {
      Role = string.Empty,
      AccessibilityName = string.Empty,
      TagName = @"A",
      InnerText = @"Sängar",
      Index = 0
   }).Click();

}

[Given(@"I choose the field ""(.*)""")]
public void GivenIHaveChoosenTheField(string p0)
{
   browser.Describe<ILink>(new LinkDescription
   {
      Role = string.Empty,
      AccessibilityName = string.Empty,
      TagName = @"A",
      InnerText = @"Lakan",
      Index = 0
   }).Click();

}

[Given(@"I choose the ""(.*)"" item")]
public void GivenIHaveChoosenTheItem(string p0)
{
   browser.Describe<IImage>(new ImageDescription
   {
      Alt = @"DVALA lakan, beige Längd: 260 cm Bredd: 150 cm",
      Type = HP.LFT.SDK.Web.ImageType.Link,
```

```
        TagName = @"IMG"
    }).Click();
}

[When(@"I press the buy online button")]
public void WhenIPressTheBuyOnlineButton()
{
    browser.Describe<ILink>(new LinkDescription
    {
        Role = string.Empty,
        AccessibilityName = string.Empty,
        TagName = @"A",
        InnerText = string.Empty,
        Index = 4
    }).Click();

}

[When(@"I go to the shopping cart site")]
public void WhenIGoToTheShoppingCartSite()
{
    browser.Describe<ILink>(new LinkDescription
    {
        Role = string.Empty,
        AccessibilityName = string.Empty,
        TagName = @"A",
        InnerText = string.Empty,
        Index = 2
    }).Click();
}

[When(@"I continue to the checkout")]
public void WhenIContinueToTheCheckout()
{
    browser.Describe<IButton>(new ButtonDescription
    {
        ButtonType = @"button",
        Role = string.Empty,
        AccessibilityName = string.Empty,
        TagName = @"INPUT",
        Name = @"FORTSÄTT TILL KASSAN",
```

```
        Index = 1
    }).Click();

}


[When(@"I fill in my personal information")]
public void WhenIFillInMyPersonalInformation()
{
    browser.Describe<IEditField>(new EditFieldDescription
    {
        Type = @"text",
        TagName = @"INPUT",
        Name = @"firstName"
    }).SetValue("Sara");
    browser.Describe<IEditField>(new EditFieldDescription
    {
        Type = @"text",
        TagName = @"INPUT",
        Name = @"lastName"
    }).SetValue("Berg");
    browser.Describe<IEditField>(new EditFieldDescription
    {
        Type = @"text",
        TagName = @"INPUT",
        Name = @"address1"
    }).SetValue("Sjögatan 1");


    browser.Describe<IEditField>(new EditFieldDescription
    {
        Type = @"text",
        TagName = @"INPUT",
        Name = @"zipCode"
    }).SetValue("252 25");
    browser.Describe<IEditField>(new EditFieldDescription
    {
        Type = @"text",
        TagName = @"INPUT",
        Name = @"city"
    }).SetValue("Helsingborg");
```

```
        browser.Describe<IEditField>(new EditFieldDescription
        {
            Type = @"text",
            TagName = @"INPUT",
            Name = @"email1"
        }).SetValue("viktoria_a94@hotmail.com");
        browser.Describe<IEditField>(new EditFieldDescription
        {
            Type = @"text",
            TagName = @"INPUT",
            Name = @"email1retype"
        }).SetValue("viktoria_a94@hotmail.com");
        browser.Describe<IEditField>(new EditFieldDescription
        {
            Type = @"text",
            TagName = @"INPUT",
            Name = @"phone1"
        }).SetValue("0722569887");
    }

    [When(@"I continue to the delivery information site")]
    public void WhenIContinueToTheDeliveryInformationSite()
    {
        browser.Describe<IButton>(new ButtonDescription
        {
            ButtonType = @"button",
            TagName = @"INPUT",
            Name = @"Spara och fortsätt till leveransinformation"
        }).Click();
    }

    [Then(@"I shall come to the delivery information site where i can see the items chosen")]
    public void ThenIShallComeToTheDeliveryInformationSiteWhereICanSeeTheItemsChoosen()
    {
        var expectedURL =
"https://secure.ikea.com/webapp/wcs/stores/servlet/IrwDeliveryOptionsView?langId=-11&storeId=2&kr
ypto=KLYYf5qgBi%2BIkrw4hMuWK68hDBnRq35vRSUzuEThK8BiyTfw1jtvDhNkaIxKD0%2F4yh9z
JIKz%2FkR6wdyUIsBEzN6yKQMoeupd3JeDl4hP%2BGQzN5eqJxFxE5ojEXrEvOwExz2lhcoIkbthmg
amen4UgwgG2H64VcHUUR%2F084zYS%2BU%3D&ddkey=https%3AIrwProceedFromCheckoutAddr
essView";
        var current = browser.URL;
```

```
            Assert.AreEqual(current, expectedURL);


        }
    }
}
```

BDD Test 2 feature file/Test scenario:

```
Scenario: Buy the "GALANT skåp med skjutdörrar" item from the IKEA website
        Given I am on the IKEA homepage
        And I search for "Skrivbord" in the searchfield
        And I choose the "GALANT skåp med skjutdörrar" item
        And I choose the colour grey
        And I select the quantity 2
        When I press buy online
        And I go to the shoppingcart
        And I continue to the checkout
        And I fill in my personal information
        And i press the continue to delivery information button
        Then I should come to the delivery information webpage with a view of my choosen items
```

BDD Test 2:

```
using System;
using TechTalk.SpecFlow;
using HP.LFT.SDK.Web;
using HP.LFT.SDK;
//using HP.LFT.Verifications;
//using NUnit.Framework;

namespace BDD2
{
    [Binding]
    public class SpecFlowFeature1Steps
    {
        IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);

        private ApplicationModel1 app1;
```

```
    [Given(@"I am on the IKEA homepage")]
public void GivenIAmOnTheIKEAHomepage()
{
   browser.Navigate("http://www.ikea.com/se/sv/");
  app1 = new ApplicationModel1(browser);

}

[Given(@"I search for ""(.*)"" in the searchfield")]
public void GivenISearchForInTheSearchfield(string p0)
{
   browser.Describe<IEditField>(new EditFieldDescription
   {
      Type = @"text",
      TagName = @"INPUT",
      Name = @"query"
   }).SetValue("GALANT");
   browser.Describe<IButton>(new ButtonDescription
   {
      ButtonType = @"submit",
      TagName = @"INPUT",
      Name = string.Empty
   }).Click();
}

[Given(@"I choose the ""(.*)"" item")]
public void GivenIHaveChoosenTheItem(string p0)
{
   browser.Describe<ILink>(new LinkDescription
   {
      TagName = @"A",
      InnerText = @" GALANTSkåp med skjutdörrar   4 295 kr /styck  Unit price   "
   }).Click();
}

[Given(@"I choose the colour grey")]
public void GivenIChooseTheColourGrey()
{

   app1.GALANTSkPMedSkjutdRrarVitIKEAPage.AttributeValuesSelectListBox.Select("grå");
```

```
}

[Given(@"I select the quantity (.*)")]
public void GivenISelectTheQuantity(int p0)
{
   app1.GALANTSkPMedSkjutdRrarVitIKEAPage.QuantityEditField.SetValue("2");

}

[When(@"I press buy online")]
public void WhenIPressBuyOnline()
{

   app1.GALANTSkPMedSkjutdRrarVitIKEAPage.KPOnlineButton.Click();

}

[When(@"I go to the shoppingcart")]
public void WhenIGoToTheShoppingcart()
{

   app1.GALANTSkPMedSkjutdRrarGrIKEAPage.MinKundvagnLink.Click();

}

[When(@"I continue to the checkout")]
public void WhenIContinueToTheCheckout()
{

   app1.KundvagnIKEAPage.FORTSTTTILLKASSANButton.Click();
}

[When(@"I fill in my personal information")]
public void WhenIFillInMyPersonalInformation()
{
   app1.AdressIKEAPage.FirstNameEditField.SetValue("Sara");
   app1.AdressIKEAPage.LastNameEditField.SetValue("Berg");
   app1.AdressIKEAPage.Address1EditField.SetValue("Sjögatan 1");
   app1.AdressIKEAPage.ZipCodeEditField.SetValue("252 25");
   app1.AdressIKEAPage.CityEditField.SetValue("Helsingborg");
```

```
            app1.AdressIKEAPage.Email1EditField.SetValue("viktoria_a94@hotmail.com");
            app1.AdressIKEAPage.Email1retypeEditField.SetValue("viktoria_a94@hotmail.com");

        }
        [When(@"i press the continue to delivery information button")]
        public void WhenIPressTheContinueToDeliveryInformationButton()
        {
            app1.AdressIKEAPage.SparaOchFortsTtTillLeveransinformationButton.Click();
        }


        [Then(@"I should come to the delivery information webpage with a view of my chosen items")]
        public void
ThenIShouldComeToTheDeliveryInformationWebpageWithAViewOfMyChoosenItems()
        {
            var expectedURL =
"https://secure.ikea.com/webapp/wcs/stores/servlet/IrwDeliveryOptionsView?langId=-11&storeId=2&kr
ypto=3AjDaSudtnU9ZlRxSTIWJQTiFKGQz4dKz%2FJSEONf3BCnXRofphhK%2FPAvyqeIWYZgpcs
8Rfs1VlQjc6oZDbLvQQ%2FjBVYsVIamKU16ecnCsA3Ww7JV93GCwslBeibY8uz476qWZUoPyMT3
pgfw83q30U8mys48pIsZv8lEU801Avk%3D&ddkey=https%3AIrwProceedFromCheckoutAddressView
";
            var current = browser.URL;
            expectedURL.Equals(current);


        }

    }
}
```

Appendix E: Parallel Test A

This appendix contains the test the thesis workers did for testing parallel tests in LeanFT. They tested Test A in both Internet Explorer and Chrome at the same time.

```
using System;
using NUnit.Framework;
using HP.LFT.SDK;
using HP.LFT.Verifications;
using HP.LFT.SDK.Web;

namespace LeanFtIKEAScenario1
{
   [TestFixture]
   [Parallelizable]
   public class LeanFtTest : UnitTestClassBase
   {
      [OneTimeSetUp]
      public void TestFixtureSetUp()
      {
         // Setup once per fixture
      }

      [SetUp]
      public void SetUp()
      {
         // Before each test
      }

      [Test]
      public void TestIKEA()
      {
         //Open Internet Explorer
         IBrowser browser = BrowserFactory.Launch(BrowserType.InternetExplorer);

         //Navigate to IKEA homepage
         browser.Navigate("http://www.ikea.com/se/sv/");

         //Click button "Sängar"
         browser.Describe<ILink>(new LinkDescription
```

```
{
   Role = string.Empty,
   AccessibilityName = string.Empty,
   TagName = @"A",
   InnerText = @"Sängar",
   Index = 0
}).Click();


//Click button "Lakan"
browser.Describe<ILink>(new LinkDescription
{
   Role = string.Empty,
   AccessibilityName = string.Empty,
   TagName = @"A",
   InnerText = @"Lakan",
   Index = 2
}).Click();

//Click "Dvala"
browser.Describe<IWebElement>(new WebElementDescription
{
   AccessibilityName = string.Empty,
   ClassName = @"productTitle floatLeft",
   TagName = @"DIV",
   InnerText = @"DVALA",
   Index = 3
}).Click();

//Click "KöpOnline"
browser.Describe<IButton>(new ButtonDescription
{
   ButtonType = @"button",
   TagName = @"INPUT",
   Name = @"Köp online"
}).Click();

//Click "Kundvagn"
browser.Describe<ILink>(new LinkDescription
{
   TagName = @"A",
```

```
        InnerText = @"Kundvagn"
    }).Click();

    //Click "Fortsätt till kassan"
    browser.Describe<IButton>(new ButtonDescription
    {
        ButtonType = @"button",
        Role = string.Empty,
        AccessibilityName = string.Empty,
        TagName = @"INPUT",
        Name = @"FORTSÄTT TILL KASSAN",
        Index = 1
    }).Click();


    //Create new ApplicationModel where our links are
    ApplicationModel1 appModel = new ApplicationModel1(browser);

    //Fill in the form for Kundvagn
    appModel.AdressIKEAPage.FirstNameEditField.SetValue("Sara");
    appModel.AdressIKEAPage.LastNameEditField.SetValue("Berg");
    appModel.AdressIKEAPage.Address1EditField.SetValue("Sjögatan 1");
    appModel.AdressIKEAPage.ZipCodeEditField.SetValue("252 25");
    appModel.AdressIKEAPage.CityEditField.SetValue("Helsingborg");
    appModel.AdressIKEAPage.Email1EditField.SetValue("viktoria_a94@hotmail.com");
    appModel.AdressIKEAPage.Email1retypeEditField.SetValue("viktoria_a94@hotmail.com");
    appModel.AdressIKEAPage.Mobilephone2EditField.SetValue("0706169057");
    appModel.AdressIKEAPage.SparaOchFortsTtTillLeveransinformationButton.Click();




}

[TearDown]
public void TearDown()
{
    // Clean up after each test
}

[OneTimeTearDown]
```

```
    public void TestFixtureTearDown()
    {
        // Clean up once per fixture
    }
}

[TestFixture]
[Parallelizable]
public class LeanFtTest2 : UnitTestClassBase
{
    [OneTimeSetUp]
    public void TestFixtureSetUp()
    {
        // Setup once per fixture
    }

    [SetUp]
    public void SetUp()
    {
        // Before each test
    }

    [Test]
    public void TestIKEAChrome()
    {
        //Open Chrome
        IBrowser browser = BrowserFactory.Launch(BrowserType.Chrome);

        //Navigate to IKEA homepage
        browser.Navigate("http://www.ikea.com/se/sv/");

        //Click button "Sängar"
        browser.Describe<ILink>(new LinkDescription
        {
            Role = string.Empty,
            AccessibilityName = string.Empty,
            TagName = @"A",
            InnerText = @"Sängar",
            Index = 0
        }).Click();
```

```
//Click button "Lakan"
browser.Describe<ILink>(new LinkDescription
{
   Role = string.Empty,
   AccessibilityName = string.Empty,
   TagName = @"A",
   InnerText = @"Lakan",
   Index = 2
}).Click();

//Click "Dvala"
browser.Describe<IWebElement>(new WebElementDescription
{
   AccessibilityName = string.Empty,
   ClassName = @"productTitle floatLeft",
   TagName = @"DIV",
   InnerText = @"DVALA",
   Index = 3
}).Click();

//Click "KöpOnline"
browser.Describe<IButton>(new ButtonDescription
{
   ButtonType = @"button",
   TagName = @"INPUT",
   Name = @"Köp online"
}).Click();

//Click "Kundvagn"
browser.Describe<ILink>(new LinkDescription
{
   TagName = @"A",
   InnerText = @"Kundvagn"
}).Click();

//Click "Fortsätt till kassan"
browser.Describe<IButton>(new ButtonDescription
{
   ButtonType = @"button",
   Role = string.Empty,
```

```
            AccessibilityName = string.Empty,
            TagName = @"INPUT",
            Name = @"FORTSÄTT TILL KASSAN",
            Index = 1
        }).Click();


        //Create new ApplicationModel where our links are
        ApplicationModel1 appModel = new ApplicationModel1(browser);

        //Fill in the form for Kundvagn
        appModel.AdressIKEAPage.FirstNameEditField.SetValue("Sara");
        appModel.AdressIKEAPage.LastNameEditField.SetValue("Berg");
        appModel.AdressIKEAPage.Address1EditField.SetValue("Sjögatan 1");
        appModel.AdressIKEAPage.ZipCodeEditField.SetValue("252 25");
        appModel.AdressIKEAPage.CityEditField.SetValue("Helsingborg");
        appModel.AdressIKEAPage.Email1EditField.SetValue("viktoria_a94@hotmail.com");
        appModel.AdressIKEAPage.Email1retypeEditField.SetValue("viktoria_a94@hotmail.com");
        appModel.AdressIKEAPage.Mobilephone2EditField.SetValue("0706169057");
        appModel.AdressIKEAPage.SparaOchFortsTtTillLeveransinformationButton.Click();




    }

    [TearDown]
    public void TearDown()
    {
        // Clean up after each test
    }

    [OneTimeTearDown]
    public void TestFixtureTearDown()
    {
        // Clean up once per fixture
    }
  }
}
```