The Impact of Test Automation on Software Testers

Caoimh O'Broin

A dissertation submitted to the University of Dublin
in partial fulfilment of the requirements for the degree of
MSc in Management of Information Systems
1st September 2015

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work, and has not been submitted as an exercise for a degree at this or any other university. I further declare that this research has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics

Signed: _____

Caoimh O'Broin

1st September 2015

Permission To Lend And/Or Copy

I agree that the School of Computer Science and Statistics, Trinity College may lend or copy this dissertation upon request.

Signed: _____

Caoimh O'Broin

1st September 2015

Acknowledgements

I would like thank my Supervisor Diana Wilson for her help and support.

A note of thanks to all the people involved in interviews and assisting with making data available at the Cloud Services Company where the research took place.

Also, I would like to say a very special thank you to Aisling, Dylan and Matt for all their patience, love and support over the last two years.

Abstract

This research examines the effect of test automation on software testers. Over time testing has carved out a separate discipline distinct from software development. Software has become increasingly more complex and distributed. The pressure to release into production multiple times a day has raised the strategic importance of test automation. The history and background of testing are considered in light of a critical analysis of the current body of research. Using both qualitative and quantitative methods, a picture emerges that is in contrast to the development strategy being promoted by the Cloud Services Company where the semi-structured interviews took place. After analysis of the data, the fact that a lot of testing carried out is still manual was unexpected and means that manual testers are still required. However, the problems that are preventing automation being more successful and widely implemented are solvable and can be addressed in the next five years or less. The conclusion to this research suggests that people currently in manual software testing are in danger of being replaced. To survive in the software industry, they will need to retrain and learn to code. Otherwise, their positions will become obsolete.

TABLE OF CONTENTS

List of Tables and Figures

Tables

Figures

Abbreviations / Definitions

CSC	Cloud Services Company - refers to the company studied in the research
TDD	Test Driven Development
BDD	Behavioural Driven Design
ERP	Enterprise Resource Planning
Legacy	Original customer system
MED	Medium Enterprise Division
QA	Quality Assurance
Scala	A functional programming language
Jira	Task and bug management system from Atlassian
Bamboo	A continuous integration build server from Atlassian
False Positive	A test reports a pass when it may actually be doing nothing or actually failing
Hot Patch	The application of applying software fixes without shutting down the system

1. INTRODUCTION

1.1 Context And Rationale For The Study

“If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization” (G.Weinberg, 1998).

Software continues to evolve and spread into every aspect of daily living. The discipline of software engineering is tasked with releasing a combination of code that fuels increasing parts of human life (Kitchin and Dodge, 2011). Whether that software executes its intended functionality is the concern of software testing and forms an essential part of the development process. Software testing is the part of software engineering that is concerned with the quality of the delivered product (Page et al., 2010).

There has been little agreement on an all-encompassing vision for software testing. Conceiving a unified theory of testing is one of the longstanding dreams of the discipline (Bertoloni, 2007). Software testing is not an industry at the forefront of people’s minds. As a practice, it began within the structure of development. Awareness of the importance of software quality grew and with it the amount of people and time dedicated to testing steadily increased from the late 1970s (Grier, 2011). The automation of testing activities has been a part of the story from the earliest days. However, it is only in more recent years that automation’s effectiveness has become more successful. While there has been extensive academic research carried out in relation to testing and automation, little attention has been given to the effect advances in test automation would have on software testers themselves. Deak (2012) notes that there is still a demand for more data on how testing is actually performed in industry.

Investigating how test automation will affect testers over the coming years is the subject of this research. The software tester is the invisible force behind many successful software products (Bach, 2002). The industry is worth an estimated 30 billion dollars and steadily rising (V. S., Nadar, 2015). Every piece of code that was produced had to be tested somehow. In the early days that may have simply meant the first testers were actually the first users or the developers themselves, as debugging was synonymous with testing (Hetzel, 1988). Software testing is an expensive activity, but corporations realize that the price incurred by undetected bugs can be much greater (Taipale et al., 2011).

1.2 Background

Every aspect of daily human life today interacts with and is infused by software. It is shaping our world and is a subject that matters (Kitchin and Dodge, 2011). Software has changed how we capture and analyse data, and we are now well within the era of Big Data. Many of the impacts of software in a daily routine are important but have become so domesticated as to disappear (Kitchin and Dodge, 2011). People generally do not marvel at the magic happening behind the scenes when they withdraw money from an ATM, they just expect it to work. So software forms an invisible foundation to human life today and usually people only become aware of software when it stops functioning (Kithcin and Dodge, 2011). When the quality of software systems results in failures, there can be catastrophic results.

Table 1.1 - Recent Cases of Software Failure

Year	**Issue Description**	**Effect of Bug**
2014	Heartbleed Bug	Exposed details of 4.5 million patients in US Health Systems.
2012	Operation Olympic Games	Cyber attack against Iran nuclear facilities that went public due to a bug in the code.
2012	JP Morgan Trading Bug	A $2 Billion dollar trading loss due to a lack of testing of a new model.
2012	Knight Capital Bug	A $440 million trading loss due to buggy and untested software.

Largely users and consumers have grown accustomed to poor quality software, and most tend to blame themselves rather than finding fault with an application or service (Lanier, 2010).The severity of software bugs can run from minor interface flaws on a website that requires the user to find a workaround to catastrophic events that causes the loss of human life. Estimating the expense of software bugs is complicated but the US National Institute of Standards and Technology estimates that the overall cost is over $60 million annually (Zhivich and Cunningham 2015).

These dramatic examples of software failure can make software bugs into news headlines. However, most of the time issues of software quality remain firmly below the radar of the average user. Software testing has grown from a reactive development activity into a $50 billion industry (Bertolino, 2007). Despite the overall lack of visibility software testing may have, it is a vitally important part of getting working software into systems and ultimately in front of users. The pace of technology adoption has been globally speeding up (McGrath, 2013). In the graph below Michael Felton shows the penetration levels in US households of devices. Cell phones took less than five years to reach levels that the telephone took decades to reach.

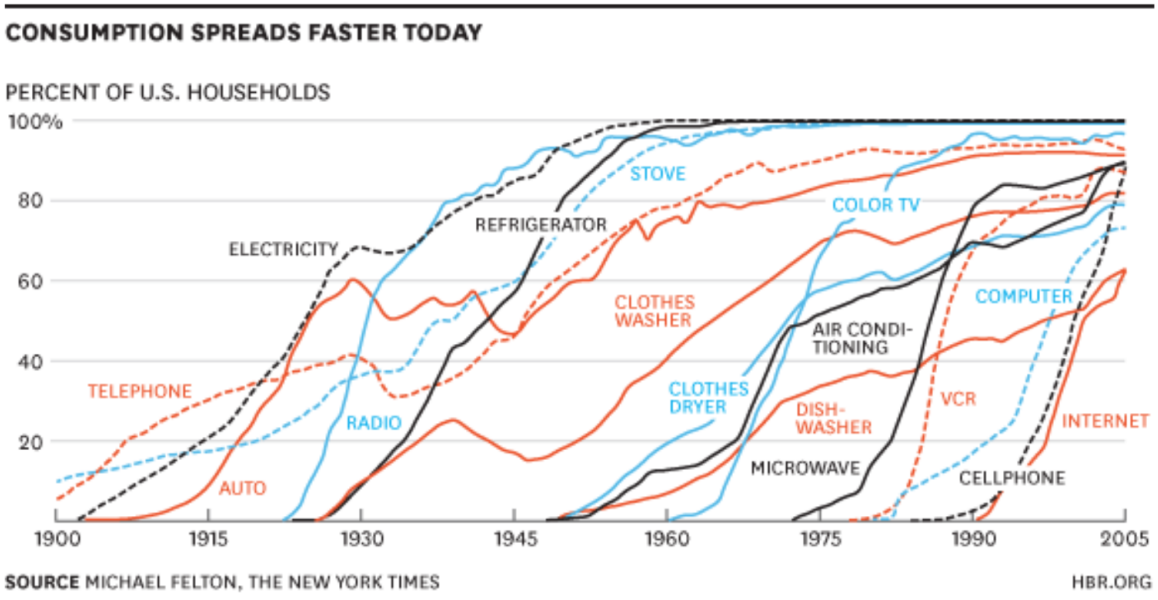

Figure 1.1: Technology Adoption Over Time - Michael Felton, New York Times

Software testing is a difficult term to define, Bertolini notes that software testing is still largely ad hoc, expensive, and unpredictably effective and the term itself can refer to a variety of activities in relation to software development (Bertoloni, 2007).

This study will investigate what effect software testing and in particular automated software testing is having on the people working in the test industry. There has been much research done on testing and automation (Deak, A., and Stalhane, T. 2013). However, the research question highlights a gap in the existing body of academic research. Bertolino (2007) notes that the majority of testing literature is focused on functionality testing.

1.3 Research Question

The research question is what effect will advances in software test automation have on software testers over the coming years. For the testers who have been manually performing functional testing what will happen to their roles if automation becomes more effective and pervasive in software production and delivery?

1.4 Why And To Whom Is This Study Important

This research will be of interest to testers who have mainly been working in a manual functional role. There is a danger for testers to become so busy in the detail of their day to day tasks that they lose sight of what is coming at them in the short term. The hope of this research is that the findings may enable testers to prepare for the significant changes coming in the software industry in relation to automation. Any person involved in the delivery of software may be interested in the findings of this study, including development and test managers, products owners and business leaders in the software sector.

1.5 Scope

This study is an investigation to see what effect advances in test automation may have on software testers. A case study approach was taken, focusing on a cloud services company. Semi-structured interviews and analysis of secondary data took place at the European headquarters of this organisation in central Dublin. Interviews with eight staff members from a range of teams and projects were undertaken to try to discover what the true state of testing currently is and where testing will go in the next five years.

1.6 Timeframe

The research for this project was conducted from January to August 2015. Ethics Approval was received in May 2015, and the interviews were carried out during late May and early June 2015.

Table 1.2 - Activities Related to Research

Timeframe	*Activity*
Early November	Ethics approval sought from CSC
Mid November	Research question proposed and accepted
December	Literature review began
April	Ethics Approval submitted to Trinity College Ethics Committee
May	Ethics Approval received
Late May - June	Interviews carried out on site at CSC
June - August	Analysis and writing

1.7 Roadmap Of Chapters

Chapter 1 Introduction: This chapter gives an introduction and background to the topic of software testing. The research question is introduced, and the context for the study is set.

Chapter 2 Literature Review: An overview of the literature relevant to the research question is looked at. The body of work is analysed through sections relating to the development of software testing as a discipline up to where the industry is currently today.

Chapter 3 Methodology and Fieldwork: In this chapter the research methodology used in this project is described and explained in the context of the literature reviewed. The other methods considered are also discussed.

Chapter 4 Findings and Analysis: Here the findings of the research are presented and discussed detailing the analysis of the primary and secondary sources.

Chapter 5 Conclusions and Future Work: The claims of the research project are discussed and the ways the research question has been answered. Limitations of the study are outlined, and a number possible directions for future research outlined in this final chapter.

2. LITERATURE REVIEW

2.1 History Of Software Testing

The origins of software testing as a practice are rooted in the early days of software development. At this time in the late 1960s and into the early 1970s the activity recognised as testing was called debugging (Shapiro, 1997). A programmer wrote some code, installed it somewhere to see if it ran. If the program crashed, the programmer would try to debug the problem to try and understand where it failed, initiate a fix and try again.

In 1968, a group of engineers and computer scientists met at Garmisch in Germany. This was the first North Atlantic Treaty Organization (NATO) conference. It was held in response to a software crisis that was emerging at the time. The phrase "software crisis" was coined to encapsulate the problems of complexity and poor quality (Grier, 2011). At the conference, the notion of software being an engineering discipline was debated. Unlike other forms of engineering, there were no physical products being produced just an abstract notion of a software structure. The second NATO conference the following year was held in Rome (Shapiro, 1997). While there was little agreement on a way forward for the emerging discipline, there was a lot of energetic debate amongst the participants. There were calls for more formal methods of inspection and validation (Grier, 2011).

During the late 1970s, ideas about what a reliable system may look like were being debated and theories introduced. This period marks the birth of the literature on software testing (Bertolino, 2007) and then in 1972 the first testing symposium took place at the University of North Carolina (Grier, 2011). The proceedings of the symposium were published as the book Program Test Methods by Wiliam Hetzel. So there was a sense of a slowly dawning enlightenment amongst developers and managers that testing code as a task was just as sophisticated and challenging as developing it.

Different approaches to 'doing' testing were being proposed - there were voices that mathematical theorems could be used to guarantee correctness (Grier,2011). The problems of complexity that software engineering had been battling with were now being seen in testing and the debate about the best method to achieve a high degree of product quality.

Susan Gerhart and John Goodenough can be seen as having developed the basic concepts that formed software testing in the mid-1970s. Their theory of software errors had two rules – one was to determine when enough testing had been done and the second was a measure of remaining errors in the code (Grier, 2011). Their work formed the foundation of the path that led to testing being seen in the 1980s as less a task for error detection and more to a form of quality engineering in a broader sense (Bertolino, 2007).

In 1985 at the International Conference on Software Engineering, Britain's then Minister of State for Industry and Information Technology, Geoffrey Pattie stated, "to put it very bluntly too much-delivered software is still unsatisfactory. It is still too often delivered late, costs more than expected, sometimes fails to work in the way required and quite often consumes excessive resources in what is euphemistically called maintenance" (Shapiro, 1997). As costs, complexity and pressure mounted during the mid and late 1980s on large software projects IT managers and business leaders became increasingly interested in software quality and the methods of software testing (Hetzel, 1988).

Table 2.1 - Pivotal Points in History of Software Testing (Jorris Meets)

Year	**Testing Event**
1958	Gerald Weinberg forms the first test team on Project Mercury the first spaceflight program
1961	Computer Programming Fundamentals - by Weinberg contains a chapter on testing
1972	First symposium on testing - as mentioned above - William Hetzel
1975	Toward a Theory of Test Data Selection - John Goodenough and Susan Gerhart
1979	The Art of Software Testing - Glenford Myers - first book entirely dedicated to software testing
1983	IEEE 829 published - first standard for software test documentation specifies the form of test documentation to be used in eight defined phases of testing.

1988	Exploratory Testing - the phrase is introduced by Cem Kaner in his book Testing Computer Software
1992	STAR Conference - first Software Testing Analysis and Review (STAR) Conference takes place in Las Vegas
1998	ISEB - First Software Testing Certificate
2000	Professional Tester Magazine launched
2003	Association for Software Testers founded
2004	Selenium Framework developed by Jason Huggins
2006	First GTAC held - Google Test Automation Conference in London
2008	uTest Launched - Crowd source testing
2011	Test is Dead - Alberto Savoia - in his keynote address at 2011 GTAC conference states that traditional testing is dead.

In this section, a brief history of testing has been presented. Coming out of software engineering software testing became a discipline in its own right with conferences, specific testing jobs, books and certifications. However, the most recent trends appear to suggest that a period of transition is occurring for software testing. That is the change from the traditional view of testing to a more continuous, automated and developer focused vision of testing and quality.

2.1.2 Testing Industry And Employment

The Irish software sector has been driven since the early 1970s by government policy making the country attractive to foreign direct investment. Many global corporations have significant presence in Ireland - Microsoft, Google, Linkedin, Facebook, Workday, America OnLine, EMC amongst others. However, over 80% of software companies in Ireland are Irish owned (Landscape Study, 2014). There has been a 39% growth in people employed in software development in indigenous companies and 23% growth for multinational companies (Fitzgerald, 2014). The shortage of suitably qualified Irish candidates for software jobs has forced many companies to inward migration somewhere between 40 - 55% of software jobs are filled in this way (Andreosso-O'Callaghan et al., 2015).

The software testing industry has been referred to as IT's invisible giant (MicroFocus, 2011). The testing tools industry alone is worth an estimated $2 billion a year, while the testing industry itself is worth $30 billion and rising (MicroFocus, 2011). In an industry where only 32% of projects are successful and over half of all development efforts are wasted, any potential for gain is of great importance (MicroFocus, 2011).

The MicroFocus report (2011) asserts that there will be a growing shortage of software testers over the coming years as the software industry continues to expand. However, this is not clear from the evidence of this study. Software is continuing to become more pervasive but this fact alone does not mean that more testers will be needed. If test automation can become more sophisticated and successful, in that case, another possibility is that less testers will be required for an expanding software industry.

IT forms part of the Irish Government's Action Plan for Jobs – 2015. In it the areas highlighted for IT are in Internet of Things (IoT) and Big Data. There is a presumption that the only way employment in IT is going to go is up. IoT is seen as critical to Ireland's economic future competitiveness (Action Plan for Jobs, 2015). The government has a task force on Big Data and this group will be looking at appropriate responses to IoT in Ireland in order to stimulate enterprise and job creation (Action Plan for Jobs, 2015). The problem with this strategy is that IoT may be one of the future mechanisms that cause drastic reductions in job numbers and not in fact the stimulus for job creation. Frey and Osborne (2013) examined a range of professions to see how susceptible they were to computerisation. They looked at seven hundred and two detailed professions. According to their estimates, 47% of total US employment is in the high-risk category for computerisation (Frey and Osborne, 2013).

The World Quality Report is a yearly report produced by the Capgemini Group. In the light of the fact that it is produced by one of the largest test consultancies (Sogetti owned by Capgemini), the findings need to be parsed with some degree of scepticism. Over fifteen hundred IT executives involved with an involvement in software testing were surveyed from around the world. There were some expected patterns of a slight recovery in confidence in IT spend after the great recession from 2008. QA budgets continued to increase from 18% in 2012 to 26% in 2014. Agile testing leads the way as the most important trend, as well as cloud based testing solutions. However, there has been a decrease in the percentage of companies sending work offshore to countries like India,

down from 28% in 2013 to 21% in 2014. The World Quality Report interprets this as a signal that testing activities are being brought back in-house. This could also signal that more testing is being done by developers as more automation is implemented.

In this section the idea that software is an integral part of the Irish economy has been described. In addition Government policy is focused on trying to maximise job potential from the latest technology trends in IoT and big data. Testing as an industry is a significant business globally in its own right and a large employer. The reason for illustrating these points is to set the context for one of the conclusions of this study. The idea that an expanding software industry will continue to require more people is not as clear-cut as it may seem. If test automation means that over time less and less people are going to be employed as testers, then this runs counter to popular opinion of the software industry and calls into question the soundness of the Irish Government's policy in promoting areas of technology that may end up replacing workers as opposed to creating more jobs for them.

2.2 Types Of Testing

Testing can be broadly broken down into two main approaches - manual testing and automated testing. Before considering each of these broad areas in turn, a definition of software testing is discussed.

2.2.1 What Is Software Testing?

Glenford Myers in his seminal book on software quality defined testing as the process of executing a program with the intent of finding errors (Art of Software Testing, 1998). The International Software Testing Qualifications Board (ISTQB) defines testing as "the process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects" (Morgan et al., 2010).

2.2.2 Manual Testing

Manual testing occurs when a human tester loads or installs or access the application or service under test. This type of testing can be scripted, that is following a predefined

series of steps in a document. The traditional workflow a manual tester may follow on a non-agile project would be to review the requirements or functional specification document. From this analysis, the tester would highlight areas that need to be tested. If there is an early prototype of the software available, then they might do an initial exploration in light of the information from the specification document. Then a series of tests could be written in a document relating to the specific stages or areas of the functional document. When the testing phase of the project starts the progress of testing can be measured by how many of the tests have been run, passed or failed, and the number of bugs can be reported. This is what a traditional model of testing might look like.

Alternatively this testing can be exploratory in nature. Cem Kaner first used the phrase in 1988 in his book *Testing Computer Software* and refers to a combination of techniques that an experienced tester can employ based on an assessment of a given context and then selecting the right tool or technique. While exploratory testing can be scripted, usually the tester drives the session using their own previous experience and knowledge how to break software and find bugs. In many ways testing is about trying to make a list of things about the product that we do not yet know (Armour, 2004).

James Whittaker in his book Exploratory Software Testing (2009) describes an approach to this style of testing using the touring metaphor. The idea is that a tester uses the application in the way a person might take a tour of a city. So on one journey called the Garbage Collection Tour, the tester would concentrate on deleting any data that they could during the session. In the Landmark Tour, you concentrate only on the main features the customer is interested in. The Back Alley tour looks at those features of the product that are not used that heavily in production.

Whittaker notes that the test industry has not been good at manual testing, giving rise to its reputation as the ugly stepchild of software engineering (Whittaker, 2009). Whittaker advocates a future for testing where testing does involve careful preparation but also leaves room for the creativity of an experienced tester to make informed judgments based on the context of the task. This point is reinforced by Itkonen et al. (2009) in a study on manual testing practices. Itkonen at al. identifies twenty two testing practices used. They observed that experienced testers use numerous techniques to find bugs and do not simply rely on test documentation.

In a 2014 research survey Kanij et al. showed that testers value testing tools to aid them in testing. However, other factors such as intelligence, dedication, communication skills and motivation were seen as crucial for success as a tester.
Our survey results indicate that software testers do indeed think that testing-specific tools and techniques are an important part of being a good software tester.
Factors like intelligence, dedication, interpersonal skills, and motivation were viewed as crucial in being an effective software tester.

There are numerous ways to approach test planning for a product. Pure functional testing is the approach most associated with the term software testing (Page et al., 2008). However, there are many other ways to think about how to test a piece of software, an app or a service. Crispin and Gregory (2014) advocate the use of mind mapping to try to capture thoughts about the different types and approaches that can be taken with software testing. This is a useful technique that can help clarify the complex landscape of testing in a continuous delivery environment.

As well as functional tests the product could be tested in a non-functional way to include performance, security, load, stress and soak tests amongst others:

- Unit testing - testing at the smallest level
- Integration testing - combining elements and testing together to make sure they integrate correctly
- System testing - testing the system as a whole in an end to end fashion to ensure integrity of the entire system
- Resilience testing - testing that the system will hold up when a required service goes down
- Soak testing - immersing the systems in a period of sustained test over a week or a month on defined schedule
- Load testing - initiating heavy load onto the system to see how it reacts
- Performance testing - setting a baseline and measuring how the system responds in terms of throughput - can be measured in a number of different ways.

Equivalence Class Partitioning (ECP) - this technique can be useful where there are a large number of input variables and testing every combination is not feasible given time and resource constraints. Initially, an analysis of the input data set should be carried out to determine the scale of the testing landscape. These can then be mapped into a table

and determined into valid classes. For example, if a project had a storage solution to test, and there were multiple services that could run on each storage unit, and each unit could run on multiple versions of Windows Server and use multiple versions of Microsoft SQL, then the tester might draw up a table with all the options and map a subset of class combinations to test. These class combinations must be reviewed and make sense in light of customer use and project risk (Page et al., 2008). Boundary Value Analysis (BVA) is a technique is based on the premise that errors tend to occur around the boundaries of values, so if a given range of values goes from 1 to 10 it is likely bugs may occur just inside or just outside that ranges of values. Regression testing is running existing tests on new versions of software to ensure nothing has been broken and that the fix is in place. Abhilasha (2013) notes that regression testing saves time because only the modified part is tested.

As illustrated in figure 2.2 below, using test quadrants can help frame the problem into a useful context to help move towards implementing the right kind of test in a given situation.

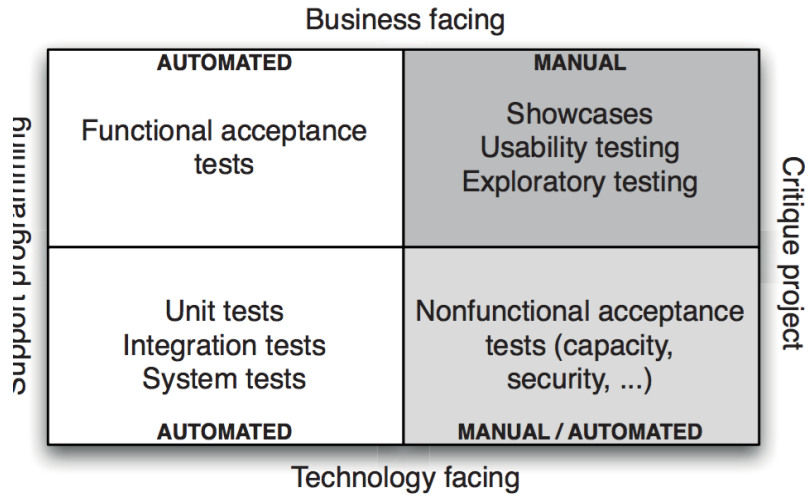

Figure 2.1 - Testing Quadrants (Crispin and Gregory, 2009)

The quadrants can help inform test estimation, which Abhilasha and Sharma (2013) note can take over half of the development costs of a project.

2.2.3 Test Cases

Test cases are the structure in which testers describe what the test will do. Test cases are used in both manual and automated testing. Writing test cases for a large

development project is a time-consuming and important task. If the test case is written poorly then no matter how many times the test is run, be it manual or automated there will be little value in it. A poor test case may be one that tests only the happy path, that does not think of all the other permutations of the situation under test, one that puts the test focus on the wrong part of the system, one that compares invalid values and may lead to false results.

Fewster and Graham (2012) note that a good test case should primarily detect defects and should aim to cause as little maintenance as possible. While the defect and maintenance points Fewster and Graham (2012) make are sound, having test cases that try and do multiple tests can cause needless complexity. The downside of this complexity is seen when tests fail and there are multiple potential points of failure. If the test case calls out one main task to test then when it fails, the process of finding that error becomes easier. Writing efficient and powerful test cases is a skill that comes with training and experience. Hemmati (2015) highlights that identifying the right tests to be selected from a large test suite and run in regression tests is a crucial test task. If this selection is done without the right information about what the strategy of a particular run of tests is trying to achieve, the effectivness of the test run will be decreased. The area of test case generation has attracted a lot of research attention, and there are numerous ways to approach how to produce test cases. Anand et al. (2013) has produced a survey of the most prominent techniques for automation test case generation. Whether test cases are needed at all is a point that has been looked at (Itkonen et al., 2014) in comparison to using an exploratory style approach.

2.3 Test Automation

“Asking experts to do boring and repetitive and yet technically demanding task is the most certain way of ensuring human error that we can think of “ (Humble and Farley, 2009).

The problems of software quality that were clearly seen in the 1980s and 1990s focused a lot of IT manager’s minds on the task of testing. The belief that automated testing would solve many of the software quality issues like increasing test coverage and an overall improvement in the quality of testing (Mulder and Whyte, 2013). However, the history of test automation is full of failed test automation projects.

What is test automation? Collins (2012) explains that test automation means to automate

all parts of the test effort from test script development and execution, to verification of requirements and the use of tools. Bertolino (2007) asserts that test automation is necessary to keep pace with the growing scale and complexity of software. According to Fewster and Graham (2012) trying to jump in and automate everything will result in a lot of activity but not much value. Automation projects need to be approached with the same level of planning and governance as any other project (Crispin and Gregory, 2014). Fewster and Graham (2012) advocate that designing automated tests is a demands a different approach than creating manual tests. Test automation should concentrate on the types of tests that a person cannot do, they contend. For example running thousands of authentication requests overnight on a web server. Bach et al. (2001) maintain that it makes no sense to assert that an automated test could replicate what a human tester could do. There are however situations where manual test cases can be transformed into automated tests.

Taking this a step further, Alegroth (2013) conducted a study at SAAB that researched a test project that transitioned relatively successfully from manual testing to an automated testing tool. The benefits included increased coverage and shorter time for test runs to complete, a point also noted by Fewster and Graham (2012). Tests that previously took hours to complete could in theory now be run in minutes once set up correctly in an automated framework or tool. These tests are then repeatable, running exactly as written each time. Once a manual test is run once its value is lost, and any code changes after that mean the test will need to be run again. Added to this is the human factor, which may mean that mistakes creep in during the execution of the test. Automation can relieve testers of boring, repetitive work and free them to do more important testing (Fewster and Graham 2012). Overall in Alegroth's study at SAAB the results were positively in favour for the adoption of test automation. Automated test suite maintenance is a common point of failure on many automation projects (Karhu, 2009, Fewster and Graham, 2012). Changes to dependencies outside the control of the team can cause failures in the tests which can go unnoticed and when discovered may be very complicated to debug, particularly if there hasn't been adequate logging put in place.

A test automation project needs support and executive sponsorship in order to launch and then to continue to succeed. Allowing changes to be made quickly and safely, knowing that a battery of automated regression tests are running to ensure nothing has been broken in existing functionality (Page et al., 2008). A clear strategy for the test automation effort is thus required in order of the project to be successful. Questions that need to be

addressed before beginning an automation project may include:

- What are the objectives of the automation project?
- What language or tool or framework will be used - what has informed the decision?
- Where will the tests run - are they going to be part of an automated build - if so how long will they take and what resources are required?
- What coverage of the product is planned - is it 100% automation or just to concentrate on a specific area of functionality?
- What type of tests are going to go into the framework - are they straightforward functional tests or performance tests or security tests?
- What sorts of training is needed, how will results be reported? Underestimating the training required can hamper efforts to implement an automated testing strategy? (Fewster and Graham, 2012).
- What type of data will be used and where will it be stored and who will own curating the data, maintaining its integrity?

Test automation engineers who do not understand testing can be damaging to a project (Crispin and Gregory, 2009), they may be able to write a lot of code but without the guidance of an experienced tester they may just be adding to the technical debt of the project. Michael Feathers (2004) describes technical debt in terms of legacy code, code that does not have any test associated with it. Therefore, as more and more code is added to a project with the aim of adding functionality or more tests, this may, in fact, be simply adding to the problem. Knowing what type of tests and where to add them in a project is a difficult challenge to overcome. Testing and coding experience are both needed to ensure that a suite of tests is covering the right issues in the context of a given project. Concentrating on too narrow a type of test, for example only having functional, happy path tests and no non-functional or negative tests can lead to disappointment and failure. This often occurs because the effort and time it takes to get some one batch of positive tests running and passing means that there are no resources left to look at other types of testing (Mulder and Whyte, 2013).

Whyte and Mulder (2013) make the point that manual and automated testing have different objectives. Therefore it does not make sense to argue that automated testing could replace manual as they are performing different jobs. However, different objectives do not technically block a suite of manual tests being automated. The fact that there may be higher costs and perceived increased risks with manual testing may mean more

organisations will look at trying to reduce their manual testing and automate as much of it as possible. Kalinowski, Teixeira and Van Oppen (2007) assert that manual testing is focused on new functionality, and new bug detection and that automated testing is designed to discover regression type defects and they conclude that automated testing cannot replace manual testing. This argument does not take into account that increased levels of unit and integration automated developer testing may catch a lot of new bugs, better coding practices and tools will also mean the code that gets committed into the deployment pipeline will be less buggy. Therefore, it is entirely possible to create banks of automated tests that replicate user behaviour in an automated way. For some companies there is no alternative but to consider automated testing as the challenges and scale of the task is simply too great to be completed manually (Ali and Saha , 2012)

Automated Acceptance tests are derived from acceptance criteria from customer or business analyst. A style of writing acceptance tests that forms part of the case study at CSC is known as Behavioural Driven Design (BDD). BDD is a way of agreeing the desired behaviour of the application, defined with the customer, business and engineering team together (Wynne and Hellesoy, 2012). Once the list of desired behaviours has been agreed, a set of tests is written to check for these behaviours. Then finally the code to implement these behavioural tests is written. There are frameworks that do a lot of the setup work for free, for example Cucumber or ScalaTests. The tester or developer needs then to specify the scenarios and then write the code to make the tests pass. In this way, as the project progresses the specification for the BDD tests forms a type of living documentation (Wynne and Hellesoy, 2012). An example of a scenario may read:

Given I am an admin user
When I log into the system
And I have a certain permission
Then I expect to see a setup button on the menu

The syntax reads in a way like a strange form of English, it is known as Gherkin syntax. One of the main benefits of this approach is that it allows business analysts or customers or other non-technical people to specify scenarios that they need in the application directly within the code project (Wynne and Hellesoy, 2012). The developer or automation engineer needs then to take that scenario and write the steps in the background.

2.3.1 Conclusions on Automation

Implementing test automation in an organization is a complicated undertaking. Simply purchasing expensive tooling rarely meets all the needs and expectations of software companies in relation to quality and testing. The automation project requires a collaborative approach from management, development and testing teams (Mulder and Whyte, 2013). A clear strategy needs to be openly discussed and agreed upon with tasks and owners and clear milestones. There must be a level of quality stability in the existing test approach and system before it is automated to avoid further disappointment (Mulder and Whyte, 2013).

2.4 Testers And Developers

There are technical problems that testers face, some of which can be surmounted. There are also and in some ways more complex sociological and psychological factors that can affect a tester's ability to do their jobs. Deak (2012) notes that there are not enough studies focussed on the testers themselves or on the socio-technical environment in which testing takes place. Myers (2004) alludes to the importance of psychology in trying to understand and improve the testing process.

The nature of the work that testers do is bound to lead to conflict situations between testers and developers (Deak, 2012). Communication and collaboration are key elements of the Agile process. Zhang et al., (2014) collected and analysed fifty developer-tester conflict scenarios, and their findings indicate that conflict between the two groups falls into three main categories: Process, people and communication. Complete eradication of conflict between testers and developers is impossible, but it can be improved if managed at the source (Zhang et al., 2014). Often tension arises from a feeling that a tester does not understand in depth a particular technical issue. One way to address this is through training a tester in programming so that they can confidently speak to a developer in the same language and earn respect that way (Zhang et al., 2014).

It can occur that a team transitions to an Agile methodology and starts to practice Scrum. However, changing privately held views that testers are less than and not as capable can be a difficult and invisible task to overcome. Software developers and testers are very different from each other in terms of mindsets, goals, experiences, and perceptions of their relative importance (Cohen et al.,, 2004; Pettichord, 2000; Rothman, 2004). The role

of an individual can be very significant in either making or breaking a team or relationship (Zhang et al., 2014). James Bach and Cem Kaner recommend that testers should approach the relationship with developers with respect (Bach et al., 2001). Recognising that writing code is complicated work and understanding that programmers can take bugs personally - all these factors can ease the relationship.

The image testers carry of themselves, and their discipline is worthy of mention in the context of the relationship between test and development. Many see testing as a negative and repetitive career choice that amounts to checking others work (Deak, 2012). However, through training and acknowledgement of the importance of testing, this negative image can be overcome. A tester on a project is often the go-to person, the one source of knowledge on a broad range of issues related to the whole project (Itkonen, 2013). Leveraging this experience and knowledge to build relationships in the team and will go a long way to more productive developers.

2.5 Agile Testing

In this section, the differences between traditional testing and Agile testing approaches will be outlined and discussed. What does a traditional approach to software testing look like? The project design and specification have been completed between business owners, system architects, developers and managers. A detailed functional specification and then higher level requirements document is produced. The development team begin coding up the product using the specification as a guide. This can take many months or years in some large projects. The test team use the requirements document to write up test cases into an excel spreadsheet. Once the developers are finished with their coding, the testing phase can begin. Often this will occur later than planned, and the test team have to work under extreme pressure to find all major bugs in the code before the impending release. As the product grows in features and functionality over the course of its lifetime, the test landscape continues to increase. To try and remedy this more testers are added, the development cycle is longer and the test time is getting shorter. This scenario would be familiar to many testers who may have worked on projects prior to Agile implementation. The idea behind Agile testing is integrate testing into the team. The team own the quality even if there are dedicated test people on the team. The development cycles are now divided into sprints of perhaps two or three weeks in duration. The Scrum ceremonies form the structure so before the sprint starts the planning session would have selected the items from the backlog to be worked on in the sprint.

The estimation for each task would include time for development of the feature and also time for testing. Test automation is one of the foundation stones of Agile testing. By adding an automated test for each piece of functionality developed, every time the build is run with new changes that automated test can run and ensure no breaking changes have been committed.

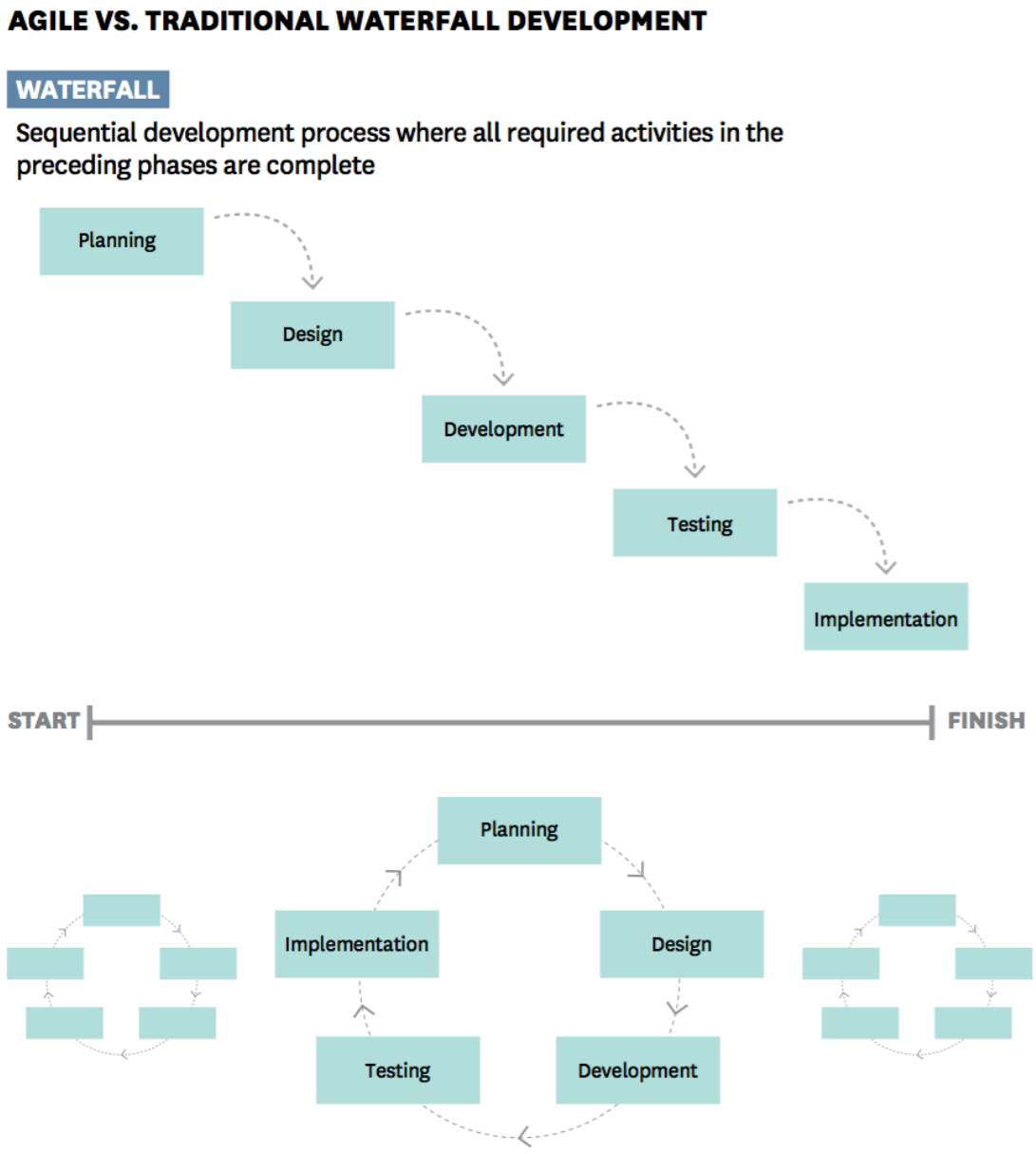

Figure 2.2 – Agile vs Traditional Waterfall Development, Harvard Business Review, 2015.

The philosophy of agile testing is that anyone on the team can take a test automation task, it does not have to be the test team. The fact that the test team is integrated with development should mean that testers are involved and invited to all meetings and design

discussions at the early stages of the project. The benefit of this is that testers can start to think about ways a proposed system could be tested. In addition, testers could make suggestions why a certain feature may not work in the proposed way based on their experience and customer knowledge.

There are many challenges in transitioning to Agile testing (Crispin and Gregory, 2014). Training testers that previously tested the product manually and now have to write automated tests. Learning programming can be a very difficult task for some people who have not coded before. This change can be a very difficult transition for some testers to make:

- There is also the cost of acquiring new tools and hiring test automation staff.
- Constant changes to code during the sprint
- Falling behind in automation tasks
- Legacy code not in a state for automation to be added - too fragile and brittle to be refactored
- Programmers with no experience in testing
- Testers falling back on manual tests when automation becomes too complex or time-consuming to implement.

Factors that need to be implemented to help Agile testing succeed:

- Close collaboration with the customer
- Short feedback loop
- Discuss each automation task with interested parties
- Whole team approach
- Use the daily-standups to ask for help when blocked (Collins, 2012).

2.6 Continuous Delivery

Continuous Delivery is a concept that has surfaced in the last number of years as a part of the evolution of processes arising from Agile implementation in software engineering teams (Humble and Farley, 2009). Continuous delivery if implemented successfully can allow teams to deploy code into production multiple times a day with peace of mind that nothing major has been broken (Crispin and Gregory, 2014). In the past, on traditional development teams in a waterfall style approach, a strange fact was that for most of the project the software being developed did not work (Humble and Farley, 2009). All the

effort and work that goes into planning testing and development carries no value until the software is in the hands of the customer. The means of getting the product to the customer is clearly a critically important piece of the software engineering picture (Humble and Farley, 2009).

Making that process of releasing software into a fast, repeatable process is the aim of continuous delivery (Humble and Farley, 2009). Every time a developer makes a change in the system a new instance of the pipeline should be initiated. A deployment pipeline is an automated version of building, testing, deploying and releasing the software (Humble and Farley, 2009).

- Developer writes code and unit tests
- Code is reviewed by peer and approved
- Code is committed, and unit and integration tests are run
- If all tests pass, the next build is initiated running in a production like environment and runs the acceptance tests
- Then once these tests are passed, the user acceptance tests (UAT) can be run
- Then the capacity or load style tests that take longer can be run at the end.

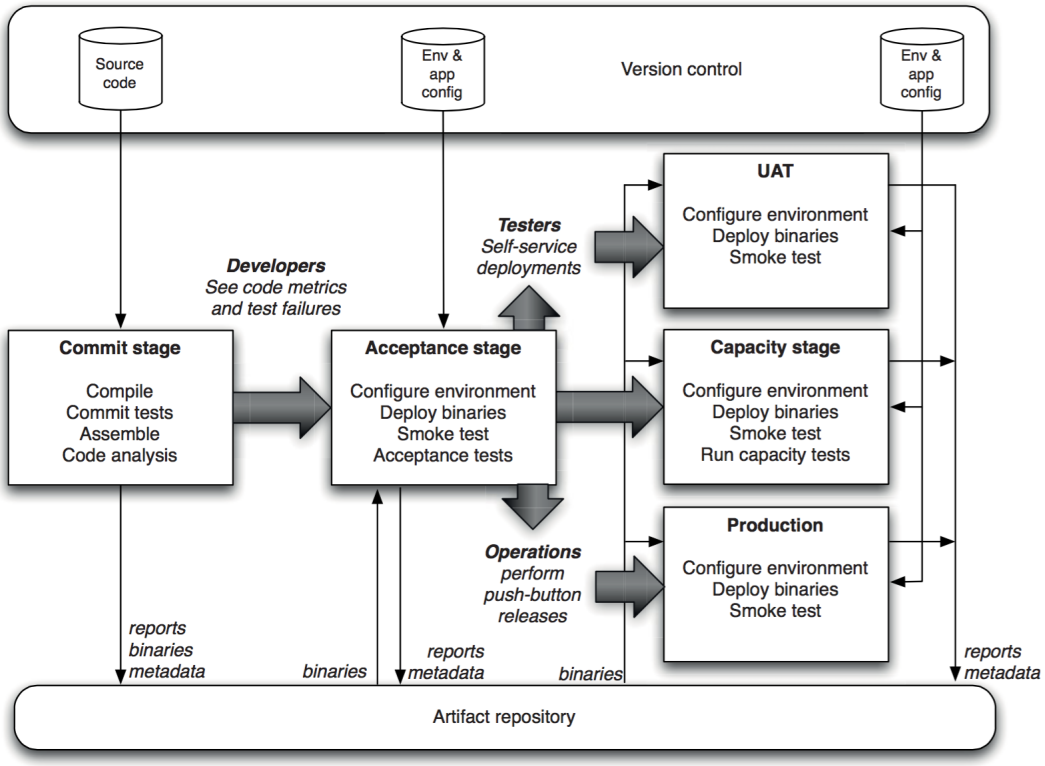

Figure 2.3 - An Example of a Continuous Delivery Pipeline (Humble and Farley, 2009)

3. METHODOLOGY AND FIELDWORK

3.1 Methodological Approaches Considered

In this chapter, the methodology and fieldwork carried out for this study will be discussed. The different approaches that were considered will be examined, and the reasons for the approach adopted will be looked at in detail. Having clarified the question to be answered and then critically assessed the existing body of research, identifying the means for data collection and analysis were the next steps to be completed in this process. The word research is commonly used in popular media to refer to a period of study or a collection of data. However for academic purposes when the word research is applied it has more specific meanings (Bryman, 2004). Research does involve data collection but simply collecting data by whatever means will not on by itself provide an answer to the research question. To understand what patterns the data may or may not be displaying a strategy and design for analysis must be employed (Saunders et al., 2012). In this way, research can be understood as a process to systematically look at a particular topic, gather data and from that data uncover some new knowledge or form a new theory (Saunders et al., 2012). Part of the research effort will involve explaining the methods used to collect the data, how the data was analysed, how the conclusions were reached and any limitations of the chosen approach (Saunders et al., 2012).

This can be a challenging part of the study given the variety of approaches that are available to researchers. With each way there is a level of anxiety that committing to a particular direction may lead down the wrong path in a given context and thus put the study in jeopardy by not achieving results in a given timeframe. This is most pertinent for part-time postgraduate students whose time and resources are limited. Therefore, this point of selection needs to be thought through carefully, all the while trying to project how the particular approach may fully answer the research question.

3.1.1 Philosophical Considerations

Given the pressure of time in part-time studies there is an understandable desire just to get on with it – that is the research effort. So with the refined research question in hand, a review of pertinent literature done the impetus to jump into data collection and analysis can be strong. However, before choices are made about surveys or interviews, time reflecting on the context and motivations for a particular approach and strategy is

necessary. The philosophical structure that holds the research is a complicated tapestry and in part reflects the way the researcher understands the world and how knowledge is developed (Saunders et al., 2012). The choice of topic, the research question and literature read and reviewed will form part of the path that leads in a particular philosophical direction, even though this may not be something that is obvious to the researcher.

Positivism – researchers adopting this approach work with an observable social reality which will result in law-like rules that govern the structures studied (Remenyi, 2002). Realism – Realism holds that reality exists independently of thought. Realism attempts to understand people's views and behaviours by trying to understand socially constructed interpretations and meanings (Saunders et al., 2012). Interpretivism – researchers in this school would find the idea of law-like generalisations as too constricting for understanding the complex world of business organisations (Saunders et al., 2012). Interpretivists are not overly concerned with making theories generalisable – with the rate of change of business strategy approaches and the every changing nature of the marketplace – the value of
all-encompassing theories can be called into question. Interpretism can help the researcher uncover the complex reality of a situation through analysis of the details (Remenyi, 2002).

Throughout the process of selecting a strategy and design for this study, the research question served as a guiding light to return to when confusion arose about the correct direction to take. The first approach considered was the deductive strategy, with a large-scale questionnaire survey as the data collection means. This could have been distributed through a network of some 100 testers on LinkedIn. The benefit here was a definable means of access to a large number of potential respondents from a varied background (Remenyi and Money, 2004). However, going from the position of testing a theory about the future of software testing in the light of test automation advances seemed from the outset problematic. Having drafted some potential survey questions that could be asked of testers it became clear that an approach that would facilitate an intimate and subjective portrait would potentially yield a richer and more meaningful answer to the research question. The type of standardized questions of a survey would not give enough flexibility to gain important insights. Also after more consideration of the topic of automation and what the future held it became clearer that there was no clear theory at this stage of where this would lead. In that way, an inductive approach appeared to be called for in this

case. Concentrating on a much smaller set of people but with more flexibility in the range of questions and with the potential to spend more time where necessary seemed the best approach as a way to arrive at a satisfactory answer to the research question. Allied to this was the fact that the research question was essentially seeking to uncover the effect of technology on people in a given situation. In this way, it was felt a qualitative approach that facilitated interviews with testers would be the preferable research strategy.

3.2 Approach Chosen For This Study

The overall research strategy chosen for this study was the Case Study. The methods employed for data collection were semi-structured interviews and employing Narrative Analysis for data analysis. In addition data from a 2013 survey was made available, significant internal documents and access to a bug database. A mixed methods approach was therefore adopted. Triangulation is an approach used in case studies where data from different sources and viewpoints is used to form a holistic picture (Kolhbacjer, Florian 2006). One of the main strengths of the case study is the ability to use a wide variety of evidence that can be either quantitative or qualitative or both, as in this study (Yin, 2003).

The case study follows an in-depth look at one organization, community or person (Bryman, 2004). In this instance, the case study would be focused on Cloud Services Company (CSC) the cloud services company, at its European HQ offices in central Dublin. Within the structure of the Case Study, a qualitative approach was adopted, and semi-structured interviews were selected as the best means to collect primary source data. At this point, the data from a 2013 internal survey at WD were made available for use in this study. This was a strong addition to the effort as this covered over 80 testers at CSC. Finally, bug reports from bug databases from the projects that the eight interviewees were involved in were analyzed as a measure of effectiveness between manual and automated testing. So there were three main sources of information feeding into the data collection part of this study.

3.3 Why This Was Chosen

An inductive approach appeared the best fit to work from the outside in – so without a clear theory or hypothesis to test from the start – it was clear that data would need to be collected and analyzed and then from conclusions new theories or knowledge could be drawn. Given the strong social and human element to the research question a qualitative

approach was chosen as the most appropriate to get in close to the testers and try to uncover the reality behind the world they operated in at this organization. To that end, semi-structured interviews were selected as the best means to give flexibility in style and content of questions and themes to be asked. Also, given the prior knowledge of this company there was a number of potential interview candidates who had expressed interest in the study and a tentative acceptance to take part in the study. Access to the candidates and interview rooms would be accessible on site. The availability of the survey and bug database data were welcome additions to the collection of data as part of this case study. A case study is said to be appropriate when "a how or why a question is being asked about a contemporary set of events over which the investigator has little or no control." (Yin, 1994, p.9). Yin (1994) defines a case study as an empirical inquiry that investigates a contemporary phenomenon within its real-life context. The sources of data for a case study can be multifaceted. Case studies can be used to explain, describe or explore events or phenomena in their own context (Yin 1994). An important aspect to using the case study is to understand that it is essentially about telling a story (Remenyi et al., 2002). Through the interviews, documents and data analysed a story needs to emerge about the patterns and peculiarities of this given situation. The approach of case study has been used in the context of investigating testing practices in a number of studies, Alegroth (2013) used the case study to research the transition of manual to automated testing at SAAB. In 2014, Anca Deak used a case study approach to investigate software testers' motivation in Agile versus traditional style projects. (Deak, 2014).

3.3.1 Limitations of Methodology

One of the challenges in using the Case Study approach is that in a way anything that is studied could be called a case study (Bryman, 2004. The case study tries to distinguish itself by highlighting the unique features of the particular situation (Saunders et al., 2012). In that way, there is less of an interest in developing general, all-encompassing theories. Rather the focus is on the quality of the specific theory or hypothesis in relation to the subject of the study. A number of conclusions and patterns emerged in time while concentrating on some completely unrelated task or indeed even while on vacation. This approach requires some diligence and caution. Given the familiarity of the topic and environment and also potentially the candidates for interview there is a risk that the researcher may not see some patterns or observations due to the effect of familiarity (Saunders et al., 2012). In addition, there are known limitations in using secondary data

including the fact that the data may have been collected for purposes other than the objectives of the research project and that there is no control over the integrity and quality of the data (Saunders et al., 2012).

3.4 How Was The Research Conducted

The primary data for this research was collected through semi-structured interviews with IT professionals who work at CSC the Irish office for a multinational Cloud Services company.

3.4.1 Candidate Selection

Careful consideration was given to the choice of interview candidates. Initially, the earliest idea was to select all candidates from a single project, which would be relatively easy to set up and complete. However, the main disadvantage to this approach was the risk that the focus would be too narrow, and the study would in effect be a report on one project and less a case study reflective of the whole organization. To mitigate this risk, more planning was done to select a more diverse range of interview candidates. This led to consideration of the projects the candidate were involved in and what way testing and automation was involved in each situation. The candidate's status in the organization was considered, time with the company and experience in Information Technology (IT).

After a period of investigation on the internal employee HR system (which is open to all employees – this is one of the core product offerings from CSC) where these details were available a list of eight potential candidates was drawn up. The list of potential candidates was drawn up. They were chosen based on their professional test experience and knowledge of both manual and automated testing. A mix was chosen from those starting out in their careers to those who were much more experienced. Also, candidates were from different roles with a mixture of test and development and management to give the widest possible view from this small sample. An email was sent to each to invite them to partake in the research effort by taking part in the semi-structured interviews. In the email, the research question was clearly outlined and the fact that the process was entirely voluntary and candidates could choose the leave the process at any time without any problem or cost. All eight candidates accepted the invitation.

3.4.2 Scheduling

The next step was to schedule time in people's calendars. This proved quite challenging, particularly with the management level candidates as their schedules were so busy. Also, time would be needed in between interviews to capture observation notes and capture any themes that came across. It was important to do this while the content and ideas were fresh and not rely on the same insights striking twice was listening to the playback of the audio interview. Suitable times were scheduled for the interviews that were carried out from 14th May until the 5th of June 2015.

3.4.3 Where

Large, private meeting rooms were made booked in advance. All documentation that candidates needed for Ethical information was sent prior to interview and printed copies were available in the room at the interview.

3.4.4 Interview Preparation

Prior to the first interview a pilot interview was conducted. The recording mechanism was tested, and the means of data transfer were confirmed to be working. A test interview was also trialed with a person not associated with the research. The purpose of this was to practice the pacing of questions and where and how to add in helper questions depending on the responses of the interviewee. The questions were designed into themes that would facilitate analysis afterward.

- Background and experience
- Journey into IT and in particular Testing
- Levels of contentment in current area
- Manual and Automated Testing experience
- Agile / Waterfall
- Future directions of Testing

The interviews lasted for approximately 40 minutes. The themes around which questions would form were prepared in advance. A number of broad helper questions were developed in case there was a lull or gap in the conversation. The initial few moments were spent talking about the course with a view to putting the interviewee at their ease.

Where appropriate a number of interjections were made during the interview when an interesting point or theme emerged in the conversation. When areas of specific interest were touched upon the interviewee was guided to focus on that particular area and drop down a level of detail. The flexibility of semi-structured interviews allowed this approach to work well for both interviewer and interviewee. Notes were also taken during the interviews to aid transcription.

3.4.5 Post Interview Tasks

After each of the interviews, the recording was checked and transferred onto a storage space in a private encrypted cloud. Then post interview notes were taken to capture and elaborate on any points that came up during the dialogue. Later the same day of each interview the recording was replayed to assess the quality and to take again any further notes and to jot down any strongly emerging themes.

At a later point, each interview was played back, and the main points were taken down and transcribed. This was a lengthy process of pausing and replaying to ensure that the correct points were being noted. Once the first round of transcribing was undertaken, each participant was met informally to follow up on the interview and to present the main themes arising from each conversation. The reason for doing this was to check the validity of the understanding and to facilitate the candidate to elaborate or revise their answers. Two of the eight significantly changed some of the content of their answers.

3.5 Type Of Data Gathered

Case studies use multiple sources of data, both qualitative and quantitative approaches. Qualitative approaches to analysis will be used in looking at the data from semi-structured interviews. As there were a number of potential candidates available at the researcher's organisation, this was the approach used. The detailed analysis of the data arising from semi-structured interviews consists in building a thematic narrative of each session and uncovering patterns (Remenyi, 2013). These patterns are then built upon to form a final narrative from which key theories can be made. The secondary data was sourced from a survey carried out at CSC in 2014 amongst 83 testers. The results had been analysed, and that data was made available and appeared to be a good fit for the research topic.In addition, a number of documents that were authored by test professionals and management at CSC were made available as well. Finally, an analysis of a bug database

at CSC was made available to gain an understanding of the difference in bug rates arising from manual and automated testing. Database records from bug bases will be appraised over a period to try to gain a deeper understanding of the patterns, if any, emerging from the changes in test approach from manual to automated

3.6 Ethics Approval

As soon as the potential for interviews became a potential approach an application was made to see if this would be at all possible at the company selected. This application was initiated by an email to the Legal and Compliance Department at CSC and there followed a series of meetings and a conference call with a team in the US to agree the terms and usage of any data from the company in the study. An application for permission to interview employees at the company was made in November 2014 and was quickly granted. The preparation for the Ethical Approval application helped to shape the initial set of questions and themes that would form the structure for the interview conversations. Each candidate was made clearly aware that their involvement was entirely voluntary and that they were free to leave the process at any time. In addition, assurances were made that all data was confidential and would only be used in the research study, that no names would be used to identify any participant and that all data would be destroyed in October 2015.

3.7 Problems And Lessons Learned From The Research Process

Employing semi-structured interviews proved more complicated than first anticipated. The importance of interviewer appearance, the first few moments of conversation of the interview, developing rapport and trust within a very short period of time with the person came out as critically important elements in executing successful interviews. Getting the timing of the probing questions right was difficult, as was resisting the temptation to provide answers or approval and so facilitating a leading approach on a certain topic of interest. Limitations with approaches implemented in this study:

- Shortcomings with semi-structured interviews - bias - generalisations - lack of control over the data
- The interviewer's ability to convey meaning of questions in a standardised way
- Motivation of the interviewees in participating may affect their responses
- Ethical considerations - had to get approval from company to undertake the study
- Applied to Trinity Ethics Committee

4. FINDINGS AND ANALYSIS

In this chapter, the findings and analysis of the study will be presented. The three main sources of data were primary source data from semi-structured interviews, data from an internal survey with over 80 software testers and an analysis of bug databases related to the four main projects which were the focus of the interviews. By means of an introduction to the case study a company profile of CSC follows. Understanding a little of the history and context of CSC will assist in analysing and understanding the data. This is part of building the story of the case study (Bannister and Remenyi, 2002).

4.1 CSC Company Profile

CSC was founded in 2005 as a cloud services company focused on the Human Capital Management (HCM) sector. HCM refers to all human resource related tasks that large organisations have to undertake throughout the lifecycle of employees and applicants within their system. The founders of CSC previously owned a successful software company that operated in the same market but not as a cloud solution, rather as a traditional installed software offering. Their previous company was bought in a hostile corporate takeover. Over six thousand people lost their jobs as a result of the takeover. Out of these ashes, CSC was formed as a cloud company, and the mission was to reimagine the HCM sector and become the number one provider.

This area of the IT industry is largely dominated by two main traditional Enterprise Resource Providers (ERP) at the enterprise level. The type of customers in this market would typically have 20,000 - 100,000 plus employees. CSC wanted to leverage the cloud to enable it to offer implementation cost and time savings. Many large enterprise-scale customers of the existing providers were well experienced in paying millions of dollars on multi-year consultant engagements to upgrade or expand the ERP system across their organisations. CSC wanted to put the customer first and in effect remove the need for the consultant middleman. As as example of a case of the costs and complexity of a traditional ERP project, consider Plenia Locatel, the Venezuelan pharmacy chain that spent millions on an SAP implementation that took years to complete and still was bug ridden and difficult to use (Gibson and Levy, 2012).

So CSC had the chance to do things right from the start. Nearly all the early employees came from the original company, so culturally there was a mix of old and new ways of

doing business. The test department grew from an initial team of just three Quality Assurance (QA) people to today where there are over three hundred full-time QA staff. The early days of software testing at CSC were executed in an entirely manual way. The internal IT projects were arranged along a traditional waterfall model approach. Although the mechanism for updating customers was cutting edge the development and test methodologies were traditional and initially at least avoided an Agile approach for the engineering teams.

As this pattern continued it became more deeply ingrained in the company development culture. Silos formed between teams and between QA and development. There was a strong emphasis on software delivery whereby all customers were kept on a single version of the software. CSC decided not to maintain multiple versions so as to simplify the maintenance and upgrade cycles. All customers received two main software upgrades each year, in addition to any necessary hot patches. In effect, CSC customers pay a fixed price and get the one version with all functionality. As sales success followed the company was silently building up a mountain of technical debt (Feathers, 2004).

4.2 Semi-Structured Interviews

As discussed in the Methodology and Fieldwork chapter careful consideration was given to the selection of interview candidates at CSC. The people interviewed spanned across four separate projects. If the research was focused on just one project, the scope would be too narrow, and there would be a risk of becoming a project journal and not the foundation for a case study.

Table 4.1 - Interview Participants and Profiles

Ref.	Pseudonym	Title	Experience	Project	Education
A	Cian	Junior Dev	6 months	CloudStack	CS Degree
B	Barbara	QA Manager	11 years	CloudStack	Arts and IT
C	Erica	Sn Developer	16 years	iLoad	CS Degree

D	Kevin	Junior QA	1 year	iLoad	Arts and IT
E	Enda	Dev Manager	12 years	OX	Masters CS
F	Alan	Senior QA	8 years	OX	Arts and IT
G	Michael	QA	3 years	HCM Web	Arts and IT
H	Dave	Dev Manager	22 years	HCM Web	CS Degree

The questions were grouped into sections and followed a similar pattern of delivery in each interview as outlined below:

- Background - length of time current position - in IT in general - educational background
- Involvement with testing and understanding of what testing is about
- Experiences with Waterfall and Agile methodologies
- Test automation experience
- Programming experience
- Test tools and frameworks they may have used
- Has testing changed over time?
- What does the future of testing look like?

At each of these sections sub-questions were prepared to drill into more detail on specific experiences the candidate may have had.

4.3 Project Profiles And Interview Findings

4.3.1 Project A - CloudStack

- Data migration project
- Migrating customer data was a bottleneck for sales
- CSC bought a third party application
- Project goal was to refactor the existing code and add new functionality
- Test tasks were to add unit tests and fix existing ones that were broken
- And to build a functional test framework and write automated acceptance tests for the areas that were completely rewritten

- Team of fifteen developers, two automation engineers and one manual tester
- Product written in Java - UI in Google Web Toolkit (GWT) - Tests written in Scala using ScalaTest framework.

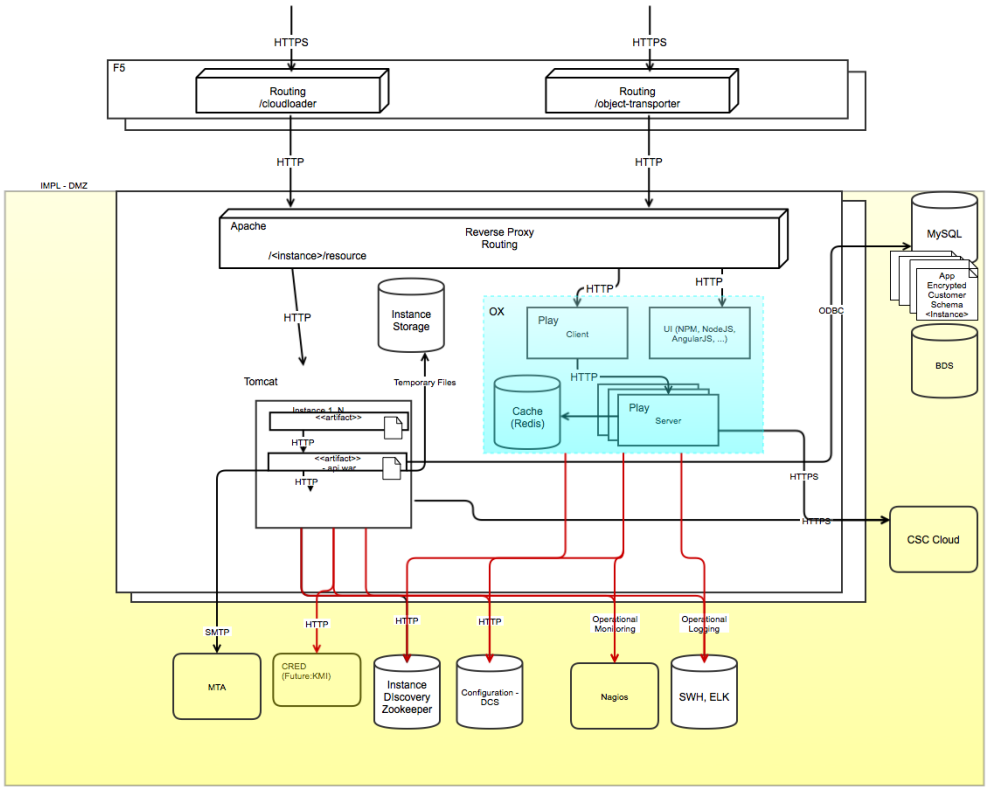

Figure 4.1 - Architecture of CloudStack

Interview A - Cian - Junior Developer on CloudStack

The first interview was with Cian, who was a junior developer on the CloudStack project. He recently graduated from college with a degree in computer science. As this was his first job and project, Cian had not done any testing or professional development before. He had a few lectures on unit testing, but he remarked that he now regrets that testing wasn't required as part of their final year project. The reason for this was he now sees how important testing is and how many problems occur when it is absent or done badly. Despite the fact that Cian had little experience, he was tasked with building the test framework for the CloudStack project. When asked how he felt about that he replied

"I was really scared to be honest. I had no idea why they were asking me to do this really important job, given I had no experience whatsoever. But on the other hand, I told myself it showed they believed in me, so I just jumped in, I guess."

Cian's experience seems to reinforce the point Deak (2012) makes that often testing is viewed as secondary activity in order of importance. If testing was seen as a critical piece of the project, why would this role be given to a new graduate with no commercial programming experience and no testing knowledge. Cian had to learn Scala on his own in the evenings, which is a relatively recent functional programming language. Once he had a good enough grasp of Scala he had to construct a test harness using the ScalaTest framework.

"I really didn't have anyone to ask as no one else in the team or even in the company had done this before, so I was leaving questions on forums and looking up books. But there are not that many resources for Scala beginners as most of the resources are for experienced programmers." The framework used at CSC is called ScalaTest as it uses the Scala programming language. One element to note that was implemented at CSC was the idea of building a vocabulary in the test framework. So this means that in addition to the words Given, When, And, Then – a business analyst may also choose from a defined list of other words for example – tenant or data or move or update. The code behind each of these specially defined words in effect means each word is a method and facilitates an implementation of a powerful paradigm for automated testing incorporating non-technical people.

Once the basic foundations of the framework were in place, he had to show the two test automation engineers how to use it and how to create BDD style acceptance tests. When asked how that process went?

" It took them quite a while to figure out what way I had built things - I mean it was hard for them 'cos I knew what I had done in my head, but I didn't write any of it down as I was too busy. The played around with it and learned a bit of Scala I think. But we didn't really have any follow ups, so I wasn't sure ." This implementation of BDD seems to be at odds with how Wynne and Hellesoy (2012) describe the approach. On the CloudStack project there was no interaction with the customer or product owner to work out what the desired behaviours were meant to be. In effect what happened on this project was they used the BDD framework to write standard automated tests. This was not BDD and hence the positive results described by Wynne and Hellesoy (2012) were not seen on CloudStack.

Cian was surprised by some of the comments from more senior developers on the team about the testers.

"There was a lot of derogatory comments about testers being dumb and not able to code properly. It seemed really unfair and I was a bit shocked because in team meetings everyone would appear comfortable together." The negative comments Cian heard

confirm that view that the relationship between developer and tester is complicated and at times one of conflict (Zhang et al., 2014).

When asked about the test strategy Cian wasn't sure if there was one. There was a lot of pressure to hit deadlines so everyone was focussed on delivery, but there appears to have been little planning in relation to testing. This scattered approach to testing confirms what Humble and Farley (2009) report as evidence of problematic testing patterns that lead to poor results and stressful relesases.

Can was asked what would he improve - " Include testers in more of the meetings and get rid of the nasty vibes. Some of the automation problems are really complex, so it seems to me we don't have time for nonsense."

On the future of testing "*It's only automation, I mean Michael saved the day with his manual testing skills, but that only happened because we didn't plan or design the framework correctly."*

Interview B - Barbara - QA Manager on CloudStack

Barbara was the QA Manager for CloudStack. She has been with the company since the very early stage. She is passionate about the company and its values. She started her career as a manual tester on the core product and then became a test team lead and was recently promoted to manager. This was her first full project as a manager. In a previous company Barbara has worked in human resources but then completed a conversion course in IT. Because of her background in HR Barbara got a job testing the HCM product at CSC. This project has been very different for Barbara. The relationship with the development manager has been tense from the start. He wasn't impressed that she didn't know any programming languages and had asked her how was she going to be able to report on the state of the automation if she couldn't read the code. She was happy that in fact Michael had come in and done such a great job in testing the UI manually, otherwise they would be in a terrible position now. She felt the test planning was adequate but that there were decisions made about the choice of Scala and the framework that were never discussed. Bach et al. (2001) note that choosing the wrong tool in the wrong context will always yield poor results. The testers struggled to learn Scala and was one of the reasons the progress was so slow. Communication with developers was difficult throughout the project. Initially, the developers were blaming testers for taking too long on the automation tasks. Then when one of the programmers tried to complete a test automation task it took him nearly two weeks. The task had been estimated at two days. The pressure to deliver to strict deadlines meant that test planning and strategy were not enforced

properly. Without the adequate test automation in place, the CloudStack team were adopting an Agile project structure but maintained a waterfall style methodology. This is a common anti-pattern where resistance to change can be the main blocker to overcome (Cohn, 201).

When asked about how she would approach testing on future projects

"Well you will always need manual testers, this project was evidence of that. Automation serves testers, not the other way around. The choice of language and framework were incorrect."

Barbara feels that the company is pushing automation from outside pressure to conform but that in reality on a lot of projects the testing is still done manually. Barbara's view is in stark contrast to Whittaker's take on the future of testing (Whittaker, 2014) at Google and further afield. Barbara would be at risk of falling into the trap of irelevance according to Whittaker.

4.3.2 Project B - OX

- New project - no legacy code involved
- A tool for customers to move HCM objects safely and securely between cloud instances
- New engineering manager was the head of the project
- Team of six developers and two automation engineers
- Fully Agile style project

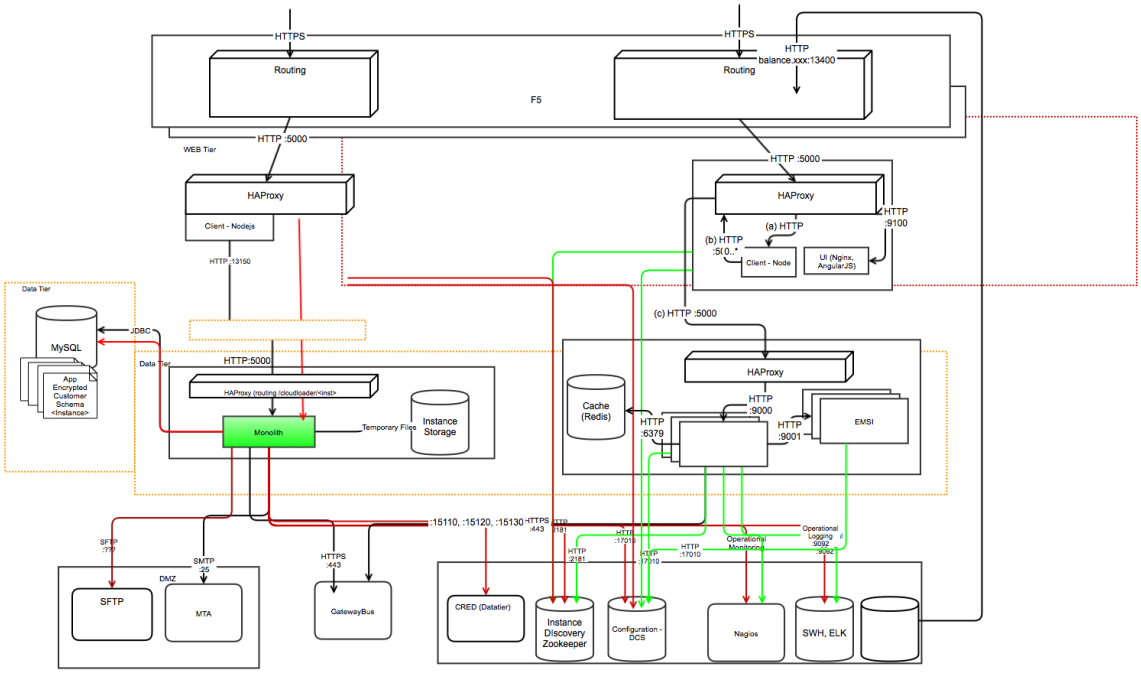

Figure 4.2 – Architecture of Ox Project

Interview E - Enda - Development Manager on OX

Enda is an experienced development manager with over twelve years leading large teams of programmers. He has been with CSC for just under a year. In his time with CSC he has impressed the Head of Development in California so much that he has been promoted to director of engineering for Europe. He is a strong advocate of Agile and in particular Scrum. The OX project was strategically important as it was his first full project in charge since his promotion and was a chance to prove the benefits of Agile and his people-centered leadership style that he had been talking about at CSC since he arrived.

The ratio of developers to testers was important for Enda, he didn't want testers swamped with work to do at the end of a sprint. Cohn (2013) The choice of tool and language was selected with the team in a series of open discussion meetings including the testers. The skills and background of the team fed into the decision of Python and Selenium. He encouraged the testers to take ownership of developing a test strategy and keeping it up to date on Confluence so that the whole team could access the resource and make changes when necessary. Enda put into practice the best elements of what Cohn (2013) outlines for successful Agile projects. In addition, the model of continuous delivery as a means of getting feedback quickly has been implemented on the Ox project with successful results (Humble and Farley, 2009).

He has been on death marches with old waterfall style projects and when the company he was then working for transitioned to Agile he has not looked back since. Enda has experience of trying to retrospectively add tests into legacy code but found this to be a risky endeavour, as clearly outlined by Feathers (2004). For Agile to succeed there has to be automation right the way through the project. Implementing a routine of defined Scrum ceremonies put the onus on the team to work through issues and to be able to demonstrate the sprint deliverable at the demo. Any issues arising were dealt with quickly and openly. For Enda, there is no defined role for manual testing in the near term future strategy of the company. This view of testing, as expressed by all the developers interviewed is the opposite of how Fewster and Graham (2012) see the situation. They say that one cannot replace the other as they are driven by different motivations doing different jobs. Petersen et al. (2012) surveyed 115 software professionals, 80% disagreed with the vision of automated testing replacing manual testing.

Interview F - Alan - Senior QA on OX

Alan was the senior test engineer on the OX project. He has been at CSC for two years. Having completed a degree in business he then did a masters in Information Technology (IT) and a few years later completed a second masters in cloud computing in Dublin. Alan made a decision to go into testing as he thought it was more interesting trying to see if he could break products than make them. He has done some automation already in Java using the Selenium web testing framework. He was excited to be on this project and had asked to be moved onto it because there was a buzz about the new Director of Engineering, Enda.

"He has a great energy about him, I mean he expects a lot, but it's never unfair or unclear so its just a matter of working hard." Alan was fully consulted on the choice of language and framework, and he feels his experience was taken into account. *"The developers were really friendly and helpful on this project, which is a change from how other projects have been run."* There appears to have been a lot of collaboration on the OX project, where all team members were based in Dublin, which was a first for the company.

The testing strategy was asked for, and Alan mentioned he felt empowered to take ownership of driving the test strategy forward.

On the future of testing, *"I hope manual testing will still be part of the picture in a few years time. I mean there are tests that I do now with a good few years experience that*

would be really complicated to automate, its just much quicker and easier for me to do it manually. But I actually think if you were to come back to this company in 5 or 6 years there would be no manual testing going on. I sit in on a lot of the tester interviews and if they don't have coding experience they won't get in - simple as that - it wasn't like that a few years ago." Alan's views on the future of testing stand out in contrast to the rest of the testers interviewed. He sees the future of the company as one where automation dominates and where testing is 100% automated. Bertolino (2007) discusses the idea of total test automation. At that time, there were too many technical barriers. Today, some of the technical pieces have been solved but the resistance to change by employees is still a persistent blocker from achieving complete test automation.

4.3.3 Project C - iLoad

- Legacy application over five years in existence
- It is an ETL (extract, transform and load) tool used by CSC
- Java based application
- Project is now in maintenance mode as a replacement product is nearly ready to launch
- Team of three developers and one manual tester

Interview C - Erica – Senior Developer on iLoad

Erica has over sixteen years experience in software development and has been as at CSC for the last six years. She has been on iLoad for over two years. She studied computer science, and her first job was in writing the tests for the development team at the time. *"None of the guys wanted to do their unit or integration tests, so that was always assigned to the newest person and for quite a while that was me. It was good in that I got to learn a lot about the product before I had to write production code. And it has instilled in me a great respect for testers ever since, it is a really hard job."* Erica has experienced what it feels like to work as a tester and as a consequence has more understanding and compassion for the job they do. Zhang et al., 2014 discuss ways to alleviate conflict between testers and developers. Erica has demonstrated that spending time in one anothers world can ease tension and foster understanding. On the current project, Erica's work has been mostly in maintenance mode. *"There are plans to replace the product we are working on but nothing concrete yet so we keep on fixing bugs but not really adding*

new features. There are a lot of old tests in Bamboo, but Kevin has been trying to deal with that - it's a tricky one."

There is a lot of documentation on Confluence (an online wiki from Atlassian) about test strategy and test meetings but they have not been updated. The original test automation engineer added a lot of automated tests, but he has since left the company. The data the tests were relying upon is now deprecated, so the tests stated to fail. Initially they were quarantined and then eventually they were deleted. This is one of the anti-patterns that are signs of serious alarm for quality according to Humble and Farley (2009). Her opinion on the future of testing *"On a project like this, which is nearing the end of the road there is no point in adding new tests but the way the original automated tests became ignored is a pattern I have seen elsewhere in this company and in other places I have worked. I can see why someone would need to manually check the product but I'm not sure a team needs a dedicated person to do that. I mean anyone really can run through a quick checklist and see if it's working or not. Testers need to be writing tests in the build really - that's what gives me peace of mind, knowing that Kevin or someone else is writing tests that ensure my changes haven't broken anything ."*

Interview D - Kevin – Junior QA on iLoad

Kevin studied arts in Galway and then did a conversion course in IT, he has been at CSC for over one year, and this is his first job after completing the course. He didn't cover testing on the course except for a small mention of unit testing. He had applied for a programming job, but he didn't have sufficient experience, so he was offered a test position instead. Kevin sees this role as a stepping stone into programming He thinks that people end up in testing by mistake. Kevin's views echo what Deak and Stalhane (2013) discovered in his study of nineteen companies. Deak reported that testing was seen as a negative career choice.

On the current project, Kevin was told to maintain an automated suite of tests written in Java. But when he went through the tests he was surprised to find very few of them running in the pipeline. *"The were mostly commented out or in quarantine. So there was very little impact, some of the managers would talk about 200 tests running at meetings but I knew that only a fraction of those tests were doing anything."* Kevin spends most of his time verifying the fixes the developers have done to solve existing bugs in the system that are impacting customers. On the future of testing *"I think there will always be manual testing to some degree, but I can see automation becoming much bigger."*

4.3.4 Project D - HCM Web

- Part of the core HCM product family
- Written in proprietary language developed by CSC
- Sixteen developers based in the US
- Four manual testers based in Dublin
- Waterfall style project methodology

Interview H - Dave – Dev Manager om HCM Web

Dave has been with CSC since the early days of the company and has many years experience as a manager. He is in charge of a large team of developers working with the company's proprietary programming language. All the developers have to go on a six-week boot camp to learn the language, it's object oriented, so it has similarities to Java and C#. The testing and development groups have always been separate entities and even have different cost centers and reporting structures within the organisation. Which is different to how the tools teams operate, where they all report directly to the development manager. So although Michael is on the project that Dave is in charge of, he actually does not report to Dave. Michael's manager is in California, and he has weekly one to one meetings over Skype. Mostly the project has followed a waterfall style methodology and largely this has worked. Dave feels that this approach is working, yet there appears to be a mountain of technical debt being built in the form of code with no tests being committed. McConnell (2009) discusses the high cost of not taking enough care with the quality of the cod. Martin (2011) in the The Clean Coder clearly calls out steps that when followed can lead to clean and bug free code. These practices have not been followed on HCM Web. When asked about the testing Dave admits that this is the one area he is concerned about. When pushed on how much he stumbled a bit but eventually settled on 75% was still tested manually (Michael one of the test team said later that the number would be close to 90%). There are automated tests but they are written by the developers in the same language and as none of the testers are sent on the boot camp, they don't understand what the tests are meant to be doing and cannot take ownership for them. If a team has a clear picture of what test coverage they have, decisions about where to put more test effort can be made with confidence and good sense (Crispin and Gregory, 2014). This was lacking on the HCM Web project.

When asked about his thoughts on the future of testing at CSC, Dave said that there would have to be more training for the testers. He advocated automated testing wherever possible, and the senior management in California do not want to hear about manual testing. *"Just make it happen is what they say - but it's pretty tricky,"* Dave said in conclusion. Cohn (2013) notes that adopting Agile practices requires a heavy investment in automation. CSC have invested in the planning but the actual execution of automation is in contrast to the strategy.

Taipale et al. (2011) observed that if a company is considering test automation, they should first consider whether their products or processes are suitable for automation. CSC need to follow this advice in looking at the company strategy that only sees test automation. HCM Web is a project that is not currently geared to achieve automation success.

Interview G - Michael – QA on HCM Web

Michael was a maths teacher at secondary school, he moved into IT attracted by the higher salaries. He has been manually testing HCM Web and raising bugs with the development team. Michael has been with CSC for three years. One of the challenges is with the time difference between California and Dublin getting feedback and answers on bugs can't take a number of hours, or it can mean working late at night.

Michael has friends who are on Agile projects and in his opinion they sound better. On this project, the testers and developers are totally separate, apart from a weekly team call. So everything in Michael's experience is tested by hand. The only automation he knows of on the project are the unit tests some of the developers write, but there is no project policy on that, so some developers do not check in unit tests at all. Michael wasn't sure if there was a test strategy when asked. He said he had spent over two months writing a test plan, which included all the test steps that were needed to functionally test the product. In terms of other types of testing, like performance or load testing Michael wasn't aware of any efforts within his team to test in that way.

Michael appears to be experiencing all the worst elements of testing in a pre-Agile world. He is not involved in project meetings, he doesn't know of changes coming to requirements, he is busy writing long documents that are almost certainly out of date by

the time they are complete. His chances for success with this methodology are very poor. Collins (2012) found that collaboration is a key element to test automation success in an agile project. Crispin and Gregory (2014) share that without a whole team approach, developing software becomes a divided and fraught activity. These patterns of weak communication of expected requirements to testers will lead to insufficient validation of expected behaviours in the application (Bjarnason et al., 2013).

4.4 Testing Survey

The secondary source of data comes from an internal survey that was carried out at CSC in 2013. Eighty software test engineers were interviewed. The population was a mixture of manual and automated testers across all divisions, teams and projects in both the American and Irish offices. The primary driver for taking the survey and investing in the analysis of the data was customer satisfaction. In a measurement based on customer surveys, the company had an 87% customer satisfaction rating. When this rate began to dip in response to a number of highly visible bugs in the main HCM application, the leadership team decided to take action. This survey was one of the results of that action. The survey was sent to 98 engineers, 83 responded giving an 85% response rate.

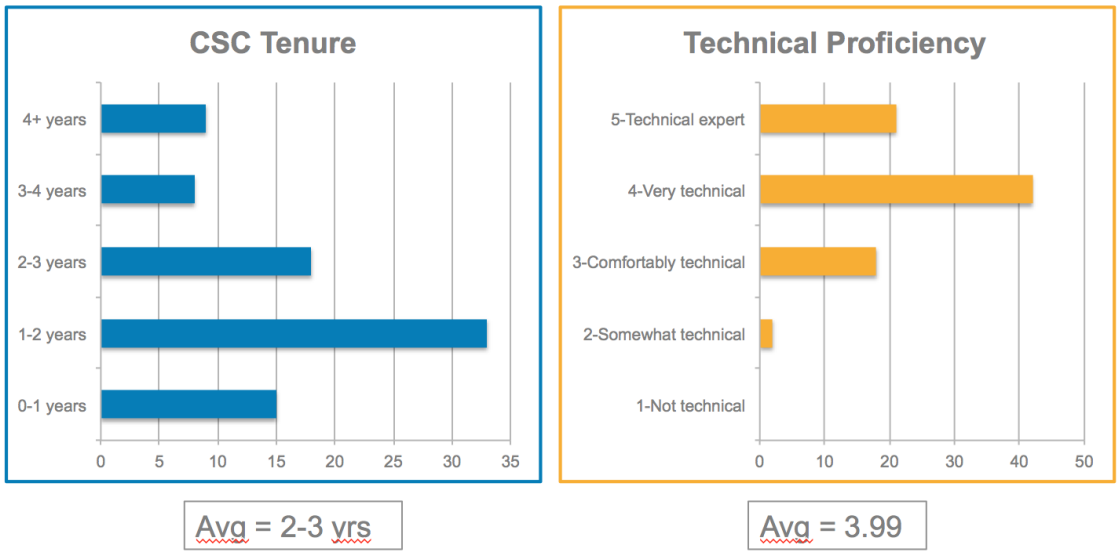

Figure 4.3 – Profile of CSC Testing Survey Candidates

Main points raised by survey candidates:

- “Stop lumping our test suites together with other teams.”

- “Pick one way of organizing test plans and stick to it. It's very difficult to find tests when they keep moving around.”
- “It is becoming extremely painful to figure out why the same scenario is failing. To replicate the failure I need to run the entire test bed which takes like 5 to 6 hours.”
- “Pipeline blockers make life miserable! Sometimes it takes days for automated tests to make the build.”
- “I always get the feedback if we add tests we will increase the build time, well as software grows so does testing/test automation. Having longer build times should not hinder adding auto tests.”
- “The biggest pain point is that most existing test cases are a mess as they were not written in a maintainable manner.”“From my experience Scrum teams still see automation as a nuisance. Phrases such as *‘this should be easy’* and *‘why is this taking so long?’* are said too often by non automation people.”
- “It would help if automation was included in planning discussions early on, separate from acceptance testing.”
- “Automation and its importance needs to be more accepted by Dev and Scrum teams. Frequently they are being told that it is important, but they aren't truly understanding that. It's seen as more of a hindrance then valued.”
- “I think Automation needs more attention from the Product Managers who actually work on scoping the work and creating stories. Automation team would know only when those stories are brought into the sprint.”

4.4.1 Main Findings From The Survey

The main issues that came out of the analysis of the survey data were :

Test Data - the recurring issue raised by respondents was that of incomplete and out of date test data. This has been causing automated tests to fail and, as a result, a lot of automated tests are ignored in the pipeline because no one is tasked with the maintenance of the tests.

Test and Automation roles not well understood by management - many of the surveyed testers felt that their role is not understood by managers in development. This manifests itself through unrealistic expectations of how many automation tasks a tester could complete in a two week sprint.

Test Plan Organisation and Visibility - many testers felt that there was little thought put into how tests documents were organised or structured. So many teams had very different methods of dealing with the planning and storing of test artifacts within the internal private cloud. So the result was a lot of confusion and many cases of duplication of effort.

Build Pipeline Performance - with the increased pressure to write more and more automated tests, the structure and performance cost of those tests was often overlooked in favour of speed. So now the build is taking longer and longer with many more failures. Because best practice in relation to logging is often not followed, failures in the pipeline in relation to the tests can be very hard to debug and discover the source of the issue.

Enhancements to proprietary language - the internal programming language is used by many teams to build out large parts of the core product offering. However, it has not been updated in a number of years and is falling behind current advances in development.

Insufficient Training - for those testers who have been put into new roles that require that they now write automation, many of these employees felt that they received very little training and in some cases, none at all.

Post Survey Recommendations

Arising from the survey six recommendations were made to take action on:

- Create a Testing Task Force to continue driving productivity results
- Improve the process for test data creation and maintenance
- Increase the level of training for all testers in the organisation
- Create a Test Build Team to take responsibility for the increasing build times and failures
- Make the necessary updates to the proprietary language
- Roll out Agile training for all teams

Looking at the problems highlighted in 2013 by the candidates in this survey and the situation today as evidenced by the semi-structured interviews, a number of the same problems persist. Some of the actions have been taken as a means to improve the situation. The training courses have been initiated for testers in relation to the proprietary language, and this has helped understand and write the associated automated tests.

However, the data issues are still persistently causing problems in the pipeline in the same way they were in 2013. A number of the testers interviewed in this study reported a lack of adequate estimation and time for testing and automation within sprints. This was also a problem in 2013. Also, the understanding of testing in a wider context amongst management and developers does not seem to have greatly changed. A persistent theme in the CSC interviews in this study was the fraught relationship between development and test.

The testers' concern that development and management did not fully understand the challenge of test estimation and execution, fits into the negative view of testing that Deak (2012) has reported. The lack of action taken on the issues arising from the survey is a clear example of the gap between what CSC states as their automation strategy and the reality on the ground. Humble and Farley (2009) describe the elements that can cause a companies efforts at achieving test automation success and continuous delivery. They note that relying on manual intervention at any significant stage is a bad smell and a sign that the organisation has not fully accepted or adopted the required changes to fully transition to a world of continuous delivery.

4.5 Bug Databases

Once the candidates for the semi-structured interviews had been selected from the four projects, permission was sought to run queries on the bug databases related to each project to see if any further insight could be drawn about the effect of manual and automated testing on reported bugs in the system. In isolation reports from this bug database would provide only a very narrow view of the full landscape of automated and manual testing on these projects (Ciupa, 2008). However, when combined with the data from the semi-structured interviews in relation to the same projects and also the survey questionnaire, this data takes on a much more meaningful and richer significance.

When a bug or problem is found in an application or service a number of steps need to be followed. The title of the bugs is very important and will aid searching greatly if appropriately titled. A screenshot if possible should be taken and stored to be attached to the bug report later. Any logs that could assist the developer debug the issue should be captured. If the log is very long, only the pertinent sections relating to the issue should be included. The other information that should be entered into the appropriate fields on the bug report could be: Operating System version, browser type and version and service

pack, any other software that was running that may potentially have had an influence on this issue, the version of the software under test, was it found in production, was it found in regression testing, crucially for our purposes was it found in manual or automated testing. In the description of the bug the exact steps that need to be followed should be described which when followed should lead the developer to the point of failure and they should then be able to see the same issue. The logs and screenshots should be attached directly to the bug. If a particular virtual machine was used and still available, then that name and location should be detailed as well.

If bugs are not logged as soon as they are seen then the likelihood is they will be forgotten. So the discipline of getting bugs logged into the database is very important. The sooner a bug is caught, the less expensive it will be to fix. However, not all bugs will be fixed. Some will be resolved as not a bug, which may be a misunderstanding on the part of the tester about the intended functionality. Some will be set as will not fix due to a newer version being released which may have deprecated the buggy feature. Some will be deferred to a later release owing to a lower severity.

The data for these reports are generated by creating queries in Jira, which is similar to creating SQL queries. For the purposes of this study, the bugs that were reported in each of the four projects that had the label manual or automated filled in for the analysis field were of most interest. To get a report with the desired results, a number of other criteria needed to be filtered out of the search.

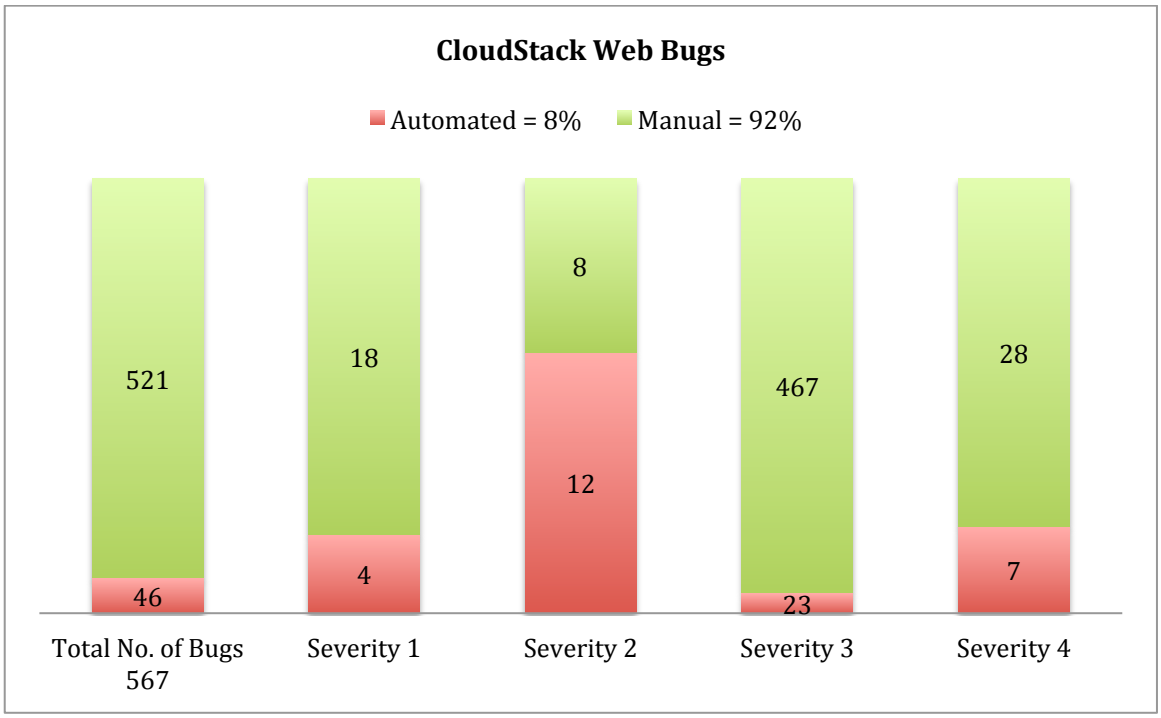

Figure 4.4 – Comparison of CloudStack bugs: Manual vs Automated Testing

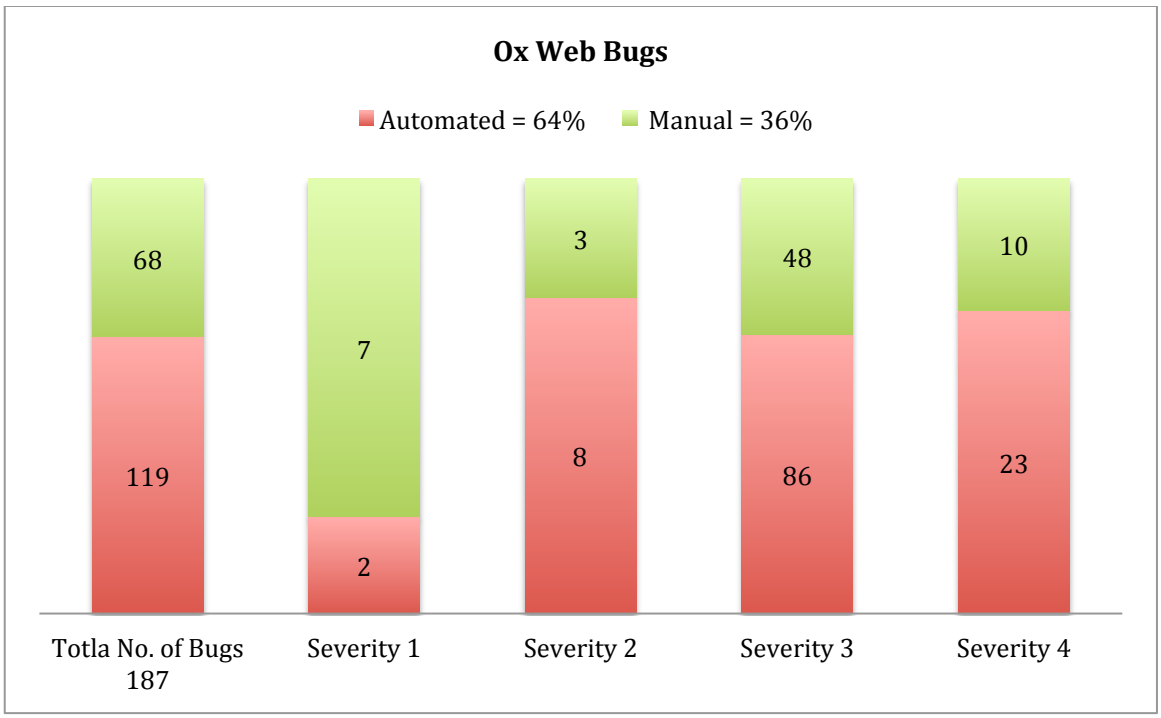

Figure 4.5 – Comparison of Ox Bugs: Manual vs Automated Testing

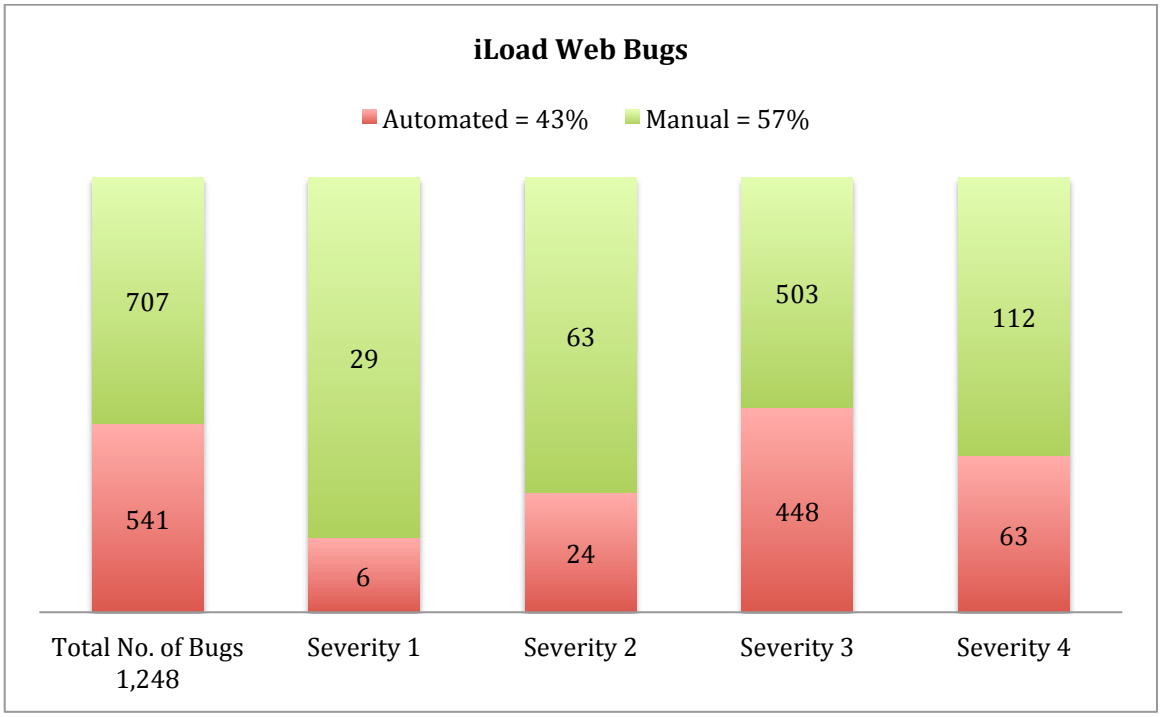

Figure 4.6 – Comparison of iLoad Bugs: Manual vs Automated Testing

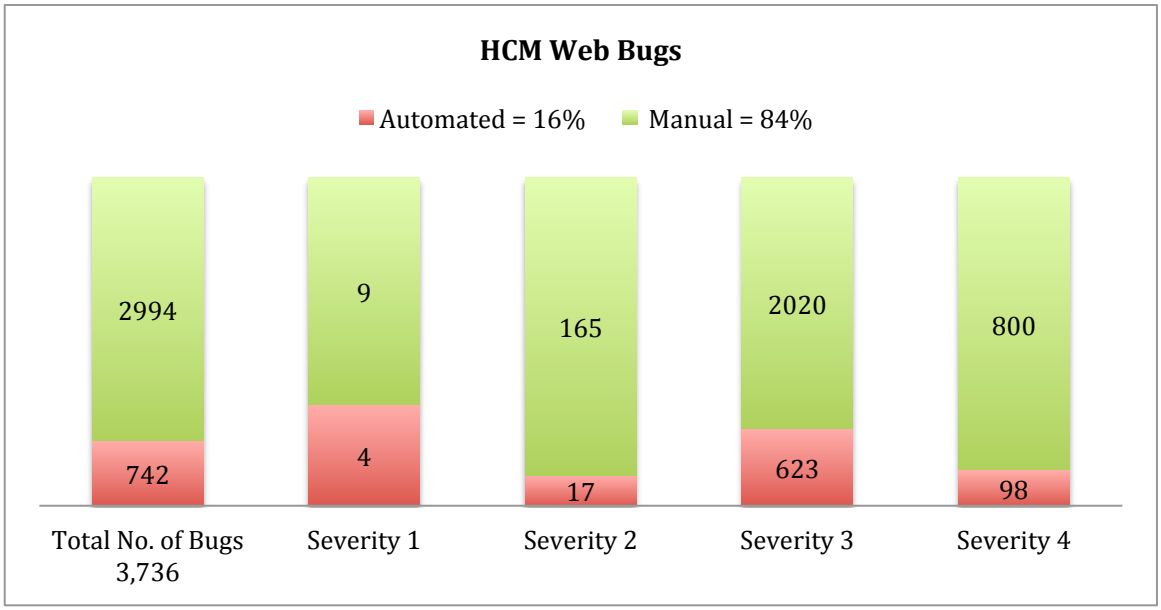

Figure 4.7 – Comparison of HCM Bugs: Manual vs Automated Testing

4.6 Analysis

4.6.1 Relationships Between Testers And Developers

As noted in the literature review the nature of the work testers do is bound to bring elements of conflict into dealings with developers (Zhang et al., 2014). If the vision of test work is seen as a way of judging a developer, then that will frame any conversations around bugs or other issues. The influence of an individual's personality and social skills are significant elements in how well this key relationship goes. In the 2013 survey of testers at CSC one of the highlighted problems was the lack of understanding development and management had in relation to testing. While this survey was introduced to staff as an anonymous survey, the names of the respondents were in fact recorded and stored. Two of the interview candidates said that this fact was known by the staff but not officially. So it is possible to infer that statements about problems in relation to management may actually have been downplayed by respondents, given that they knew their answers were not anonymous. In addition during the interviews, Cian mentioned his shock at hearing other developers talking in a mean way about the testers on the team.

So the problem of relationship has not gone away and appears to be still a problem despite having been highlighted in the survey over two years ago. The challenge for management is that the solution to this issue appears to be quite complicated. Deak (2012) addresses the issue of the perception of testing as a less important activity than development and the caricature of a tester as a failed programmer. This could be addressed by more technical training for testers (Deak, 2012). In addition to this technical challenge, there are other issues of resistance to change. On the CloudStack project, an agile methodology was chosen. However, during the interviews with Barbara and Cian from that project, the culture was more akin to a waterfall style of project, where test and development are separate entities and development throw code over the fence to test at the very last minute.

4.6.2 Future Of Manual Testing

All of the testers in the interviews spoke about their belief that manual testing will always be needed. The CloudStack project came to rely on the skills of a manual tester and this saved the testing side of the project. So in that case the tester's opinions appear to be vindicated. However, the use of a manual test resource on the CloudStack project only

came about when the automated testing effort was taking too long and was far more complicated than expected. The reasons for this situation appear to have more to do with poor planning and technology choices rather than an out and out failure of automation.

On the iLoad project, manual testing predominates in place of the automated tests that are written but not running in the build. Given that this project is in maintenance mode there is a lack of motivation to drive automation forward on this project. Kevin has not got a positive image of the work he is doing. He feels that testing is the poor cousin of development, and his career of choice is programming. It seems strange then that he does not take more interest in digging into the code of the tests as opposed to testing through the user interface.

Michael on the HCM project seems powerless over what type of testing he can perform. There appears no possibility for him to do automation. To that end it is possible that his view that manual testing will always be needed may be grounded in a fear of change rather than a holistic view of what is coming down the line technically within his discipline.
The OX project appears to have got a lot of elements right. The mix of people on the team seemed to have worked very well with a charismatic and motivating leader and a technically talented team. The developers and testers were well integrated and decisions about technology were openly discussed. As a result, the automation effort on this project was much more successful than the other projects.

During the case study at CSC a number of the interview candidates referred to previous experience where companies had invested heavily in an automated testing product, only to be sorely disappointed by the actual results some months or years later. Increasing the speed of tests and breaking the cycle of repetitive manual tests are often the main reasons for trying to adopt automation (Amaricai, 2014).

So at CSC the near term future may well be one where manual testing in reality still predominates. To that extent, the testers may be correct. However, this may have more to do with the dysfunctional manner in which automation has been implemented at CSC. If more projects were run like the OX project then the future of manual testing would be a lot more uncertain at the company.

4.6.3 Delivery Of Software

Humble and Farley (2009) note that software only carries a value when it has got to its users. During the semi-structured interviews only the candidates involved in development mentioned delivery or pipelines or builds. This is significant in that it may show that the testers view was focused on tasks and detail but not as much on the wider picture of how the work they do influences elements in the delivery of the software. Enda, the development manager on OX, spoke about how everything that they do in terms of testing has to be put into the pipeline. The testers never mentioned where or how their tests would run.

4.6.4 Educational Backgrounds

All of the candidates in development had primary degrees in computer science. None of the testers had their primary degrees in computers but all of them had some post-graduate qualification in IT. Alan on the OX team had two masters in IT, most recently in cloud computing. So is there any significance is this difference of background? The difference can initially be explained by the hiring policy of CSC, for a development position a degree in computing is the minimum standard to be considered for a role. However, when hiring into the testing roles, in the past CSC have taken in people from a variety of backgrounds. If, like Barabara who had a background in HR, they can prove that that have a rounded experience coupled with an IT qualification than that was enough. However, CSC is changing the policy in relation to requirements for hiring new testers to include more programming experience. Dave, the development manager on the HCM Web project, noted that in his opinion a conversion post graduate course in IT is not the same as doing a four-year degree in computer science. So perhaps the significance may have more to do with perception by peers and how that may affect relationships on teams.

4.6.5 Difference Between Test Automation Strategy And Reality

A number of the candidates mentioned that senior management in CSC in the USA did not want to hear any mention of manual testing. To that end, all testing strategies that were published internally were totally focussed on automation. However, all four of the projects looked at as part of the case study had elements of manual testing to different degrees. CloudStack's test automation was too slow and complicated, and the majority of the testing was done by manual testing. The iLoad project had automation in place, but

the tests have been ignored for some time, so any active testing on that project is now completed manually. Over 90% of the testing on the HCM project is completed manually. Even on the most successful project, in terms of test automation, had elements of manual test.

So it seems that there is a marked difference in what the company is saying it is doing in testing and the actual reality on projects. In addition, when the statistics for the bug databases are considered it is clear that the vast majority of bugs are still being found by manual testing. If, at some point in the future all of the testing at CSC is automated testing, then they will have to find a way to ensure that bugs are still being found before a customer sees them.

5. CONCLUSIONS AND FUTURE WORK

5.1 What Are The Claims Of This Research?

This research journey started with forming a question. That question was how would advances in test automation affect software testers? The research on software testing has tended to concentrate on the technical aspects and methodologies of the discipline. However, the literature review showed that the question for this study had highlighted a gap in the research knowledge. The story of testing at CSC began to emerge as the interviews began. Having analysed the data from the interviews, the testing survey and the bug database, the main claim that this research is making is that test automation will eventually replace the software tester at CSC. The four projects that were studied had very different approaches to automated testing and consequently different results were delivered. The testers in the interviews expressed the view that manual testing will always be required in the future. Given the poor performance of automation on the CloudStack project, manual testing on projects run in this way will be needed. However, CSC is a large, well-financed organisation, and the success of the OX project was noted, with senior executives in California receiving regular status updates on progress. It was, perhaps, the first project that showed positive results in alignment with the company strategy on test automation. Over time, perhaps in as little as five years, iterations of improvements could mean that an automation project will require no manual testing at all. Then the role of writing automated tests could, as a number of the developers in the interviews suggested, simply be another development task that is rotated in a sprint.

This projection of a future without testers at CSC is dependent upon a number of changes and improvements being implemented. Firstly, better project planning and governance needs to be put in place for test and development projects. Secondly, test tool selection and task estimation needs to be discussed by the entire team to make an informed choice based on suitability to team members' skills and abilities. Thirdly, CSC needs to address the conflict that is present between testers and developers at the company. In the hope that developers can view testing in a fresh light and appreciate its significance to the overall success of a project. This is a significant challenge for CSC as a number of their core products are still being built using a waterfall style methodology, where test is totally separate from development. The human element in testing needs to be understood and taken into account, where motivation, perception of testing and influence of senior testers are key elements in a successful test team (Shah et al., 2010). Fourthly, CSC needs to

remedy the issues highlighted by testers during the 2013 survey. While a number of task forces and meetings resulted from the analysis of the results, very little action was implemented. The test data issue, in particular, highlighted in the survey and by a number of interview candidates, is a critical issue for test automation advancement.

If these issues are resolved and a project template, along the lines of the Ox project, is agreed then the chances of wide-scale test automation success are high at CSC. The biggest barrier potentially is the resistance to change amongst staff at the company. Implementing the technical solutions requires capital investment and agreement, which can be achieved. However, changing old attitudes and building trust where none exists will require skillful leadership and a willingness from staff to make the necessary effort. If this proves too much of a challenge, then CSC risks remaining in the state of limbo it currently is in with regard to testing. Where, on the one hand, the company strategy espouses a vision of total automation and plans projects with this goal in mind. While the working reality on the ground is that projects are still heavily reliant on manual testing to make deadlines and fill in gaps that currently take too long or are too complicated to achieve with automation.

There is clear evidence that test automation is central to CSC, two of the interview candidates spoke of how senior management did not want to hear any mention of manual testing in documents or job requirements. Looking at the sixty-five open test related jobs currently at CSC, fifty-nine of these require programming experience.

5.2 Any New And Interesting Findings

A number of new findings emerged from this study at CSC that will be of interest to the company and to those working on any of the development and test teams there.

- That company policy is not aligned with the reality of day to day testing
- There is conflict in relationships between testers and developers that is not historical but is a current problem across the organisation
- Testers are not aware of changes coming to their chosen careers
- Depending on CSC addressing a number of issues - within five years there may be very little work for manual testers and certainly the current crop of manual testers, would not get a job at the company given the requirements in place for test positions requiring programming experience.
- Having been born within development, testing as a discipline, appears to be

returning to development and merging back in, to become a development task to add tests to the pipeline within a sprint.

- Testers believed they could test in a way that incorporated complexity and subtlety that an automated test could not replicate.

5.3 The Question Of The Generalisability Of The Findings

This research was conducted using the case study approach at CSC. In the light of that there are challenges in saying that the findings of this study are applicable in a general sense across the software development and test industry. This is a limitation with the case study research approach. By the nature of the approach, a narrative is uncovered through an analysis of the data that has elements that will be specific to CSC. However, there are other elements that are common across the industry.

The findings of this research are that within five years there will be significant changes in the way testing is done and the skills needed to gain employment. The elements that have led to this conclusion do have elements that are specific to the structure and operation of the organisation studied in this research. However, the main conclusions are derived from aspects of the research that form part of nearly all software engineering teams.

5.4 How Has This Research Advanced The Field Of Testing

Currently, much of the focus of research related to testing is focused on technical activities. This study has highlighted that while this technical effort is on going, there is going to be a significant human cost to the development of test automation. That cost will be the loss of jobs for people currently engaged in manual testing. By presenting the evidence from this small case study, the hope is that more research could now focus on the transition from manual testing to automation, to assist those whose jobs will be at risk.

There is a lot of literature that says that automated testing cannot replace manual testing because they are different activities with different motivations. This is a flawed argument; different motivations will not stop sophisticated software finding a way to mock human interaction in the testing process. If this vision of a totally automated testing future comes to pass, it will change significantly many aspects of the testing industry. Many testing texts will become redundant, testing conferences will need to be redesigned, and testers

will leave the industry.

This research, within the context of this case study at CSC, has highlighted that significant changes are imminent within the discipline of testing. From the conversations with the testers during the semi-structured interviews, there is a level of denial or unawareness of the enormity of the future changes to their chosen career. The next important step to be taken for testing research is to try to look in more detail at this transition to a new future of test automation. This case study has tried to begin to fill in the gap identified at the beginning of this project, to investigate the tester more than the technique.

5.5 Limitations Of The Study

Given the time and resources associated with a part-time research effort, the fact that only eight candidates were interviewed from a company of over four thousand is a limitation on how representative this sample was of the company as a whole. However, the aspect this study was focused on was only in relation to testing and automation strategy. To address this limitation in population sample for the interviews, great care was taken to select candidates from different projects, from different levels of hierarchy and with varying amounts of company experience and IT experience in general.

5.6 Possible Future Directions For Research In This Area

The research in this study took a small sample of employees at CSC within a case study approach. There are many ways this area of research could be expanded in ways that have not been explored as yet.

- Eight employees at CSC were interviewed for this study. This number could be increased significantly. Depending on the resources available to the research team, a broad survey type approach could be used to contact software testers globally to investigate the influence of advances in test automation globally.
- The case study methodology is well suited to study in greater depth the complicated relationship between developers and testers. This could be looked at in the context of the changing world of testing and development. A possible approach would be to quantitatively map patterns of productivity before and after agile transformations have occurred within a team.
- Studying the effects of automation over time could yield significant and interesting

results. If a study was conducted every year at one organisation to measure and monitor how test automation was changing the approach to testing over time.

- A number of different approaches to testing have been mentioned in this study. A future work could choose a set of test methodologies, manual vs. automated, and compare and contrast the perceived and actual effectiveness of the different ways of doing testing.
- Martin Ford in his books Lights in the Tunnel and Rise of the Robots warns of a future of large-scale unemployment as a result of pervasive automation. Andrew McAfee and Erik Brynjolfsson in their book Race Against The Machine argue something similar about how production is increasing but for the fist time in history this is not being matched by a rise in employment. A possible future direction for test automation research would be to site a study within the context of a global move towards automation. This study would look at possible future directions for programming languages and whether, in fact, automation might not just replace testers but also replace programmers as well.
- If test does go back into the discipline of development and testing becomes a development task - will that mean the apathy about the topic that was reported in the interviews and survey results and referenced in literature disappear?
- A long-term case study would be interesting to research within the context of the Irish Software industry. A number of testers could be chosen at different companies and interviewed over a number of years to analyse the changes they experience in the industry and in their companies.

There are voices within the test industry that acknowledge there are significant changes happening that will effect how testers do their work. In the book How Google Tests Software, James Whittaker (the then Head of Test Engineering at Google) notes that the combination of continuous delivery, agile development and early user-involvement have fundamentally altered development and heralded the end of testing as we know it. He predicts that every tester's world will have to adjust to these new realities or face irrelevance. While many within the industry may share these opinions or feelings, more research is required to gain a deeper understanding of the transition and practically to assist testers cope with the coming changes.

6. REFERENCES

Abhilasha, and Sharma, A. (2013). Test effort estimation in regression testing. *Innovation and Technology in Education (MITE), 2013 IEEE International Conference in MOOC*, 343–348.

Alegroth, E., Feldt, R., and Olsson, H. H. (2013). Transitioning Manual System Test Suites to Automated Testing : An Industrial Case Study.

Ali, M., and Saha, T. (2012). A proposed framework for full automation of software testing process. *Informatics, Electronics and Vision (ICIEV), …*, 436–440.

Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., … McMinn, P. (2013). An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software*, *86*(2013), 1978–2001.

Andreosso-O'Callaghan, B., Lenihan, H., and Reidy, P. (2015). The Development and Growth of the Software Industry in Ireland: An Institutionalized Relationship Approach. *European Planning Studies*, *23*(June), 922–943.

Armour, P. G. (2004). Not-Defect: The Mature Discipline of Testing. *Communications of the ACM*, *47*(10), 15–18.

Bach, J. Kemer, K., Pitchford, B., Lessons Learned in Software Testing - A Context-Driven Approach (2002), Wiley.

Bannister,F., and Remenyi,D. (2002). *The Creation of Knowledge Through Case Study Research, Henley Management College, Working Paper Series.*

Beizer.B. Software Testing Techniques (2nd ed.). Van Nos- trand Reinhold Co., New York, NY, USA, 1990

Bertolino, A. (2007). Software Testing Research : Achievements , Challenges , Dreams. *Future of Software Engineering. FOSE '07*, (September), 85–103.

Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B.,Feldt, R. (2013). Challenges and practices in aligning requirements with verification and validation: a case study of six companies. *Empirical Software Engineering*, 1–47.

Bryman, Alan (2004). Social research methods (2nd ed.). New York: Oxford University Press.

Capgemini, Sogeti, and Hp. (2014). World Quality Report 2013-14, 1–64. Retrieved from http://www.capgemini.com/resources/world-quality-report-2013-14

Ciupa, I., Meyer, B., Oriol, M., and Pretschner, A. (2008). Finding Faults : Manual Testing vs . Random + Testing vs . User Reports. doi:10.1109/ISSRE.2008.18

Cohn, Mike (2013). *Succeeding with Agile - Software Development Using Scrum.* Addison Wesley.

Collins, E. F. (2012). Software Test Automation Practices in Agile Development Environment : An Industry Experience Report, 57–63.57, IEEE.AST 2012, Zurich, Switzerland.

Crispin, L. and Gregory, J. Agile Testing – A Practical Guide for Testers and Agile Teams.2009. Addison Wesley.

Crispin, L. and Gregory, J. More Agile Testing – Learning Journeys for the Whole Teams.2014. Addison Wesley.

Deak, A., and Stalhane, T. (2013). Organization of testing activities in Norwegian software companies. *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2013*, 102–107. doi:10.1109/ICSTW.2013.18

Deak, A. (2012). Understanding socio-technical factors influencing testers in software development organizations. *2012 IEEE 36th International Conference on Computer Software and Applications*. doi:10.1109/COMPSAC.2012.103

Fearthers, M. (2004). *Working Effectively with Legacy Code*. Robert.C.Martin Series. Prentice Hall.

Fewster, M. and Graham, D., Experiences of Test Automation: Case Studies of Software Test Automation, Addison Wesley. 2012.

Fitzgerald, B., Lenihan, H., Lopez-Gomez, C., and O'Sullivan, E. (2014). Irish Software Landscape Study. doi:10.1016/S0022-3913(12)00047-9

Frey, C.B., Osborne, M.A., The Future of Employment: How Susceptible are Jobs to Computerisation? (2013), Oxford Martin publication, Oxford University press.

Gelperin, D., and Hetzel, B. (1988). The growth of software testing. *Communications of the ACM*, *31*(6), 687–695. doi:10.1145/62959.62965

Gibson, C. Levy, A. (2012). Plenia Locatel Group: Globalizing from Venezuela. *CISR Research Briefing*.

Grier, D. A. (2011). Software Engineering : History. *Center for International Science and Technology Policy, George Washington University.*, 37–41. doi:10.1081/E-ESE-120044178

Agile Practice *THE Competitive Advantage for a A Digital Age, Harvard Business Review (2015).* https://hbr.org/resources/pdfs/comm/atlassian/atlassian2015.pdf [Last accessed 21st August 2015].

Heartbleed (https://en.wikipedia.org/wiki/Heartbleed)[last accessed 13th July 2015]

Hemmati, H., Fang, Z., and Mantyla, M. (2015). Prioritizing Manual Test Cases in Traditional and Rapid Release Environments.IEEE

Hunter, M. J. (2010). You Are Not Done Yet, (c).

Itkonen, J., and Lassenius, C. (2013). The Role of the Tester ' s Knowledge in Exploratory Software Testing, *39*(5), 707–724.

Itkonen, J., and Mäntylä, M. V. (2014). Are test cases needed? Replicated comparison between exploratory and test-case-based software testing. *Empirical Software Engineering*, *19*, 303–342. doi:10.1007/s10664-013-9266-8

Itkonen, J., Mäntylä, M. V., and Lassenius, C. (2009). How do testers do it? An exploratory study on manual testing practices. *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, 494–497. doi:10.1109/ESEM.2009.5314240

ITQSB (http://astqb.org/glossary/search/testing) [Last Accessed 26th August 2015]

Kalinowski, M., Teixeira, H. V., and Van Oppen, P. J. H. (2007). ABAT: An approach for building maintainable automated functional software tests. *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, 83–91. doi:10.1109/SCCC.2007.4396980

Kanij, T., Merkel, R., and Grundy, J. (2014). A Preliminary Survey of Factors Affecting Software Testers. *2014 23rd Australian Software Engineering Conference*, 180–189. doi:10.1109/ASWEC.2014.32

Karhu, K., Repo, T., Taipale, O., and Smolander, K. (2009). Empirical observations on software testing automation. *Proceedings - 2nd International Conference on Software Testing, Verification, and Validation, ICST 2009*, 201–209. doi:10.1109/ICST.2009.16

Kaur, K., and Kumar, S. (2014). Analysis of various testing techniques, *5*(3), 276–290. doi:10.1007/s13198-013-0157-6

Kitchin, R., and Dodge, M. (2011). Code / Space: Software and Everyday Life. *The MIT Press*.

Knight Capital - http://www.bloomberg.com/news/articles/2012-10-17/knight-capital-reports-net-loss-as-software-error-takes-toll-1-) [Last accessed 13th July 2015]

Mäntylä, M. V., Itkonen, J., and Iivonen, J. (2012). Who tested my software? Testing as an organizationally cross-cutting activity. *Software Quality Journal*, (Elsevier). *20*, 145–172. doi:10.1007/s11219-011-9157-4

Martin, R. C. (2011). *The Clean Coder - a code of conduct for professional programmers*. Prentice Hall (Robert.C.Martin Series).

Mcconnell, S. (2009). *Code Complete 2*. Microsoft Press.

Meerts, J. The History of Software Testing,

http://www.testingreferences.com/testinghistory.php [Last accessed 18th August 2015].

Focus, M. (2011). Software Testing : IT ' s Invisible Giant.

JPMorganTradingBug
https://en.wikipedia.org/wiki/2012_JPMorgan_Chase_trading_loss
[last accessed July 13th 2015].

Morgan, P., and Samaroo, A., and Hambling, B., (2010), Software Testing: An ISTQB Foundation Guide, British Computer Society.

Mulder, D. L., and Whyte, G. (2013). A Theoretical Review of the Impact of Test Automation on Test Effectiveness. *Proceedings of the European Conference on Information Management and Evaluation*, (2005), 168–179. Retrieved from http://search.ebscohost.com.ezproxy.liv.ac.uk/login.aspx?direct=trueanddb=bthandAN=87385174andsite=eds-liveandscope=site [Last accessed 26th August 2015]

Mulder, D. L., and Whyte, G. (2011). Mitigating the Impact of Software Test Constraints on Software Testing Effectiveness. *Ejisecom*, *14*(2), 254–270. Retrieved from http://www.ejise.com/issue/download.html?idArticle=776 [Last Accessed 26th August 2015]

Myers, G. (2004). *The Art of Software Testing, Second edition*. *Software Testing, Verification and Reliability* (Vol. 15, p. 151). doi:10.1002/stvr.322

Lanier, J. (2010). You Are Not a Gadget: A Manifesto. *Renaissance Quarterly*, *30*, 224. doi:10.2307/2860049

V. S., Nadar, S., Exchange, N. S., Sundari, V.,. (2015). Software Industry Snapshot. *FRPT- Software Snapshot. 1/12/2013, p18-18. 1/2p.*, 25–27.

Operation Olympic Games Software Bug (Last accessed 13th July 2015) https://en.wikipedia.org/wiki/Operation_Olympic_Games

Page, A. Johnson, K. Rollinson, BJ. Microsoft Press, How We Test Software At Microsoft, 2008.

Petersen, K., and Mantyla, M. V. (2012). Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. *2012 7th International Workshop on Automation of Software Test (AST)*, 36–42. doi:10.1109/IWAST.2012.6228988

Remenyi, D., Research Strategies – Beyond the Differences
Electronic Journal of Business Research Methods, Volume 1 Issue 1 (2002) 38-41

Remenyi, D. and Money, A. (2004) .*Research supervision for supervisors and their students.* Academic Conferences Limited, Oxford, pp280. ISBN 0954709608

Remenyi, D., Money, A., Price, D. and Bannister, F. (2002) research. Irish Journal of Management, 23 (2). pp. 1-17. ISSN 1649-248X

Remenyi, D., Field Methods for Academic Research: Interviews, Focus Groups and Questionnaires, 2013, ACPIL.

Saunders MNK, Lewis P, Thornhill A. (2012) *Research Methods for Business Students*. 6 Edition. Harlow : Pearson

Shah, H., and Harrold, M. J. (2010). Studying human and social aspects of testing in a service-based software company: Case study. *Proceedings - International Conference on Software Engineering*, 102–108. doi:10.1145/1833310.1833327

Shapiro, S. (1997). Splitting the difference: The historical necessity of synthesis in software engineering. *IEEE Annals of the History of Computing*, *19*(1), 20–54. doi:10.1109/85.560729

Taipale, O., Kasurinen, J., Karhu, K., and Smolander, K. (2011). Trade-off between automated and manual software testing. *International Journal of Systems Assurance Engineering and Management*, *2*(June), 114–125. doi:10.1007/s13198-011-0065-6

Weinberg, G. The Psychology of Computer Programming, 1998, Dorest House Publishing Company.

Whittaker, J., How to Break Software, Pearson, 2002.

Whittaker, J,. Exploratory Testing, Addison-Wesley Professional, 2009.

Whittaker, J., How Google Tests Software, Addison-Wesley Professional, 2012.

Wynne, M., and Hellesoy, A., The Cucumber Book: Behaviour-Driven Development for testers and Developers, 2012, Pragmatic Programmers Bookshelf.

Yin, Robert K. (2003). *Case study research, design and methods* (3rd ed., vol. 5). Thousand Oaks: Sage.

Zhang, X., Stafford, T. F., Dhaliwal, J. S., Gillenson, M. L., and Moeller, G. (2014). Sources of conflict between developers and testers in software development. *Information and Management*, *51*, 13–26. doi:10.1016/j.im.2013.09.006

Zhivich,Michael, Cunningham, Robert. (2015). The Real Cost of Software Defects. *MIT Lincoln Laboratory*, 1–8. (IEEE)

Appendix 1 Interview Questions

1. What is your role in the organisation?
2. How long have you been in your current position?
3. How long have you worked in software testing?
4. Did you study Computer Science?
5. How did you get into testing?
6. What way would you describe software testing to someone not involved in IT?
7. What types of testing do you perform as part of your work?
8. Have you worked on Waterfall structured projects?
9. Have you worked on Agile structured projects?
10. In your opinion what changes has Agile brought to software testing?
11. Have you been involved in test automation?
12. Have you a learned programming language?
13. What tools have you used in relation to automation?
14. What problems have you encountered with test automation?
15. Do you think the role of the tester has changed during your career?
16. Do you think test automation will become increasingly more important?
17. What do you think the future for software testers looks like in terms of employment?
18. Do testers need to adapt? In what ways?