Behavior Driven Development in a Large-Scale Application: Evaluation of Usage for Developing IFS Applications

by

Payman Delshad

LIU-IDA/LITH-EX-A--16/005--SE

2016-02-16
Final Thesis

Behavior Driven Development in a Large-Scale Application: Evaluation of Usage for Developing IFS Applications

by

Payman Delshad

LIU-IDA/LITH-EX-A--16/005--SE

2016-02-16

Supervisors: Mahder Gebremedhin, IDA, Linköping University
Oscar Rydberg, IFS World Operations AB

Examiner: Martin Sjölund, IDA, Linköping University
Abstract

Nowadays, Agile software development methods are often used in large multisite organizations that develop large-scale applications. Behavior Driven Development (BDD) is a relatively new Agile software development process where the development process starts with acceptance tests written in a natural language. The premise of BDD is to create a common and effective process of communication between different roles in a software project to ensure that every activity can be mapped to the business goal of the application. This thesis work aims to find an effective and efficient BDD process and to evaluate its usage in a large-scale application in a large multisite organization through a series of interviews, a controlled experiment, and an online survey. Furthermore, by means of the aforementioned experiment, the study measures the impact of an experimental usage of BDD on testing quality. To discover an effective and efficient BDD process, two alternatives with automated tests that run on different architectural layers, namely client layer and web service layer, were examined. Based on the defined metrics, the alternative with automated tests that ran directly on the web service layer was chosen as the more efficient process which was compared against the existing Agile-based baseline that used automated client tests. The results show that an efficient BDD process improves the testing quality significantly which can, in turn, result in a better overall software quality.

**Keywords:** Behavior Driven Development, BDD, Agile, Large-Scale Applications, Large Multisite Organizations, Testing Quality, Automated Testing
Acknowledgements

This thesis work has been carried out at IFS Linköping, between September 2015 and February 2016, as the last part of my master studies at Linköping University.

I am really grateful to the members of the “Methods and Processes” team at IFS Linköping for the idea behind this thesis and for giving me the opportunity to work on it. I would like to thank the participants in my experiment, survey, and interviews, who willingly shared their precious time to help me with this study.

Furthermore, I would like to express my gratitude to my examiner Martin Sjölund at IDA and my supervisors Mahder Gebremedhin at IDA and Oscar Rydberg at IFS for their continuous guidance and feedback.

Special thanks to my mother for everything she has done for me. I would not be the person I am today without her eternal love and support.

Finally, a big thanks to my friends Vaheed and Farrokh who helped me with this thesis report and to all my friends in Linköping who helped me keep my morale high during challenging times with their beautiful friendship.
## Contents

### CHAPTER 1  INTRODUCTION ........................................................................................................... 1

1.1 Motivation..................................................................................................................................... 1
1.2 Problem Description ................................................................................................................... 2
1.3 Research Question ..................................................................................................................... 2
1.4 Delimitations.............................................................................................................................. 3
1.5 Thesis Outline............................................................................................................................ 3

### CHAPTER 2  BACKGROUND ........................................................................................................... 5

2.1 IFS................................................................................................................................................. 5
2.2 IFS Applications .......................................................................................................................... 6
2.3 R&D.............................................................................................................................................. 6
2.4 Process Models and Workflows .................................................................................................... 7
2.5 Development Process ................................................................................................................... 8
2.6 Manual Testing ............................................................................................................................ 9
2.7 Automated Testing ....................................................................................................................... 9
  2.7.1 Current Setup .......................................................................................................................... 10
  2.7.2 Current Test Execution Time .................................................................................................. 11
  2.7.3 Desired Setup ......................................................................................................................... 11
  2.7.4 Desired Test Execution Time .................................................................................................. 12
2.8 Technical Architecture ................................................................................................................ 12
  2.8.1 IFS Applications ..................................................................................................................... 12
  2.8.2 IFS Web Client ....................................................................................................................... 13

### CHAPTER 3  BEHAVIOR DRIVEN DEVELOPMENT .................................................................... 15

3.1 Test Driven Development .......................................................................................................... 15
3.2 Behavior Driven Development .................................................................................................... 16
3.3 Defining the BDD Process .......................................................................................................... 16
  3.3.1 Requirements ........................................................................................................................ 17
  3.3.2 Design and Implementation .................................................................................................. 17
  3.3.3 Testing .................................................................................................................................. 17

### CHAPTER 4  THEORY ................................................................................................................... 19
List of Figures

Figure 2.1: Overview of IFS ........................................................................................................5
Figure 2.2: The structure of IFS and its business model.........................................................6
Figure 2.3: Overview of IFS Applications...............................................................................7
Figure 2.4: The structure of R&D .........................................................................................8
Figure 2.5: The three levels of IFS process models.................................................................8
Figure 2.6: Different phases of the AQUA process.................................................................9
Figure 2.7: The manual testing process at IFS.....................................................................9
Figure 2.8: The process of automating BFTs using CUIT tools...........................................10
Figure 2.9: The desired test automation pyramid of R&D....................................................11
Figure 2.10: Technical architecture of IFS Applications.......................................................13
Figure 2.11: Technical architecture of IFS web client...........................................................13
Figure 3.1: The TDD process ...............................................................................................15
Figure 3.2: The BDD process for a feature........................................................................17
Figure 4.1: The conceptual framework of the thesis............................................................21
Figure 5.1: The Customer Order workflow .......................................................................27
Figure 5.2: Testing possibilities for the web client...............................................................32
Figure 5.3: Specification for Enter customer order ...............................................................34
Figure 5.4: Jenkin setup for the experiment .......................................................................35
Figure 5.5: Initial test results for customer order feature ....................................................36
Figure 5.6: Intermediate test results for customer order feature ........................................37
Figure 5.7: Final test results for customer order feature ......................................................38
Figure 5.8: Test results as living documentation ..................................................................39
Figure 6.1: Test Execution Time .........................................................................................50
Figure 6.2: The functional roles of survey respondents .......................................................53
Figure 6.3: Business Goal Alignment – Specification vs. Feature ......................................55
Figure 6.4: Business Goal Alignment.................................................................56
Figure 6.5: Test Failure Traceability .................................................................58
Figure 6.6: Test Readability..............................................................................59
Figure 6.7: User Satisfaction............................................................................59
List of Tables

Table 6.1: Interviewees of Project 1 .................................................................45
Table 6.2: Interviewees of Project 2 .................................................................47
Table 6.3: Timing results for the experiment ...................................................49
Table 6.4: Test Automation ROI .........................................................................50
Table 6.5: Tester’s Productivity ..........................................................................51
List of Listings

Listing 5.1: Feature written in Gherkin.................................................................29
Listing 5.2: A BDD test suite in Mocha ...............................................................30
Listing 5.3: A BDD test suite in Jasmine.............................................................30
Listing 5.4: The customer order feature..............................................................35
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AQUA</td>
<td>Agile Quick User-friendly Adaptable</td>
</tr>
<tr>
<td>BDD</td>
<td>Behavior Driven Development</td>
</tr>
<tr>
<td>BFT</td>
<td>Business Flow Test</td>
</tr>
<tr>
<td>BSA</td>
<td>Business Systems Analyst</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>CoS</td>
<td>Corporate Services</td>
</tr>
<tr>
<td>CUIT</td>
<td>Coded UI Test</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain Specific Language</td>
</tr>
<tr>
<td>e2e</td>
<td>end-to-end</td>
</tr>
<tr>
<td>HP ALM</td>
<td>HP Application Lifecycle Management</td>
</tr>
<tr>
<td>HR</td>
<td>Human Resources</td>
</tr>
<tr>
<td>LCS</td>
<td>Life Cycle System</td>
</tr>
<tr>
<td>PD</td>
<td>Product Director</td>
</tr>
<tr>
<td>PSM</td>
<td>Product Solution Manager</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>Research and Development</td>
</tr>
<tr>
<td>ROI</td>
<td>Return of Investment</td>
</tr>
<tr>
<td>S&amp;A</td>
<td>Service and Assets</td>
</tr>
<tr>
<td>SE</td>
<td>Software Engineer</td>
</tr>
<tr>
<td>SM</td>
<td>Support Manager</td>
</tr>
<tr>
<td>TDD</td>
<td>Test Driven Development</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Effective and efficient software development has always been an important goal in software engineering. Several software development processes such as waterfall model, software prototyping, iterative and incremental development, spiral development, and rapid application development have evolved over the years. Agile methodology [1] is a group of software development processes with a focus on collaboration between self-directed cross-functional teams.

One of the latest software development processes that stemmed from Agile methodology is Behavior Driven Development (BDD) [2] which provides tools and methodology for collaboration between business and technical roles in a software project. BDD advocates a development process that is test-driven with a strong focus on behavior specification using the domain language of the project.

1.1 Motivation

It is generally believed that effective and rigorous testing is an essential contributor to software quality and reliability [3]–[7] and software quality is one of the major characteristics of an effective development process.

Dan North, the creator of BDD, claims that it “has evolved out of established Agile practices and is designed to make them more accessible and effective…” [2]. It is desirable to do a study on whether BDD can live up to its reputation of being effective, especially in the context of large-scale applications in a large multi-site organization. In other words, one can try to discover and evaluate an effective development process based on BDD.

---

1 Large-scale systems are complex systems comprising of large amounts of hardware, lines of code, numbers of users, and volumes of data.
2 One definition for a large organization is having more than 500 employees. Unites States Patent and Trademark Office defines companies with less than 500 employees as small business: http://www.uspto.gov/web/offices/com/sol/og/2015/week52/TOCCN/item-82.htm
3 A multi-site organization has several offices in different geographical locations.
1.2 Problem Description

IFS AB (Industrial and Financial Systems), the company where the thesis work was performed, used an automated testing tool for testing the workflows in its main product. There were some areas of improvement with the existing testing process:

1. Testing took a long time because there were too few automated tests and too many manual tests.
2. Making automated tests took a long time.
3. It was a complicated task to create and maintain automated tests and only developers could do it.
4. Results from the automated tests were sometimes unstable. Unstable refers to results that cannot be reproduced when running the test manually.
5. It was difficult to debug a failure and find the root of the problem.
6. It was not easy to map the tests to the business goal of the application.

As a result, the company needed an effective process that could address the issues above.

1.3 Research Question

The thesis tried to answer the following question:

“What can be an effective and efficient development process with BDD to improve the testing quality of a large-scale application in a large multi-site organization?”

The question and hence the thesis work needed to deal with two challenges. Firstly, it needed to discover an effective and efficient BDD process. For this purpose, two BDD processes were compared to find the one that was more effective and efficient. Secondly, the process that was deemed more effective and efficient was compared to existing development process to investigate whether it improved the testing quality.

Effectiveness, Efficiency, and Testing Quality are defined in a conceptual framework later on in 4.2 and a number of variables are presented based on these definitions. Qualitative and quantitative data for the aforementioned variables were collected from different sources such as experiment, interview and survey. The data was analyzed to try to answer the thesis question.
1.4 Delimitations
The initial idea for the thesis was to investigate whether the usage of BDD results in software quality improvements. Software quality is a multifaceted notion that can be measured using several different metrics such as defect density and customer satisfaction. However, most of these measurements can only be done in retrospect from the statistical data that is accumulated in a period of time. Large-scale application projects in large organizations take years to complete. Due to time constraints, it was not possible to determine if BDD improves software quality.

Therefore, this thesis mostly focused on (automated) testing and potential improvements to that process. Nevertheless, it is commonly believed that improvement in testing process results in better software quality as mentioned earlier in section 1.1.

The scope of this research is limited to an individual workflow within a certain ongoing project, a series of interviews, and a survey. The teams selected for the interviews were the team where the experiment was performed and another team which was selected at random.

1.5 Thesis Outline
A background to the organization, its main product, and current development and testing processes is provided in Chapter 2. An overview of BDD is presented in Chapter 3. To lay the foundation for the study, Chapter 4 explores the latest related research and describes the theory of the thesis. Chapter 5 explains how the thesis work was performed. Results obtained from the thesis work are summarized in Chapter 6. Finally, Chapter 7 presents the conclusions and proposes directions for possible future research.
Chapter 2

Background

This chapter provides some information about IFS and its main product. Moreover, the existing processes of development and testing at IFS are explained through analytical study of the current methods and processes.

2.1 IFS

IFS is a global enterprise software vendor with 2700 employees at offices distributed all over the world. The company has more than 2400 customers worldwide. The number of end-users is estimated to be more than 1 million. As summarized in Figure 2.1, IFS can be categorized as a large multi-site organization based on this information.

IFS consists of three parts or organizations: Research and Development (R&D), Consulting, and Corporate Services (CoS). IFS and its three sub-organizations are displayed as rectangles with thick outline in Figure 2.2.

The main product of IFS is called IFS Applications (latest release V9.0) which is developed in the R&D unit as displayed with a solid arrow. The product is sold to customers either by IFS consulting or through other partner consulting companies as demonstrated with dashed arrows. The bugs and issues with the product are reported by the customer to the support in the
consulting team which can be imagined as a two-level team: *Support one* which is local (can speak the customer’s language) and *support two* which triages the bugs and issues and determines the nature of the problem. The issues might be anything from bugs in the product, which is then reported to R&D Support, to customer’s difficulty to employ the product to their needs.

![Diagram of IFS Applications](image)

**Figure 2.2: The structure of IFS and its business model**

### 2.2 IFS Applications

IFS Applications [9] is a single, integrated, and component-based extended enterprise application suite that enables global manufacturing, project-based, and asset-intensive industries to successfully handle their core processes. IFS Applications manages more than 8000 forms in different categories including financials, human resources, quality management, document management, customer relationship management (CRM), business intelligence, sustainability management, and other core functionalities to facilitate full life cycle management of products, assets, customers, and projects. Figure 2.3 displays an overview of the product. It can be deployed on Windows, Unix and Linux platforms, as well as private and public clouds such as Windows Azure. Certain functionalities of the application are also available as mobile apps on iOS, Android, and Windows. This information can establish the fact that IFS Applications is a *large-scale application*.

### 2.3 R&D

R&D organization is where the product development happens. It is comprised of several self-managing product groups such as Human Resources (HR), Service and Assets (S&A), and
Financials. These groups are in charge of one or more products or applications. These applications which often share the same name with their product group will be referred to as *components*. Moreover, there are supporting units with separate responsibilities. For instance, the task of overseeing software development and testing process for the organization is done in the *Methods and Processes* unit.

There are three managing roles in each product group: Product Director (PD), Support Manager (SM), and Product Solution Manager (PSM). The work is later done in smaller product teams that are one of the two types: *Support team* which usually does the maintenance and bug fixing, and *Product team* which are in charge of feature development. There are two roles in each team: Software Engineer (SE) and Business Systems Analyst (BSA). Usually each team consists of one (or more) BSA(s) and several SEs. The teams are often distributed geographically across multiple offices. Figure 2.4 summarizes the structure of R&D.

### 2.4 Process Models and Workflows

User scenarios for different components of IFS Applications are specified in form of business process models and documentation.
The business process models are already documented in three different levels for most of the business processes as displayed in Figure 2.5: level 1 or *business solution* is the highest level. Each business solution is usually comprised of several level 2 or *business process* items. Business processes consist of several level 3 or *application as activity diagram* items. These level 3 items are referred to as *workflow* throughout this report.

**2.5 Development Process**

The current development process at IFS is AQUA which stands for *Agile Quick User-friendly Adaptable* and is a Scrum-based development process with focus on shortest delivery
timeframe and small and self-managing teams. Figure 2.6 displays the different phases of AQUA. Development is done in iterations and some testing is done continuously in each iteration. However, the final acceptance test is run after the final iteration.

![Figure 2.6: Different phases of the AQUA process](image.png)

### 2.6 Manual Testing

To test the software, there are tools that translate process models to test cases and docs. Figure 2.7 provides an overview of this process. Using Hewlett Packard Application Lifecycle Management (HP ALM)’s [10] manual testing tool (HP Sprinter[11]), the testers can run manual tests and report bugs in HP ALM.

![Figure 2.7: The manual testing process at IFS](image.png)

### 2.7 Automated Testing

To get an idea about the automated testing situation, the current setup in terms of test types and test execution time is looked at first. Then, the desired setup imagined by the R&D team is explained. Failures are reported to the list of committers on certain intervals. Someone on that list will then need to isolate and fix the culprit commit.
2.7.1 Current Setup
There are four main categories or types of tests: general application test, business flow test, unit test, and static code test. These test types will be described in more detail in this section.

2.7.1.1 General Application Tests
To test the general functionality of the application, an in-house tool called *Application Tester* is used. The tool is written and integrated into the application. The tests run on one dedicated test machine. To give an idea of the number of tests and their runtime, a sample test run of 74,271 tests takes 161 minutes (roughly 0.13s per test).

2.7.1.2 Business Flow Tests
To test the workflows in the application, the BSAs write manual tests known as Business Flow Tests (BFT). Some of these tests are then automated by SEs using Microsoft Coded UI Test (CUIT) tools. CUIT can record the user actions and convert them into test code which can be played back. It is important to note that, throughout this report, BFT is used to refer to this automated version. You can see the process of automating BFT tests in Figure 2.8. As highlighted in the picture, the link between the input (BFTs written by BSAs) and the output (automated tests generated using CUIT) is quite fragile because every change in the process model or the manual tests will require changing the automated tests. Similar to general application tests, the tests run on one dedicated test machine and the entire test suite takes approximately three hours. As for the average runtime of the BFTs, a sample test run of 183 BFTs takes 84 minutes (roughly 27.5s per test).

![Figure 2.8: The process of automating BFTs using CUIT tools](image_url)
2.7.1.3 Unit Tests
Unit tests are occasionally written by developers to test small pieces of code without starting the application.

2.7.1.4 Static Code Tests (Code Analysis Tests)
This type of tests is used before compiling the code to find possible issues and vulnerabilities with the static source code.

2.7.1.5 Other Test Types
There are also types of test like Installation tests that do a fresh install test and Documentation tests that check for HTML sanity of the documentation files.

2.7.2 Current Test Execution Time
Running entire test suite takes approximately three hours for each of BFT and application tester test types. The two test suites run in parallel on two machines every night. There is also a test run of some sanity tests that run every two hours during workdays and takes roughly 30 minutes to complete. Failures are reported to the list of committers at certain intervals. Someone on that list will then need to isolate and fix the culprit commit.

2.7.3 Desired Setup
Figure 2.9 portrays the desired test automation pyramid [12] for the R&D team. The area for each test type represents the number of tests. In other words, it is desirable to have a very good coverage for static code analysis and unit tests, but only a small number of UI or end-to-end (e2e) acceptance tests are needed. This is because moving up in the pyramid, the tests become more expensive both in terms of test execution time and maintenance as well as less reliable in terms of results.

![Figure 2.9: The desired test automation pyramid of R&D](image)
2.7.4 Desired Test Execution Time
There is a three-hour limit for the total test execution time or else the results will not be available the next morning when the development teams start working with the latest build from the day before. Ideally, it is desirable to improve the current situation and have a faster overall test run performance.

As for the sanity tests, it is desired to improve the current execution time from 30 minutes to under 10 minutes. In the ideal case, one can imagine a setup where the sanity tests can be integrated to the development process as post-commit hooks or Continuous Integration (CI) hooks for merge requests and hence provide a quick and precise feedback; that means there will be no need to go through a list of commits to isolate a bug because the tests are run after each single commit or as part of every single merge request.

2.8 Technical Architecture
To get a better picture of the product, an introduction to the technical architecture can be useful. A high-level description of the architecture for IFS Applications and IFS web client is provided in the following sections.

2.8.1 IFS Applications
The data lives in an Oracle database. Higher up in the design there are several middle tier servers that talk to the DB using PL/SQL and to the clients using IFS’s own protocol on HTTP(S). Figure 2.10 demonstrates the technical architecture of IFS Applications.
Figure 2.10: Technical architecture of IFS Applications

2.8.2 IFS Web Client
The experiment carried out in this thesis was done in a team that was in charge of creating the web client for IFS Applications. Figure 2.11 demonstrates the architecture/technology for the web client.

Figure 2.11: Technical architecture of IFS web client
Chapter 3

Behavior Driven Development

Behavior Driven Development (BDD) which is an increasingly popular\(^4\) Agile process was originally developed by Dan North [2] to address the issues with Test Driven Development (TDD). In the following section, a brief introduction to TDD is provided to help understand the test-first cycle which is also used in BDD. Subsequently, a description of BDD and relevant practices is presented. This chapter ends by outlining the BDD process which was used and followed in the experiment conducted in this study.

3.1 Test Driven Development

TDD [13] is a test-first approach, meaning that the development process starts by writing an (initially failing) automated test case and making it pass by writing the minimum amount of code and re-factoring later if needed. Figure 3.1 shows the TDD’s test-first cycle.

Figure 3.1: The TDD process

\(^4\) Google trends shows breakout (growth of more than 5000%) for BDD-related terms such as:
BDD test: [https://www.google.com/trends/explore#cmpt=q&q=%22bdd+test%22](https://www.google.com/trends/explore#cmpt=q&q=%22bdd+test%22)
Cucumber (software): [https://www.google.com/trends/explore#cmpt=q&q=%2Fm%2F0c4z18h](https://www.google.com/trends/explore#cmpt=q&q=%2Fm%2F0c4z18h)
There are, however, certain problems with TDD. For instance, general confusion among developers trying to use it, “programmers wanted to know where to start, what to test and what not to test, how much to test in one go, what to call their tests, and how to understand why a test fails” [2]. The other issue is that TDD is focused on testing the state rather than behavior of the system [14].

3.2 Behavior Driven Development

BDD was originally designed to extend TDD by using semi-formal user scenarios that are close to natural language to describe the behavior of the target system. The premise of BDD is to create a common and effective process of communication between the business interest upheld by Business Systems Analysts (BSAs) and the technical insight provided by Software Engineers (SEs). Borrowing from Agile software development’s desired behavior that has a business value, the focus is on behavioral specification of software using the domain language of the situation.

BDD is quite similar to other practices such as Acceptance Test Driven Development, Specification by Example [15], Example Driven Development, and Story Driven Development in that they all try to help the team members to understand the customer needs before the development by conversing in the domain language of the customer.

The desired behavior is specified in semi-formal behavioral specifications of user stories which are called features. Each feature consists of multiple scenarios. For this purpose, collaboration is needed between business analysts and software engineers. There are no formal requirement for this process, but it is important to make sure that the acceptance criteria, also known as scenarios, are declarative rather than imperative. In other words, one needs to state what needs to happen instead of how to do something, so the focus is on the business language with no mention of specific technical aspects such as UI elements.

3.3 Defining the BDD Process

The BDD process for a single feature is displayed in Figure 3.2. The items in the rectangles are artifacts and the numbered lines are activities. According to the BDD process, these activities are iterated for different features until the business goal is met. The feature which is driven from a business goal is described as multiple scenarios. Scenarios are concrete examples showing how a feature should work. Scenarios are then converted to executable specification using a BDD tool. These executable specification are the embodiment of features in code and can be executed automatically. The result of the execution is initially fail and software
engineers need to implement the code to make all the scenarios pass. Finally, the test result reports serve as living documentation which can be consulted by the members of the team at any time. The rest of this section describes each activity marked with numbers in Figure 3.2.

![Figure 3.2: The BDD process for a feature](image)

### 3.3.1 Requirements
The first two activities (1 and 2) are collaborative tasks for all roles in the project to derive the requirements from the business goal and define them in form of features and scenarios. The artifacts for these activities are feature files that consist of scenarios.

### 3.3.2 Design and Implementation
In this set of activities the behavior specified in the previous activities is implemented. Developers automate the acceptance tests by making them *executable* (activity 3), implementing all scenarios and iterating over them with a TDD approach until all the scenarios in a story pass (activity 4). The TDD approach requires designing and implementing the application code iteratively. The artifacts for these activities are executable specification and application code.

### 3.3.3 Testing
This activity involves running the executable specification or the automated acceptance tests by developers or testers. Developers can use the test results to report that a feature is implemented and the test suite can also be used for regression-testing when refactoring the code as a part of TDD. The test results will serve as living documentation (activity 5) that is tightly coupled with the behavior specification.
Chapter 4

Theory

This chapter begins with a summary of latest research on Behavior Driven Development (BDD) and other relevant Agile processes such as Test Driven Development (TDD). A conceptual framework, built on the literature study and the thesis question, is then introduced as a foundation for the methodology of this study. The conceptual framework defines the variables used throughout this report and illustrates how they relate to each other.

4.1 Related Work

Even though BDD is a relatively new methodology, there is a fair amount of literature on this subject in form of published papers, books, blog posts, and other online material. Rahman et al. suggested that running BDD acceptance tests in parallel can cut down on the execution time which results in faster test feedback [16]. In another paper [17], the authors introduced a reusable architecture for acceptance tests in BDD in order to cater for certain needs such as reusability of step implementation for BDD scenarios, separation of concern among different roles in a project, and ease of auditability while dealing with challenges such as maintainability, system integration complexity, and emulating production-like execution environment. Lai et al. combined BDD with iterative and incremental development and proposed a quality measurement model for security functional requirement items [18].

One might think that by following BDD where code is written to make the test cases pass, a coverage of 100% is achieved. However, in an empirical study [19], Diepenbeck et al. showed that, contrary to the common belief, the code coverage decreases over time. Subsequently, they proposed an algorithm to generate BDD scenarios based on uncovered code.

Drechsler et al. presented the concept of Completeness-Driven Development (CDD), which uses BDD for behavioral abstraction level, as an essential methodology for correctness and efficient development [20].

Morrison et al. evaluated the feasibility of using BDD to verify compliance of electronic health record systems with governing regulations [21], [22].

Agile software development requires frequent changes which implies maintaining the tests to reflect the changes. To deal with these changes, Sathawornwichit & Hosono presented an
approach to maintain consistency among design models, system under test, and test components using metadata and BDD-style acceptance tests [23].

More research has been done on TDD probably due to the fact that is relevantly older than BDD. Because TDD is a predecessor to BDD, it is interesting to summarize some of the TDD research relevant to the subject of this thesis. In an experiment at IBM, Williams et al. found that the code developed using TDD showed 40% fewer defects compared to baseline [24]. Some of the studies focused more on the effectiveness of TDD: One study saw an increase of productivity for the students who wrote more tests [25]. Another research by Gupta and Jalote defined and used metrics such as development efforts, and developer’s productivity to evaluate effectiveness and efficiency of TDD and observed an improvement in those metrics [26]. Janzen & Saiedian did several researches on the subject of TDD and could show that it decreases code size and complexity [7] and increases code-related features such as object decomposition, test coverage, and external quality, and aspects including productivity and confidence which are more related to developers [3].

There seemed to be a shortage of related work and research on the subject of effectiveness and efficiency of BDD and its effect on the overall software or testing quality. Therefore, it was interesting to conduct a study to focus on these aspects.

4.2 Conceptual Framework

The hypothesis in this thesis is that an effective and efficient development process with BDD improves the testing quality. In order to examine the hypothesis there needed to be a conceptual framework where all relevant variables and the possible relationship between them were described.

This study examined the Development Process that served as the independent variable [27] and Effectiveness, Testing Efficiency, and Testing Quality as three dependent variables as displayed in Figure 4.1.

As explained in the next chapter, some of the variables described the data acquired through the means of interviews and a survey. Others were presenting quantitative data from an experiment. The rest of this section defines the aforementioned variables.
4.2.1 Development Process
The development process was the independent variable to which different treatments were applied. These treatments included BDD on two different layers and the company’s Agile-based process.

4.2.2 Effectiveness
Effectiveness for a process is the degree to which it is successful in producing a desired result. The desired result of a development project is to deliver business value. Therefore, an effective development process guarantees that both the project team and the delivered product do the right thing. In other words, every activity performed as a part of a project serves the purpose of achieving a predefined business goal. Based on this, Business Goal Alignment (BGA), defined as the degree to which different activities in the development process can be mapped to the business goal, was used as the variable representing effectiveness.

Effectiveness can also be measured by the end-product quality measurements such as defect density and functional tests pass rate of the product. However, measuring the software quality was, as discussed in section 1.4, outside the scope of this thesis.

4.2.3 Testing Efficiency
An efficient system or machine achieves most productivity using least wasted effort or expense. As discussed earlier, testing is an integral part of an Agile development process and therefore a more efficient testing process results in a more efficient development process. In a study on
the effectiveness and efficiency of Test Driven Development (TDD), Gupta and Jalote used the overall Development Efforts and Developer’s Productivity as metrics to measure efficiency of TDD [26]. In a similar approach, overall Testing Efforts (TE) and Tester’s Productivity (TP) were chosen as a variables representing the testing efficiency of the development process. Given the following definitions:

- **Testing Efforts (TE):** The overall testing efforts from specification to the final automated test which was the total time spent on authoring the specification or feature, and implementing the automated test in person-hours.
- **Non-Commented Lines of Code (NCLOC):** Total lines of code written for specification or feature, and the automated test ignoring comments and empty lines.

Tester’s Productivity was defined as:

\[
TP = \frac{NCLOC}{TE}
\]  

(4.1)

To define other measures for testing efficiency, the process through which an incoming error or failure in the test results was dealt with needs to be explained. The process started by looking at the test results and determining the source of the failure. The time needed to investigate the source of an error or failure was defined as Investigation Time \( (t_i) \). The source of the error can often be recognized as one of the followings:

- **Bug in the Application:** The failure was due to a bug in the application which should be analyzed and fixed by a developer.

- **Test Environment Problem:** The failure was due to problems with the test environment such as database setup and issues with the test machine which should be fixed by the systems support personnel.

- **Testing Issue:** The test code might have needed to be modified because the application had changed or the test was not stable enough. Alternatively, it could be the testing framework that needed to be modified to accommodate for changes or to remedy a (transient) problem. Test Maintenance Time \( (t_u) \) was defined as average total time spent for single maintenance of an automatic test.

An efficient test could be imagined as having a smaller \( (t_i + t_u) \). It was possible to quantify and measure \( t_i \) from the statistical data available for the Business Flow Tests (BFT). However, to have a similar set of data for BDD tests, there needed to be a considerable amount of test results
for different workflows which was outside of the scope of this study. One possible way to work around this limitation was to consider the fact that test investigation time was directly related to Test Failure Traceability (TFT), which was defined as the end-user’s opinion about on the ease of discovering the reason for a failing test. Therefore, an increase of test failure traceability would result in shorter investigation time.

In a nutshell, an efficient testing process was defined as having low $TE$ and $t_a$ while having high $TP$ and $TFT$.

4.2.4 Testing Quality

4.2.4.1 Test Automation ROI

There are several cost models for test automation. From the simplistic cost model described by Hoffman [28], to opportunity cost model proposed by Ramler & Wolfmaier [29]. For the purpose of this study, the fixed automation costs such as hardware, environment setup, and software licenses were ignored and only time was selected from the list of variable automation costs to calculate the ROI for the first year. Time is expressed as following dependent variables:

- Test Creation Time ($t_c$): Average total time spent to create an automatic test. This included time need for reading the specification and creating the test using the existing tool(s). For the BDD treatment, this includes the time needed to create the feature file as well as the automated test.
- Test Maintenance Time ($t_m$): This is already defined in the previous section.
- Test Execution Time ($t_m$ or $t_a$): Average total time needed to run a single test written for a workflow; $t_m$ was the average total execution time for a manual test and $t_a$ was the average total execution time for an automated test.

Given $n_a$ as the average maintenance occurrences in a year and $n_a$ as the average automated execution occurrences in a year, $Cost$ and $Gain$ were calculated as:

\[
Cost = t_c + (t_a \times n_a) \tag{4.2}
\]

\[
Gain = (t_m - t_a) \times n_a \tag{4.3}
\]

Finally, a definition of Test Automation ROI was proposed as:

\[
ROI = \frac{Gain - Cost}{Cost} \tag{4.4}
\]
4.2.4.2 Test Feedback

Test feedback was expressed as a combination of following dependent variables:

- *Test Execution Time (tₐ)*: This is already defined in the previous section.
- Test Result *Business Goal Alignment (BGA)*: A dependent variable which described the end-user’s opinion about the test results’ alignment with the business goal. The data was the average value obtained from a survey.
- *Test Failure Traceability (TFT)*: This is already defined in section 4.2.3.

4.2.4.3 Test Usability

The definition for usability was based on the interpretation of usability described by Abran et al. [30]. For practical reasons it was decided to ignore certain measures such as security. The measures considered were:

- *Test Readability (TR)*: A dependent variable, based on the code readability scoring proposed by Buse & Weimer [31], which was the end-user’s opinion about the readability of the test on the scale of 1 to 5. Readability affects the ease of learning and modification. This data was the average value obtained from a survey.
- *User Satisfaction (US)*: A dependent variable which described the end-user’s satisfaction with tests. The data was the average value obtained from a survey.
Chapter 5

Methodology

This chapter explains the research method and presents a discussion on its validity, reliability and research ethics. The study used methodological triangulation by using more than one method to gather data to evaluate the usage of a development process. Quantitative data was obtained by means of an experiment where it was possible to conduct an experiment in a controlled environment and a survey of the project members when it was not feasible to conduct an experiment. Moreover, interviews were conducted to provide some qualitative data for the research. In addition to the aforementioned methods, company’s documentation, and personal contact were used to gather some of the required data.

5.1 Discussion on Chosen Methods

A company seeking to evaluate a new method or tool before introducing it to improve a process or way of working is a typical use case for empirical studies. The three empirical strategies that are widely used are surveys, experiments, and case studies. The choice of a suitable strategy for a research depends on the characteristics and limitations of the required study.

Surveys are often used at the end of a study to investigate something in retrospect like when a tool has been used for a while. They can also be used before the research to get a snapshot of the current situation [27]. For the current research, an interview was performed in the beginning to get a good understanding of current development and testing processes and a survey was conducted at the end of the thesis work to get some feedback and data related to the performed research.

Experiments are used when the researcher needs to apply more than one treatment to objects to compare the output. They need to be performed in a controlled environment [27]. For the current project, the requirement was to evaluate the effect of Behavior Driven Development (BDD) without causing much risk to the ongoing project which was on a tight schedule. It was desirable to perform the study in a laboratory setting to have full control over the situation. Furthermore, the objective was to compare the results of the application of the BDD process and the existing Agile-based development process. Based on this discussion, conducting an experiment was a good candidate for the purpose of this study.
The third empirical strategy is case study. Case studies are suitable for industrial evaluation of software engineering methods [27]. A case study would have been a perfect strategy for the current thesis work. However, the scope of the thesis work would imply a small or simplified case study due to time limitations and the fact that the work was carried out in an unreleased product and that it was important to keep the research work separate from the ongoing project to avoid unnecessary risks to product delivery schedule. Such case studies do not scale well according to Wohlin et al. [27].

5.2 Interviews with Stakeholders

The thesis work started with interviews with the different roles or stakeholders to get a picture of current development and testing processes at IFS. Two teams were selected for the interviews: the team in which the thesis project was carried out and one of the product teams. In each team, three interviewees with different roles were contacted via email to book a meeting for the interview at their earliest convenience and participation was voluntary. Fortunately, everyone who was contacted agreed to participate.

The structured interviews were designed with six open-ended questions. Current development and testing processes were the subject of the first four questions in an attempt to obtain some qualitative data on opinions about current situation of these processes and possible improvements to the current situation. A short presentation of BDD was then provided by the interviewer before asking the interviewees about their opinion about BDD and how it might be useful for improving software quality. The interviews ended with asking the participants for any additional comments as a final question. A full list of the questions used in the interviews can be found in Appendix A. The following aspects were taken into consideration when designing the interviews:

1. **Roles**: In each team, three different roles were selected for the interviews.
2. **Geographically Distributed Teams**: At least one of the interviewees worked at a different office.
3. **Large and Complex Application**: The interviews were conducted as open-ended interviews to allow for a more in-depth discussion.

The interviews were recorded and transcribed. The full version of transcripts is available in Appendix B. A discussion on the results of the interview is presented in the next chapter.
5.3 Experiment
As the next part of the study, an experiment with focus on a possible usage of BDD was conducted. The experiment was based on the conceptual framework introduced in section 4.2. Quantitative data from the experiment were used to measure the effectiveness and efficiency of the two BDD approaches, find the better approach based on the results and compare it to the existing process.

In section 3.3, a BDD process was introduced to create executable specifications and eventually automated tests from high-level scenarios which could be understood by all members of a team. The experiment started off with an existing workflow which is the subject of the next section.

5.3.1 The Customer Order workflow
The experiment conducted during the course of this thesis work was to automate an example workflow which was the object of the experiment. The workflow chosen for this purpose was the Customer Order workflow that described the process of ordering products by a customer. This was a simplified version of the Enter Customer Order workflow that was a part of the Manage Customer Order process of the Sales component. Figure 5.1 shows the activity diagram for the Customer Order workflow.

![Figure 5.1: The Customer Order workflow](image)

5.3.2 Experiment Design
In this experiment the independent variable was the development process which is described in section 4.2.1. The object of the experiment was the Customer Order workflow which was introduced in the previous section. Two different treatments were considered for the independent variable:

1. **Existing Development Process**: Following the existing Agile-based development process, the existing BFT tool was used to convert the workflow to an automated test.
2. **BDD Process**: Following the BDD process described in section 3.3, a BDD tool was used to convert the workflow to an automated test. Moreover, as described in the
research question, it was desirable to find an effective and efficient BDD process. Therefore, more than one alternative for BDD treatment were needed. Since BDD is heavily influenced by testing, a good approach to find BDD alternatives was to focus on testing and consider tests that run against different layers of the application. Looking at the three layers in Figure 2.10, the following three BDD treatment alternatives were possible:

a. BDD with tests on the client layer (e2e testing)
b. BDD with tests on the web service layer (REST API testing)
c. BDD with tests on the database layer (PL/SQL testing)

The first two alternatives were chosen because they could be done using the same technology (JavaScript and Node.js) for writing the tests and after some discussion with the thesis supervisor at the company about which alternatives would be more valuable for the company in the future.

For each treatment, the required process was followed to create the automated tests. All dependent variables needed to calculate ROI other than Test Execution Time for automated tests ($t_a$) were either measured by the subjects of the experiment which were the people who applied the treatments or were obtained from others sources such as company’s documentation. To calculate $t_a$, the automated tests were run a number of times and an average of the measured execution times was used as $t_a$. The results of the experiment are presented in the next chapter.

### 5.3.3 BDD Tools Overview

Before the experiment was started, it was necessary to examine the current BDD tools. The choice of tools was based on the technology used in the web client for those layers that were chosen in the experiment design. Since the client used the AngularJS stack, JavaScript on Node.js was a suitable candidate for client layer tests. For the web service layer tests, the choice of the languages and technology was less important as long as the test could speak with the web service using the OData protocol; however, for the sake of simplicity and uniformity it was decided to use JavaScript here as well. The other reason for this decision was to eliminate the possibility that a different technology would affect the experiment results.

In order to make an informed decision on the choice of the BDD tool for the experiment, an evaluation of the popular and relevant tools was needed. The focus here was on JavaScript tools because the tests needed to be written in JavaScript. The BDD tools that were examined are explained in this section.
5.3.3.1 Cucumber

Cucumber [32] is a popular testing tool (JavaScript implementation for Node.js and modern browsers is called Cucumber.js) that runs automated tests written in a BDD style. Behavior is described in plain text as features written with Gherkin [33] syntax. Gherkin is a Domain Specific Language (DSL) that can be used to describe the behavior of the software. It uses the Given-When-Then format as you can see in Listing 5.1.

The Gherkin parser in Cucumber converts the features to Steps Definitions in the target language that later needs to be implemented to turn the phrases into concrete actions. To set up the environment in which the steps will be run, there are certain support files like the World constructor and hooks.

---

**Feature:** Filtering the components of IFS Applications
As a user of IFS Applications
I want to be able to search for components by name
In order to find a certain component

**Scenario Outline:** Filter by name
Given I am on the home page of IFS Applications
   And I haven't filtered any components by name
When I search for a component by '<searched_name>'
   Then I should see a list of <hits> components that match that name
   And '<found_name>' should be the top component

**Examples:**

<table>
<thead>
<tr>
<th>searched_name</th>
<th>hits</th>
<th>found_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>cust</td>
<td>2</td>
<td>Customer Order</td>
</tr>
<tr>
<td>lobb</td>
<td>1</td>
<td>Lobbies</td>
</tr>
</tbody>
</table>

Listing 5.1: Feature written in Gherkin

5.3.3.2 Mocha

Mocha [34] is probably the most popular JavaScript test framework running on Node.js and browsers which supports asynchronous testing, test coverage reports, and results in various formats. To write effective tests, one needs to use assertions libraries such as Chai [35] and libraries for creating test doubles (spies, stubs and mocks) such as Sinon [36]. Mocha supports writing tests using different Interfaces (or DSLs) such as BDD and TDD. Listing 5.2 shows an example BDD test suite with the popular describe-context-it syntax of RSpec.
describe('Array', function() {
  before(function() {
    // ...
  });

  describe('#indexOf()', function() {
    context('when not present', function() {
      it('should not throw an error', function() {
        [1,2,3].indexOf(4).should.not.throw();
      });
      it('should return -1', function() {
        [1,2,3].indexOf(4).should.equal(-1);
      });
    });
    context('when present', function() {
      it('should return the index where the element first appears in the array', function() {
        [1,2,3].indexOf(3).should.equal(2);
      });
    });
  });

Listing 5.2: A BDD test suite in Mocha

5.3.3.3 Jasmine

Jasmine [37] is a BDD framework for testing JavaScript code. It provides assertions and test doubles and does not depend on any other JavaScript frameworks for this purpose. It is the default testing framework used and supported by the AngularJS project and is quite similar to Mocha in terms of Syntax as displayed in Listing 5.3.

describe('A suite', function() {
  describe('contains spec with an expectation', function() {
    expect(true).toBe(true);
  });
});

Listing 5.3: A BDD test suite in Jasmine

5.3.4 Other Testing Tools

A number of tools that were not BDD-specific but were needed to write certain types of tests or are mentioned later in this report, are listed in this section.
5.3.4.1 Selenium WebDriver
Selenium WebDriver [38], [39] is a tool to introspect and control user agents (browsers on desktop and mobile devices). This is done by using the platform- and language-neutral Wire protocol that can remotely drive the browsers.

5.3.4.2 Protractor
Protractor [40] is an end-to-end test framework for AngularJS applications. Protractor runs tests against your application running in a real browser, interacting with it as a user would. It has support for different BDD frameworks such as Jasmine, Mocha, and Cucumber.

5.3.4.3 Karma
Karma [41] is essentially a tool that spawns a web server that executes source code against test code for each of the browsers connected. The results for each test against each browser are examined and displayed via the command line to the developer such that they can see which browsers and tests passed or failed.

5.3.5 Testing Possibilities for the Web Client
After the tool research phase resulting in a list of available tools provided in the two previous sections, it was needed to find the testing possibilities for the web client to determine which of the possibilities are suitable for a BDD treatment. To create automated tests, revisiting the client architecture overview displayed in Figure 2.11, the following testing possibilities could be imagined for the client layer:

1. end-to-end (e2e) testing with Protractor
2. Unit testing with Karma and Jasmine

For the middle-tier or web service layer the possibilities were:

3. REST API testing
4. Unit testing with Junit

The possibilities above are summarized in Figure 5.2. Options 1 and 3 were suitable for BDD and options 2 and 4 were suitable for low level unit testing with a possible Test Driven Development (TDD) approach.

As described in section 5.3.1, the experiment focused on a sample workflow and automated that using the current Business Flow Test (BFT) tool (See 2.7.1.2) and also with BDD processes with tests mentioned in option 1 and 3 from the list above. The final results were then analyzed to answer the main thesis question.
5.3.6 Environment for the Experiment

In this section, the control environment used for the purpose of this experiment is explained in detail. To achieve acceptable levels of control on the context, a laboratory-like environment was needed. Therefore, the experiment was performed on a dedicated test machine. The test machine was a 64-bit Microsoft Windows Server 2008 virtual machine with 4 GB of RAM and an Intel Xeon E5-2660 v3 @ 2.60 GHz processor. The machine was accessible using Remote Desktop Connection. The environment created on this machine, which is referred to as sandbox throughout this report, had the following local installations:

- **Oracle database**: version 12.1.0.1 64-bit
- **Oracle WebLogic Server**: version 12.1.3
- **IFS Applications**: client version 9.0.9.0 server version 6.90.3.0
- **Oracle Java**: version 8 update 66
- **Apache Maven**: version 3.3.3
- **Node.js**: version 5.0.0
- **Git**: version 2.6.3.windows.1
- **Jenkins CI**: version 1.639
- **Google Chrome**: version 47.0.25.26.106 m
- **Other dependencies needed for the BDD test runners**: versions are listed in package.json files that can be found in Appendix C and Appendix D.
5.3.7 Existing Development Process

The existing Agile-based development process, which was the experiment baseline, was the first treatment to be applied to the object of the experiment: The test created by BFT tools ran against the IFS Applications windows client. The process consisted of the following activities:

1. The process started with the thesis supervisor creating a manual test case for the *Customer Order* workflow in form of a word document which was the closest thing to a specification. The specification is displayed in Figure 5.3. The thesis supervisor was asked to record the time spent for this activity.

2. Based on the document from the previous step, a BFT test was written for the *Customer Order* workflow. A developer which was familiar with the BFT tool was asked to create the automated test case. The developer was asked to record the time that was spent for creating the test ($t_c$). The test was then executed a number of times and the execution time ($t_a$) was collected for all the test runs.

3. As the last part of this treatment, the specification was modified to add an extra order line and the developer was asked to modify the test to conform to the new specification and report the time spent for this modification which was regarded as the maintenance time ($t_u$).

5.3.8 BDD on Client Layer

The first BDD treatment alternative was BDD with e2e tests that ran on the client layer: The workflow was expressed as a feature containing two scenarios which were written in Gherkin. This feature file was written by the author of this report with some help from the thesis supervisor and the time spent for this activity was recorded. The feature file which is displayed in Listing 5.4, was used for both BDD treatment alternatives. This pertains to activities 1 and 2 of BDD activities defined in section 3.3.1.
Document Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>By</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>151118</td>
<td>OSEYSE</td>
<td>Created first version of test case.</td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Test Case Name:** Enter Customer Order  
**Owner:** Supply Chain  
**Product/Component:** Sales / ORDER  
**Title:** Create Customer Order header with order lines.  
**Prerequisite:** That we are logged in as ALAIN in a RACE database.  
**Description:** Create Customer Order header with order lines.  
**Purpose:** To Create a Customer Order, with Order lines to validate that it's possible.

<table>
<thead>
<tr>
<th>Step number</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Navigate to Sales / Order / Customer Order</td>
<td>The form Customer Order should be loaded.</td>
</tr>
<tr>
<td>2.</td>
<td>Click on the New button</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Enter the value 1000 in the field Customer</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Click on the Save button</td>
<td>Validate that the field Status has the value Planned.</td>
</tr>
<tr>
<td>5.</td>
<td>Select the Order Lines Tab</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Click on the New button in the Order Lines tab grid.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Enter the value BP1 in the Sales Part No field</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Enter the value 1 in the Sales Qty field.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Click on the Save button</td>
<td>Validate that the new Order Line was created.</td>
</tr>
</tbody>
</table>

Figure 5.3: Specification for Enter customer order
**Feature:** Customer Order component in IFS Applications

As a user of IFS Applications
I want to be able to use the Customer Order component
In order to manage the customer orders

**Scenario:** Enter customer order

*Given* I am the user "ALAIN"
*When* I add a new customer order with customer 1000
*Then* a customer order with status "Planned" should be added

**Scenario:** Enter order lines

*Given* I have a customer order
*When* I add a new order line with sales part no "BP1" and sales qty 1
*Then* the order line should be added

---

**Listing 5.4: The customer order feature**

The feature was converted to step definitions using *Cucumber*. The steps were implemented using *Protractor* to run the tests directly on the web browser and the assertions were written with the help of *Chai* and *Chai as Promised* assertion libraries. This setup and code were implemented as a Node.js package called **client-e2e-tests** living in a git repository on the company’s GitLab server. This is activity 3 defined in section 3.3.2. and the time spent for this activity was recorded as test creation time \( t_\text{c} \).

The test runs were defined as jobs in a *Jenkins Continuous Integration Server* installed on the sandbox. Figure 5.4 displays the Jenkins jobs defined for the purpose of the experiment. The results of the test run which were in JSON format were fed to a Jenkins plugin called *Cucumber Reports* [42] to get a better visual report and feedback. Implementing the steps and setting up the continuous integration (CI) were parts of activity 4 from section 3.3.2. The time spent for setting up CI was ignored similar to the time spent to install and set up the BFT test runner.

![Jenkins setup for the experiment](image)

**Figure 5.4: Jenkin setup for the experiment**
Two jobs were defined for testing on e2e level as displayed in Figure 5.4:

1. **Deploy_web_client**: This job pulled the latest web client code from Git, built the web client and deployed it to Oracle WebLogic server.

2. **e2e_tests**: This job triggered `Deploy_web_client` first to deploy the latest web client and waited until it finished successfully. The latest e2e tests pulled from Git were then run and the results were reported back to Jenkins.

The scenarios were failing initially as displayed in Figure 5.5. Some of the steps were passing since operations like navigating to customer order component worked. Some steps were skipped because, using Cucumber, the steps that follow undefined, pending, or failed steps in a scenario are not executed and are skipped.

![Cucumber Reports](image)

*Figure 5.5: Initial test results for customer order feature*

A software engineer familiar with the client code was asked to start developing the code to implement the first scenario and ran the Jenkins job to get feedback. As seen in Figure 5.6, the first scenario was passing at this point.
The TDD process of implementing the feature continued, as defined in activity 4 described in section 3.3.2, until the feature was complete as displayed in Figure 5.7. To eliminate the potential noise and fluctuation in the test execution time ($t_a$), the test was run a number of times to obtain an average value for $t_a$.

As a last part of this treatment, the feature was modified to add an extra order line and the time spent for this modification was recorded which was regarded as the maintenance time ($t_u$).

### 5.3.9 BDD on Web Service Layer

The second BDD treatment alternative was BDD with REST API tests that ran on the web service layer: The process for REST API testing was quite similar to e2e testing because it started with the same feature file and generated the step definition file using Cucumber. However, the steps were defined using Node.js ‘request-promise’ library to test against the web service REST API directly. The same set of libraries used by e2e tests were used for assertions.
This setup and code were implemented as a Node.js package called \texttt{client-rest-tests} living in a Git repository on the company’s GitLab server.

![Cucumber Reports](image)

**Figure 5.7: Final test results for customer order feature**

Similar to e2e tests, the REST API tests were defined as a Jenkins job and the results were fed to the \textit{Cucumber Reports} plugin. The data was obtained by running the Jenkins job multiple times. Similar to e2e tests, two jobs were defined for the tests on this level as displayed in Figure 5.4:

1. **Deploy\_web\_service**: This job pulled the latest web service code from Git, built the web service and deployed it to Oracle WebLogic server. The web service was deployed using a different port from that of the web client.

2. **REST\_API\_tests**: This job triggered \texttt{Deploy\_web\_service} first to build the latest web service and waited until it finished successfully. The latest REST API tests pulled from git were then run and the results were reported back to Jenkins.
A developer familiar with the web service code was asked to follow the BDD process to develop the code in the web service needed for the feature. Once again, similar to BDD on e2e level, the developer was asked to run the Jenkins job after each code change to verify the fix and get feedback on the progress. The process continued until the feature was complete. The test was then run a number of times similar to e2e layer test to obtain an average for test execution time (t_a).

As a last part of this treatment, the feature was modified to add an extra order line and the time spent for this modification was recorded which was regarded as the maintenance time (t_u).

**5.3.10 Living Documentation**

As mentioned earlier in section 3.3.3, activity 5 results in test results that could be used a *living documentation*. An example of test results as living documentation is displayed in Figure 5.8.

![Figure 5.8: Test results as living documentation](image)

**5.3.11 Limitations**

Effort was put into keeping all the independent variables other than development process at fixed levels:

1. The experiment was performed on a development sandbox:
   a. Dedicated database which means complete control over the data.
b. The client and server were on the same machine to minimize the network latency problems.

2. Same workflow was used for all three treatments.

3. Same tooling (Cucumber/Chai) was used for both BDD treatment alternatives.

However, the BFT tool worked with the windows application and the two BDD tools worked with the web client. This might have induced some limitations on how the results will be analyzed since different technologies were used for these clients. The windows application, written in Visual C#, was compared to the web client, implemented using AngularJS web stack including Typescript/JavaScript, HTML5, and (S)CSS. Nevertheless, the limitation will be on the test execution time only. For test execution time comparing the results from the two BDD test runs on different layers is more meaningful. Other measurement seem to be meaningful regardless of the client technology.

5.4 Survey

Based on the conceptual framework introduced in section 4.2, an online survey was designed and conducted, as the last part of the thesis work, to gather quantitative and qualitative data for some of the variables described in the conceptual framework. A survey is a suitable strategy to gather information from a specific population. The survey was conducted in form of an online questionnaire. This method was chosen over interviews because it was desirable to get a large number of responses which was hard to achieve with individual interviews due to scope and duration of the thesis project. Apart from respondent information and additional comments, the variables gathered through the survey were Test Readability (TR), Test Result Business Goal Alignment (BGA), Test Failure Traceability (TFT), and User Satisfaction (US).

5.4.1 Data Collection

The survey was designed using Google Forms [43]. A summary of the survey questions can be found in Appendix E. The tentative design was presented to the thesis supervisor at the company and one of the project managers for feedback before being sent out to other employees on a larger scale. Invitation emails were sent to a sample of 26 employees. The survey was open for one week. The response data was downloaded in form of a Comma Separated Values (CSV) file which was imported into Microsoft Excel 2013 for further analysis. The statistical analysis and hypothesis tests were performed using Real Statistics Resource Pack (Release 4.3) [44] add-in in Microsoft Excel 2013.
5.5 Internal Documentation

Some of the information used in this study was obtained from the company’s documentation such as wiki pages and help files. The data was mostly used for understanding the current situation, project descriptions, software development process, and the testing process.

5.6 Personal Contact

One of the best ways to access the tacit knowledge in a company is through contact with personnel. Some of the data was gathered using this kind of communication which included talking to project members in person or contacting them via e-mails, instant messages, and video conferences.

5.7 Validity and Reliability

There can be several threats to validity and reliability of empirical results. Wohlin et al. explained four aspects of validity of the result of an experiment [27]:

1. **Conclusion Validity** which means that when drawing conclusions on relationship between treatment and outcome, there needs to be a statistical relationship with a given significance. This is considered when discussing the results in the next chapter.

2. **Internal Validity** which is about making sure that the observed relationship is not due to some factor other than the treatment. To achieve this, the experiment was carried out on a sandbox to achieve a controlled environment. All requirements for the IFS Application such as Oracle database and all requirements for the testing such Oracle WebLogic server, the web client, and Jenkins CI were installed locally on the sandbox to eliminate the unwanted effects that might be the result of transient issues such as network or database problems. Furthermore, tests were executed several times and the average value for test execution time ($t_a$) was used in order to filter out the invalid results.

3. **Construct Validity** which is about validity of the relationship between theory and observation and whether it is okay to generalize the result of the experiment to theory. Construct validity includes issues related to experiment design (design threats) and issues related to social factors (social threats). Some of the measures taken to combat design threats were:

   a. The variables such as what *effectiveness* or *efficiency* were defined clearly in conceptual framework.
b. Even though the object of the experiment was a single workflow which might under-represent the construct, the workflow was generic enough to be a good representative for workflows in the application.

c. Additional methods and observations such as interviews and survey were used to collect data on different variables defined in the conceptual framework. This methodological triangulation helped with cross-checking measurements for quantitative variables obtained via the experiment against other measurements for quantitative variables collected by a survey or the qualitative empirical findings from the interviews.

d. When choosing subjects to apply the treatments by writing automated tests or implementing a feature, it was important to make sure that the subjects were experienced in the required tool or programming language to eliminate the confounding constructs.

e. All constructs were presented by more than one dependent variable which is very important when generalizing results. For instance, software testing quality was presented by test automation ROI, test feedback, and test usability. This helps to avoid or detect situations where a treatment may unintentionally affect other constructs negatively.

With regard to social threats, even though the subjects might have had some guesses about the hypothesis, there were no indications that they were biased or behaved any differently. The researcher which also carried out the testing part for the BDD treatments tried to be neutral while applying the treatment as well as with analysis of the results.

4. **External Validity** which deals with generalization of the discovered causal relationship outside the scope of the study and to industrial practice. To battle the risks to external validity, extra care was taken in selecting relevant BDD tools, having the same environment and settings as the industrial practice, selecting subjects with similar experience level, and in short making the experiment as realistic as possible.

Several other activities was performed to increase the validity of the report: the project report (this report) was updated continuously. The project was carried out in an Agile team with daily standups which guaranteed constant reporting and feedback from the thesis supervisor at the company on research methodology, experiment, and survey questionnaires. To acquire review by peer researchers, the report author attended the course “Scientific Methods” at Linköping University while working on the report and got valuable feedback from course teachers and
fellow classmates. Furthermore, the report was reviewed by the examiner and the supervisor at
the university at different stages.

To make the result reliable, so that others can produce similar results by following the
research method, careful attention was paid to explain the research method exhaustively. All
material such as interview questions, survey questionnaires, the source code, and setup used in
the experiment were provided in the report.

In order to increase the response rate of the survey, pre-notification emails was sent by the
thesis supervisor at the company before sending out the survey invitation emails. The invitation
email was reviewed by insightful people such as the thesis supervisor and one of the team
leaders at the company. The survey questions were also reviewed by the aforementioned people
as well as the supervisor at the university. The survey was designed to take less than 10 minutes
to complete, in order to avoid invalid results due to tired and unmotivated respondents. The
last, open-ended, question in the survey was filled out by many respondents which proves the
fact that the motivation was quite high.

5.8 Research Ethics
The thesis work followed the guidelines for research and experimentation ethics proposed by
Wohlin et al. [27] and Runeson and Höst [45]. The thesis project was proposed by IFS which
implies informed consent of the company for participation in the study. The purpose of the
study was communicated with the team leader and team members where the study took place,
as well as interviewees, survey respondents, and experiment subjects. The survey responses
and interview data were collected anonymously so that the participants can express their
opinions freely. When designing the survey, attempt was made not to include culturally
sensitive questions.

The company was informed of the results through direct and constant reporting to the thesis
supervisor, one presentation for half-time control, and the final presentation of the results and
analysis at the end of the thesis work.

The information that could be regarded as confidential and sensitive such as project names
and unnecessary product details was removed from the report published publicly through
Linköping University. Furthermore, the thesis supervisor at the company was consulted when
in doubt about confidentiality of a material that needed to be included in the report.
Chapter 6

Results

In this chapter, the results obtained from the interviews, the experiment, and the survey are presented in separate sections. At the end of each section, there is a discussion on the results. For the interviews, the attempt was to understand whether an evaluation of Behavior Driven Development (BDD) was reasonable according to the obtained qualitative data. As for the experiment and survey results, the discussions are mainly focused on testing the thesis hypothesis by analyzing the quantitative data.

6.1 Interviews with Stakeholders

6.1.1 Project 1

The description is provided based on three interviews conducted in September 2015. Table 6.1 lists those interviewed from Project 1.

<table>
<thead>
<tr>
<th>Role</th>
<th>Role experience</th>
<th>IFS experience</th>
<th>Date</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project manager</td>
<td>5 months</td>
<td>17 years</td>
<td>2015-09-28</td>
<td>30 minutes</td>
</tr>
<tr>
<td>SE</td>
<td>5 months</td>
<td>10 years</td>
<td>2015-09-25</td>
<td>30 minutes</td>
</tr>
<tr>
<td>BSA</td>
<td>5 months</td>
<td>8 years</td>
<td>2015-09-28</td>
<td>30 minutes</td>
</tr>
</tbody>
</table>

*Table 6.1: Interviewees of Project 1*

6.1.1.1 Project description

Project 1 works on developing the web client for IFS Applications. The work on the project started in April 2015 with preliminary preview release in early 2016.

6.1.1.2 Interview findings

For the sake of brevity, the interview findings are summarized in the two tables below. Refer to Appendix B for full text of transcripts.
### 6.1. Interviews with Stakeholders

<table>
<thead>
<tr>
<th>Issues with development process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclear requirements (business goal)</td>
</tr>
<tr>
<td>Overlapping of tasks definitions (SE need to BSA-specific tasks)</td>
</tr>
<tr>
<td>Insufficient specification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Possible improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>More communication between roles in team</td>
</tr>
<tr>
<td>Focusing on one’s primary task</td>
</tr>
<tr>
<td>Better planning of sprints</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issues with testing process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing after development leads to bugs being found late in the process</td>
</tr>
<tr>
<td>Insufficient testing</td>
</tr>
<tr>
<td>Unclear process for test result follow-up</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Possible improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using test first approach</td>
</tr>
<tr>
<td>More test suites and better coverage, specifically for new and ongoing projects</td>
</tr>
<tr>
<td>Including testing as part of development process</td>
</tr>
</tbody>
</table>

After being presented with an introduction to BDD, most interviewees were positive on the idea of trying to evaluate its usage. It was mentioned that it would be interesting to try testing on different layers and find the sweet spot for the number and coverage of tests. Quick feedback and reliable results were listed a desirable features for the tests. The main concerns were the learning curve of Gherkin and BDD tools. Furthermore, there deemed to be a risk of BDD being overly ambitious and it was believed that for BDD to succeed, there needs to be very good collaboration between the stakeholders which could be difficult to achieve in a large multi-site organization like IFS.

### 6.1.2 Project 2

The description is provided based on three interviews conducted in September and October 2015. Table 6.2 lists those interviewed from Project 2.

#### 6.1.2.1 Project description

Project 2 belongs to a team that deals with the wide area of engine and equipment handling: (pre)maintenance of normal equipment, lifecycle development, fleet management. One of the main component that the team manages, for example work order management, has been in
RESULTS

place for 30 years. Project 2 is trying to throw it out and replace that which is a huge undertaking. Work on project 2 is about to start and is expected to take about 2 years.

<table>
<thead>
<tr>
<th>Role</th>
<th>Role experience</th>
<th>IFS experience</th>
<th>Date</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project manager</td>
<td>5 years</td>
<td>18 years</td>
<td>2015-10-05</td>
<td>30 minutes</td>
</tr>
<tr>
<td>SE</td>
<td>Starting on a new project</td>
<td>20 years</td>
<td>2015-10-01</td>
<td>30 minutes</td>
</tr>
<tr>
<td>BSA</td>
<td>5 years</td>
<td>20 years</td>
<td>2015-09-29</td>
<td>30 minutes</td>
</tr>
</tbody>
</table>

Table 6.2: Interviewees of Project 2

6.1.2.2 Interview findings

The interview findings are summarized in the two tables below. Refer to Appendix B for full text of transcripts.

<table>
<thead>
<tr>
<th>Issues with development process</th>
<th>Possible improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Specifications and requirement in task descriptions on the bug tracking system</td>
<td>• More and better specification</td>
</tr>
<tr>
<td>• Not following Agile all the time</td>
<td>• Adjusting process to team’s needs</td>
</tr>
<tr>
<td>• Overlapping of tasks definitions</td>
<td>• Automated regression testing</td>
</tr>
<tr>
<td>• Inefficient communication methods for geographically dispersed teams</td>
<td>• Encouraging conversations</td>
</tr>
<tr>
<td>• Sprints are sometimes too short</td>
<td>• More efficient methods of communication such as video chats for geographically dispersed teams</td>
</tr>
<tr>
<td></td>
<td>• More focus on product quality of the final product</td>
</tr>
</tbody>
</table>
6.1. Interviews with Stakeholders

<table>
<thead>
<tr>
<th>Issues with testing process</th>
<th>Possible improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Testing after development leads to bugs being found late in the process</td>
<td>• Including testing as part of development process</td>
</tr>
<tr>
<td>• Insufficient unit testing</td>
<td>• More dialog between team members</td>
</tr>
<tr>
<td>• Inaccurate test setup information</td>
<td>• More end-to-end testing such as business flow tests and in more teams</td>
</tr>
<tr>
<td>• Business flows are not tested properly</td>
<td>• More unit testing</td>
</tr>
<tr>
<td></td>
<td>• More education for users on how to use different types of tests</td>
</tr>
</tbody>
</table>

The interviewees of project 2, after being provided with an introduction to BDD, were also mostly in favor of the evaluation study. The fact that one of the characteristics of BDD is that it is possible to map the tests to the business goal of the final products, if proved by means of an experiment, was believed to be very helpful. It was generally agreed that testing is important and they company should always look for new testing methods to improve the software quality.

6.1.3 Discussion

Looking at the interview findings for the two projects, there seemed to be a general agreement on the importance of including testing as a part of development process instead of running some functional tests at the end of the process. The other interesting finding is that the teams could benefit from more conversation between the team members. There was also a consensus on that fact that a working product that adheres to requirements, which is the business goal of a product team, should be the primary focus. Finally, it was mentioned that better documentation, requirements, and specification would benefit the project teams. Most of the aforementioned improvements and findings were quite similar to characteristics of BDD. It was quite clear that an evaluation of usage of BDD in a development project could be well justified.

The feedback received on BDD was mostly positive. However, it was suggested that an evaluation would be important to try to determine whether BDD can live up to its reputation on improving software and testing quality, and resulting in a product that delivers business value.
6.2 Experiment
When presenting results in this section, BFT is used to refer to the baseline treatment described in section 5.3.7. e2e and REST are used to refer to the two BDD treatment alternatives described in sections 5.3.8 and 5.3.9 respectively.

6.2.1 Test Automation ROI
Table 6.3 summarizes the timing results for the experiment.

<table>
<thead>
<tr>
<th></th>
<th>BFT</th>
<th>e2e</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Creation Time ($t_c$)</td>
<td>18 h</td>
<td>5 h</td>
<td>3 h</td>
</tr>
<tr>
<td>Test Maintenance Time ($t_m$)</td>
<td>40 m</td>
<td>1 h</td>
<td>30 m</td>
</tr>
<tr>
<td>Test Execution Time for automatic tests ($t_a$)</td>
<td>15501 ms</td>
<td>15388 ms</td>
<td>588 ms</td>
</tr>
</tbody>
</table>

Table 6.3: Timing results for the experiment

As mentioned previously, the automated test created via each treatment was executed a number of times to eliminate the potential noise and the results for Test Execution Time ($t_a$) are the average values for these measurements. Figure 6.1 shows the $t_a$ measurements for all the test runs. It can be observed from the obtained results that the first test run is slower than the average test run. Furthermore, there are fluctuations in the results for e2e and BFT test runs. That is why it was decided to have multiple measurements. The average $t_a$ for the aforementioned treatments converged to approximately 15 seconds for 10 test runs. The number of test runs deemed to be sufficient based on the obtained measurements. It is also important to mention that $t_a$ cannot be used to compare the efficiency of the BFT test with the other two tests because it tests a different client. However, $t_a$ is a good indication of efficiency when comparing e2e and REST treatments: Based on these measurements, REST was 26 times faster than e2e counterpart.
Figure 6.1: Test Execution Time

Given that average Automated Execution Occurrences \( (n_a) \) was 365 (the BFT tests were run daily), average Maintenance Occurrences \( (n_m) \) was 12 (approximately once per month), Test Execution Time for manual tests \( (t_m) \) was three hours, and the equations from section 4.2.4.1, then Cost, Gain, and Test Automation ROI can be calculated for each treatment as displayed in Table 6.4.

<table>
<thead>
<tr>
<th></th>
<th>BFT</th>
<th>e2e</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>26 h</td>
<td>17 h</td>
<td>9 h</td>
</tr>
<tr>
<td>Gain(^5)</td>
<td>1093.42 h</td>
<td>1093.43 h</td>
<td>1094.94 h</td>
</tr>
<tr>
<td>Test Automation ROI</td>
<td>41.05%</td>
<td>63.31%</td>
<td>120.8%</td>
</tr>
</tbody>
</table>

Table 6.4: Test Automation ROI

6.2.1.1 Discussion

Looking at timings in Table 6.3, the average Test Maintenance Time \( (t_u) \) was one hour for e2e test and 40 minutes for BFT test. The value of \( t_u \) for REST test was 30 minutes which is half of the baseline. e2e and BFT have higher \( t_u \) probably due to the fact that both treatments run on the client layer. The test maintenance time largely improved by removing the client layer

---

\(^5\) Compared to manual testing
and testing directly against the web service. This confirms the general perception that by moving up in the technology stack and increasing the complexity, tests get harder to maintain.

However, the e2e test had a higher $t_a$ compared to BFT even though they both tested against the client layer. It is important to mention that the BFT tool had been in use for quite a while and had a framework (or a library of utility functions), while the BDD treatments were created from scratch for this experiment and did not have any utility libraries. Therefore, some of the time spent for creating and maintaining the BDD tests was spent on the utility libraries instead of the actual tests.

A similar observation can be made regarding the average automated Test Execution Time ($t_a$). While BFT and e2e treatments had very similar $t_a$ measurements, REST had an extremely faster run time, most probably due to the fact that by moving down the technology stack some of the complexity and latency introduced by higher levels such as client is removed resulting in better test execution times.

For Test Automation ROI displayed in Table 6.4, REST had twice more ROI than e2e and three times more than BFT primarily because it had a lower $t_c$ and $t_a$. In addition, both BDD alternatives provided more ROI compared to BFT.

### 6.2.2 Tester’s Productivity

The total Testing Efforts (TE) and Non-Commented Lines of Code (NCLOC) from the experiment were used to calculate Tester’s Productivity (TP) using the equation \((4.1)\) for each treatment as summarized in Table 6.5.

<table>
<thead>
<tr>
<th></th>
<th>BFT</th>
<th>e2e</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing Efforts (TE)</td>
<td>18 h</td>
<td>5 h</td>
<td>3 h</td>
</tr>
<tr>
<td>Non-Commented Lines of Code (NCLOC)</td>
<td>44</td>
<td>92</td>
<td>56</td>
</tr>
<tr>
<td>Tester’s Productivity</td>
<td>2.4</td>
<td>18.4</td>
<td>18.6</td>
</tr>
</tbody>
</table>

**Table 6.5: Tester’s Productivity**

### 6.2.2.1 Discussion

Looking at the result presented in Table 6.5, more code needed to be written for both e2e and REST test compared to BFT. One reason for this, as discussed earlier in section 6.2.1.1, might be the lack of utility libraries for the BDD treatments which requires more code to be written.
for them compared to the existing BFT framework. Nonetheless, both BDD alternatives had lower overall Testing Efforts (TE) than BFT which resulted in higher Tester’s Productivity (TP) compared to BFT.

### 6.3 Survey

In this section, **AQUA** is used to refer to the existing Agile-based development process and **BDD** refers to BDD process followed in the experiment. There were no differences between the two BDD alternatives as much as the survey was concerned because same activities were performed as a part of the BDD process.

The survey was responded by 19 persons out of 26 who received the requests for participation which is a response rate of 73%. A summary of the survey results can be found in Appendix E. The survey contained three questions about respondents’ primary role, total experience, and experience in the current role. This information was helpful to able to analyze the samples based on respondents’ role and experience. Figure 6.2 displays the functional roles of the survey respondents. The roles are categorized under three categories based on the function of the role:

1. **Technical Roles**, which includes Software Engineers (SE), Software Architects, and Product Architects. The functional description of this category is mainly design and development of products.

2. **Managing Roles**, which includes Product Managers, Project Managers (PM), Product Solution Managers (PSM), and Product Director (PD). While these roles might be having different job descriptions, they all have one thing in common: They focus mainly on the managing activities of the development process.

3. **Domain Experts**, which are the Business Systems Analysts (BSA) function as experts in different business domains. They represent the business insight in a development process.

Respectively, there were ten, six, and three respondents in each of the categories mentioned above. The sample size was sufficient for the intended hypothesis test. Unfortunately, there were not enough respondents in each role category to be able to perform separate comparisons for each category for the purpose of hypothesis test.

In the following sections, the survey results are analyzed for those dependent variables defined in the conceptual framework for which the survey questions were designed. The main aim of the survey was to obtain quantitative data for the aforementioned variables from the
RESULTS

project members’ answer to the survey’s single-choice questions. The questions used an interval scale of range one to five for the answers. Furthermore, the survey was designed with pairs of questions for the dependent variables. The following is an example for such a question pair:

1. The respondents were presented with the test result for a failed BDD test and were asked to rate how easy it is to find the reason for the failure.

2. The test result for a failed Business Flow Test (BFT) was shown to the respondent and the same question about ease of tracing the failure cause was asked.

Based on this design, a paired t-test could be a good candidate for hypothesis test if the data was normally distributed. To test if the survey data had a normal or Gaussian distribution, a Shapiro-Wilk test with $\alpha=0.05$ was performed. Only the data for three out of 13 single-choice questions was normally distributed. As a result, the non-parametric Mann-Whitney U-test was chosen for hypothesis testing to investigate if it was possible to reject the null hypothesis by proving that there was a statistically significant difference in the samples for a pair of questions. In other words, if the asymptotic significance ($p$) was smaller than $\alpha=0.05$.

![Functional Roles of Respondents]

**Figure 6.2: The functional roles of survey respondents**
Each of following sections, which present the results for some of the dependent variables defined in the conceptual framework, starts with a comparison of mean values for the corresponding question pair grouped by functional role and years of experience. The Mann-Whitney U-test, if suitable for the sample size, is then performed to see if the null hypothesis can be rejected for that variable.

### 6.3.1 Business Goal Alignment

Some of the survey questions were specifically designed to measure the Business Goal Alignment (BGA) of different artifacts pertaining to activities performed in the experiment as parts of the development processes that were being investigated. Once again, with the focus being mostly on testing activities, the questions dealt with BGA for specification and test result artifacts.

#### 6.3.1.1 Specification vs. Feature

The respondents were asked to rate how easy it was to understand the business value delivered by the specification document used for AQUA compared to the BDD feature written in Gherkin. Figure 6.3 displays the mean values of BGA for this artifacts grouped by respondents’ functional role and work experience. Except for the domain experts, all other roles thought that it was easier to map the feature file to the business goal of the application under test. Moreover, more experienced respondents gave a higher rating to BDD feature. Even though the overall mean value for the BDD feature file was 4.0 compared to 3.7 for the AQUA specification document, a Mann-Whitney U-test on the data samples could not show any significance to reject the null hypothesis.
6.3.1.2 Test Result Business Goal Alignment

The respondents were provided with screenshots of the test result overview page and test result page for a single test for the two treatments and were asked to rate the ease of mapping these pages to business goal of application under test. Figure 6.4 displays the mean values for BGA for the aforementioned artifacts. For both pairs of questions, all respondents gave a higher rating to BDD test results. The data samples for the two pairs of questions were tested with Mann-Whitney U-test and the null hypothesis was rejected for both questions meaning that the test result BGA for BDD results was significantly higher than the AQUA results.

6.3.1.3 Discussion

The overall BGA for the different testing activities which was defined as a representative for testing effectiveness was not proven to have been improved using BDD because it was not possible to show a significant difference in project members’ opinion for the specification vs. feature. As a result, it cannot be stated that the testing process for BDD is more effective than AQUA. Other variables such as defect density and functional test pass rate could be used to investigate the effectiveness of testing and development processes. Unfortunately, getting such information requires a longer study as discussed in section 4.2.2.
Nevertheless, the test result \textit{BGA}, which is defined as one of the variables representing \textit{Test Feedback} in the conceptual framework, is proven to be higher for BDD.
The other interesting notion which was mentioned in one of the respondents’ comments was the fact that \textit{Business Goal} was not defined in the questions:

“...Some of the questions were hard to answer, and I am thinking especially about those that mentioned 'business goals'. Too fuzzy a term for me, without more context. I think that the way they write 'test cases' in BDD is interesting, and might be a good way for us to define tests. I understand the goal is that a BSA should be able to write the scenarios. I would like that to be true, but perhaps there needs to be some training on what kind of things you can express in that language, for example when it comes to how I can specify that a certain value is to be entered in a certain file or part of the UI. I think the JavaScript BDD code example thing had too much 'noise' (or 'overhead' or 'boilerplate') to be written by a non-technical person. I hope we can perhaps generate that code from that other, simpler, language...”

[Product Architect]

While the term \textit{Business Goal} was not directly defined, it was not difficult for respondents to guess the definition. The survey started by displaying a specification document and asking respondents to rate it on how much business value it delivers. The exact same question was asked for a Gherkin feature afterwards. This should have given some hint on the business goal of the specific feature of the application which dealt with customer orders. It is true that there was a risk of disparate interpretations by different roles in a project. However, the idea with the survey design was to try to not be too specific on definitions to allow for slightly different interpretations from dissimilar functional roles.

\textbf{6.3.2 Test Failure Traceability}

The other survey question was whether the respondents could easily detect why the test had failed by looking at the test result page. This question was designed to measure \textit{Test Failure Traceability (TFT)} for a failing test. Figure 6.5 displays the mean values for \textit{TFT} for BDD compared to AQUA. The null hypothesis was rejected with a Mann-Whitney U-test suggesting a higher \textit{TFT} for BDD.

\textbf{6.3.2.1 Discussion}

One of the respondents provided the following comment via the last open-ended question:

“\textit{it is good if the video generation is also considered on BFTs on this survey}”

[Software Engineer]

The existing testing framework recorded a video of the testing session as well as a screenshot of the application that could be useful for tracing the test failure. The above comment was not entirely correct because a screenshot of the application and \textit{a link to the recorded video} was visible in the survey question. Nevertheless, video recording can be mostly helpful to trace the
failure for graphical user interface (GUI) testing. One of the BDD alternative was on web service layer where the video was less helpful. The question focused on tracing the reason for test failure from the result page and not all other artifacts such as video or server logs.

![Test Failure Traceability](image)

**Figure 6.5: Test Failure Traceability**

### 6.3.3 Test Readability

Based on *Test Readability (TR)* definition from section 4.2.4.3, the overall readability of the feature file and the step definitions for BDD was compared to the readability of the BFT. The BDD feature written in Gherkin is more than just a specification file. Gherkin has a formal syntax and can be regarded as a high-level test. In other words, in certain cases such as changing a variable in the step or duplicating a step, as long as the step definition is available, it is sufficient to edit the feature file only to change a test. While the specification document created in the existing Agile based process was just a document file which did not have a strong link with the resulting automated test as displayed earlier in Figure 2.8.

Figure 6.6 compares the average *TR* of the aggregate samples for BDD’s feature and step definition files to BFT. It is important to mention that Mann-Whitney U-test can be used for unequal sample sizes which was the case for the readability results. The test showed statistically significant difference between the samples indicating a higher *TR* for BDD.
6.3.4 User Satisfaction

One of the measures for usability, as defined in section 4.2.4.3, was User Satisfaction (US). Figure 6.7 displays the average US for BDD and AQUA. Because of the small sample sizes for this question, it is not possible to perform a Mann-Whitney U-test for the data. For cases like this, the data samples should be discussed individually.
6.3.4.1 Discussion
There were 9 respondents who had experience with BFT and they were from all three major functional roles. One interesting observation is that non-technical roles were less satisfied with BFT. Unfortunately, only three respondents had worked with BDD which were the two software engineers who helped with the BDD experiment and the thesis supervisor. The thesis supervisor was also the domain expert for the experiment. Therefore, the average result obtained for US can be interpreted to represent the opinion for the domain experts and the technical roles.

The result for BDD is normally distributed which suggests that the mean value (4.0) can constitute the general opinion of the users from different roles who have used BDD. The mean value for US for BFT was 3.67. Based on this discussion, it can be stated that the end-users are at least equally satisfied with the two treatments.
Chapter 7

Conclusions

This study evaluated two experimental usages of Behavior Driven Development (BDD) with tests that ran on different layers of the application architecture. The research aimed to find an effective and efficient development process with BDD and to investigate its impact on testing quality of a large-scale application in a large multisite organization. The conclusions start with a comparison between the two BDD alternatives in an attempt to find the one that is more effective and efficient based on the results presented in the previous chapter. Subsequently, the chosen BDD process is compared to the existing Agile-based development process to determine whether it improves the testing quality.

7.1 Comparison of BDD Alternatives

According to definition of effectiveness for a development process provided in section 4.2.2, alignment with business goal was chosen as a representative for effectiveness. However, because using the same feature file and result reporting system meant that both BDD alternatives aligned equally with the business goal of the application, no conclusions could be made as to which BDD alternative is more effective. More research and alternative variables for software quality such as defect density is needed to be able to detect the effectiveness of BDD.

On the other hand, section 4.2.3 defined an efficient testing process as having low overall Testing Efforts (TE) and average Test Maintenance Time (tm) while having high Tester’s Productivity (TP) and Test Failure Traceability (TFT). As presented in the previous chapter, BDD with tests on the web service layer is more efficient than BDD on the client layer because it had better measurements for the aforementioned variables with the exception of TFT where none of the BDD alternatives is proven to be superior.

Thus, BDD on web service layer is chosen as the candidate to be compared with the existing Agile-based process.

7.2 Impact on Testing Quality

In the conceptual framework, Test Automation ROI, Test Feedback, and Test Usability were defined as variables representing testing quality:
1. **Test Automation ROI**: According to the results from the experiment discussed in section 6.2.1.1, BDD was found to increase the *Test Automation ROI*.

2. **Test Feedback**: Test Feedback was defined as a combination of *Test Execution Time* ($t_a$), test result *Business Goal Alignment (BGA)*, and *Test Failure Traceability (TFT)*. Based on the results presented in section 6.2.1 and the discussions in sections 6.3.1.3 and 6.3.2.1, it can be concluded that BDD provides better test feedback.

3. **Test Usability**: *Test Readability (TR)* and *User Satisfaction (US)* were chosen to represent usability of tests. BDD tests were shown to be more readable as presented in section 6.3.3. However, the discussion in section 6.3.4.1 suggested that the end-users were equally satisfied with both development processes. It can be concluded that BDD enhances test usability.

In conclusion, the result from this study showed that BDD on the web service layer improved the testing quality for the application under test.

### 7.3 Transferability

The discussion in the section is about transferability of the results of this study to similar large-scale applications in similar large multisite organizations. It is possible to consider the conclusions of this study for analytical generalization with the help of the conceptual framework. The survey respondents had different functional roles and were from geographically dispersed teams which makes it possible to generalize the thesis findings to a certain degree to large multisite organizations. The workflow used in the experiment was regarded to be a suitable candidate for a typical business workflow in a large-scale application; there were no indications that using a different workflow would have entailed unpredictable changes to the study results.

Regarding *Test Feedback*, it can be deduced that test result reports serving as *living documentation* by mapping the requirements to business goal are highly appreciated by different roles in a project, especially those with more experience, and that the BDD-style test result reports facilitate failure traceability by linking the test results to the actual features and scenarios.

BDD process of specifying behavior in terms of high-level feature in Gherkin, and converting it to step definition might be more labor-intensive than creating automated tests directly from some specification document. However, the fact that the overall test readability increases for all different roles in a project makes it worthwhile to follow the BDD process.
7.4 Future Work

Due to limited scope of the study and the constraints of a master thesis project, it was not possible to work on some aspects of BDD in the theoretical framework. These aspects are listed in this section as ideas for future research.

One suggestion for future work is to conduct a study to compare the quality of the software produced with the BDD process and the current Agile-based process using metrics and variables such as code coverage, defect density, customer problems and customer satisfaction.

This study investigated BDD tests on two different application layers. It would be interesting to examine testing on the database or database abstraction layer. This is specifically interesting because (most of) the business logic is perform in the aforementioned layer.

As mentioned in section 5.1, a case study is preferable in order to be able to introduce a method or tool in a company on a larger scale. Therefore, one idea is to conduct a (multiple-) case study to evaluate BDD in its natural context in order to observe and investigate complex real life issues which otherwise would be difficult to do with an experiment. The scope of this study was not sufficient for performing such a case study.
Appendix A

Interview Questions

Role:

Experience in current role:

Experience in IFS:

Date and duration of interview:

Questions:

1. What do you think about the current development process? Pros and cons?

2. How can the development process be improved?

3. What do you think about the current testing process?

4. How can the testing process be improved?

(A brief presentation for BDD is provided by the interviewer)

5. What do you think about BDD and its premise for software quality?

6. Any other remarks/thoughts?
Appendix B

Interview Transcripts

Project 1
Current Development Process
The company has adopted Agile process for development for the last two years. However, the Agile process is not followed exactly by the book. According to the SE, they do standups, but not on a weekly basis rather than daily and in the current project the standup meeting is done every 2 to 3 days. The development work is done in sprints/iterations and at the end of each sprint they do the retrospective meeting that has been helpful. The SE believes that the retrospective meetings could be a bit longer so more people can give feedback.

The project manager finds the development project to be a bit rigid and sometimes they do not have the clear view of what exactly they have done or who their customer is. Therefore, the manager thinks that the AQUA process is mostly focused on the D(evelopment) part of the Research and Development (R&D). Even though the project manager is quite fond of the Agile method, but would like to have a guidance rather than a strict process. The team tries to follow the Agile method manifesto, but it is not followed strictly. The project manager believes that the larger the project is the more strict one needs to be, but they often have projects with two or three people, so it is not feasible to have all the formal meetings since they members are sitting next to each other and the communication happens naturally. One major problem in the project is that sometimes the SEs need to analyze the business model (which is the BSA’s job description) in addition to their main task that is developing features and it doubles the amount of work they need to do. The current situation is that you get a demand that is more like a cloud, for example We need to improve security that is very obscure and there is no BSA that can break that down into proper specification. Also there is a lot of legacy code that sometime influence the process for change requests.

The BSA has been mostly involved with support projects where they follow a different process (similar to Kanban), but is going to start working with projects that follow the AQUA process from now on. The BSA has been working with the Agile method to some extent, doing the daily standups and working in iterations of one month. When it comes to specification, the BSA does not think they actually have any specifications, rather there are issues and user stories
created in the bug tracking system (Jira). As for specifications, the BSA comes from the consultancy department where a lot of formal documentation should be created early on in the project rather than having the iterative approach. Currently, working with Jira, specifications for epics are created for each iteration and they stay on the issue and do not get turned into formal specifications elsewhere. The BSA believes that it is quite good to have a close cooperation with the other members of the team with the Agile process, but it can also be stressful at times with short sprints that results in things getting solved too quickly sometimes.

Potential Improvements to Development Process
To improve the current development process, the SE suggests having the standard 4 hour planning meeting before a 2 weeks iteration (current process is a 2 hour planning meeting for a 1 month sprint), so that they can discuss the details and time estimation. It is a recurring problem that they either have too many or too few tasks in an iteration and they need to adjust the backlog.

The project manager thinks that they need to have a set of guidelines for dos and don’ts, but the team should have some freedom to choose how to do things. Furthermore, the manager believes that there needs to be more focus on the R(esearch) part of the R&D. As for the collaboration between BSAs and SEs, it would be much better if the SEs can focus on their main task, which is developing the software and leave the business analysis part to the BSA. Specifically with the current project, being in early stages, there is a possibility to specify the requirement more clearly for the new client (there are still some legacy parts on the server/middle-tier layer though).

According to the BSA, it is good to let some features mature more, both in the idea and the implementation. The BSA thinks there is a redundancy of information since the company uses two different systems for bug tracking, which is Jira and is used by support and engineering teams and Life Cycle System (LCS) used by the customers and it would be much better with just one system.

Current Testing Process
At IFS, according to the SE, one does not start with writing a test and the developers usually do their own testing. The testing process has always been at the final stage after everything is ready. However, when it comes to deployment there is a substantial area of testing that is missing and lots of bugs are found regarding end-user testing. So toward the end of the project there is a lot of bug-fixing activities which puts more pressure on the developers.
The project manager regards testing as one of the biggest problems of the company. The manager’s view is that they simply do not have proper testing. Nevertheless, the project manager believes that they know about this issue and that they have big holes they have been trying to fill in by means of current test types such as Business Flow Test (BFT). So in a nutshell, they have tried different things, but they do not have a testing process that works. When it comes to testing, it is not following the Agile testing process, rather it is done as a separate process at the end of the development cycle.

The BSA thinks that when one looks at the current test types, for instance code analysis tests and BFT, one sees a very extensive test process. However, when it comes to looking at the results of testing and following up, things can be improved from a process perspective. For instance, if there are a number of failures in BFT results, it is not clear to the BSA if people follow up on the errors or what the process actually is.

**Potential Improvements to Testing Process**

According to the SE, Test Driven Development (TDD) can improve the current testing process. TDD has not been used in any of the previous project at IFS, but it can be very helpful to have comprehensive test suite for the builds at the end of the project.

To improve the testing process, the project manager thinks it is not going to be easy to add tests to the existing or finished projects. For ongoing and new projects it will be much easier to introduce testing processes. The project manager believes that they need to start including testing into the development process.

The BSA suggests that the testing should be a part of the development process and someone should follow up on the errors and failures. Furthermore, the BSA thinks that IFS can profit from having a separate QA role rather than having everyone do the QA. Furthermore, different teams have different requirements for testing and central components like the web client that is used by all other projects will need to be tested thoroughly.

**Perception of Premise of BDD**

After a brief presentation of BDD is provided by the interviewer, the interviewees are asked about what they think about this approach and whether it can help the teams to provide a better software.

The SE believes, that BDD promises quite a lot. When using TDD the test feature and scenarios still need to be written by someone who has technical or programming knowledge. However, if they use BDD the BSAs could actually write specifications which is quite
convenient because those are the ones who have the domain knowledge and know about the business requirements. The specifications are then translated into empty method stubs that the developers can fill in and turn into automated tests. So in general the SE was quite positive about BDD promises. The only concern for the SE was that the learning curve of the Gherkin language and BDD tools could be quite steep.

The project manager is certain that they will try out TDD or that they actually already do. The manager sees BDD as a continuation of TDD and thinks that it would be good for them to try it out. Regarding different layers such as unit tests and UI tests, sometimes the differences between these different layers and test types is quite blurry and a bit confusing at times. Moreover, the manager does believe that one should never try to test everything, rather the teams should find the sweet spot for the number and coverage of tests. All that being said, the project manager likes the idea with BDD and sees the point. What the project manager wants to see is a large basket of possible ideas and tools to use. As long as they can get a quick feedback and reliable test results it should be possible to include testing as a part of the continuous integration. Finally, the manager believes that with the current project they have the opportunity to start introducing some kind of QA process. It is, however, important to not start too big, but try with a small step and gradually work on extending the test coverage.

The BSA has some experience with BDD tools like Cucumber [32] and is fond of automation. However, the BSA feels that BDD is a bit idealistic meaning that for it to succeed it will need good collaboration between stakeholders in a big company like IFS. The BSA enjoys the fact that it is based on use cases/user scenarios which is different from unit tests that run on a low level. Since the BSA believes that the business value is the most important goal, this could be a very effective method. Finally, the BSA has some gripe with TDD and believes that it has got some critics lately. The idea with TDD is to write code to make a failing test suite to pass, rather than writing a software that delivers business value. BDD can help to avoid this issue, but it is interesting to see how it will work out in practice.

Project 2
Current Development Process

Nowadays the development process is better than the early days with the waterfall process when the BSA started, when they had to select a lot of requirements before the project started, followed by a long project (1-1.5 years). After the release was done, the customer would come with requirement changes. Currently, they often have some customer representative when a project is ongoing that can see the demos as the project develops and give the feedback that is
necessary for the iterative process. The Agile and the AQUA process, has affected the BSA in the sense that they work more as a team now and the focus is on working in iteration and the constant evaluation. However, the BSA believes that the habit of specifying the requirement in the task description on Jira is not very helpful. The other issue is that even though they try to work according to AQUA and Agile, sometimes if things are not going as planned and they get into difficulties, then they go outside the methodology.

The current development process has both pros and cons according to the SE. An advantage with the AQUA process is that it focuses on unit and automation testing which is quite useful for the developer because it gives them hints about the quality of the code. The problem with AQUA, however, is that projects come up with new ideas and improvement to the current process, but they do not get added to the process. Today, everything rests on developer’s shoulders and how well they are testing their own code. Sometimes after a new feature is checked in and enters the environment, the first thing functional testers run into is that the code is not working as it should. When it comes to collaboration between different roles in a team, the conversation is quite easy as long as the team members are on the same site. However, if the team members are in different locations, using the inefficient methods of communication like emails or comments in the bug tracking system can be very counter-productive.

The project manager is working on getting the development process completely rewritten. However, the manager thinks Scrum is much better than all different waterfall methods they have tried for the following reasons:

- constant feedback
- no need to design upfront so it is easier to adjust if the requirements changes
- at least one demo per month to the stakeholders
- possible to get help for testing from the outside world, for instance consulting organizations

In their team they adhere fairly strictly to Scrum apart from the roles. As for the AQUA process, the manager believes that they still have the tendency to push bugs and errors to the end of the project and that is when the testing happens.

**Potential Improvements to Development Process**

The BSA believes that there needs to be more specifications similar to what they had earlier, so that you could go back (even after some years) and look at a specification to find the purpose of a certain feature and relearn how it should work. Regarding the development process, the
BSA thinks that sometimes the process needs to be adjust to the team needs for them to be able to follow it properly or else they will end up not using it altogether.

The SE thinks that the development process should evolve as it is being used within the projects. Obvious and simple things should be tested with automated tests so that simple bugs do not creep in to production. For collaboration in teams, the members should have the feeling that it is an open discussion and people are not discouraged from asking questions. No question is a stupid question. It might only mean that the person asking such a question has not understood the requirement or functionality which means that the question is still relevant and it is better that the question is asked and answered. For geographically dispersed teams it is very important to make use more efficient methods of communication like instant messaging and video chat.

The product manager does not think that the methodology is really bad in itself, but there needs to be more emphasis on the fact that a working software is the primary measure. There should always be good quality in the final product, but that is not always followed and sometimes they need to argue on priorities of bug fixing and development tasks. The other problem is that the sprints are sometimes too fast and it is better to focus more on quality.

**Current Testing Process**

The BSA thinks that sometimes the testing is performed and the bugs are found a little too late in the process because there is not a good support for unit testing for the developers, causing easy errors to not be avoided.

The SE thinks that some of the tests need a certain setup and the information on how to do this setups is not always accurate. Furthermore, the RACE environment (which is a prefilled version of the database used for demonstration and internal usage) is created more from a sales perspective and is not for the developers. As a result, when a developer adds new functionality the SE needs to add new data for it as well so that the test systems can continue to work.

In their team, according to the project manager, they do not have any business flow tests. They do not have any support for automatic functional workflow testing at all. The only types of tests they have are manual tests and some automatic technical tests like upgrade testing. The business flow tests are apparently not working since there has been work on it for many years, but it is still not possible to use in development projects.
Potential Improvements to Testing Process

The BSA believes that the developers should have some basic testing on what they have developed and that the testing should not be done as a separate task at the end of the project. Communication between software engineers and BSAs is very important. It will be useful to have more dialog instead of just filing a bug on Jira and watch it go back and forth between the team members. In general, the BSA believes it will be good to introduce the tests like BFT to even more teams and use them when merging new features.

Having good unit tests will help developers to find errors earlier in the process according to the SE. Moreover, end-to-end tests like BFT are very important since they always find errors that is not normally discovered by the developers. Nevertheless, people need to be educated about what kind of tests there are, what they can do, the requirement for the testing tools, how the tools work and how to interpret and use the test results.

The project manager thinks that they should focus on automatic business flow test to achieve a good quality. Therefore, there is a need for a good automated test tool that is easy to maintain so that when they develop something they can test it directly, specifically with regards to business flow tests. They have done some usability testing on some of the project, but it is not used in all their projects, so that is probably somewhere they could improve. However, they were to prioritize, the manager would do the automated tests first and then focus on some usability testing.

Perception of Premise of BDD

The BSA does not think it will be difficult to use BDD instead of using other tools like BFT. Regardless of the tools used, the BSA believes that testing is very important and they should always look for new ways to test the software to be able to improve its quality.

According to the SE, if they could use BDD to test the basics of the forms in IFS Applications, it can be quite useful. Furthermore, the SE thinks that it is very promising and hopes they get more of the similar tests. Specifically because in most cases one knows what they really want from the product, but one does not care what happens in the background and that is where this approach can be useful. The idea is good, but it depends on how it is used since for IFS this is quite new and needs to be experimented.

The project manager does not directly perform the development or testing, be thinks that BDD looks quite good to have as a principle and a positive change to the current process.
Appendix C

Source Code for client-e2e-tests

```
environment.js
package.json
protractor.conf.js
README.md

├───output
│       README.md

└───test
    ├───features
    │       customer_order.feature
    │
    │       steps_definitions
    │       customer_order_steps.js
    │
    │       page_objects
    │       customer_order_page.js
    │
    └───support
            hooks.js
            world.js

package.json:

{
    "name": "client-e2e-tests",
    "version": "0.1.0",
    "description": "",
    "scripts": {
        "start": "cd client-framework & npm start",
        "update-webdriver": "node_modules/.bin/webdriver-manager update --standalone",
        "start-webdriver": "node_modules/.bin/webdriver-manager start",
        "test": "protractor",
        "debug": "protractor debug",
        "interactive": "protractor --elementExplorer"
    },
    "repository": {
        "type": "git",
        "url": "http://[IFS Git server address]/client-e2e-tests.git"
    },
    "license": "UNLICENSED",
    "private": true,
    "devDependencies": {},
    "dependencies": {
        "chai": "^3.0.0",
        "chai-as-promised": "^5.1.0",
        "cucumber": "^0.9.2",
        "fs": "0.0.2",
        "mkdirp": "^0.5.1",
        "path": "^0.12.7",
        "protractor": "^3.0.0",
        "protractor-cucumber-framework": "^0.3.0",
        "sanitize-filename": "^1.4.5"
    }
}
```
protractor.conf.js:

```javascript
var env = require('./environment.js');

exports.config = {
  seleniumAddress: env.seleniumAddress,
  baseUrl: env.baseUrl,
  specs: ['./test/features/customer_order.feature'],
  framework: 'custom',
  frameworkPath: './node_modules/protractor-cucumber-framework',
  cucumberOpts: {
    format: ['json:output/cucumber_report.json', 'pretty'],
    colors: true,
    source: true,
    require: ['./test/features/steps_definitions/*.js']
  },
  // wait for every app to be stable before each action,
  // and search within all apps when finding elements.
  // useAllAngular2AppRoots: true,
  rootElement: 'app'
};
```

environment.js:

```javascript
// Common configuration files with defaults plus overrides from environment vars
var webServerDefaultPort = 8090;

module.exports = {
  // The address of a running selenium server.
  seleniumAddress: (process.env.SELENIUM_URL || 'http://localhost:4444/wd/hub'),

  // Capabilities to be passed to the webdriver instance.
  capabilities: {
    'browserName': (process.env.TEST_BROWSER_NAME || 'chrome'),
    'version': (process.env.TEST_BROWSER_VERSION || 'ANY')
  },

  // Default http port to host the web server
  webServerDefaultPort: webServerDefaultPort,

  // Protractor interactive tests
  interactiveTestPort: 6969,

  // A base URL for your application under test.
  baseUrl: 'http://' + (process.env.HTTP_HOST || 'localhost') + ':' + (process.env.HTTP_PORT || webServerDefaultPort)
};
```
customer_order_steps.js:

```javascript
var chai = require('chai');
chai.use(require('chai-as-promised'));
var expect = chai.expect;
var CustomerOrderPage = require('./page_objects/customer_order_page');

module.exports = function() {
    this.World = require('../support/world.js').World;
    var startPage = new StartPage();
    var customerOrderPage = new CustomerOrderPage();

    this.Given(/\^I am the user "([^\"]+)"\$/, function (user) {
        var url = 'http://localhost:8080/ifs/app/index.html#/page/CustomerOrder/MasterDetailPage';
        return browser.get(url);
        // Make sure user is logged in
    });

    this.When(/\^I add a new customer order with customer \(\d+)\$/, (timeout: 60 * 1000), function (customerNo) {
        browser.sleep(1000);
        customerOrderPage.newButton.click();
        this.verifySentKeys(customerOrderPage.customer, customerNo);
        return customerOrderPage.saveButton.click();
    });

    this.Then(/\^a customer order with status "([^\"]+)" should be added\$/, function (state) {
        browser.sleep(500);
        return expect(customerOrderPage.state.getAttribute('value')).to.eventually.equal(state);
    });

    this.Given(/\^I have a customer order\$/, function (callback) {
        this.orderNo = customerOrderPage.orderNo.getAttribute('value');
        callback();
    });

    this.When(/\^I add a new order line with sales part no "([^\"]+)" and sales qty \(\d+)\$/, (timeout: 60 * 1000), function (salesPartNo, salesQty) {
        customerOrderPage.newOrderLineButton.click();
        customerOrderPage.salesPartNo.clear().sendKeys(salesPartNo).sendKeys(protractor.Key.TAB);
        this.verifySentKeys(customerOrderPage.salesPartNo, salesPartNo);
        return customerOrderPage.salesQty.clear().sendKeys(salesQty);
        this.verifySentKeys(customerOrderPage.salesQty, salesQty);
        return customerOrderPage.saveNewOrderLineButton.click();
    });

    this.Then(/\^the order line should be added\$/, (timeout: 60 * 1000), function () {
        browser.sleep(500);
        return expect(customerOrderPage.orderLineNo.getAttribute('value')).to.be.eventually.equal('1');
    });
}
```
customer_order_page.js:

```javascript
var CustomerOrderPage = function() {
    var group = element.all(by.tagName('group')).first();
    this.customer = group.element(by.id('Customer'));
    this.state = group.element(by.id('state'));
    this.orderNo = group.element(by.id('OrderNo'));
    this.newButton = element(by.id('page-toolbar-new-button'));
    this.saveButton = element(by.id('page-toolbar-save-button'));
    this.saveNewOrderLineButton = element(by.id('saveNewOrderLineButton'));
    this.newOrderLineNo = element(by.id('newOrderLineNo'));
    this.salesPartNo = element(by.id('salesPartNo'));
    this.salesQty = element(by.id('salesQty'));
};

module.exports = CustomerOrderPage;
```

world.js:

```javascript
function World() {
    // Config vars
    this.Config = {
        reportDir: process.env.TEST_RESULTS_DIR || process.cwd() + '/output'
    };

    // Sometimes sendKeys() does not work correctly, let us know in that case
    this.verifySentKeys = function(element, str) {
        element.getAttribute('value').then(function(sentKeys) {
            if (sentKeys != str) {
                console.log('WARNING: sendKeys() was not successful! Sent value was: ' + sentKeys);
            }
        });
    };
}

module.exports.World = World;
```
hooks.js:

'use strict';

var fs = require('fs');
var path = require('path');
var sanitize = require("sanitize-filename");

var hooks = function () {
    this.World = require('./world').World;

    this.BeforeFeatures(function (event, callback) {
        // setup
        callback();
    });

    this.Before(function (scenario, callback) {
        // before each scenario
        callback();
    });

    this.After(function (scenario, callback) {
        // after each scenario
        // take a screenshot of the browser if the scenario failed
        if (scenario.isFailed()) {
            var screenshotDir = path.join(this.Config.reportDir, 'screenshots');
            if (!fs.existsSync(screenshotDir)) {
                fs.mkdirSync(screenshotDir);
            }
            browser.takeScreenshot().then(function (data) {
                var base64Data = data.replace(/\s+/g, "");
                var file_name = path.join(screenshotDir, sanitize(scenario.getName()) + ".png").replace(/\s+/g, "");
                fs.writeFile(file_name, base64Data, 'base64', function (err) {
                    if (err) console.log(err);
                });
            }).then(function () {
                callback();
            });
        }
        callback();
    });

    this.AfterFeatures(function (event, callback) { // teardown
        callback();
    });

};

module.exports = hooks;
Appendix D

Source Code for client-rest-tests

```json
package.json

{  
  "name": "client-rest-tests",  
  "version": "0.1.0",  
  "description": "",  
  "scripts": {  
    "test": "cucumber-js test/features/customer_order.feature --format json:output/cucumber_report.json --format pretty"  
  },  
  "repository": {  
    "type": "git",  
    "url": "http://[IFS Git server address]/client-rest-tests.git"  
  },  
  "license": "UNLICENSED",  
  "private": true,  
  "devDependencies": {  
    "request-debug": "^0.2.0"  
  },  
  "dependencies": {  
    "chai": "^3.4.0",  
    "chai-as-promised": "^5.1.0",  
    "cucumber": "^0.8.1",  
    "mocha": "^2.3.3",  
    "request-promise": "^1.0.2"  
  }
}
```
customer_order_steps.js:

```javascript
var chai = require('chai');
chai.use(require('chai-as-promised'));
var expect = chai.expect;
var serviceRoot = 'http://localhost:8080/ifsapp/OrderManagement.svc/';
var request = require('./../../../../lib/util.js').request;
var customerOrderPayload = {};
var customerOrderLinePayload = {};

module.exports = function() {
  this.Given(/I am the user "([^"]+)"$/, function (user, callback) {
    customerOrderPayload.AuthorizeCode = user;
    callback();
  });

  this.When(/I add a new customer order with customer (\d+)/, function (customerNo, callback) {
    customerOrderPayload.CustomerNo = customerNo;
    this.dataPromise = request(serviceRoot + 'CustomerOrders',
    customerOrderPayload);
    callback();
  });

  this.Then(/a customer order with status "([^"]+)" should be added$/, function (state) {
    this.dataPromise.then(function (res) {
      this.orderNo = res.body.OrderNo;
      return
      expect(this.dataPromise).to.eventually.have.deep.property('body.state', state);
    });
  });

  this.Given(/I have a customer order$/, function (callback) {
    // NOOP, since the OrderNo is already in this.orderNo
    callback();
  });

  this.When(/I add a new order line with sales part no "([^"]+)" and sales qty (\d+)/, function (partNo, desiredQty, callback) {
    customerOrderLinePayload.OrderNo = this.orderNo;
    customerOrderLinePayload.PartNo = partNo;
    customerOrderLinePayload.DesiredQty = desiredQty;
    this.dataPromise = request(serviceRoot + 'CustomerOrderLines',
    customerOrderLinePayload);
    callback();
  });

  this.Then(/the order line should be added$/, function () {
    return
    expect(this.dataPromise).to.eventually.have.deep.property('body.OrderLineNo');
  });
};
```
util.js:

```javascript
var rp = require('request-promise');
//require('request-promise').debug = true;
//require('request-debug')(rp);
/*
* Performs a GET requests to a webservice.
* @param {string} uri - The address of the webservice. e.g. 'https://some.webservice.svc/'.
* @param {object} options - Options for the request. Example:
* { qs: {
*   access_token: 'xxxxx xxxxx' // -> uri + '?access_token=xxxxx%20xxxxx'
* },
*   headers: {
*     'User-Agent': 'Request-Promise'
*   }
* }
* @returns {object} The server response as a Promise */
module.exports.getURL = function(uri, options) {
  if (typeof options === 'undefined') options = {};
  if (typeof options.json === 'undefined') options.json = true; // Automatically parse the JSON string in the response
  if (typeof options.jar === 'undefined') options.jar = true; // enable cookies
  options.simple = false;
  options.resolveWithFullResponse = true;
  options.uri = uri;
  options.jar = true; // enable cookies
  return rp(options);
}
/*
* Performs a POST/PUT/PATCH/DELETE request to a webservice.
* @param {string} uri - The address of the webservice. e.g. 'https://some.webservice.svc/'.
* @param {object} payload - The data to send with the request.
* @param {object} options - Options for the request. Example:
* { qs: {
*   access_token: 'xxxxx xxxxx' // -> uri + '?access_token=xxxxx%20xxxxx'
* },
*   headers: {
*     'User-Agent': 'Request-Promise'
*   }
* }
* @returns {object} The server response as a Promise */
module.exports.request = function(uri, payload, options) {
  if (typeof options === 'undefined') options = {};
  if (typeof options.json === 'undefined') options.json = true; // Automatically stringifies the body to JSON
  if (typeof payload !== 'undefined') options.body = payload;
  if (typeof options.method === 'undefined') options.method = 'POST';
  if (typeof options.headers === 'undefined') options.headers = {};
  if (typeof options.headers['odata-version'] !== 'undefined')
    options.headers['odata-version'] = '4.0';
  if (typeof options.headers['content-type'] !== 'undefined')
    options.headers['content-type'] = 'application/json; odata.metadata=minimal';
  if (typeof options.headers['accept'] !== 'undefined')
    options.headers['accept'] = 'application/json';
  options.simple = false;
  options.resolveWithFullResponse = true;
  options.uri = uri;
  options.jar = true; // enable cookies
  return rp(options);
}
```

80
Appendix E

Survey Questions and Result Summary

**Q1:** How many years have you been working at IFS? (Including consultant work)

**Q2:** Which functional role corresponds to your primary work assignments?

**Q3:** How many years have you been working in your current role?

[Screenshot of specification document (see Figure 5.3) was provided for Q4]

**Q4:** It is easy to understand what business value is being delivered by the feature for which test above was written.

[Screenshot of Gherkin feature (see Listing 5.4) was provided for Q5 and Q6]

**Q5:** It is easy to understand what business value is being delivered by feature above.

**Q6:** What do you think about the readability of the feature above?

[Screenshot of a Business Flow Test (BFT) was provided for Q7]

**Q7:** What do you think about the readability of the test code above?

[Screenshot of a step definition file create with Cucumber was provided for Q8]

**Q8:** What do you think about the readability of the test code above?

[Screenshot of a BFT test result overview page was provided for Q9]

**Q9:** It is easy to map the test results above to the business goal of the application under test.

[Screenshot of a test result page for a failing BFT was provided for Q10 and Q11]

**Q10:** It is easy to know why the test has failed from the test result above.

**Q11:** It is easy to map the test result above to specifications and business goal of the application under test.

[Screenshot of feature overview page from Cucumber Reports plugin was provided for Q12]

**Q12:** It is easy to map the test results above to the business goal of the application under test.

[Screenshot of a result page for a failing feature was provided for Q13 and Q14]

**Q13:** It is easy to know why the test has failed from the test result above.
Q14: It is easy to map the test result above to specifications and business goal of the application under test.

Q15: Have you worked with Business Flow Tests (BFT) before?

[Q16 was only asked if the answer to Q15 was ‘Yes’]

Q16: How satisfied are you with Business Flow Tests (BFT)?

Q17: Have you worked with Behavior Driven Development (BDD) before?

[Q18 was only asked if the answer to Q17 was ‘Yes’]

Q18: How satisfied are you with Behavior Driven Development (BDD)?

Q19: Additional comments
<table>
<thead>
<tr>
<th>Q1</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Sample Variance</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Sum</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>11.74</td>
<td>11.00</td>
<td>7.96</td>
<td>63.43</td>
<td>23.00</td>
<td>1.00</td>
<td>223.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Q3</td>
<td>5.89</td>
<td>4.50</td>
<td>5.81</td>
<td>33.75</td>
<td>20.00</td>
<td>1.00</td>
<td>106.00</td>
<td>18.00</td>
</tr>
<tr>
<td>Q4</td>
<td>3.68</td>
<td>4.00</td>
<td>1.06</td>
<td>1.12</td>
<td>5.00</td>
<td>1.00</td>
<td>70.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q5</td>
<td>4.00</td>
<td>4.00</td>
<td>0.58</td>
<td>0.33</td>
<td>5.00</td>
<td>3.00</td>
<td>76.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q6</td>
<td>4.26</td>
<td>4.00</td>
<td>0.87</td>
<td>0.76</td>
<td>5.00</td>
<td>2.00</td>
<td>81.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q7</td>
<td>2.74</td>
<td>3.00</td>
<td>1.15</td>
<td>1.32</td>
<td>5.00</td>
<td>1.00</td>
<td>52.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q8</td>
<td>2.95</td>
<td>3.00</td>
<td>1.13</td>
<td>1.27</td>
<td>5.00</td>
<td>1.00</td>
<td>56.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q9</td>
<td>2.58</td>
<td>2.00</td>
<td>1.30</td>
<td>1.70</td>
<td>5.00</td>
<td>1.00</td>
<td>49.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q10</td>
<td>2.63</td>
<td>2.00</td>
<td>1.21</td>
<td>1.47</td>
<td>5.00</td>
<td>1.00</td>
<td>50.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q11</td>
<td>2.50</td>
<td>2.00</td>
<td>1.15</td>
<td>1.32</td>
<td>5.00</td>
<td>1.00</td>
<td>45.00</td>
<td>18.00</td>
</tr>
<tr>
<td>Q12</td>
<td>3.53</td>
<td>3.00</td>
<td>1.12</td>
<td>1.26</td>
<td>5.00</td>
<td>2.00</td>
<td>67.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q13</td>
<td>3.67</td>
<td>4.00</td>
<td>1.24</td>
<td>1.53</td>
<td>5.00</td>
<td>1.00</td>
<td>66.00</td>
<td>18.00</td>
</tr>
<tr>
<td>Q14</td>
<td>3.79</td>
<td>4.00</td>
<td>0.92</td>
<td>0.84</td>
<td>5.00</td>
<td>2.00</td>
<td>72.00</td>
<td>19.00</td>
</tr>
<tr>
<td>Q15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Q16</td>
<td>3.67</td>
<td>4.00</td>
<td>0.50</td>
<td>0.25</td>
<td>4.00</td>
<td>3.00</td>
<td>33.00</td>
<td>9.00</td>
</tr>
<tr>
<td>Q17</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Q18</td>
<td>4.00</td>
<td>4.00</td>
<td>1.00</td>
<td>1.00</td>
<td>5.00</td>
<td>3.00</td>
<td>12.00</td>
<td>3.00</td>
</tr>
<tr>
<td>Q19</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Bibliography


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Uphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Payman Delshad