2013

Using Grounded Theory to Develop a Framework for Software Testing Best Practice in a Telecommunications Company

David Hendrick
Technological University Dublin, david.hendrick@tudublin.ie

Using Grounded Theory to Develop a Framework for Software Testing Best Practice in a Telecommunications Company

David Hendrick

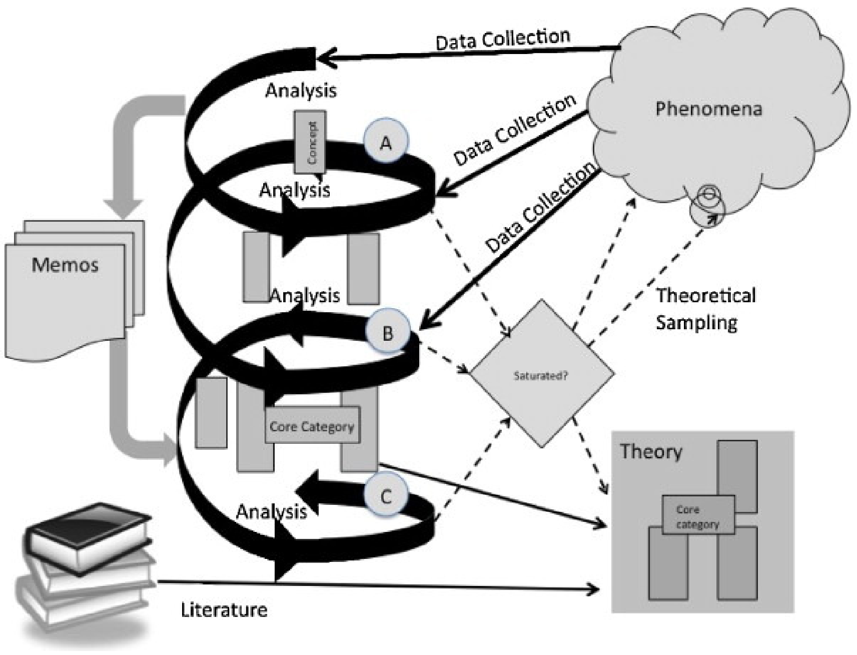
A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Information Technology)

January 2013

Grounded Theory

Strauss and Corbin (1998) version of Grounded Theory:

1. Open Coding
2. Axial Coding
3. Selective Coding



I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Knowledge Management), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: ___________________________________

Date: ***28 January 2013***

ABSTRACT

Software testing is a key part of the software development process, irrespective of the development methodology being followed. Estimates suggest that testing can account for more than 50% of the cost of a software development project. However, the cost of inadequate testing or verifying software can be far greater; this can result in losses that could total more than 10% of an organisation's turnover. It is clear that software testing is essential; but while many companies implement different forms of testing, there is often no structure or best practice followed to this testing. There appears to be a gap in terms of research into software testing best practice.

It has been shown in previous research that qualitative research methods can be used successfully when conducting research in the software engineering domain. In particular Grounded Theory was seen as very suitable for research in this area. The basis of Grounded Theory is in the formation of a theory from the data that is gathered. As such it is considered reflective of the reality of situations.

This dissertation aimed to create a model for testing best practice in the telecommunications industry using a qualitative research methodology. It aimed to develop a Grounded Theory that could be used to guide testing within a large telecommunications company.

Results suggested that the Grounded Theory developed could be used to improve project quality and reduce time to market for projects within this organisation.

Key words: *grounded theory, software testing, telecommunications, user acceptance testing, qualitative research methods*

ACKNOWLEDGEMENTS

Firstly I would like to express my sincere thanks and gratitude to my supervisor Damian Gordon. His enthusiasm, support and guidance helped me immensely over the course of this dissertation.

I would also like to thank my employer who was very supportive of this research. Particular thanks go to the IT Delivery Lead for his supportive attitude and allowing me the space and time I required to complete this dissertation.

Special thanks also to my parents for their constant support and encouragement over the years.

Finally, thanks to Emer for her understanding and support while I worked on this dissertation.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1 Background

The process of developing and releasing software is a complex and expensive one and is fraught with risk (Adolph *et al.* 2012). It is also a big business with estimates of it being a $1.6 trillion industry (Bartels *et al.* 2006 in Adolph *et al.* 2012). Because of this, we can imagine that organisations would want to ensure that the process for developing and testing this software that they have implemented are correct and being used properly.

As the complexity (and cost) of these software projects increases areas such as quality, reliability and the customer acceptance are critical to the organisations that produce this software (Huang 2005). The methods to ensure there is this reliability and quality is the process of verification and validation of the software; software testing being the most widely used method (Whyte and Mulder 2011). Undertaking software testing can potentially be a manual and tedious process; yet it is an essential phase in software development models (e.g. the V-model). Estimates suggest that testing can account for more than 50% of the cost of a software development project (Whyte and Mulder 2011). However, the cost of inadequate testing or verifying software can be significantly greater; according to Engel and Last (2007) this can result in losses that could total more than 10% of an organisation's turnover. When viewed in this light it is clear how important the testing of software is.

What is a little less clear is how human and social factors can effect efficient and cost effective testing. In fact this is an area that is somewhat neglected in Software Engineering research, which normally tends to focus on processes, producers, models tools etc (Adolph *et al.* 2012). However, there have been a number of studies that show that societal factors and individuals abilities can highly significant in the development process. It is suggested that social factors can be significant cost drivers on software projects, often eclipsing other factors (Cockburn 2002).

There is potentially significant benefit from investing time into an investigation of the role social factors play in Software Engineering and especially the areas of software development and testing. One method that has been used by a number of researchers is that of Grounded Theory; which is a qualitative research method that was first proposed by Glaser and Strauss in 1967 where researchers construct theories from the data that they collect (Glaser and Strauss 1967). The theoretical foundations that underpin Grounded Theory can be traced back to the area of *Symbolic Interactionism*, a theory that '*sees humans as key participants and 'shapers' of the world they inhabit*' (Coleman and O'Connor 2007, p.4). Grounded theory has been shown to be useful in several social science fields and can help to explain how certain behaviours and actions shape social processes through human interactions (Glaser and Strauss 1967). The thrust of approach is that theories grow organically from the data that is collected and it is important that researchers don't have pre-conceived theories. It is nearly the opposite of the existing scientific method where researchers will have a hypothesis or theory and then go about conducting experiments to hopefully fail to disprove the theory. In grounded theory you conduct the experiment (or data collection) and the theory should emerge from this (Coleman and O'Connor 2007). It is argued that the theories derived from this gathered data are more realistic and likely to resemble the actuality of the situation (Strauss and Corbin 1998).

Within the telecommunications industry there are a number of methods used in software development practices. Within the large telecommunications company that this project will concentrate on there is a currently a "one size fits all" approach to processes for project testing in the SDLC. In this company there are a number of processes and procedures that were developed and implemented when the team was first created over 2 years ago. This was a "one size fits all" model that didn't cater for the different types of projects that the team are engaged on. Thus, over time these have been found to be inflexible, laborious and no longer applicable to the teams work. Thus, it is feasible that new processes would be required that would match the way in which testing is currently undertaken, ones that would take into account the varied nature of the projects undertaken by the team. This thesis will attempt to tackle the problem as to whether or not Grounded Theory can be used to develop a framework that be used to guide and inform types of testing that should be undertaken on the various projects the team are engaged in.

Grounded Theory can be thought of as being quite simple conceptually but is rigorous and scientific in its practice (Adolph *et al*. 2012). The figure below highlights the basic processes involved in developing a Grounded Theory.

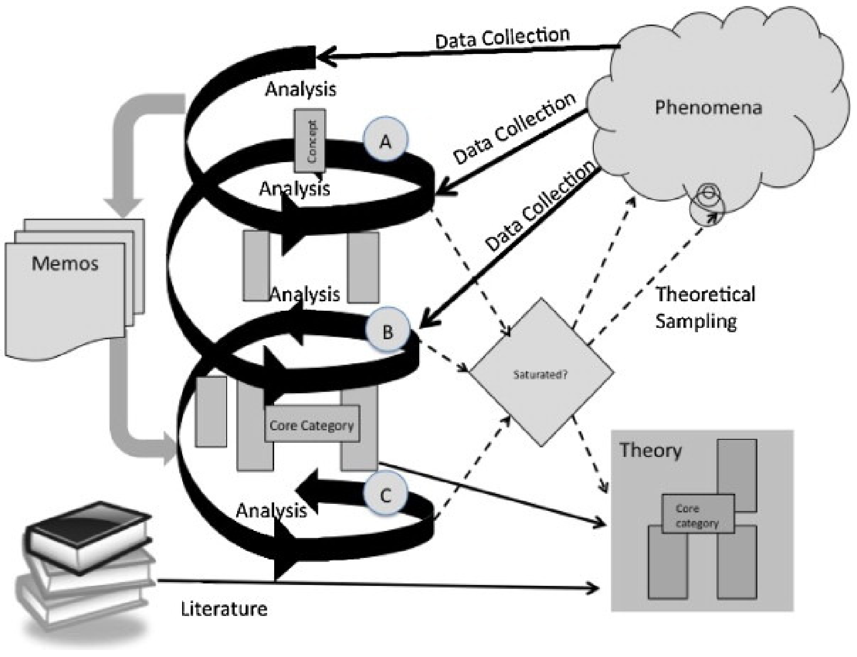


Figure 1.1: The Grounded Theory Method (Adolph *et al*. 2012)

There are a number of stages to the process, the first step involves gathering (usually via interview) and coding data; this is followed by breaking down the data into distinct units or concepts (Coleman and O'Connor 2007). These concepts are the fundamental building blocks of Grounded Theory (Adolph *et al*. 2012). The concepts are then re-interpreted and re-evaluated and inter-relationships between theories are recorded. Through a process of repetition gradually more and more sub-concepts are subsumed by overarching core concepts. From these, gradually, the theory emerges (Coleman and O'Connor 2007).

Traditionally the concept of Grounded Theory has been used to explain socio-cultural items of interest (e.g. in fields of sociology, nursing and psychology) there is a growing body of research that suggests that it can be used to explain phenomena in the information systems development domain. Research suggests that Grounded Theory is gaining a greater acceptance in the areas of Software Development and Information Systems as it can be seen to be an effective way of explaining phenomenon under study in a context based and process orientated fashion (Myers 1997). A number of researchers have used Grounded Theory to investigate various aspects of Software Engineering such as studying how IT is used in practise (as opposed to the theory behind it) (Goede and De Villiers 2003) and for requirements documentation (Power 2002). It has also been concluded by Hansen and Kautz (2005 in Coleman and O'Connor 2007) that Grounded Theory was a methodology that is well suited to research in the Information Systems.

This dissertation will propose to apply Grounded Theory research to the area of software testing. Software testing is one the of the ways that of ensuring quality in a software product (Whyte and Mulder 2011). The testing process shouldn't be considered as one phase in a project but something that will take place over the entire software development lifecycle (Baresi and Pezze 2006). Software testing should start as early as possible, requirements should be analysed to ensure they are testable, this is known as static testing(ISTQB 2011). There are a number of other types of testing that can be under taken over the duration of a project (e.g. Unit/Module Testing, System Testing, Integration Testing, Performance Testing, User Acceptance Testing). The selection of the types of tests to be completed during a testing event is a hard process and '*requires deep experience of software validation*' (Baresi and Pezze 2006, p.90).

1.2 Research problem

The research problem being investigated in this project is to explore if Grounded Theory can be used to develop a framework for best practice in Software Testing in a large telecommunications company.

1.3 Research objectives

The aim of this project is to create a Testing framework for applying to software projects within a large telecommunications company. In order to achieve this there are a number of objectives that need to be completed:

1. Perform a literature review in the area of Software Development methodologies with a particular focus on Software Testing

2. Perform a literature review on Research Methodologies (both Qualitative and Quantitative) with a particular focus on Grounded Theory and the application of this theory to software engineering projects
3. Develop a framework and theory for best practice in software testing in a telecommunications company
4. Test the validity of the framework and theory by retrospectively applying it to two recently completed projects in the company
5. Evaluate the results and if appropriate recommend adoption and roll out of framework across IT projects.

1.4 Research methodology

Primary and secondary research methods will be used to conduct this research, the primary method will be structured interviews with Test team members. These interviews will be used as a method of developing the Grounded Theory. Grounded Theory involves manual transcriptions of the entire interviews verbatim and picking out relevant themes so that the theory emerges from the data. The secondary research methods that will be used was extensive literature reviews that will be conducted to help achieve the research objectives.

1.5 Scope and limitations

The scope of this project will specifically focus on the development of a framework for a telecommunications company. Some of the findings may not be applicable to other industries, but many will be.

It will not be developing a new methodology for software testing. Neither will it be developing a new approach to software testing; rather it will be looking at the best available testing techniques and the best sequencing in which those techniques should be used.

1.6 Organisation of the dissertation

Chapter 2 will be an introduction to the areas of software project development. There will be discussion around the more rigid models such as the Waterfall model. As an alternative to this; more modern practices such as Agile development will be discussed. What should be clear from these discussions is the importance that testing plays in the process of software development. Following this a detailed description of software testing will be presented.

Chapter 3 will outline the Research Methodologies with an emphasis on Ground Theory. It will first outline the different types of Research Methods (Qualitative Vs Quantitative). It will then detail the 5 types of Qualitative Methods (of which one is Grounded Theory). Next there will be a detailed investigation of Grounded Theory.

Chapter 4 will give an introduction to the problem to be resolved and an outline of the ways in which Grounded Theory can be used to develop the framework. This Chapter will involve discussion of the company within which the study took place. It will detail how testing currently takes place within the organisation to give an "As-Is" view of the situation.

Chapter 5 will contain an outline of how the data was gathered. An explanation of the interviews and how these fed into the Grounded Theory will be discussed in detail. There will also be an evaluation of the existing documentation within the organisation.

Chapter 6 will outline the creation of the Grounded Theory. This will be the experimentation chapter. It will outline in detail how the framework was created, what the recommendations are and how these should be employed.

Chapter 7 will be the evaluation chapter where the success of the objectives of the dissertation will be discussed. In order to evaluate the validity of the Framework it will be applied retrospectively to two recently completed projects within the organisation.

Chapter 8 will be the conclusion chapter. This chapter will re-visit the research objectives and evaluate if the project has met those objectives. It will also outline how the dissertation contributes to the body of knowledge on this subject and outline recommendations for future work.

2. TESTING AND SOFTWARE DEVELOPMENT METHODOLOGIES

2.1 Introduction

The aim of this dissertation is to develop a framework or Grounded Theory for software testing best practice within an organisation. Before this can be assessed it is necessary to have some background knowledge of how companies develop and release software and where Testing sits within this. Thus, this chapter will detail the different activities that are involved in the process of developing software. It will then highlight the various methodologies used to develop software. Finally, there will be an in depth analysis of the various types of testing that can take place on a project. Once this is complete and an understanding of how software is developed then the place software testing takes in the overall software development lifecycle should be clear.

2.2 Traditional Software Development Lifecycle

The process of developing software involves a number of different steps and activities. There are also a number of philosophies and methodologies that underpin how software is developed. The type of organisation that you work in, the type of software being developed and the experience and preference of team members will all contribute to the methodologies used. The first that we shall examine is the traditional software development lifecycle where projects are modelled following rigid stages from Requirements to Analysis to Development to Test to Deploy.

2.2.1 Waterfall Model

The traditional way of thinking about software development was first proposed by (Royce 1970). In this seminal paper Royce discussed the methods that were used in software development (later to be termed the "Waterfall Model) and highlighted the major flaws with this system. In this traditional system there are major phases that all projects "flow" through, from Requirements to Deployment (or Operations).

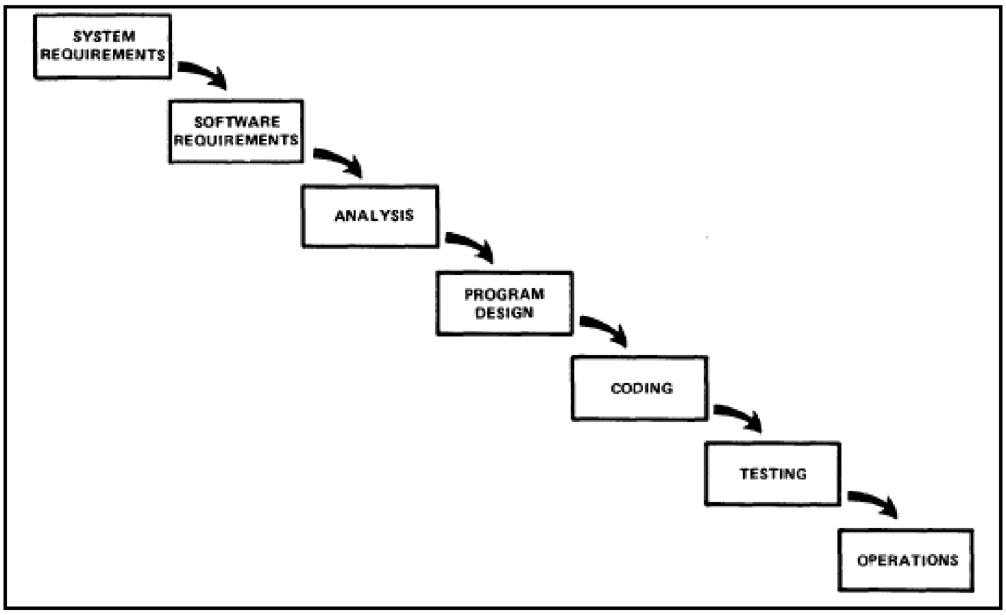


Figure 2.1: Traditional Waterfall Development Model (Royce 1970)

This is thought to be the first incarnation of the waterfall model (although the term 'Waterfall' is never used (Weisert 2003)). In the article Royce outlines the way that software is traditionally built. In this paper he argues that the two central pieces of software development are the design and coding steps. But there are other key steps that must be also be undertaken (analysis, test etc). The main criticisms of the model by Royce are:

1. The fact that testing happens at the end of the project where it may be too late or costly to fix some design issues
2. The sheer inflexibility of the model, there is no scope for changes to requirements etc as once you have passed one stage you can't go backwards (much like a water can't flow back "up" a waterfall)

While Royce was against the traditional ways of software development as outlined in this review, he proposed ideas to make the process better. Some of these ideas can be used if you were to follow the Waterfall model (e.g. that of Document the Design where Royce espouses the need for rigorous documentation) and others if you were to follow the newer, leaner approaches (e.g. that of iterative design to build to test cycles) (Royce, 1970).

2.2.2 Spiral Model

The Spiral Model is a software development model proposed by Barry Boehm that blends an iterative approach to development with the linear approach of the Waterfall Model (Boehm 1986). It has evolved from the experience and refinement of the Waterfall Model when it was applied to large American Government projects. It is seen that the Spiral Model can incorporate most of the previous models developed. The Spiral Model proceeds iteratively through four key phases.

1. Determine the objectives: This involves identifying the objectives of this portion of the project and any alternative ways of implementation.

2. Identify and Resolve Risks: This step involves identifying areas of uncertainty and make recommendations for mitigation of risks.

3. Development and Test: In this phase the prototypes are developed and tested in a linear fashion as in the Waterfall model where there will be a phase of Design, then Coding followed by Integration, Test and Deployment.

4. Planning for the Next Phase: This is the final step in the model and involves completing the planning for the next iteration of the model (Boehm 1986).

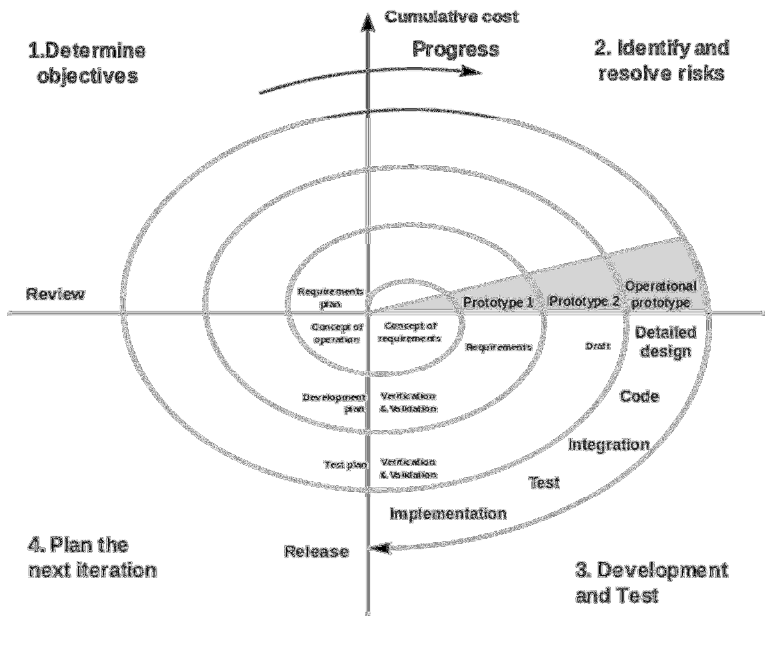


Figure 2.2: Spiral Model (Boehm 1986)

2.2.3 V-Model

The typical V-model represents software development processes similar to the traditional Waterfall Model. The major difference is that rather than move down the waterfall process steps climb back up following the coding phase. The V-Model attempts to represent how the phases before the test phase relate to the different test phases on a project. Each phase in the development stages maps back to a testing phase (ISTQB 2011). It also outlines what test activities should be taking place during the related development activities (e.g. during the Design phase the design of the application should be being developed along with the design of the test phase).

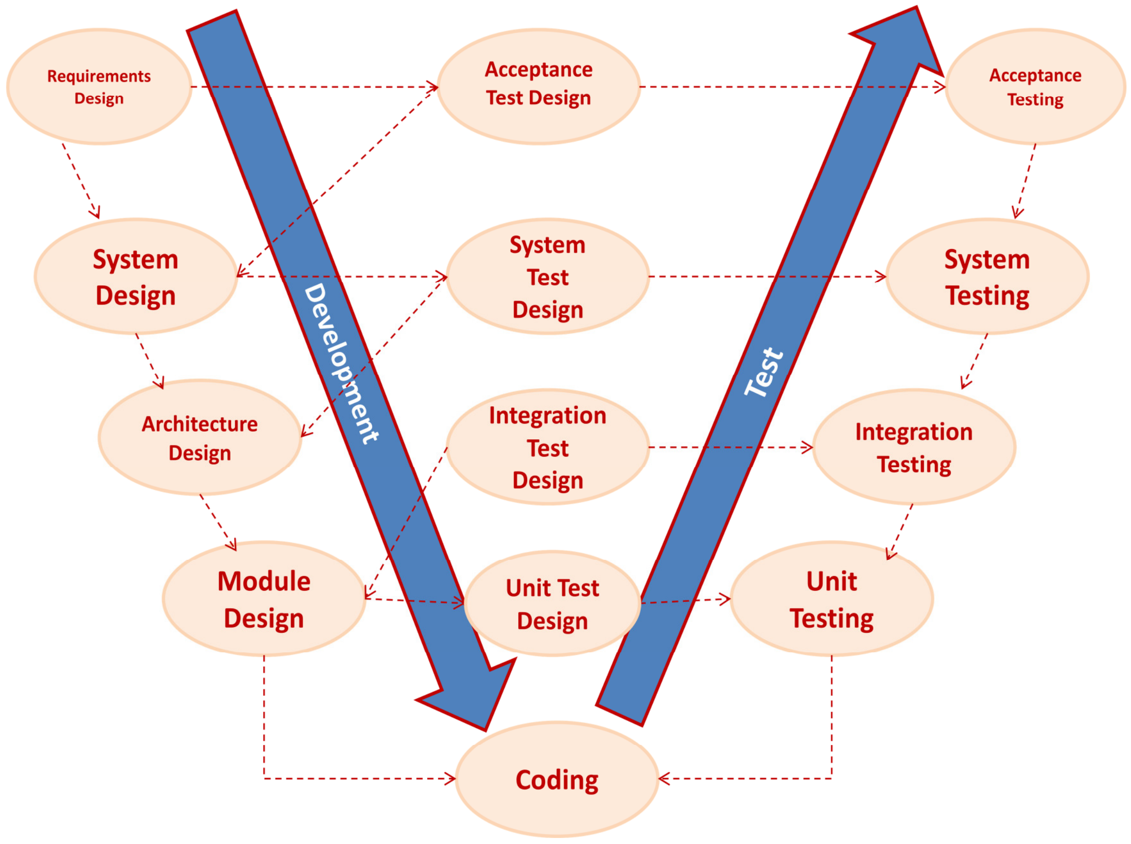


Figure 2.3: The V-Model (One Stop Testing n.d.)

2.3 Agile Software Development

What does it really mean to be Agile? We understand that in the traditional sense of software development we get some requirements, build a piece of software, test it and release it. However, by the 1990's the increased complexity of software and the inability of customers to state upfront their full list of requirements this traditional model became hard to work with. It was becoming clear that these traditional, plan driven methods were too rigid and inflexible and that they can't respond fast enough to changing project requirements and new project environments (Erickson *et al.* 2005).

Thus, there was a need for a new model, one that embraced change and recognised uncertainty. One set of ideas to overcome this became known as "Agile" methods, ones which place an emphasis on people and creativity rather than processes (Dybå and Dingsøyr 2008). A good definition of what is means to be agile is given by Erickson *et al.* (2005 p.89):

> *At its core, agility means to strip away as much of the heaviness, commonly associated with traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines*

The beginnings of the Agile manifesto dates back to the mid-1990's where a number of researchers working independently came up with different methodologies such as Dynamic Systems Development, Feature Driven Development and XP (Williams and Cockburn 2003). These practitioners all met in 2001 to discuss the similarities of their methods. They found that there were a number of similarities in their various methodologies and used the term "agile" to categorise them (Williams and Cockburn 2003).

The participants then wrote the "Manifesto for Agile Software Development". In this they described the four core categories that describe the similarities between the methods:

- ***individuals and interactions over processes and tools,***
- ***working software over comprehensive documentation,***
- ***customer collaboration over contract negotiation,***
- ***responding to change over following a plan.***

(Williams and Cockburn 2003)

This then gave rise to The Agile Manifesto website. On this website the contributors outline their 12 principles of Agile Development:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Figure 2.4 Agile Manifesto (Manifesto for Agile Software Development 2001)

In the model of Agile development the focus is on teams continually delivering software in short iterations (Moe *et al*. 2012). There is an emphasis on close collaboration, people are seen as more important than processes and the whole operation should be "leaner". This was taken by some to mean that there should be no documentation on the project. In reality what it means is that there should only be the minimum amount of documentation; only spend time documenting what is important (Dingsøyr *et al*. 2012). We can look at planning as an example of this. In the traditional context of a plan driven environment the "plan" would include documented processes and tasks that need to be completed by certain milestone dates etc and all formulated in various plans (e.g. project plan, design phase plan, test plan etc). In the agile world, the emphasis would not be on the output documents but more concerned with the planning process, thus, this is why these methods can appear to be less planned than they actually are (Boehm 2002). Figure 2.3 below shows where the agile methods sit on the planning spectrum. To the far left are the hackers, with no plans at all and to the far right are heavily contracted, milestone driven projects, agile methods sit closer to the left hand side, but still consider planning in their processes (Boehm 2002).

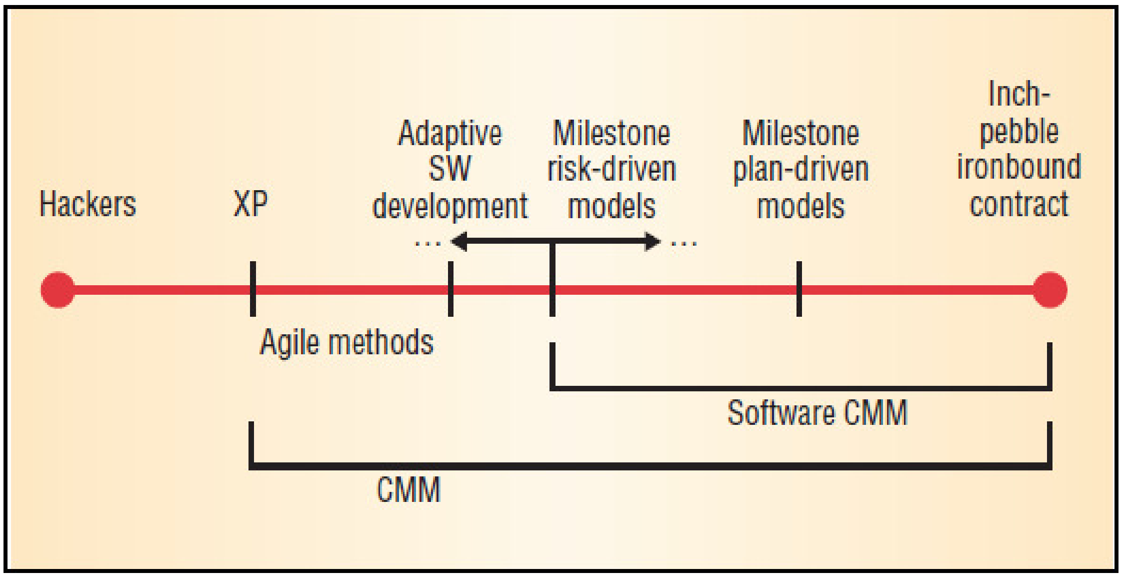

Hackers
XP
Adaptive SW development
Milestone risk-driven models
Milestone plan-driven models
Inch-pebble ironbound contract
Agile methods
Software CMM
CMM

Figure 2.4: Outlines where Agile methods sit on planning spectrum (B. Boehm 2002)

Another important factor was the customer (Beck *et al*. 2001). They shouldn't lie at the fringes of the development process, but at the heart of it. Finally, there was agreement and an acceptance that there was a level of uncertainty that came with the process of software development and that to try to control and eliminate all elements of uncertainty was futile (Dingsøyr *et al*. 2012).

When the agile method was first proposed over 10 years ago naturally the first number of discussions were around the merits of when you should use a the traditional Waterfall approach over when you should use the agile methods (Boehm and Turner 2004). However, there are also concerns over the various methodologies of Agile development. The main methodologies will be covered in the next section of the dissertation.

2.3.1 Scrum

A popular Agile framework is the Scrum framework and was first proposed by Schwaber (1995). The term is derived from the game of rugby where a 'scrum' is a method of getting a ball that is out of play back into play as quick as possible with teamwork (Schwaber and Beedle 2001). The Scrum is a method for managing the software development process and it does not prescribe the use of any particular software development technique (Abrahamsson *et al.* 2002). It is about team-members working together to deliver working software in a flexible environment (Schwaber 1995).

The Scrum involves three phases: Pre-Game, Development and Post-Game. During the Pre-Game Phase high-level planning and design take place as well as the development of the list of requirements for the system in the Product Backlog. Next there is the Development Phase, where the requirements from the Product Backlog are divided out into separate Sprints (or min-release cycles). The aim being that at the end of each Sprint a working piece of software is delivered. Sprints will last from one week to one month and there may be between 3 and 8 Sprints per release. Once the Sprints are complete we move to the Post-Game phase which consists of system testing and releasing the product as well as completing all required documentation (Abrahamson *et al.* 2002).

While Scrum does not prescribe any particular software development techniques Scrum does have some software practices and tools to help the development phases (Schwaber and Beedle, 2001):

> The first that will be mentioned has been outlined earlier, that is the Product Backlog. This "defines everything that is needed in the final product based on current knowledge" (Abrahamson *et al.* 2002, P.34). It can contain new features or functionality, bugs, defects, customer requests for functionality or technical upgrades. Generally the Product Backlog will be created with input from many sources e.g. customers, Quality Assurance, Sales, Product Owners, Management (Schwaber 1995).
>
> The next element in the process we will discuss in more detail is the Sprint. These can be considered a procedure for adapting to the constantly changing

environmental variables in a software development project (Abrahamson *et al.* 2002). A Sprint is a process whereby the team undertakes development work to deliver working software. There tolls used in the Sprint are: Sprint Planning, Sprint Backlog and the Daily Scrum Meeting. Sprint Planning involves a two-phase meeting organised by the Scrum Master (project manager) and includes users, management, development team and the Product Owner. During this session the functionality to be delivered in the Sprint is finalised. The output from this phase is the Sprint Backlog which lists all items to be included in that Sprint. Finally, there are daily stand-up meetings to track progress. The meetings are very shot in duration (less than 10 minutes) and involve each member of the team detailing three things: what they have done since the last meeting, what they will do until the next one and any problems they have been having. (Schwaber and Beedle, 2001).

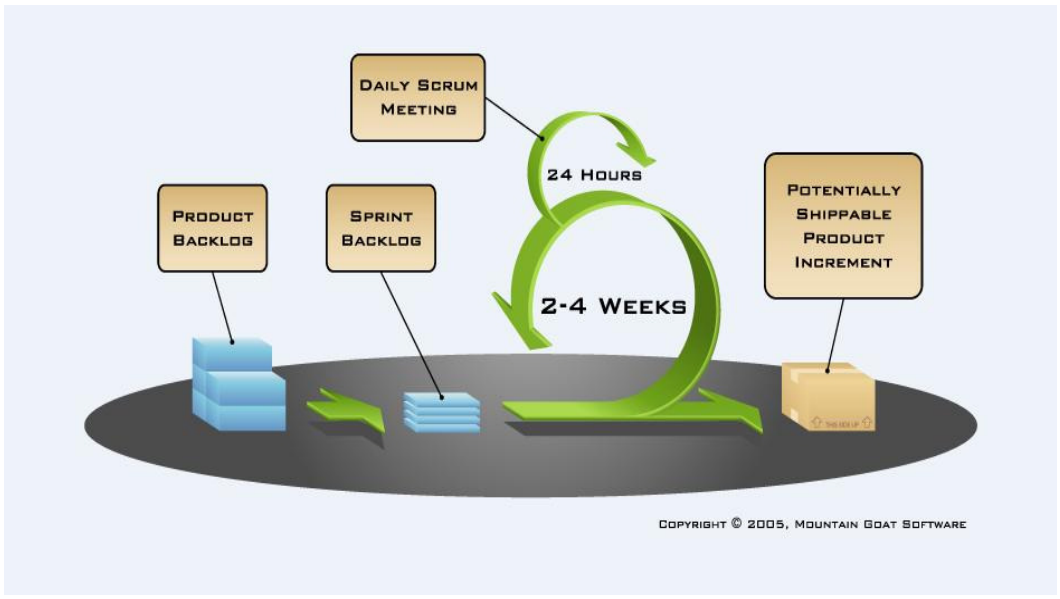

Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

Figure 2.5: Sprint Cycle (Kumana and Dickinson 2009)

In terms of successful experiences using Scrum Rising and Janoff (2000) report its successful use in three development projects. But the use of Agile and Scrum are not seen to be fully applicable to large complex projects (Abrahamson, *et al.*, 2002) it has been argued that small teams within large projects could use elements of the methodology to great success (Rising and Janoff 2000).

2.3.2 Extreme Programming (XP)

The idea of XP is often credited to Kent Beck and his book Extreme Programming: Embrace Change (Beck 2000).. XP consists of 4 key ideas:

- continual communication with the customer and within the team;
- simplicity, achieved by a constant focus on minimalist solutions;
- rapid feedback through mechanisms such as unit and functional testing; and
- the courage to deal with problems proactively (Beck 2000)

What can be seen from looking at XP in practice is that many of their key principles (such as minimalism, simplicity and user involvement etc) are present in any disciplined process. What makes the difference in XP is that these are brought to the "extreme". For example the idea of simplicity means focusing on the key, high priority requirements first and worry about the big design and solutions to problems that haven't occurred later (Paulk 2001).

There are typically 12 basic elements to XP:

1	Planning Game:	This ensures that the scope of upcoming releases can be quickly scoped
2	Small Releases:	Should have short development iterations and put a working piece of software out on a short cycle (e.g. every 2 weeks)
3	Metaphor:	This should be used to guide development, a simple story that outlines how system should work
4	Simple Design:	Keep the design as simple as possible
5	Testing:	Testing should take place continually throughout development
6	Refactoring:	This involves removing duplication, simplification, adding flexibility etc by restructuring systems but not making changes
7	Pair Programming:	Code is written in pairs, one programmer writes, the other reviews what is being written
8	Collective Ownership:	The system is free to be improved by anyone at anytime
9	Continuous Integration	This involves continually building, integrating and regression testing the system many times a day
10	40-hour week:	Only work 40 hours in a week (when possible)
11	Onsite customer:	You should have the customer onsite to answer queries
12	Coding standards	Ensures that code is adequately commented

Figure 2.6: 12 Practices of XP (Beck 2000)

Thus, by using these practices together ensures that XP is a '*coherent method*' (Paulk 2001 p.4) for developing software.

2.3.3 Crystal Methods

The next method in the Agile stack is that of the Crystal Methods, which are a family of methods that vary based on size and complexity of project (Coffin and Lane 2006). This approach was developed by Alastair Cockburn developed from interviews and discussions with many project teams about the best way to develop software (Cockburn 2004). This approach puts an emphasis on close communication between team members working in small teams (Dybå and Dingsøyr 2008). There are several "flavours" of Crystal that should be chosen depending on project criticality, these are colour coded to represent geological crystal's hardness (Coffin and Lane 2006). The figure below shows how the type of Crystal method for the project should be selected. As the size of the project increases (denoted by the numbers at the bottom of the model, these represent the upper limit to project team size) then it becomes harder and moves to the right in the figure, and therefore a more comprehensive method is required, such as Maroon (Cockburn 2004). As the criticality of the project increases (i.e. moves from bottom to top) then additional aspects of the method need to be put in place to accommodate the increased complexity (Coffin and Lane 2006).

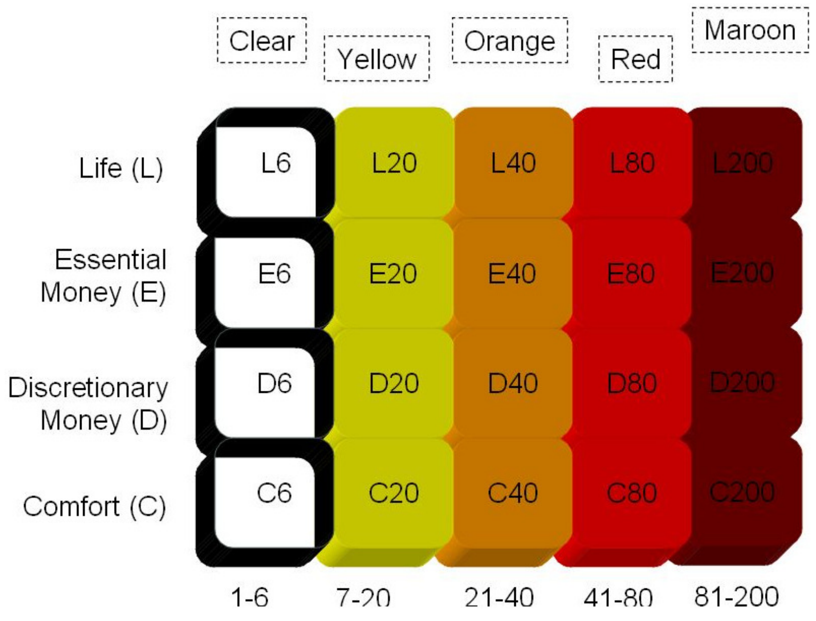


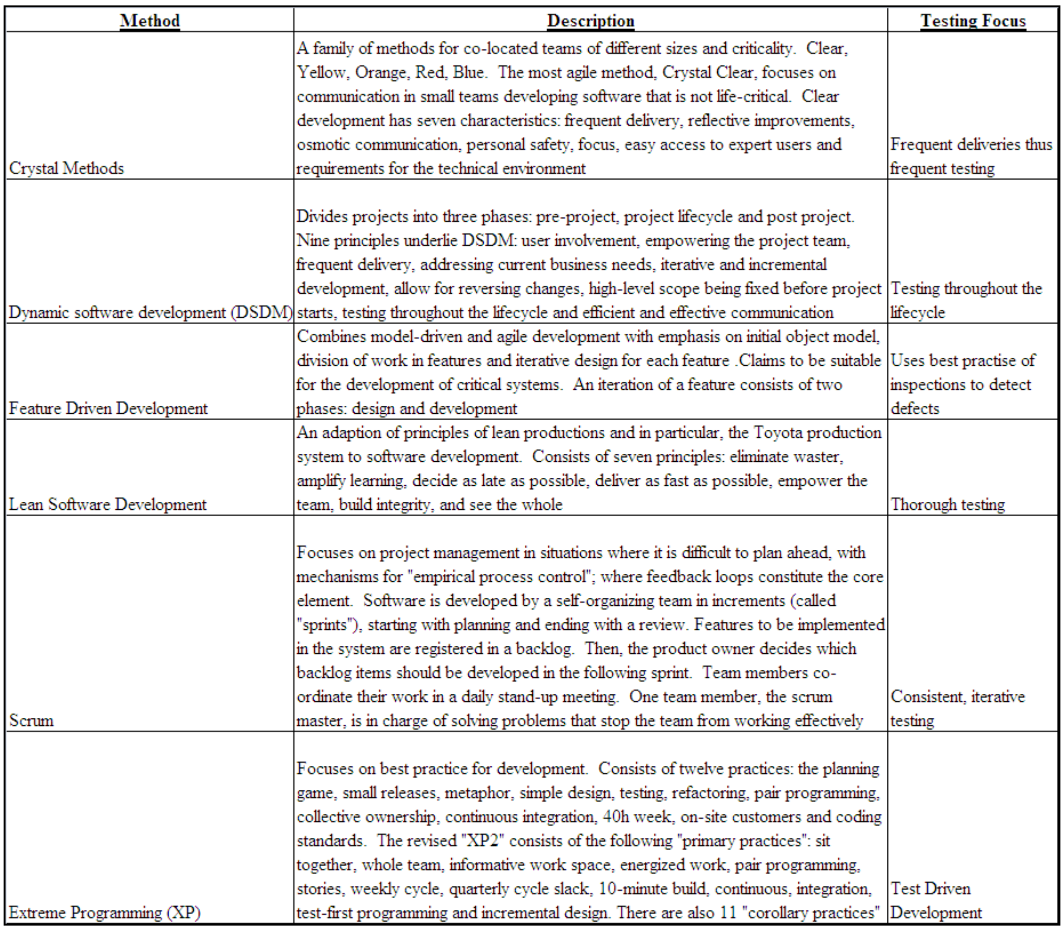
Figure 2.7: The Crystal Family of Methods (Coffin and Lane 2006)

As with XP, the Crystal methods are based on key principles (seven this time):

- **Frequent Delivery:** Working software should be delivered every couple of months
- **Close Communication:** Small teams should be working in the same room and talk often
- **Reflective Improvement:** The project team should meet continual and discuss progress to ensure project is on track.
- **Personal safety:** there are two elements to this. The first is in relation to people mush feel safe in their ability to express the truth on projects without fear of recrimination. Secondly, safety refers to the fact that not all projects are delivering the same type of software, e.g. and Air Traffic Control software would be more critical than an app for organising shopping lists
- **Focus:** Teams should know the top priorities for their teams and be given time to work on this
- **Easy Access to Expert Users:** As with XP (and most Agile methods), this method assumes that customers are based on site to answer queries quickly.
- **Frequent Integration:** The same concept as XP, that is, continually building, integrating and regression testing the system many times a day (Cockburn 2004).

2.3.4 Recap on Agile Development

As noted at the outset there are a number of software development methodologies that can be called Agile. These were developed by the group known as the Agile Manifesto in 2001. This group published their manifesto which included findings from the various methods that each was working on (as they found they shared many of the same methods). Thus, in order to recap and outline the different methods in the Agile family.



Method	Description	Testing Focus
Crystal Methods	A family of methods for co-located teams of different sizes and criticality. Clear, Yellow, Orange, Red, Blue. The most agile method, Crystal Clear, focuses on communication in small teams developing software that is not life-critical. Clear development has seven characteristics: frequent delivery, reflective improvements, osmotic communication, personal safety, focus, easy access to expert users and requirements for the technical environment	Frequent deliveries thus frequent testing
Dynamic software development (DSDM)	Divides projects into three phases: pre-project, project lifecycle and post project. Nine principles underlie DSDM: user involvement, empowering the project team, frequent delivery, addressing current business needs, iterative and incremental development, allow for reversing changes, high-level scope being fixed before project starts, testing throughout the lifecycle and efficient and effective communication	Testing throughout the lifecycle
Feature Driven Development	Combines model-driven and agile development with emphasis on initial object model, division of work in features and iterative design for each feature .Claims to be suitable for the development of critical systems. An iteration of a feature consists of two phases: design and development	Uses best practise of inspections to detect defects
Lean Software Development	An adaption of principles of lean productions and in particular, the Toyota production system to software development. Consists of seven principles: eliminate waster, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity, and see the whole	Thorough testing
Scrum	Focuses on project management in situations where it is difficult to plan ahead, with mechanisms for "empirical process control"; where feedback loops constitute the core element. Software is developed by a self-organizing team in increments (called "sprints"), starting with planning and ending with a review. Features to be implemented in the system are registered in a backlog. Then, the product owner decides which backlog items should be developed in the following sprint. Team members co-ordinate their work in a daily stand-up meeting. One team member, the scrum master, is in charge of solving problems that stop the team from working effectively	Consistent, iterative testing
Extreme Programming (XP)	Focuses on best practice for development. Consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40h week, on-site customers and coding standards. The revised "XP2" consists of the following "primary practices": sit together, whole team, informative work space, energized work, pair programming, stories, weekly cycle, quarterly cycle slack, 10-minute build, continuous, integration, test-first programming and incremental design. There are also 11 "corollary practices"	Test Driven Development

Figure 2.8: Description of different Agile methods (Dybå and Dingsøyr 2008 p.835)

2.4 Role of Testing in Both Models

What should be clear is that while all methods may differ in how they deliver projects, all place a significant emphasis on testing. While in Traditional Waterfall models testing is a discrete phase that happens at the end of a project, in Agile it is a much more fluid and flexible task that continually happens during the development lifecycle. It is important to understand what testing actually is. The next section of the dissertation will outline and explain Software Testing in detail.

2.5 Software Testing

When developing software there are a number of “stages” that need to be completed in order to deliver a working piece of software. Royce (1970, p1) stated that in reality the two ‘*essential steps common to all computer program developments*’ are Analysis and Coding.

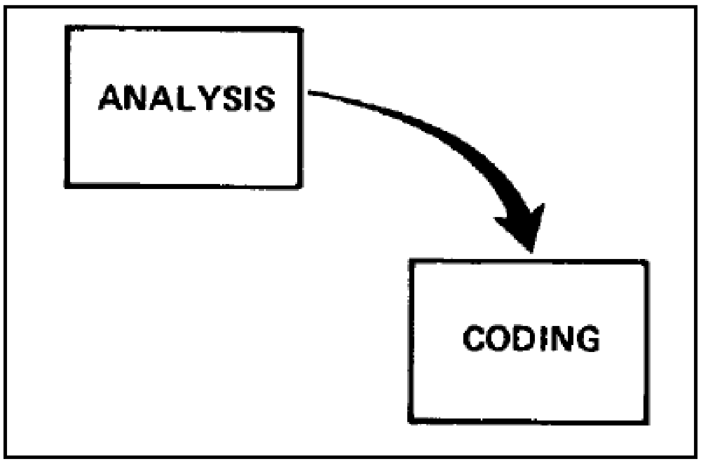


Figure 2.9: Implementation steps to deliver a small computer program for internal operations (Royce 1970)

However, in there are other key steps that all projects should iterate through no matter what methodology is used to deliver it. As noted by Royce there are fundamental steps to ensure that project is built (the analysis, design and coding) but there are also steps that aim to ensure the quality of the delivered project. This part of the process can be through of as part of the "Quality Process" and should be something that is not considered a phase but something that '*spans the whole development cycle*' (Baresi and Pezze 2006, p.90). There are various ways that quality can be ensured on a project. A key method is through the use of formal and rigours software testing. But what exactly do we mean by "software testing"?

Software Testing has various definitions in the literature; Bertolino (2007) defines it as verification that the system behaves as expected and will identify defects in the system under test. In his seminal book "The Art of Software Testing" Myers (1979) defines it as executing software to expose failures. In reality there are many more definitions and also there is much more to it than simply execution. Hamlet (1995) argues that the primary goal of software testing is to measure the dependability of the software under test. Perhaps a fuller and more complete description from the International Software Testing Qualifications Board (ISTQB®): they define it 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* (ISTQB 2012).

It should be clear that testing is about much more than simply execution and spans the entire lifecycle of a project.

Before we discuss the different methods of software testing and the types of testing that can be undertaken it is perhaps appropriate to consider why it is so important to test software. As mentioned in the introduction the process of developing and releasing software is complex, expensive and can be fraught with risk (Adolph *et al.* 2012). It is also big business with estimates of it being a $1.6 trillion industry (Bartels *et al.* 2006). Thus, we can imagine that companies would want to ensure that the process for developing and testing this software that they have implemented are correct and being used properly.

As the complexity (and cost) of these software projects increases in areas such as quality, reliability and the customer acceptance are critical to the organisations that produce this software (Huang 2005). It is also a complex and costly task with estimates that testing can account for more than 50% of the cost of a software development project (Whyte and Mulder 2011). However, the cost of inadequate testing or verifying software can be far greater; according to Engel and Last (2007) this can result in losses that could total more than 10% of an organisation's turnover. When viewed in this context it is not surprising how important the testing of software is. However, even in the face of this evidence on the need to completed full and rigorous testing, the area of software testing is coming under significant pressure due to shorter time to market requirements from customers as well as cuts in budgets (Srikanth and Williams 2005). Customers are willing to pay for analysis and coding activities as they see these as contributing to the actual product, but are less willing to pay for testing and other activities that they may not see as key to the delivery of the software (Royce 1970). However, software testing is a vital part of what can be thought of as the "Quality Process" in software development.

2.5.1 Quality Process

As noted, the Quality Process can be considered as all of the activities that take place around software development process that are related to ensuring the quality of the delivered software (Baresi and Pezze 2006). For the purposes of this dissertation we will focus on the place software testing takes within the overall Quality Process of an organisation. As the earlier definition of software testing from the ISTQB outlines testing should encompass the entire lifecycle of the project. It is too late to only have test involvement during the testing phase; the Test function in an organisation should be involved from feasibility right through to the deployment of the project. Test teams should be involved in static reviews of the requirements to ensure that these are all testable. They should be involved in review design documentation also to help detect and remove design errors before coding even begins (ISTQB 2011).

If we are to look at the quality process over the entire lifespan of a project then the test elements of this can be broken down into 5 main elements:

1. Planning and Monitoring
2. Verification of Specification
3. Test Case Generation and Selection
4. Test Case Execution
5. Software Validation and Process Improvements (Baresi and Pezze 2006).

1. <u>Planning and Monitoring</u>

This is the first step in the quality process. It should happen as early as possible in the project lifecycle. There should be early planning of all phases of the project, including the test window, early release of plans for review and this should continue throughout the entire project. There should also be close monitoring of each phase in a project to ensure delivery is reached on time and to budget (Baresi and Pezze 2006).

2. <u>Verification of Specification</u>

This step in the quality process involves the reviewing of specification documentation. This can involve internal review of specifications for clarity and external review of specifications against other specs for consistency(Baresi and Pezze 2006) This level of static analysis is a very important step in the Quality Process and is a key test method that can reveal errors early on in the project lifecycle (ISTQB 2011). The benefit of this analysis is clear when literature would state over 50% of defects are generated in the requirements phase of a project (Srikanth and Williams 2005a).

3. Test Case Generation and Selection

The problem with test case generation (and selection) is perhaps the one that attracts the most attention in the literature (Bertolino and Marchetti 2005). Test cases are usually generated from requirement specifications. They should be generated as soon as these specifications have been signed off. This early generation of test cases has the added advantage of reducing the risk of this activity becoming a bottleneck later on in the project. This means that test cases generation can happen in parallel with code development (thus removing it from the critical path on the project) (Baresi and Pezze 2006).

There are a number of methods used for test cases selection. Common sense and experience tells us that it is impossible to test every potential scenario that might occur in the use of a computer program. Thus, when selecting test cases we need to ensure that the cases selected will cover the most important areas of our program (Edgren 2011). One method of doing this is Test Case Prioritization (Sampeth *et al.* 2008). It has been shown using test suite prioritization defect detection rates in the project are increased as compared with random test selection (Rothermel *et al.* 2004). Thus, if test case prioritization can be of benefit how best should we prioritize our test cases? Srikanth *et al.* (2005b) suggest that test case prioritization should be based on 4 factors: requirement volatility, customers' priority, the complexity of the implementation and how prone the system is to faults. Thus, based on this we can prioritise a set of tests based on requirements that can '*increase test effectiveness which contributes to increased detection rate of high risk defects*' (Whyte and Mulder 2011, p. 259).

Another method for test case selection is that of Test Case Reduction. This involves a systematic review of all the created test cases to remove what are considered non-essential cases. (Whyte and Mulder 2011). The end point of this exercise is to ensure that all requirements are tested using the least amount of test cases which should reduce the cost of regression testing (Zhang *et al.* 2008) which is ultimately the goal of Test Case Reduction (Whyte and Mulder 2011). Studies have shown that the number of test cases can be reduced without reducing the defect detection (McMaster and Memon 2006). However, caution should be exercised in test case reduction as other studies have shown that this can cause a reduction in defection detection capabilities (Whyte and Mulder 2011).

4. Test Case Execution

It is not necessary to wait until the entire piece of software is completed before commencing testing. There are numerous levels of testing that will be looked at in detail in the next section but these all allow and are predicated based on the fact that testing can happen in stages. Baresi and Pezze (2006) noted that testing can be 'executed in absence of the completed system' (p. 92). In order to execute test cases without a complete system there are number of pieces of architecture that need to be in place to allow testing to begin, such as stubs to external systems and test oracles. The stubs provide simulated responses from other external systems that will allow a higher level validation of test cases (Baresi and Pezze 2006). In terms of actual execution this can be completed manually or via automation. The distinction between these types of testing will be outlined in the coming sections.

5. Process Improvement

The area of process improvements relates to reviewing current and past projects for areas that caused problems and looking at ways to improve this. These problems can either be changed by altering development activities or taking specific actions.

In the next section we will investigate the different types of testing, at a high level there are two forms of testing: Functional and Structural. We will firstly look at Functional Testing; highlighting the different types of functional tests. Then we will look at Structural Tests and the different types of structural testing methods.

2.5.2 Functional Testing

Functional testing relates to the testing of systems at a functional level. It is testing based on system requirements. As such it can be thought of as "black box testing". In this type of testing the software under test is treated like a black box, the internal structure of the system under test is hidden. The testers have no knowledge of the internal workings of the systems. The testing is based on inputs to and outputs from the system. Thus, tests are run based on externally specified required behaviour (ISTQB 2011).

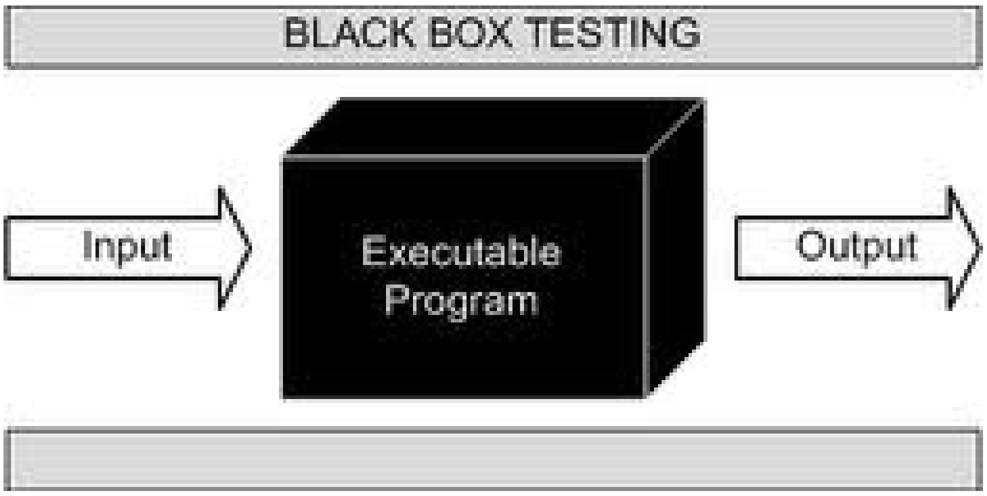
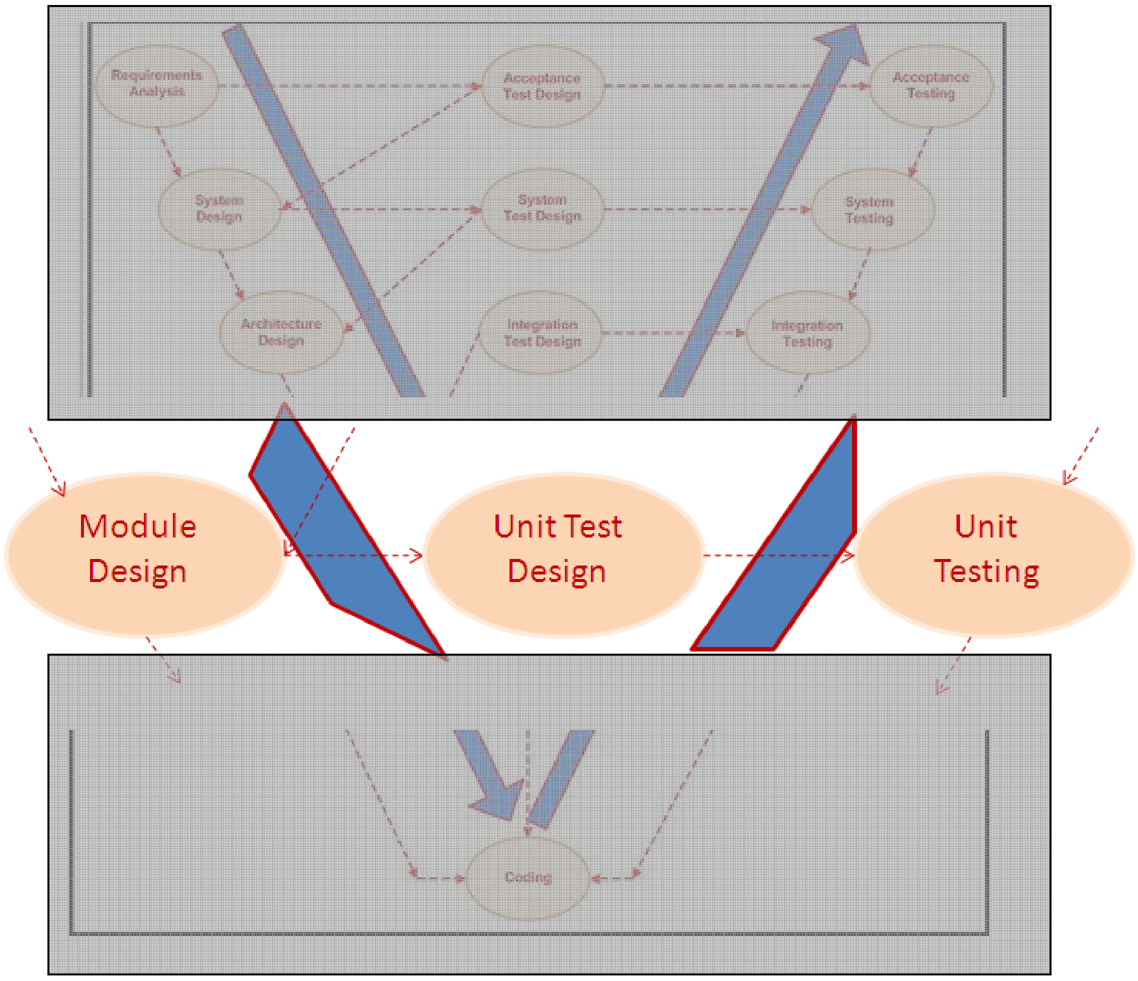


Figure 2.10: Black Box Testing (Sooraj 2011)

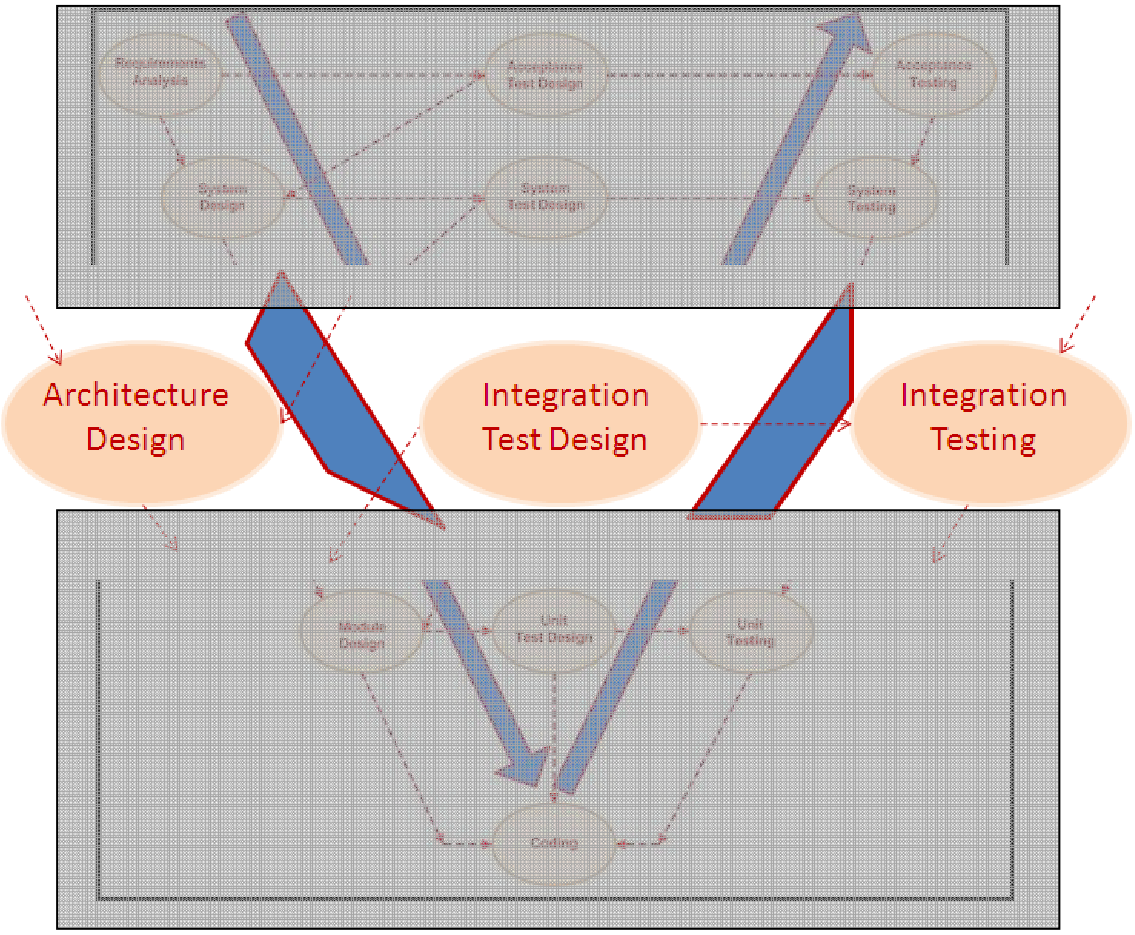
Perhaps the best way to investigate Functional Testing is to look at the V-Model again. The V-Model details how the different types of functional testing map back to different phases of requirements, design and coding on the left hand side of the model.

2.5.2.1 Unit Test



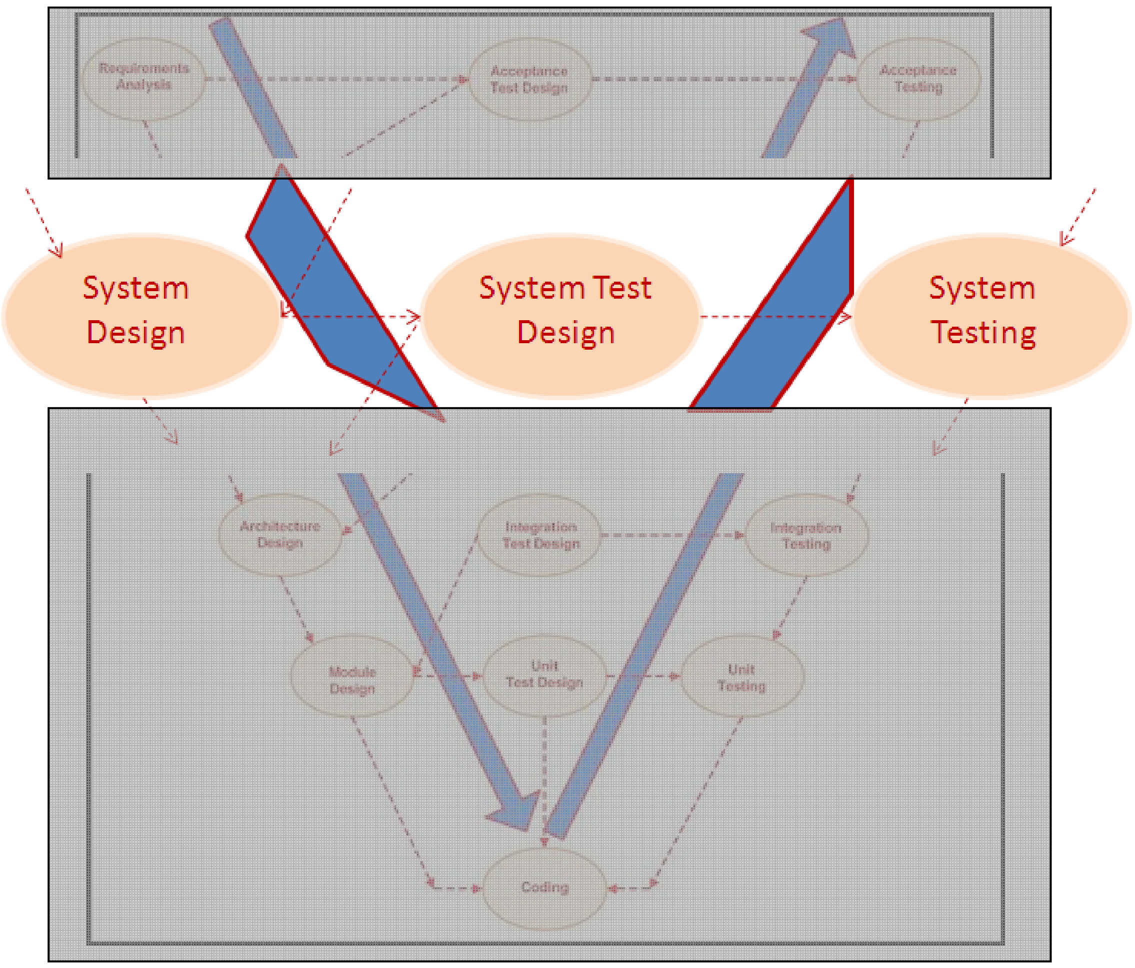
This is the first level of the V-Model and is often called Component Testing. This is the lowest level of details and is used to demonstrate that the individual components perform as specified (ISTQB 2011). Decisions about what will be tested and when to stop are generally made by the developers as this is a technical activity (Hetzel 1993). This lowest level of testing is where the module designs are validated and that the specific component requirements are met (Roper 1994). Unit testing may be informal with no structure but can also involve detailed test planning and design work (Hetzel 1993).

2.5.2.2 Integration Testing



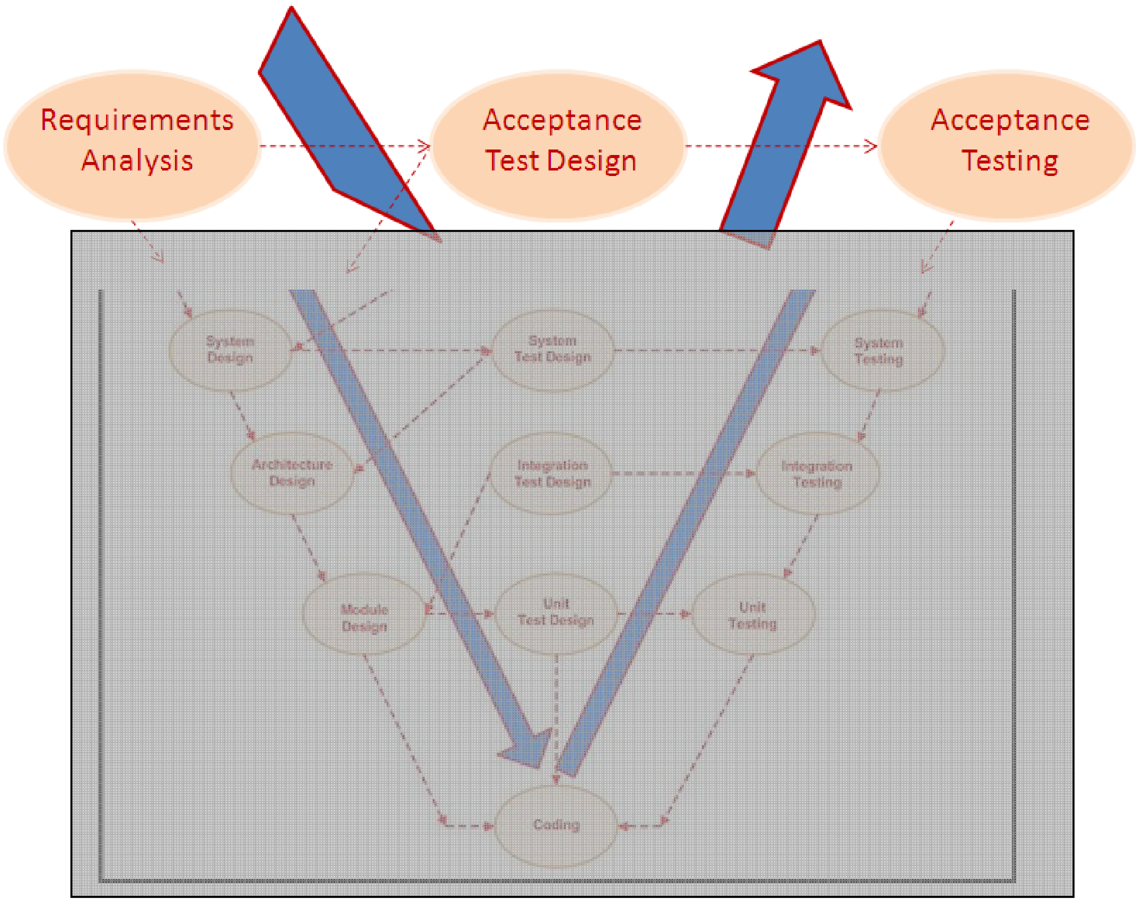
The next level on the model is Integration Testing. This is one step up from Unit/Component Testing and is where we first see the individual components of system communicate. Once again this is technical phase and is usually completed by developers (ISTQB 2011). The aim of this phase is to verify that the detailed design of various components still function when integrated (One Stop Testing n.d.). This is also where the interfaces between systems are tested (Hetzel 1993).

2.5.2.3 System Test



System Testing is the next step on the V-Model. This is the first step in the model where all modules are brought together and tested as a system (Hetzel 1993). This is also the first stage in the phase that is not developer led. This is usually run on behalf of the software suppliers by an independent team (ISTQB 2011). Test cases are designed to validate that systems and architecture designs are implemented correctly (Roper 1994).

2.5.2.4 Acceptance Testing



Once System Test phase is complete we move onto the last stage in the V-Model: Acceptance Testing. The aim of this phase is to 'provide the end user or customer with confidence and insurance that the software is ready to use' (Hetzel 1993 p. 11). It is usually completed by a representative of the business or customer and is business process focused. Thus, it will involve end users executing tests cases that are designed around how they would use the system on a day to day basis in their job (ISTQB 2011). They can be a subset of the System Test cases and usually include business transactions (Hetzel 1993). It is the highest level on the V-Model and is completed to test that the system requirements on the project have been met.

2.5.3 Structural Testing

The next set of testing to be considered is that of Structural Testing. This can be thought of as "White Box Testing", against the black box testing of functional testing. Thus, the focus of these tests is significantly different. With Black box testing the system under test is not investigated and the inputs and outputs are all that are considered. However, with the White Box testing the system itself is under test. The internal workings of the system are being investigated (ISTQB 2011).

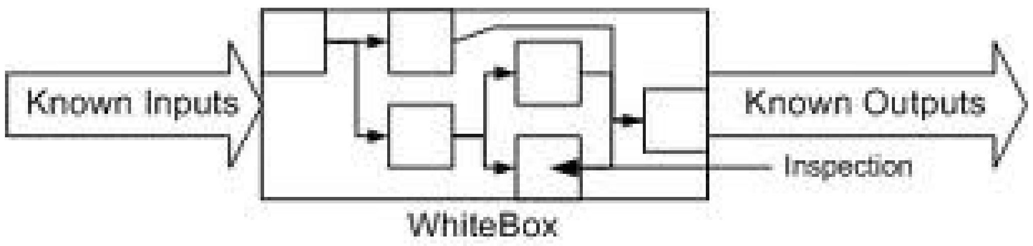


Figure 2.11: White Box Testing (Clifton 2003)

There are a number of different types of structural tests that can be undertaken. Some of the more popular methods will be outlined below:

2.5.3.1 Statement Testing

The purpose of this testing is to ensure that every source language statement in a program is executed at least once (Hetzel 1993). In order to complete this you must interrogate the code to find all the condition controlled statements (e.g. while X>y, IF, ELSE etc) and ensure that values are inserted into each condition so that all statements in the code get completed (ISTQB 2011).

2.5.3.2 Branch Testing

This is also known as decision coverage. The aim of this testing is to execute the TRUE and FALSE outcome of every decision statement in the code (Roper 1994)
There are a number of other methods that can be used that follow similar lines of detailed code-level investigation:

- State Transition Testing
- Boundary Analysis
- Decision Tables
- Equivalence Partitioning

- Path Testing (Roper 1994)

2.6 Conclusions

This chapter investigated the different software development methodologies in use in industry today. There was a discussion of the traditional software development models such as the Waterfall Model and V-Model. Next, the limitations of these models were highlighted. These limitations have led to the development of alternative methods (the so called Agile methods). A number of these methodologies were then outlined in detail.

Following this it was noted that a central phase in both methods is software testing. Thus, there was a need to give an overview of software testing in the context of the overall Quality Process in an organisation. Following this the various methods of software testing used were outlined (Functional and Structural). Functional Testing methods were discussed in detail as these are the methods that will be under investigation in this dissertation. An introduction to Structural Testing was outlined in order to give the reader an idea of the other low level tests, but was not investigated in detail.

3. RESEARCH METHODS AND GROUNDED THEORY

3.1 Introduction

This dissertation is attempting to understand how Grounded Theory can be used to develop a framework for best practice in the Testing phase of projects. In order to understand what Grounded Theory is and why it was selected it will be necessary to review the existing types of research methodologies available. This chapter will detail the different methodologies used in conducting research project. It will start with an overview of the philosophical schools of thought that can influence selection of one method over and other. Next, the two main types of research methodology will be discussed (Quantitative versus Qualitative). Then, as a follow on from that one method of qualitative research will be outlined in detail, that of Grounded Theory. Finally, the use of Qualitative methods; and Grounded Theory in particular, in software engineering research will be discussed.

3.2 Research Methodologies

When conducting any research one of the most fundamental questions is how are you going to carry out your research and collect your data? This will largely be dictated by the type of methodology that is employed. There are number of potential alternatives that researchers have and the choice of method will depend on many things such as the research question being studied, what methods previous researchers have used and the philosophical leanings of the researchers (Creswell 2008). The methods can be broadly categorised into two areas: Qualitative and Quantitative (Coleman and O'Connor 2007). At a high level the distinction between the two would be that qualitative methods rely on words and open-ended questions whereas quantitative methods rely on numbers and closed questions (Creswell 2006). However, in order to get a more complete view of the differences it is necessary to investigate both in more detail. This will first start by an investigation of the philosophical underpinnings of each method and then a discussion of each in detail.

3.3 Philosophical Underpinnings

An important factor that can guide researchers into being predisposed to certain methods over others is the philosophical assumptions that underpin those methods (Coleman and O'Connor 2007). Researchers make philosophical assumptions at the start of designing a research process that will guide the selection of qualitative or quantitative methods (Creswell 2008). Thus, the philosophical views of individual researchers can play a significant role in the methods of approach chosen. This view, termed "world view" or "paradigm", will influence the choice of research method and can be defined as 'a basic set of beliefs that guide action' (Guba 1990 p.17).

Researchers will come to the study with their own pre-existing beliefs and these can inform and guide the researchers when choosing one method over another (Creswell 2006). When looking at the "world views" that will guide selection of an appropriate method, this research focus on the two areas that have received the most academic interest in terms of literature reviews: *Post-Positivist* and social *Constructionism* (Coleman and O'Connor 2007).

3.3.1 Post-Positivist

The post-positivist view represents what might be considered the traditional form of scientific research and might be more aligned to quantitative research than qualitative (Creswell 2006). This is sometimes known as the scientific method and this would usually underpin quantitative research as it allows for general laws or principles to be established (Coleman and O'Connor 2007).

It is termed post-positivist as it came about as a refutation of the positivist approach and recognises that it is impossible for us to be absolutely positive about any claims we make about human behaviour or the world around us (Phillips and Burbules 2000). In post-positivism it is held that cause probably determines the outcomes or effects (Creswell 2008). Thus, much of the emphasis of post-positivists will be looking at the causes to determine their effect on outcomes. Any knowledge gained is from careful and objective measurement of the world and any research conducted can '*provide answers which have a provable base*' (Coleman and O'Connor 2007, p2). The basis of this scientific method from a post-positivist point of view would be for a researcher to begin with a theory, they will then collect and analyse their data such that the theory is supported or not (Creswell 2006). Thus, it is a deductive form of reasoning, whereby we have a theory and look for support within the data collected (Strauss and Corbin 1998).

Phillips and Burbules (2000) outline five key assumptions of the post-positivist point of view:

1. Knowledge is not absolute so for this reason we can't prove a hypothesis, simply fail to reject it
2. Research involves making a claim and then design an experiment to test this
3. Our knowledge is based on data, evidence and rational consideration
4. When doing research the aim is to develop relevant statements for explaining the phenomenon under study
5. There is a need to be objective in order to conduct competent research (Creswell 2008).

Therefore the basis of post-positivist view is a scientific basis involving experimentation, data and testing hypotheses. As noted this would be typically quantitative research. In contrast to this approach is the Social Constructivists view of research which is examined in the following section.

3.3.2 Social Constructivism

Another "world view" that can be used in relation to research is that of social constructivism or sometimes known as interpretivsm (Coleman and O'Connor 2007). This is the paradigm that is typically associated with qualitative research (Creswell 2008). In this view people attempt to gain an understanding of the world which they inhabit. We all place some subject meaning on our collective and individual experiences, some of this is learned from previous personal experience or from our cultural backgrounds (Creswell 2006). Thus, when researchers with this world view carry out investigations they look at the complexity of situations rather than simplicity in meaning (Crotty 1998). The aim of research undertaken under this paradigm is to '*rely as much as possible on participants views of the situations being studied*' (Creswell 2006 p.8). People construct meaning from social interactions; they aren't borne with innate points of view, these are constructed over time via elements such as cultural background, previous experience and historical norms in their society (Creswell 2008). The main difference between the post-positivist viewpoint and the social constructivist one is that researchers who work under the post-positivism paradigm will start with a theory and test it deductively, in contrast, social constructionists see theories (and meanings) as something that should develop inductively from the research and data (Crotty 1998).

When those working under social constructivist worldview will conduct research they will do so by talking to participants, listening intently and studying their behaviour in social situations (Creswell 2006). Researchers who follow this paradigm will focus on '*specific contents in which people live in order to understand the historical and cultural settings of participants*' (Creswell 2006 p.21). Researchers will also take into account their own background and should be mindful of how their pre-conceived ideas may influence the way they interpret behaviours (Neuman 2011).

When one looks at research from the social constructivist point of view there are a number of assumptions that can be made:

1. As a human we construct meaning through our interactions with the world around us
2. Cultural perspectives play an important role in how we understand and interpret the world around us
3. Any meanings we generate are based on social interactions in the community (Crotty 1998).

Some researchers will claim that the "worldview" held by those undertaking an inquiry or study will effect which philosophical view point they take and which type of research they conduct. However, in choosing a methodology there are some practical as well as philosophical considerations to take into account, such as what research methodology was employed on previous research, as well as the more mundane aspects such as cost and time available, all of which can guide researchers into using one or other type of research methodology (Coleman and O'Connor 2007).
Now that the basis for the philosophical underpinnings of the two major methods have been investigated it is necessary to look at each method in detail, starting with Quantitative Methods.

3.4 Quantitative Methods

A good definition of quantitative methods is that outlined by Creswell (2009 p.4) where he defined it as '*a means for testing objective theories by examining the relationship among variables*'. The outcome of quantitative methods can be analysed using statistical tools and can be subjected to scientific testing (Coleman and O'Connor 2007). Scientific enquiry, that makes predications or theories on general laws and principles and then tests these, would seem most compatible with Quantitative methods (Creswell 2006). In order to gather the data required to allow this scientific analysis there are a number of techniques that can be used, these are defined by Creswell (2009, p11) as strategies of inquiry and he defines them as methods that '*provide direction for procedures in a research design*'. We shall now investigate some Quantitative Strategies of Inquiry.

3.4.1 Quantitative Strategies of Inquiry.

As outlined above quantitative methods are associated with a post-positivist world view. Thus, these strategies will include elements that would be favoured from a post-positivist point of view. These are things such as experiments and quasi-experimental designs, with a more recent move towards more complex, multi-variable experiments (Creswell 2006). There are two key strategies of inquiry that lend themselves to this level of investigation, survey research and experimental research:

3.4.2 Survey Research

Creswell (2009, p.12) outlines that surveys provide '*quantitative or numeric descriptions of trends, attitudes, or opinions of a population*'. By conducting surveys researchers can take the quantitative or numeric descriptions or empirical data, they can then conduct statistical analysis on this data to deductively arrive at a position that can either support or fail to support their theory. Surveys provide a means, via questionnaires or longitudinal studies, to gather data that can allow inquirers to generalise from the small sample to the larger population (Babbie 1990).

3.4.3 Experimental Research

This is the classic quantitative method, the classic experiment as espoused in the scientific method. It involves investigation to determine if a specific action effects an outcome (Coleman and O'Connor 2007). There are a number of different types of experimental design but they all involve assessing the impact of a certain action by providing treatment/action on a certain group and withholding from another and then investigating any differences in the outcomes between the two groups (Creswell 2008).

Now that we have looked at the Quantitative Research Method it is necessary to investigate the other method, that of Qualitative Research Methods.

3.5 Qualitative Methods

When talking about qualitative research methods this refers to an research that will produce some findings '*not arrived at by statistical procedures or other means of quantification'* (Strauss and Corbin 1998, p. 10). Its primary purpose is to collect and interpret non-numeric data (Coleman and O'Connor 2007). Creswell (2006, p4) defines it as '*a means for exploring and understanding the meaning individuals or groups ascribe to a social or human proble*m'. While quantitative research will ask closed questions such as "how much?" and "how often?" qualitative researchers will more commonly ask open-ended questions such as "why?" and "how?" (Coleman and O'Connor 2007). It involves building research about the way people live their lives, encompassing their experiences as well as behaviours and emotions. It can be used to gauge feelings on certain topics such as organisations, cultural phenomenon or governments (Strauss and Corbin 1998). Qualitative research develops its final theory inductively through interactions with research participants (Coleman and O'Connor 2007).

3.5.1 Strategies of Inquiry

Similar to quantitative methods, qualitative methods have their own strategies of inquiry. Some researchers have noted that there are up to nineteen strategies available to the qualitative researchers (Wolcott 2008). However, for the purpose of this research only the five major strategies will be outlined.

3.5.1.1 Ethnography

Ethnography involves studying groups in their natural setting over a prolonged period of time (Creswell 2006). In an ethnographic study researchers will attempt to uncover and interpret many values of the group who are under study. These could be behaviours, beliefs and language sharing (Murchison 2010). Usually those involved in ethnographic studies are interested in looking at groups larger than twenty individuals (Creswell 2006). This is because ethnographic studies are generally involved in understanding the behaviours and views of entire cultural groups. In an ethnographic study the researchers will look to immerse themselves fully into the day to day lives of the participants under study. They will review and interpret the behaviours and interactions among group members in order to develop their theories on behaviour (Creswell 2006).

3.5.1.2 Grounded Theory

Grounded theory is a strategy of inquiry where the key idea is that theories are generated (or "Grounded") in the data collected (Strauss and Corbin 1998). The emphasis in Grounded Theory is to focus on generating new theories (Coleman and O'Connor 2007). Creswell (2006, p.13) defines is as '*a strategy of inquiry where the researcher derives a general, abstract theory of a process, action of interaction grounded in the views of the participants*'. The participants are key to the development of the theory (Glaser and Strauss 1967). The researchers also ensure that they don't carry pre-conceived notions or theories before the research begins; theories should evolve from multiple comparisons and sampling of different groups (Birks and Mills 2011).

3.5.1.3 Case Studies

Case Studies involve an in-depth study of an issue by exploring cases related to the area of the issue (Creswell 2006). Some argue that it is not a strategy of inquiry or a research method but rather a choice of what is studied (Denzin and Lincoln 2005) whereas others view it as a methodology in and of itself (Yin 2008). Whatever point of view is taken case studies are a popular method of qualitative research in many fields, such as psychology (Freud used this approach for generating theories) as well as Medicine, Law, and Political Science (Creswell 2006).

3.5.1.4 Phenomenological Research

Phenomenology is a strategy of enquiry where researchers will try to identify the essential elements of the human experience of certain phenomenon (Creswell 2008) Phenomenological research attempts to study the lived experience of several participants of a particular phenomenon (Manen 1990). Thus, the '*basic purpose of phenomenology is to reduce individual experiences with a phenomenon to a description of the universal essence'* (Creswell 2006 p. 58). The purpose is to identify "phenomenon" or objects of human experience, this could be anything from insomnia, to be being bullied or undergoing surgery (Moustakas 1994). The researchers then gather all the data about the different experience of this phenomenon and then use this to develop a '*composite description of the essence of the experience for all of the individuals'* (Creswell 2006 p.58). This description will outline what was experienced by these people and also how they experienced it.

3.5.1.5 Narrative Research

While phenomenological research tries to explore the meaning for several individuals Narrative Research attempts to report on the life experiences of individuals. In this strategy of enquiry the researcher develops a picture about the lives of participants through self-reported stories told by those individuals (Creswell 2008). The Narrative Research method involves the researchers taking the stories of people and re-telling them in a narrative form in chronological order (Czarniawska 2004). The methods for conducting this research involves studying one or two individuals in detail, collecting all the "data" (in this case their stories) and then outlining individual experiences in time-ordered fashion (Creswell 2006).

Now that we have outlined the major methods of Qualitative and Quantitative Research it is necessary to focus on the method that was chosen for this study, that of Grounded Theory.

3.6 Grounded Theory

At its highest level Grounded Theory is a qualitative research method where theory is generated from the data collected (Adolph *et al.* 2012). It is one of the most popular methods of qualitative research in use today (Birks and Mills 2011). The overall goal of the Grounded Theory research is to gain an understanding, from an actor's perspective, the behaviour under study (Glaser 1978).

3.6.1 Symbolic Interactionism and Grounded Theory

Grounded Theory has its roots in *Symbolic Interactionism* (Coleman and O'Connor 2007). Symbolic Interactionism itself is related to the philosophical ideals of pragmatists; this has been noted by researchers such as James, Cooley and Mead (Heath and Cowley 2004). Symbolic Interactionism is a theory that attempts to explain group life and human conduct (Blumer 1986). In this paradigm humans are seen as '*key participants and shapers of the world they inhabit*'" (Coleman and O'Connor 2007, p4.). People are able to asses and modify their behaviours as required in certain scenarios; people have a critical self-awareness (Mead and Morris 1934 quoted in Heath and Cowley 2004). By our social interactions with others we derive meanings and these help to shape society through our shared understanding of these interactions (Blumer 1986).

3.6.1.1 Symbolic Interactionism

The term Symbolic Interactionism was developed based on research carried out in the University of Chicago in the 1970's (Klunklin and Greenwood 2006) and was first described by Blumer (1986). In this he outlined key elements of integrationist inquiry as noted above; the role of concepts and symbols and development of meaning from the social interactions (Heath and Cowley 2004). This was seen as the major challenger to the traditional view of the Functionalist philosophical approach (Klunklin and Greenwood 2006).

The Functionalist views the world as one "system" or a number of complementary parts that form a functioning unit. The constituent parts and their analysis is only relevant in so far as the effect they may have on the whole of the system (Klunklin and Greenwood 2006). People involved in society (or systems) learn to internalise others expectations of them '*through socialisation; individuals are determined..rather than determining'* (Merton 1973 quoted in Klunklin and Greenwood 2006 p. 33). This means that when researchers are working from a functionalist point-of-view they will begin with some framework which then guides their research questions. These questions are then converted into hypotheses which can then be tested (Blumer 1986). It is clear that this reflects a classic deductive form of reasoning where a theory is proposed, data collected and then tested to either support or fail to support the hypothesis (Creswell 2006).

Those who approach research from a symbolic interactionist point-of-view will fundamentally disagree with this position and will differ significantly in their approach to research (Klunklin and Greenwood 2006). As noted, Symbolic Interactionism was first proposed by Blumer (1986). In this theory Blumer stressed the importance of the individual and how they are determining rather than determined, the exact opposite of that which is proposed by functionalists such as Merton (1973 in Klunklin and Greenwood 2006). Society is created '*through the purportive interactions of individuals and groups'* (Klunklin and Greenwood 2006 p.33). The focus on Symbolic Interactionism is on theories that are inductively derived (rather than via deduction) and this is a key feature of qualitative research (Creswell, 2009). The key topics in Symbolic Interactionism are: the self and the world and social action (Charon 1995).

<u>**The Self**</u>

The sense of self in Symbolic Interactionism is constructed through our social interactions with others in our society, family members and others we come into contact with throughout our life (Charon 1995). We learn how to behave and respond to others based on their feedback from our interactions (Klunklin and Greenwood 2006). Through these we develop an idea of social norms for behaviour and then allow us to elicit our sense of self control in social situations. Our self-identity emerges through social interactions (Charon, 1995).

<u>**The World**</u>

The symbolic interactionist considers the world to be a world of 'symbols' (Klunklin and Greenwood 2006). These symbols can be objects in the world, however, not all objects are symbols; objects only become symbols when meaning is ascribed to them

(Blumer 1986). Thus, objects can be physical or abstract. The key point is that objects themselves don't have any intrinsic meaning in and of themselves but meaning is '*derived from how others act towards objects*' (Klunklin and Greenwood 2006 p. 34). Thus, symbolic interactionism refers to the process of continually assigning of meaning to symbols through our interactions with the world. People continually try to interpret how others in the world are reacting to their actions and will try to arrive at how is best to act themselves. We then take this feedback from others in our world and modify our behaviours accordingly (Klunklin and Greenwood 2006).

When studying social factors Symbolic Interactionism will attempt to study behaviour through first-hand interactions (Blumer 1986). A key part of Symbolic Interactionism as a research methodology is the concept of exploration and inspection (or depiction and analysis). This involves exploring data and then continually analysing it to ensure that it remains grounded in reality (Klunklin and Greenwood 2006). It will become clear in the next sections that this is a key idea in Grounded Theory and will see why it is said that that the roots of Grounded Theory lie in Symbolic Interactionism.

3.6.1.2 Symbolic Interactionism and Grounded Theory

The core methodology of Symbolic Interactionism relates to direct observation and developing theory from this observation. Data is acquired through rigorous examination of the phenomenon under study. Then from this data the theories or hypotheses are constructed (Klunklin and Greenwood 2006). This parallels the methodological approach as outlined by Glaser and Strauss (1967) where core concepts are developed through observation and study. These are then refined through a process of on-going comparative analysis (Strauss and Corbin 1998).

A second area of convergence of Symbolic Interactionism and Grounded Theory relates to the ideas espoused by Blumer (1986): that of exploration and inspection. By 'exploration' Blumer refers to the idea that the researchers respond flexibly to the data they find would fit well with the concept of constant comparative analysis in Grounded Theory (Klunklin and Greenwood 2006). Then, if the idea of 'inspection' is looked at we see that this refers to the close analysis of data. This would be consistent with the ideas of Glaser and Strauss (1998) where data is closely studied, categorised and coded. Thus, it can be concluded that '*Grounded Theory is usefully construable as the method of Symbolic Interactionism*' (Klunklin and Greenwood 2006 p. 34).

3.6.2 History of Grounded Theory

Grounded theory was first proposed by two American sociologists Barney Glaser and Anselm Strauss in 1967. Strauss gained a degree in Sociology from the University of Virginia and then a Masters and PhD from the University of Chicago. He studied under Herbert Blumer and George Herbert Mead while in Chicago (Stern. 2009). As noted these were major figures in the field of Symbolic Interactionism and he was naturally drawn to this method as it was his background (Glaser and Strauss 1967). Glaser earned a degree in Sociology from Stanford and later a PhD from Columbia University. While there he studied with Paul Lazerfied and Robert Merton in the area of descriptive statistics, and had a more pragmatic approach to research, as can be seen in his later work (Birks and Mills 2011). At the time Strauss was working in the University of California, San Francisco's School of Nursing in the 1960's where he was appointed to assist in the development of a Doctoral Nursing degree program. Glaser first met Strauss and asked him to join him on the research he was conducting into the study of dying (Morse *et al.* 2009). Together they published a number of papers, the first of which was The Art of Dying which had an important impact on the treatment of dying patients and their families (Birks and Mills 2011). As they neared the end of their research grant both Glaser and Straus began to realise that they the methods they were using for data gathering had been different from what they had been doing previously. They now had a much more rigour and ordered attitude to the analysis of data (Stern 2009). It has been suggested that the authors' differing backgrounds played a part in the development of this new methodology. As noted Glaser's background involved more quantitative data analysis (due to his background in statistics) (Bryant and Charmaz 2010) whereas Strauss' background in Symbolic Interactionism made it natural for him to constantly re-evaluate the data that had been gathered (Morse *et al*. 2009).

From this initial position of new methods they developed their seminal work *The Discovery of Grounded Theory* (Glaser and Strauss 1967). Their aim was to set up a concise method for systematic qualitative research (Bryant and Charmaz 2010). At the time the book was published there was a preference for research that used the traditional statistical-quantitative methods. Glaser and Straus wanted to show that Grounded Theory could produce outcomes of equal importance to quantitative methods (Morse *et al.* 2009). By doing so Glaser and Straus found themselves in opposition to some quantitative methods researchers but their approach allowed for methods that replicated some of the results and methods used by other researchers. Thus, this ability to replicate quantitative methods and results gave rise to a major strength of Grounded Theory. For the first time this allowed the process and procedures of qualitative research methods to be visible and repeatable and more easily understood (Bryant and Charmaz 2010). The next section will outline the theory in more detail.

3.6.3 Introduction to Grounded Theory

At its heart and the essence of Grounded Theory is the notion of forming new theories from the data (Glaser and Strauss 1967). Glaser and Straus, when developing their theory, took exception to the foundations of the prevailing research paradigms of the time:

1. That of verification of theory by using data gathered
2. Logical deduction of theory from a-priori assumptions (Birks and Mills 2011)

They emphasised the generation of new theories over the verification of existing ones (Coleman and O'Connor 2007). Glaser and Straus were very quick to outline the faults they saw in the existing methods and their belief that data is the key focus of their work (Birks and Mills 2011). This wanting to keep data at the heart of their theory possibly arises due to Glaser and Straus' focus on ensuring that qualitative research is kept '*as a scientifically respectable practise*' (Bryant and Charmaz 2010). Thus, they felt that Grounded Theory must undergo scientific rigour in it application and should only be applied by trained sociologists (Glaser and Strauss 1967).

Grounded Theory can be thought of along as somewhat analogous to Agile software development in that as a concept they both can be understood simply but in practice are disciplined and rigorous (Adolph *et al.* 2012).

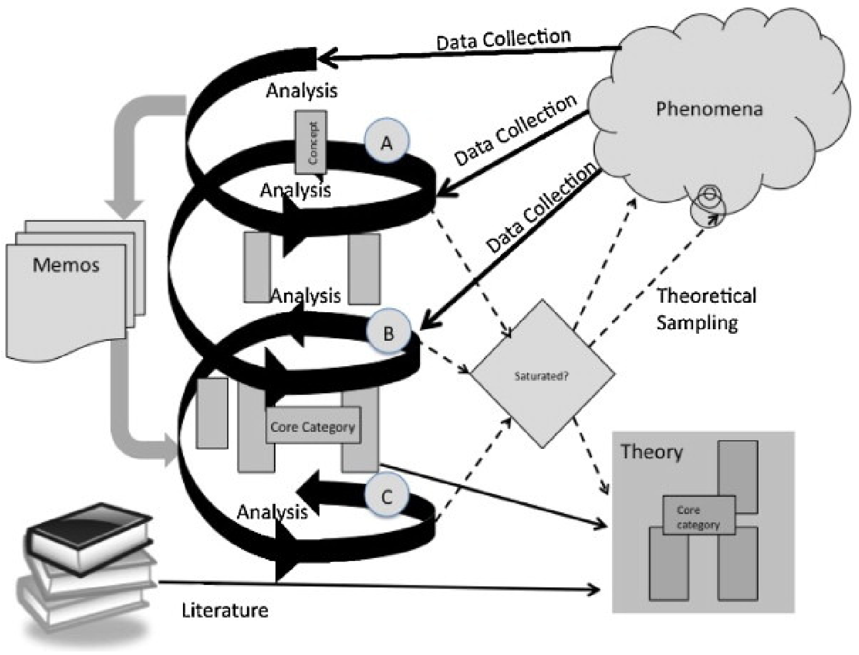


Figure 3.1: The Grounded Theory Method (Adolph *et al*. 2012 p.1271)

The diagram above outlines the basics of Grounded Theory.

A. Researchers begin by collecting data on an area of interest and these are used to build concepts, which are essential to building categories
B. The concepts are then further developed by completing constant comparative analysis which involves comparing new data with data already amassed. From this analysis core categories emerge that the collected data can fit into. This process is repeated until the categories become saturated, which means that by the addition of further information no new concepts are emerging (Coleman and O'Connor 2007). From this the theory is generated
C. After the theory is generated it may be compared against existing literature
D. During the entire process the researchers will continually write themselves memos

As the core concepts behind Grounded Theory have been introduced it is necessary to look at the methods used in generating a Grounded Theory in detail. The next section will outline these methods and how they are used to generate a Grounded Theory.

3.6.4 Methods in Grounded Theory

So far Grounded Theory has been introduced at a conceptual level. It is necessary to now examine the finer details of the actual methods used when carrying out Grounded Theory Research. The basis of Grounded Theory is in the generation of theory from data and the next sections will examine how this data is gathered and analysed to develop a theory.

3.6.4.1 Initial Coding

The first step that is carried out on the road to developing a Grounded Theory is the *Initial Coding*. Glaser and Strauss (1967) initially proposed two levels of coding: *Development* of as many categories as possible and then the *Integration* of these over-arching categories. This was then later refined by (Strauss and Corbin 1998), who proposed three levels of coding. Whether researchers follow Glaser and Straus' or Strauss and Corbin's methods the Initial Coding step is common to both and a vital step to developing a Grounded Theory (Heath and Cowley 2004). Coding involves reviewing the data that has been gathered (usually in the form of interviews) line by line, word by word, to gain an insight into what participants are saying with the purpose of '*identifying important words, or groups of words, in the data and then labelling them accordingly*'. (Birks and Mills 2011 p. 9). Codes can be considered words or groups of important words whereas the categories are groups of codes (Birks and Mills 2011). In their original text there wasn't a great emphasis on the processes of the actual coding (Heath and Cowley 2004). However, Birks and Mills (2011) note that this was later the focus of more detailed research and there have been a number of methods described for undertaking coding, from the elaborate approaches suggested by Strauss and Corbin (1998) to the more straightforward processes such as those described by Charmaz (2006).

3.6.4.2 Concurrent Data Generation and Collection

The fundamental point of *Concurrent Data Generation and Collection* is that the initial step in Grounded Theory involves the gathering of data and analysis of this data. Then from this analysis further data needs to be gathered (Glaser and Strauss 1967). It is this fact that differentiates Grounded Theory from other research methods. This is because other methods require researchers to collect data and then use this data to test out a pre-conceived hypothesis (Birks and Mills 2011). Grounded Theory lets the data generate theory (Coleman and O'Connor 2007) and then collects further data to explore that theory.

3.6.4.3 Memos

As noted, a key element of Grounded Theory is the use of *Memos*, which are notes that the researchers take during the gathering of data, or in 'fleshing out' of a theory (Coleman and O'Connor 2007). They are '*written records of a researchers thinking during the process of undertaking Grounded Theory study'* (Birks and Mills 2011 p.10). They can take any form that the researcher likes; they can be statements, notes, questions, etc. These memos can become more theoretical as the study progresses and can help in the generation of the theory (Coleman and O'Connor 2007).

3.6.4.4 Theoretical Sampling

Theoretical Sampling is the process of '*collecting, coding and analysing the data'* (Coleman and O'Connor 2007 p.4) to be used as part of the constant comparative analysis. By doing this it should become clear to the researcher where more information is required and results in saturation of categories (Glaser and Strauss 1967). Due to the nature of Grounded Theory, researchers don't know in advance what the theory will be. Due to this they may need to adjust questioning or sample different groups for new data depending on the results of initial sampling, this requires the further step of theoretical sampling (Coleman and O'Connor 2007).

3.6.4.5 Constant Comparative Analysis

When data has been collected as part of a Grounded Theory study it then needs to be constantly check and compared with existing data (Heath and Cowley 2004). This comparisons of codes, to codes, codes to categories and categories to categories is called *Constant Comparative Analysis*; (Birks and Mills 2011). As Grounded Theory is said to inductively develop theory from data (Glaser and Strauss 1967) this method of constant comparison is key to building up theory from data (Birks and Mills 2011).

3.6.4.6 Theoretical Sensitivity

Theoretical Sensitivity relates to the researches own personal insights into themselves and their area of study (Glaser and Strauss 1967). The concept of theoretical sensitivity acknowledges that researchers have their own sensitivities and it is necessary to incorporate this into Grounded Theory study (Birks and Mills 2011).

3.6.4.7 Intermediate Coding

Intermediate Coding is the next major coding step in the analysis of the gathered data. The goal of this phase is for the researchers to fully develop the categories and link them together (Coleman and O'Connor 2007). The difference between this phase and initial coding is that goal of initial coding is to break down the data into codes whereas Intermediate Coding reconnects the data in ways that are more conceptually abstract (Birks and Mills 2011). This is referred to as *Axial Coding* in later writings of Strauss and Corbin (1998).

3.6.4.8 Identifying the Categories

Through the previous steps of Intermediate Coding and linking categories should allow researchers to identify core categories emerging from the data. This can happen through ensuring that there is full category saturation (Birks and Mills 2011).

3.6.4.9 Advanced Coding and Theoretical Integration

Advanced Coding and Theoretical Integration is a crucial stage for the development of the integrated Grounded Theory (Birks and Mills 2011). Once the core category has been chosen this then gives the researchers the power to explain theoretically what their research is about and what the data is telling them (Strauss and Corbin 1998). Grounded Theory should produce an explanation of a phenomenon that is comprehensive and accepts and explains variances in data and research questions (Birks and Mills 2011). There are techniques used to develop this integration of core categories such as writing storylines; which can be used to develop and present the Grounded Theory (Strauss and Corbin 1998). This theoretical integration can give the Grounded Theory more explanatory power by placing it in context of the existing literature and body of knowledge on the subject matter under examination (Birks and Mills 2011).

3.6.4.10 Generating Theory

The final step is to produce a theory that is integrated, comprehensive and explains the situation under study (Birks and Mills 2011). Grounded theory is developed using the methods described above and can conceptually through of as '*three cogs that can drive a machine (you) to generate Grounded Theory'* (Birks and Mills 2011 p.13). They go from the larger (more general) cogs, to the smaller (more specific) cogs and advance practices (in the smaller cog) but they all contribute to the development of Grounded Theory.

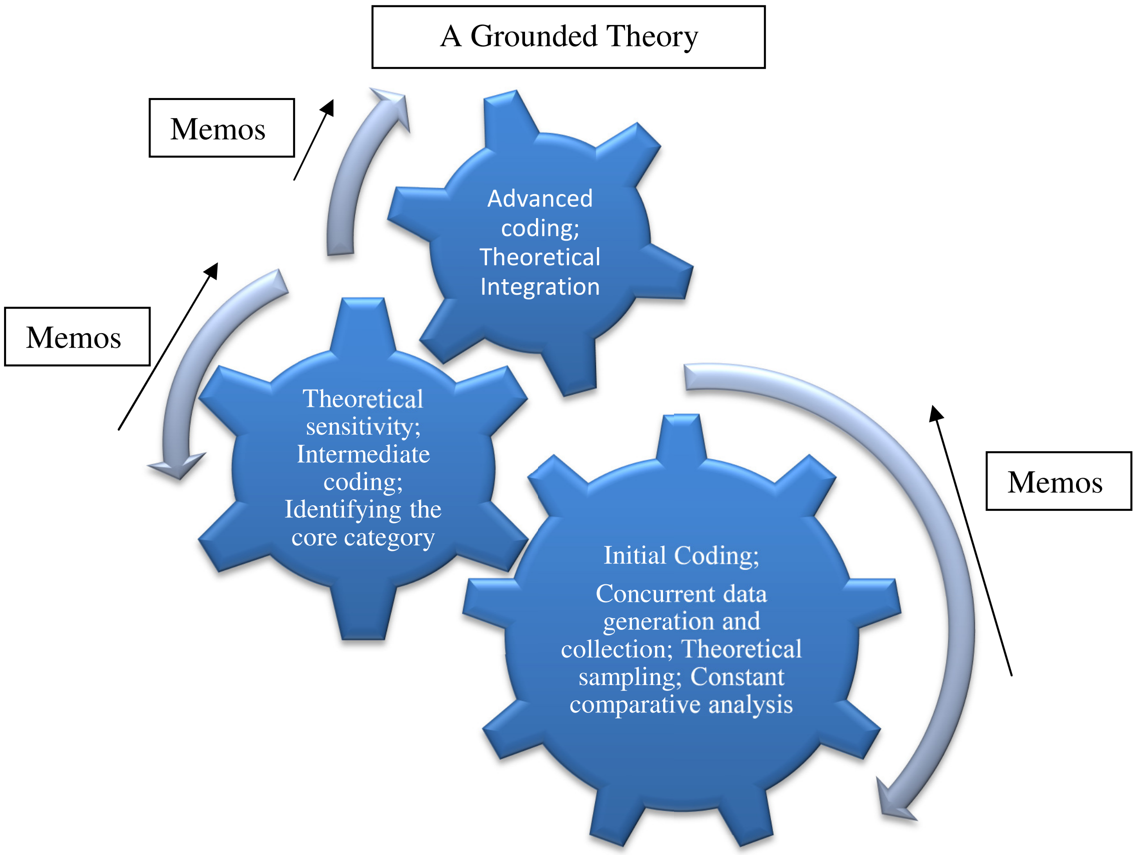

A Grounded Theory
Memos
Advanced coding; Theoretical Integration
Memos
Theoretical sensitivity; Intermediate coding; Identifying the core category
Memos
Initial Coding;
Concurrent data generation and collection; Theoretical sampling; Constant comparative analysis

Figure 3.2: Essential Grounded Theory Methods (Birks and Mills 2011 p.13)

3.7 Competing Versions of Grounded Theory

The fact that there are competing versions of Grounded Theory has been alluded to previously but it is now necessary to investigate this in detail. There have been significant developments in the research area around Grounded Theory since Glaser and Strauss' seminal text. Since the 1967 book Glaser and Strauss have gradually separated professionally and have developed differing versions of Grounded Theory

(Coleman and O'Connor 2007). As noted Strauss came from a background of Symbolic Interactionism and Glaser from a more rigorous background in descriptive statistics (Morse *et al.* 2009). It has been highlighted that this difference in their backgrounds may have led to their divergence in thinking around Grounded Theory research, but others point out that this difference was always present (Heath and Cowley 2004). In fact it wasn't until Strauss and later Strauss and Corbin (1998) published their revisions of Grounded Theory that the division was most apparent (Coleman and O'Connor 2007). Glaser (1992) criticised the work of Strauss and Corbin accusing it of being overly prescriptive in outlining how studies should be carried out. Thus, he argued this was forcing theory onto the data, rather than allowing theory to develop from the data (Coleman and O'Connor 2007). It has been noted that Glaser (in his works of 1978) was seen to stick to the original tenants of the Grounded Theory with Strauss and Corbin (1998) developing a new formulation of the model (Heath and Cowley 2004).

In response to some criticism of the vagueness of the original model Glaser himself tried to expand on concepts such as theoretical sampling and memo's in more detail (Glaser 1978). However, it was Strauss and Corbin's focus on developing analytical techniques to help novice researchers to implement Grounded Theory that drew most criticism from Glaser (Heath and Cowley 2004).

Glaser and Strauss differed in other more specific areas of Grounded Theory. Glaser for example believed that research questions should only emerge during coding but Strauss and Corbin (1998) argue that research questions need to be pre-set in order to give researchers boundaries and to guide the research process (Coleman and O'Connor 2007). They also differed in their approach to reviews of literature. Glaser (1978) believed that researchers should only review the literature after theories have emerged to avoid prior reading that may influence emerging theories (Adolph *et al.* 2012). He believed that any prior knowledge of the subject area should based on general reading of the problem area rather than on focused literature reviews (Heath and Cowley 2004). Strauss and Corbin (1998), however, are more realistic in their approach about the idea that researchers will have some prior knowledge in the field of study before entering it. Indeed (Strauss 1987) argued that the use of prior knowledge and experience can be used for theoretical sensitivity and generating theory (Heath and Cowley 2004).

There was also a difference when it comes to their approach to coding. Glaser and Strauss (1967) originally proposed two levels of coding with Strauss and Corbin (1998) adding another. The table below outlines the differences in methods:

	Strauss and Corbin	**Glaser**
Initial Coding	***Open Coding***	***Substantive Coding***
	Use of analytical techniques	Dependant on data
Intermediate Phase	***Axial Coding***	***Continuation of above***
	Reduction and clustering of categories	Comparison, with focus on data, become more abstract, categories refitted, emerging frameworks
Final Development	***Selective Coding***	***Theoretical Sampling***
	Detailed development of categories, Selection of core, Integration of categories	Refitting and refinement of categories which integrated around emerging core
Theory	Detailed and dense	Parsimony, scope and modifiable

Table 3.1: Comparison of Strauss and Corbin and Glaser in terms of levels of coding (Heath and Cowey 2004)

3.7.1 Open Coding

In the method proposed by Strauss and Corbin (1998) the first level of coding is *Open Coding*. In this level of coding the researcher uses analytical techniques to assign codes to words or text in interviews. The techniques involve '*breaking down interviews and observations into distinct units of meaning which are labelled to generate concept'* (Coleman and O'Connor 2007, p.5). The codes will later become concepts, from this initial set of codes generated these are used to code subsequent interviews and researchers should end up with a large number of codes at the end of the research process (Coleman and O'Connor 2007). Glaser (1978) calls this stage *Substantive Coding*.

3.7.2 Axial Coding

The next stage is the *Intermediate Phase* of coding. This is referred to as *Axial Coding* by Strauss and Corbin (1998). Due to the intense and significant questions that is necessitated in their version of Grounded Theory an extra layer of coding is required (Heath and Cowley 2004). In this phase the reduction and relating of categories to one another is undertaken. It is called axial because '*coding occurs around the axis of a category'* (Coleman and O'Connor 2007 p.5). In Glaser's model there is no concept of further coding in this intermediate stage and there is simply continuation of coding of substantive coding (Heath and Cowley 2004).

3.7.3 Selective Coding

The next stage is the final development of the theory. Strauss and Corbin (1998) refer to this stage as *Selective Coding*. This is the process of final integration of categories. In this stage the core is selected and integrated to develop a theory. Glaser (1978) terms this stage as *Theoretical Sampling* and this involves the refinement of emergent categories and their integration. Thus, while both deal with integration, the emphasis from Strauss and Corbin (1998) is on the selection of core categories while Glaser is concerned with refinement of categories (Heath and Cowley 2004).

In conclusion, whichever version of Grounded Theory is chosen is often determined by the preference and background of the researcher, but it is necessary to understand that there are different implementations of the theory (Coleman and O'Connor 2007). It is now necessary to highlight the use of Grounded Theory in the Software Engineering domain.

3.8 Grounded Theory in Software Engineering

As has been noted, due to the nature of it and its ability to explain social and cultural phenomenon, Grounded Theory has been traditionally used in areas such as sociology and nursing (Creswell 2006). It has been seen to be able to explain how behaviours of people shape processes and help us to understand how people interact (Glaser and Strauss 1967). Because Grounded Theory helps understand human behaviours, the technique has more recently been employed as a research method in the areas of Business and Information Technology (Coleman and O'Connor 2007).

A number of researchers have advocated the use of qualitative research methods (and specifically Grounded Theory) in the area of Software Engineering (Coleman and O'Connor 2007). For example, Silva and Backhouse (1997) claim that by using qualitative methods in software engineering research can lead to the development of theories that are grounded in an interpretative or phenomenological paradigm. This allows them to be more understandable and consistent (Coleman and O'Connor 2007). Hoda *et al.* (2010) used Grounded Theory to develop a theory and framework for organising teams in an Agile software development environment (Orlikowski 1993). In perhaps one of the best examples of the use of Grounded Theory in Software Engineering, Coleman and O'Connor (2007) show that Grounded Theory can be used to explain how software development activities in two organisations were affected by the introduction of CASE tools.

Further to these studies Coleman and O'Connor (2007) highlight a number of studies that have shown Grounded Theory has been successfully implemented in research studies in various areas of Software Engineering such as:

- Software inspections
- Systems thinking
- Requirements documentation
- Process modelling
- Virtual team development

It has also been shown that Grounded Theory is useful as a method where there is a lack of original research and theory (Adolph *et al.* 2012). Through the investigation of the literature it was found that there is a significant lack of empirical studies looking at frameworks or best practice test methodologies, it is for these reasons that Grounded Theory was selected as a suitable candidate for gathering and analysing data in this dissertation.

3.9 Conclusions

The research question this dissertation is an investigation of the use of Grounded Theory to develop a model or framework for testing best practices. In order to understand Grounded Theory as a research method it was necessary to firstly introduce the idea of research methods. This chapter firstly outlined the two major different types of research methodologies (Qualitative and Quantitative). Next, each area was outlined in detail from the philosophical backgrounds of each to various examples of the different methods. Next, Grounded Theory was examined in detail (as this is the method used in this dissertation). Firstly the background to the development of the theory was outlined. Following this the methods used in the development of Grounded Theory were discussed. Next, the fact that there was a split between the creators of the theory was discussed and the differences in their methods were examined. Finally, the use of Grounded Theory as a method of research in the Software Engineering sector was highlighted. The next chapter will outline the organisation in more detail.

4. ORGANISTAIONAL BACKGROUND

4.1 Introduction

The core aim of this dissertation is to use Grounded Theory to develop a framework for software testing best practices in a Telecommunications organisation. However, before the Grounded Theory can be developed it is necessary to understand the organisation in which the theory is to be generated. This chapter will start by outlining in detail the structure of the organisation including the various sections. It will focus in particular on the IT section and the Test Team within that section. This description will include details of how testing is currently undertaken within the organisation. This will provide background and an "As-Is" view of testing practices.

4.2 Organisational Background

This section will outline some background to the organisation in which this Grounded Theory research is being conducted. It will give an overview of the company at high level and outline the various departments. Particular focus will be placed on the IT department where the key experiments are being undertaken in this research. In this section the way projects are delivered will be discussed, with emphasis on how projects are tested. This will give the necessary "As-Is" picture which will put the interviews and their discussion into context.

4.2.1 Organisational Structure

The organisation involved in this research is a large, multi-national telecommunications company based in Dublin. There are over 800 people employed in the head-office with many more based in retail shops around the country. The organisation has many departments as might be imagined for a company this size. In order to understand the workings of the company it would be necessary to understand and outline what each area does.

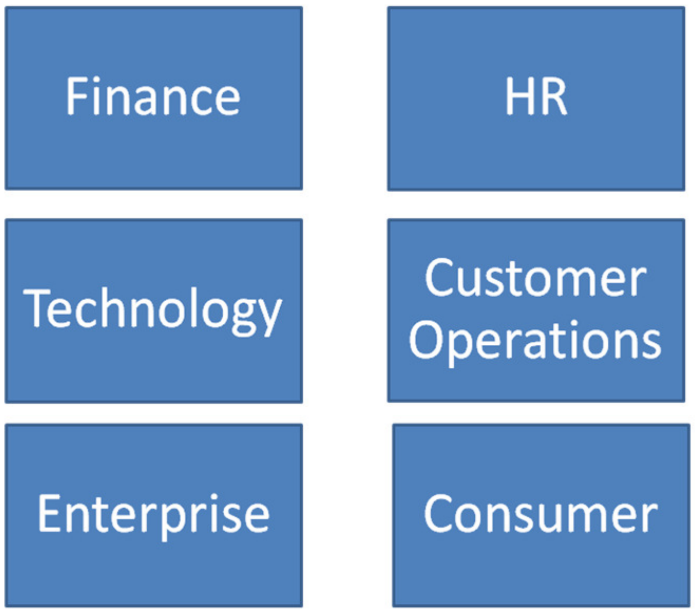


Figure 4.1: Outline of Organisational Departments

4.2.1.1 Finance Department

The finance department are responsible for all financial processes in the organisation. This encompasses all areas from Payroll to Purchasing. There are various teams within the finance department that look after specific areas. For example, the Revenue Assurance team are responsible for ensuring that no revenue loss occurs when new IT projects (such as introducing new tariffs) are undertaken. There are several elements to the Finance team, from Commercial Finance, Financial Strategy and Planning, Fraud and Risk Assurance to Supply Chain Management. The Finance team also provide direction and guidance on financial issues to various business units within the organisation.

4.2.1.1 HR Department

As with most large companies there is an entire department dedicated to HR functions. They are responsible for looking after staff needs, assisting with hiring of new staff and running the yearly review process as well as dealing with day to day HR issues. HR understands that it is the people in the organisation that drive the business success through innovation and creating the best customer experience. It is the job of HR to support the employees of the organisation to allow them to deliver these results. They are also responsible for the energising and rewarding staff in order to help them to perform most effectively and deliver on their potential.

4.2.1.2 Customer Operations Department

This group are responsible for all customer facing sections of the business. This would include the retail stores as well as the call centres. They are responsible for ensuring that customers' needs are met through ensuring customer call centres are staffed and have the right equipment and systems to allow customer service representatives to do their job. They also have a section whose sole role is in the ensuring that an IT projects do not negatively impinge on the customer experience (the customer either being a customer service representative using an internal IT system or an external customer interfacing with the company through the various channels, e.g. the company's website and self-care sites). This team is called the Customer Experience team.

4.2.1.3 Consumer Department

This group are responsible for the entire mobile and fixed line customer base of consumer customers (i.e. individual consumers of the products and services, not businesses). They are responsible for both the pre-pay (in the mobile sector only, i.e. non-contracted "Ready to Go" customers) and post-paid (in the fixed and mobile sector i.e. customers who are on bill-pay contracts for various periods of time). This team will come up with new propositions to sell to customers in order to generate revenue (e.g. new sets of tariffs). They will frequently engage with Technology to deliver innovative projects that will increase the customer base or reduce churn (customers leaving for competitors).

4.2.1.4 Enterprise Department

This team are essentially the equivalent of the Consumer team but are responsible for business customers. These customers range in size from SME's to large blue-chip companies and even Government departments. As with the Consumer team, this team is responsible for both the fixed and mobile customers. This department has a large sales team. Again, Enterprise will often engage with technology in order to deliver new products and services to their customer base.

4.2.1.5 Technology Department

At a high-level the Technology department is divided into two major teams: the *Networks* team and the *IT* team, both under the leadership of the CTO. These streams have very different responsibilities.

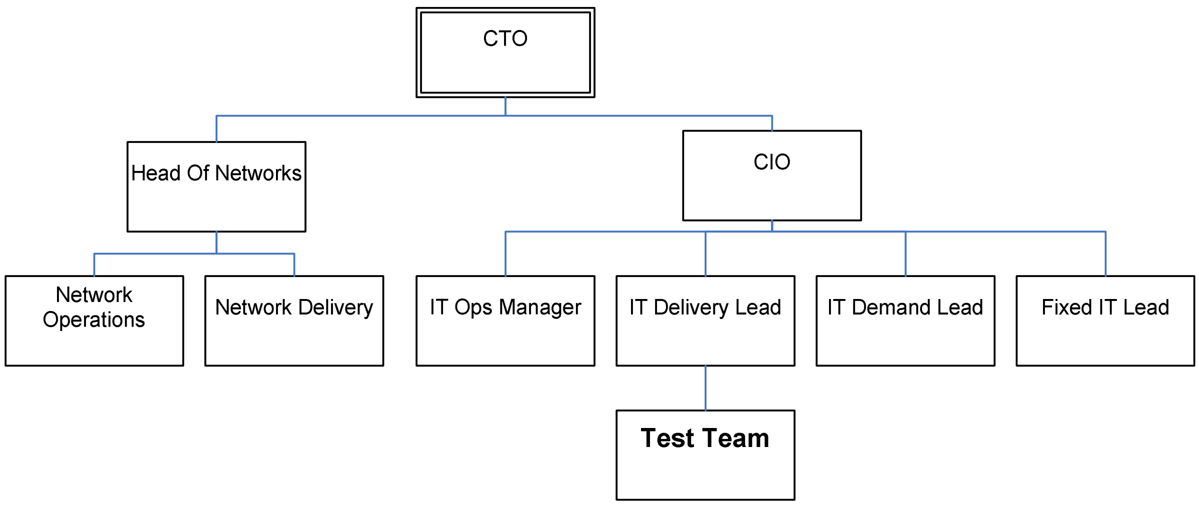


Figure 4.2: Technology Org Chart and Place of Test Team in Org Structure

4.2.1.5.1 Networks Team

The Networks team are responsible for all elements of the companies' mobile network, and is lead by the Head of Networks. They are the team that ensure people from Cork to Donegal and Dublin to Mayo can use their phones to make and receive calls and text and also (increasingly these days) can send and receive data on their Smartphones. There are a number of areas and sub-teams within this team. Some of the team work towards delivering projects for the Consumer/Enterprise teams in conjunction with the IT team. Others are purely responsible for care and maintenance on the various telecoms masts that the company own and manage. There is a Network Operations Centre that operates on an around-the-clock basis, who continually monitors the Network performance and can respond to issues of outages immediately.

4.2.1.5.2 IT Team

The other major part of the Technology department is the IT team and this team is under the leadership of the CIO. The main responsibility of this team is the care and maintenance of the IT systems within the company. There are a number of different sub-teams in the IT area, each with their own specific function:

4.3 IT Components of The Organisation

4.3.1 IT Operations

This team is similar in function to the Network Operations team but with a focus on the IT systems rather than network elements. They are responsible for the maintenance and troubleshooting of all IT systems. This involves all issues with customer care front ends (including the company's website) as well as infrastructure (e.g. servers, databases etc). This is also operates on an around-the-clock basis. There are three levels of support; first-line is typically the highest level of investigation and the basic troubleshooting investigation. If the team at this level cannot resolve an issue then it is escalated to second-line support. If the second-line support cannot resolve it gets passed to third-line (which is usually the individual vendors). This team are involved in all IT delivered projects from a number of areas. They will be the team responsible for monitoring and supporting new systems when they go live so they need to ensure that they are satisfied with the level of documentation that they are receiving from vendors to allow them to support the application in Production. They also act as a quality monitor. At the end of testing, before IT Operations will accept handover of the project into Production they will require a *Test Closure Report* that highlights all outstanding issues that will not be fixed prior to release. There needs to be a plan in place for these to be resolved in Production during project closure. If any issues are raised during a project then it will be raised with IT Operations. The types of issues raised will vary depending on the project; some may be related to tariff options being incorrectly displayed on the customer care front-ends, other may relate to hardware failures.

4.3.2 IT Demand and Delivery

This large team is responsible for planning, scheduling and delivering all new IT projects for the company (both mobile and fixed). There are multiple sub-teams in this area with individual managers who all report into the CIO.

4.3.2.1 Architecture and Demand Sub-Team

This team is lead by the IT Demand Lead and is responsible for the demand planning and scheduling of all projects in the Delivery work stream. There are two sections to this team, the Demand side and the Architecture side. The Demand side will usually interface with the Business (this will usually mean representatives from the Consumer and Enterprise teams responsible for new project delivery) to develop a pipeline of new projects that they want delivered in the coming financial year. Once there is an initial overview of the workload, the projects are handed to the Architecture sub-group. It is their responsibility to investigate and understand technical aspects of the project and to give an initial estimate on the feasibility and size of the project. They will also develop a view as to the number of vendors that will be required in order to deliver the projects. All projects then go to a Project Board who will decide which projects will be delivered in the timeframe under consideration (usually the financial year). Once projects have been approved and funding has been put in place they are handed over to the IT Delivery Group to implement and complete.

4.3.2.2 IT Delivery Sub-Team

This team is responsible for the delivery of the IT projects agreed by Demand and the Business Owner. It is lead by the Head of Delivery who reports into the CIO. This team has also multiple sub-teams that are broken down by role. There is a Project Management team which is a team of Project Managers (PM's) lead by the Senior PM. They are assigned multiple projects to lead and deliver all aspects of those projects. Then there is a team of Release Managers who are responsible for delivering releases of multiple projects (there are usually three release cycles a year). There is a team of Business Analysts who are responsible for the requirements gathering on projects. Again they are assigned to multiple projects and are lead by a Business Analyst Manager. There is also a Test Team in the Delivery side. This is made up three Test

Leads and two Test Analysts (Full Time) and supplemented by contractors as required. This team is responsible for leading the Testing phases of the projects and ensuring quality deliveries. The actual roles and responsibilities of this team will be examined in more detail in the following sections of the dissertation.

4.3.3 Fixed IT Sub-Team

The Fixed IT team is responsible for the delivery of all Fixed (i.e. landline) projects in the company. They will interface with IT Demand and Delivery team frequently as there is a very close collaboration required when delivery projects as the fixed IT systems are in the process of being integrated with the mobile systems.

Now that a background to the company has been established it will be helpful to look at the IT project delivery lifecycle in more detail.

4.4 IT Project Delivery Lifecycle

This section of the dissertation will outline the processes and procedures that are used in delivering an IT project. There are various types of projects delivered, such as introducing new suite of tariffs for bill pay customers, introducing new software in the call centres or delivery an entire new mixed mobile and fixed proposition that requires significant architecture changes. No matter what type of project it is, they will all follow the standard phases outlined below.

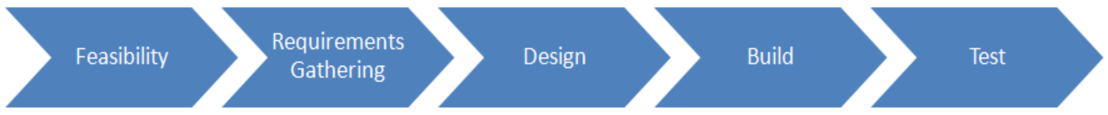


Figure 4.3: Project Phases

4.4.1 Feasibility

This is the first phase of a project and involves a number of steps and key stakeholders. At this stage the project have been signed off by the Release Board, but based on very high-level details contained in a one-page document. This first stage involves acquiring further details from the Business on what the project needs to deliver. There is also an updated project-sizing session where all relevant stakeholders (vendors, business analysts, architects) will meet to discuss any updates to initial sizing. It is here that all of the Test phases of the project will be defined. Once this is complete and all stakeholders are satisfied with sizing there is a quality gate meeting that ensures that the project can proceed into the next phase.

4.4.2 Requirements Gathering

This is the first stage in the project proper. It involves the Business Analysts working with all stakeholders to come up with a list of requirements. There are numerous workshops held with various parties (vendors, business, architects etc). During these workshops the Business Analyst needs to take into account what all parties want and write them down in a *Requirements Document* and assign an owner to each requirement. As noted previously there is no in-house development done during any projects. This is all completed by vendors it is the vendors who will sign off on the requirements assigned to them to state that they can deliver these. The output from this stage is a signed off *Requirements Document* that becomes the baseline against which all vendors will base their design, build and testing. Any changes after the *Requirements Document* has been signed off require a Change Request to be submitted. Another output that is required is the completion of Use Cases; these are also completed by the Business Analyst with input from the vendors, the Business users and a representative from the Customer Experience team. Once again at the end of this phase there is a quality gate meeting chaired by the release manager. The PM for the project must ensure that the *Requirements Documents* (as well as other pre-requisites) have been completed. If all documentation is in place then they will be allowed to proceed to next stage.

4.4.3 Design

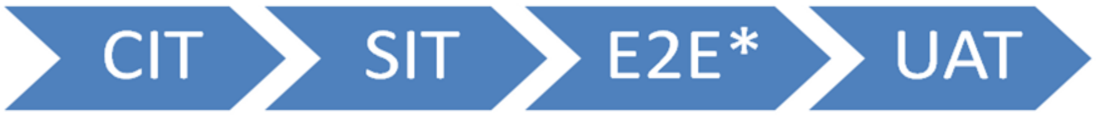
This phase is under the direction of the Project Manager (PM). However, this is usually a phase where vendors will complete all of their High-Level and Low-Level Design documentation. During this phase the PM is responsible for co-ordination of any activities and acts as a conduit for any dependencies certain vendors have on each other to allow them to complete their design documents. For example, one vendor may need to know what format to expect a message from another vendor so that they can design their interfaces to accept this message. If there are delays from certain vendors at this stage, this can delay the entire project and has to be managed closely by the PM. The output from this stage is that all vendors have completed their High-Level and Low-Level Design documentation. This are usually shared across all parties. It is during this phase that the in-house Test Team are first actively engaged. Once the CRD and Use Cases have been signed off the Test Team can begin writing test cases. This will be discussed in more detail later in this section.

4.4.4 Build

Similar to the Design phase, this phase of the project is mostly driven by vendors (under the guidance of the PM). In this phase all vendors will complete the coding and building of their various parts of the project. This is usually done in each of the vendors own environment and any calls out to other systems that are required during unit testing are stubbed. There will be weekly project meetings held by the PM. During this phase each vendor will give updates on their progress and highlight any delays or issues that they are facing.

4.4.5 Test

The Test phase is the first phase where all the vendors come together in the one environment. There are a number of test phases that a project goes through before it is released:



* Not every project has the E2E test phase; it depends on the size and functionality of the project.

Figure 4.4: Phases of Testing within a Project

The first two (CIT and SIT) phases are vendor driven phases and they are responsible for these. The next two (E2E and UAT) phases are under the responsibility of the organisation's In-House Test Team.

4.4.5.1 The CIT Phase

Component Integration Testing (or CIT) is the first level of testing in the overall Test phase. This is usually completed by vendors in their own environments. It is the first phase where they are able to test that their new components will work properly with their existing components. This can be thought of as slightly more complex unit testing process. There is very little involvement from the PM or the in-house test team (hereafter shall be referred to as "the Test Team") during this phase. The vendors will supply a weekly status report on the Test progress during this phase. Each vendor will have their own exit criteria for this phase.

4.4.5.2 The SIT Phase

System Integration Test is the next level of testing on projects. This is the first phase where all vendors will test the integration of their code with that of other vendors in a fully integrated environment. There should be no use of stubs during this test phase as all parties should deploy working code to the test environment. This phase is driven by vendors however there is more involvement by the Test Team during this phase. They act as Project Managers for this phase. Depending on the complexity of the project they may need to come up with a Test Strategy that all vendors will need to execute, they will manage inter-vendor dependencies (e.g. Vendor A may execute a test from their system that requires Vendor B to accept a message and respond with another message. If Vendor A doesn't receive the message they expect they may need to get Vendor B to investigate an issue in their code). The Test Team will act as Systems Integrator and will liaise and manage these relations between vendors. They will also run daily defect meetings and produce a daily status report (which is an amalgamation of all individual vendors' reports). There are set exit criteria here that all vendors must meet in order to allow progress to the next phase of testing. There is a quality gate at the end of this phase and if all exit criteria are met, they will be allowed to progress to the next phase of testing.

4.4.5.3 The E2E Phase

End-to-End Testing is the first phase under the direct control of the Test Team. As noted, this is not required on every project. However, currently during the wider sizing in the Feasibility, if the project is a *Large*[1] or *Extra Large*[2] in size then there is automatically six weeks of E2E testing assigned to the project.

During E2E test execution is completed either by the vendors under the direction of the Test Team or by the Test Team themselves. During the SIT phase vendors will complete integration tests so that they ensure that their systems can integrate with all others. However, there are no tests that will take flows from end-to-end in the SIT phase which is often necessary. For example in order to set up a provision (set up) a customer on the mobile network might involve one vendor (A) to initiate calls to various other vendor systems (B and C). Thus, vendor A will send out a call to B and C, once B and C respond with the correct data A is satisfied. However, this phase was introduced as it was found that for some projects there is a need to complete the entire provisioning of a customer (for example) from start to finish. Thus, in this case A would send calls to B and C. B and C would then take this data and complete all the necessary actions on their side and then send a completion note back to A. Then A would need to take all this data and ensure that the customer was set up correctly (i.e. use this customer details to make a phone call).

This phase requires close collaboration between vendors as data needs to be handed from one vendor to another at very specific points in time. This phase attempts to mimic real customer interactions with the system from start to finish. During this phase the Test Team will act again as mangers of this phase, they will be responsible for planning all the tests that need to take place and getting all pre-requisites in place (e.g. data and environments). They will then be responsible for managing vendor execution (or internal execution depending on circumstances), running daily defect meetings and reporting status. As with all phases at the end of this phase there is a quality gate here and the Test Team need to ensure that all testing has completed.

[1] A *Large* project is one that has between 500 and 1500 man-days effort

[2] An *Extra Large* project has over 1500 man-days effort

4.4.5.4 The UAT Phase

User Acceptance Testing (UAT) is the final phase of testing that the Test Team has direct responsibility for. During this phase the Test Team have written a number of test cases based on business flows, these flows are meant to capture how the end users will use the system on a day-to-day basis. This was originally conceived as a test phase that the Business should own and manage, as it is a final test phase to say that they, the users, accept the product that is being delivered to them. Thus, they business owners (or representatives from the business) should devise tests and then run them. They should be responsible for the overall sign off on the acceptable quality of the project. However, in reality this does not happen. The Test Team are the owners of this phase and will usually write test cases and then execute them. This is done in close collaboration with the business users but they don't take active part in the test phase.

Before the test phase begins a request is submitted to the Business Owner to have an end user (usually a customer service representative) to be released from their day to day to work to assist in the test execution. This request is not always able to be facilitated due to the staff shortages in the call centres and in these situations the test execution is completed solely by the test team. This phase is similar in focus to E2E however; this is completed in a pre-production environment as opposed to test environment to fully mimic the behaviours that would be expected in the Production environment.

During this phase the Test Team will manage the phase, run daily defect meetings and complete daily status reports. Once again there is a quality gate at the end of this phase. If testing is completed and exit criteria then the project can move into the next phase.
UAT is the last phase of functional tests that are carried out. Depending on the project there are some other types of testing that are under taken before the project is released to Production (e.g. ORT or Performance Testing).

Operational Readiness Testing is completed by the IT Operations team and is a phase that is used by the Operations team to ensure that they can support the project once it goes into Production. Thus, they will complete activities such as troubleshooting issues, bringing down servers and re-starting them. If there are to be no non-functional testing to be carried out on a project then it will be released from UAT (in the Pre-Production environment) to Production. Once in Production there is a Closure phase where any issues that are raised during this period will be resolved by the vendors. Once Closure is over the project is completed, various activities are completed by the PM and Release Mangers and the project is shut down.

4.5 Test Team

The Test Team is a relatively new team (only being formed in the last two years). It consists of a Test Manger, four Test Leads (three full-time, one contractor) and three Test Analysts (two full-time, one contractor). Each role carries out a particular, specific function.

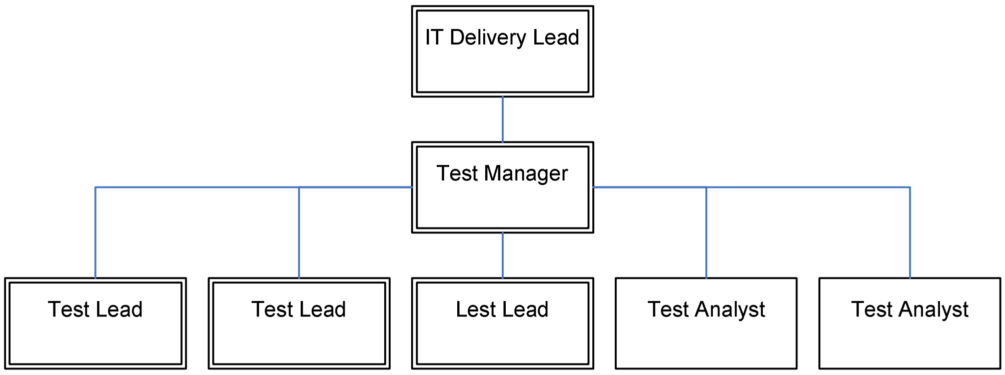


Figure 4.5: Organisational Chart of the Test Team

4.5.1 Test Manger

The Test Manager is responsible for all Testing activities that the team undertake. They are responsible for the staffing of the team and co-ordination of roles. They are the first point of contact from external teams to the Test Team. Once projects get approved the PM will approach the Test Manager to have a resource assigned to the project (usually a Test Lead and then later a Test Analyst). The Test Manger is also responsible for defining the overall Test Strategy and leading the direction of the team.

They also have People Management responsibility for the team members and will conduct one-to-one meetings with all members and undertake performance reviews. They report directly to the IT Delivery Lead.

4.5.2 Test Lead

Test Leads are responsible for the day-to-day management of the Test phases on projects. They will adopt a leadership role in projects and work closely with the PM. They should be engaged on a project as early as possible and should lead the work-stream across the SIT/E2E/UAT phase of the project. Their emphasis in those phases is on designing, planning and managing the overall Test work stream and is the single point-of-contact for test-related queries on the project. They are responsible for defining the overall testing strategy for the project. The two major deliverables that Test Leads must develop are the *Test Strategy* and the *Test Approach*. The Test Strategy outlines at a high level the test strategy for the project, this will include things like what phases of testing are undertaken on the project, defect management process, high level timelines. The Test Approach is a much more detailed document that outlines, at a lower level, the actual types of testing that each vendor will undertake on the project. It is completed by the Test Lead but takes as input Vendor Test documentations (such as their Test Plans and Test Scripts). They will also need to create a detailed Test Plan. The Test Lead is also responsible for the communication of Test progress via daily status reports. They will also run daily defect meetings during Test phases and send out test closure report and feeding into the PM's lessons learned document once the project is completed.

4.5.3 Test Analyst

The Test analyst is responsible for design test cases and writing them. As noted they will then also be responsible for execution of these test cases. They will also be responsible for supporting Business users in executing test cases if they are engaged on the project. Where Business users are to be used the Test Analyst is responsible for planning how many will be needed and submitting the initial request. They are responsible for submitting data requests and other activities related to execution of test cases. They will ensure that all test cases are uploaded into the Test Management tool (this is a piece of software that is used to manage the execution of test cases and for raising defects). This will allow the Test Lead to generate test reports. They will work closely with the Test Lead to report on progress and escalate any issues that are blocking progress. They have two major deliverables during the project are the development of the Test Case Matrix and the creation of Test Scripts. The Test Case Matrix outlines, at a high level, the types of test cases that will completed on a project. This is usually test case names (it should be clear from the names what the point of the test case is). This Matrix should also include traceability back to the *Requirements Documentation* to ensure that all requirements that can be tested are (from a user perspective). The second deliverable is the development of the Test Scripts, which are detailed step-by-step execution test scripts. These outline in very low-level detail what needs to be done to execute the test script. They will tell whoever is executing the test cases the exact steps to follow. These test scripts will be uploaded to the test management tool for the execution phase.

4.6 Conclusions

This dissertation is related to the development of a best practice framework for software testing in a telecommunications organisation. In order to understand what changes were to be recommended as part of this study it is necessary to understand the "As-Is" situation in the company. This chapter outlined a background to the company; this included detailing the structure of the organisation and focused specifically on the IT and IT Delivery teams. It detailed how projects get delivered and what teams are involved in this. It also examined in detail how testing is currently undertaken on projects. This involved outlining all the phases of test and how the Test Team is structured and how they contribute to the testing on projects. This constituted the "As-Is" view. The next chapter will outline how the data used to generate the theory was gathered and what process guidelines exist in the organisation at present.

5. GATHERING THE DATA

5.1 Introduction

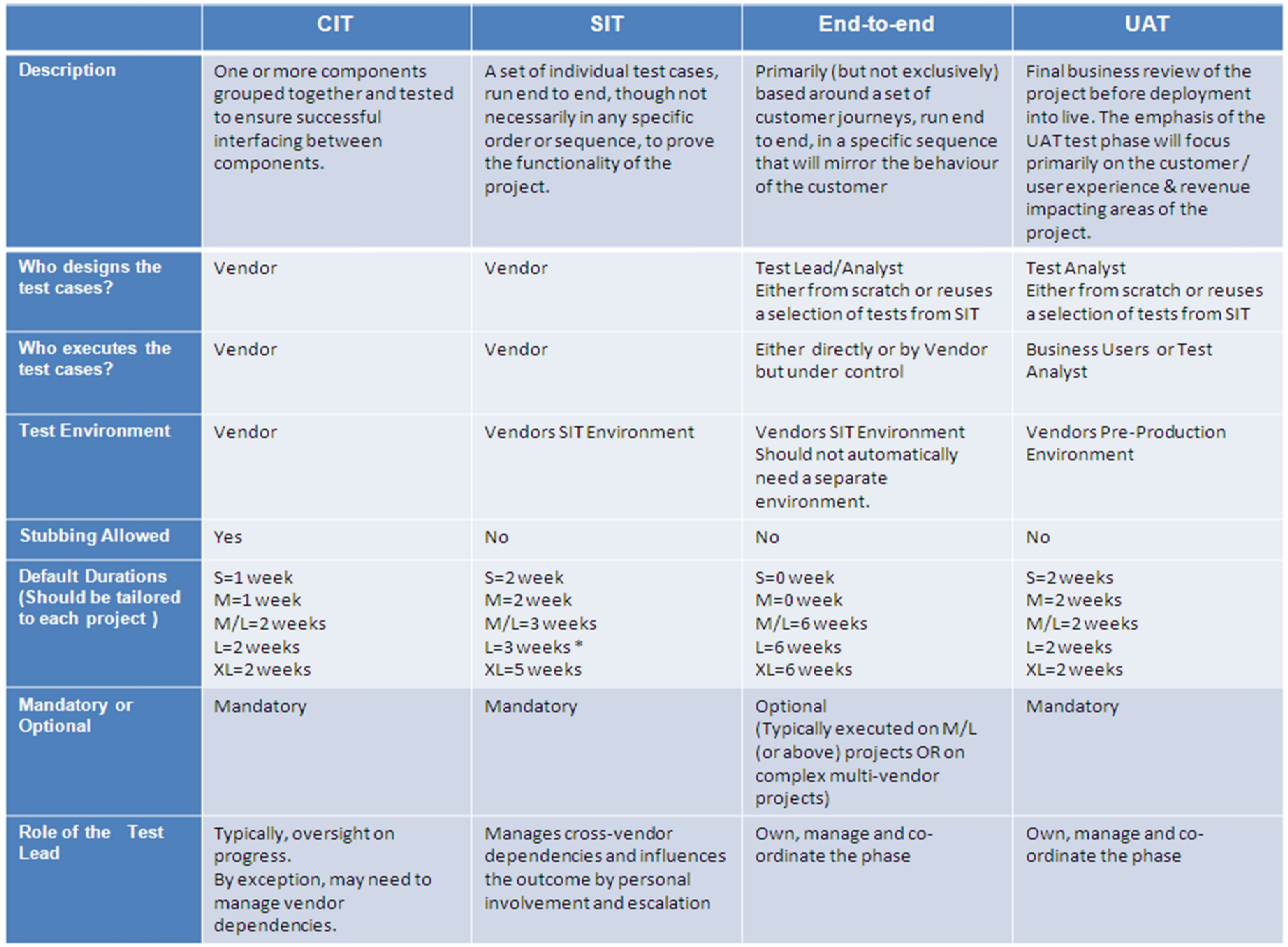
The aim of this dissertation is to create a framework for testing best practices in a telecommunications organisation using Grounded Theory. The last chapter gave a background into the company and the "As-Is" situation of current testing practices. This chapter will outline in detail how the data used to generate the new Grounded Theory was gathered. The first section will discuss the knowledge processes that exist within the organisation. The next section will outline the interview stage of the data acquisition. This will include an overview of the interview process, (data for development of the Grounded Theory was gathered using interviews) it will outline the participants interviewed and included samples of the types of questions asked and answers provided.

5.2 Existing Knowledge

There is vast array of information about various sources within an organisation of this size. This is true in the organisation understudy. There are published guidelines for how Testing should be undertaken and an intranet wiki site that outlines what is expected from the Business Users during the Testing phase. This section aims to highlight that there are processes and procedures that should be followed during the Testing phase. However, in the interview section it will become clear that they aren't always being fully followed.

5.3 Test Methodology

When the team was initially set up there was a great deal of effort invested in developing a set of clear processes and documentation to describe how the team should undertake testing. This was included in an overall Test Methodology document. It detailed what was expected of the Test Team in all the Test phases and gave an overview of these phases. On the next page there is a table taken from this document that details the test phases and the Test Teams involvement in these phases:



	CIT	SIT	End-to-end	UAT
Description	One or more components grouped together and tested to ensure successful interfacing between components.	A set of individual test cases, run end to end, though not necessarily in any specific order or sequence, to prove the functionality of the project.	Primarily (but not exclusively) based around a set of customer journeys, run end to end, in a specific sequence that will mirror the behaviour of the customer	Final business review of the project before deployment into live. The emphasis of the UAT test phase will focus primarily on the customer / user experience & revenue impacting areas of the project.
Who designs the test cases?	Vendor	Vendor	Test Lead/Analyst Either from scratch or reuses a selection of tests from SIT	Test Analyst Either from scratch or reuses a selection of tests from SIT
Who executes the test cases?	Vendor	Vendor	Either directly or by Vendor but under control	Business Users or Test Analyst
Test Environment	Vendor	Vendors SIT Environment	Vendors SIT Environment Should not automatically need a separate environment.	Vendors Pre-Production Environment
Stubbing Allowed	Yes	No	No	No

Default Durations (Should be tailored to each project)	S=1 week
	M=1 week
	M/L=2 weeks
	L=2 weeks
	XL=2 weeks
		S=2 week
		M=2 week
		M/L=3 weeks
		L=3 weeks *
		XL=5 weeks
			S=0 week
			M=0 week
			M/L=6 weeks
			L=6 weeks
			XL=6 weeks
				S=2 weeks
				M=2 weeks
				M/L=2 weeks
				L=2 weeks
				XL=2 weeks

Mandatory or Optional	Mandatory	Mandatory	Optional (Typically executed on M/L (or above) projects OR on complex multi-vendor projects)	Mandatory
Role of the Test Lead	Typically, oversight on progress. By exception, may need to manage vendor dependencies.	Manages cross-vendor dependencies and influences the outcome by personal involvement and escalation	Own, manage and co-ordinate the phase	Own, manage and co-ordinate the phase

Figure 5.1: Overview of Test Phases and Test Team Involvement

5.4 SharePoint Site

There is a SharePoint site that is dedicated to helping the business engage with the Technology team. It outlines all the various channels that are open to the Business users if they want to release a product and how they should contact the various areas within the Technology team. There is a section of this SharePoint site dedicated to detailing what is expected of the Business Owners and Users in the User Acceptance Testing (UAT) phase. Below is a sample taken from the site (areas in red, underlined, bolded and italicised by author for emphasis):

Who conducts UAT?

- UAT is performed by ***business users*** who are experienced in the systems being tested
- Users from all channels impacted by the change are involved
- ***It is really important that business users are involved, as they have the best knowledge of the flows in our systems and the nuances of dealing with upgrades, etc.***
- They are supported by the ODP Test Team who recommend the UAT tests to be executed
- The ODP test team agrees a suite of UAT tests with the Business Owner and then manage the preparation and execution of the tests
- ***Business users execute the tests and verify the results on front-end systems and record any defects that they find***
- The ODP test team manages the resolution of any defects found

The main benefits derived from executing UAT:

- It is an opportunity for business users to validate that their requirements have been delivered correctly
- It is the last chance to identify and resolve defects prior to launch, thereby minimising customer experience issues
- ***Actual business users are involved, so we verify that the solution is fit for purpose before it is deployed***
- It builds familiarity and business confidence amongst users of the changes being introduced
- IT enables informed business decision making in relation to proceeding with a deployment (Go/No Go Decision)

What is expected from the Business?

- The Business Owner engages with the ODP Test Team and Project Manager to agree the scope of UAT testing
- The Business Owner reviews and signs off the Test Approach, Test Plan and Test Matrix. These documents outline what tests are to be carried out and are prepared by the ODP Test Team
- ***Support the ODP Test Team to secure experienced users from all channels***
- ***Business users execute assigned test cases***
- The Business Owner provides input and attends daily updates during the UAT execution phase
- The Business Owner signs off the Test Closure report

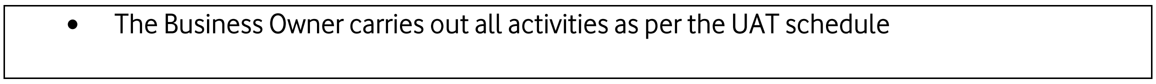

- The Business Owner carries out all activities as per the UAT schedule

Figure 5.2: Organisations UAT Procedures Sharepoint

It should be clear from this that there is an expectation set by the Technology team that the UAT phase should be executed by Business users. There are processes in place for how the UAT test phase should be being conducted. However, from the data gathered in the interviews it appears that this is procedure is not fully being followed. This will be examined in the next section in detail.

5.5 Interview Process

This section will detail the process for data gathering, as the initial step in the development of the Grounded Theory.

5.5.1 Selection Of Grounded Theory as Research Methodology

The first stage of any research process is to select a suitable research methodology. As noted there are a number of different approaches open to researchers when conducting research. In this case it was felt that a qualitative approach would be most suitable. Thus, it was necessary to select from one of the five methods outlined previously. Of these, Grounded Theory was selected as the best method for a number of reasons. Firstly, a number of other studies (as motioned in Coleman and O'Connor) have suggested that Grounded Theory can be used highly successfully in Information Technology projects. Based on this initial starting point Grounded Theory was considered a reasonable approach to take in this research project. Secondly, in this study it was felt that there was a lot of experience within the team and that by using a research method that would take their experience into account, the theory could effectively be generated by gathering data from the team, this is a key element of Grounded Theory (Glaser and Strauss 1967) and therefore it was clear that Grounded Theory would be suitable in this case. Finally, as noted previously Grounded Theory can be useful in situations where there are only a small number of existing theories (Adolph *et al.* 2011). From a review of the literature it was noted that there has not been much research carried out concerning theories of best practice for software testing within the telecommunications domain. Thus, again Grounded Theory would seem to fit as an appropriate research methodology.

5.5.2 Selection of Interview Candidates

This dissertation aims to investigate and develop a framework for the development of best practice in the Test phase of projects (not at an overall project level) it was thought that the best place to start would be the Test Team. These are a team of highly experienced professionals with a number of years experience and would have very strong opinions on how best testing should be carried out. Also, by interviewing the Test Team it was felt that this would be a good approach to start to develop a theory from the data as this team would be able generated a wide range of ideas. There were others considered as part of this such as members of the management team (Test management and also Project and Release Mangers as well as the delivery lead). However, it was felt that these individuals would be too far away from the "coalface" and may not have a clear idea of what was going on a daily basis. They may have an opinion on the ideal way things should be done but the Test Team have visibility of how it is working day-to-day so were considered as the ideal candidates. Just as a recap see below for outline of the roles and responsibilities of the Test Lead and Test Analyst:

Role	Responsibilities
Test Analyst	Create Test Scripts
	Execute Test Scripts
	Raise Defects and Retest Fixed Defects
Test Lead	Overall Test Phase Management
	Test Planning and Strategy
	Defect Triage and Assignment
	Status Reporting

Table 5.1: Test Lead and Analyst Roles and Responsibilities

5.5.3 Development of Questionnaire

Once the interview audience is chosen, the next step in the process is to formulate the questions to be asked in the interviews. For grounded theory, a semi-structured approach to the interviews makes most sense, as it allows for flexibility in the questioning and yet allows for easy comparison of themes that emerge from the different interviews. The questions developed were chosen to be as open-ended as far as possible; the aim of this was to kick-start discussion and that the interviews would be as free-flowing as possible. The goal was to use the questions as a guide for ensuring various key areas were covered, and that the interviewer was able to probe and question interviewees further to fully elicit meaning from what was being said.

Several drafts of the interview questions were developed and a set of those were piloted on one member of the Testing team who was then excluded for experiment. This piloting process resulted in the questions were further refined, clarified and reduced. For example, the initial introductory questions took up too much time in the interview with the interviewee explaining the various test phases, etc. This information is already known and clearly documented and it was felt that time in the interview needed to focus on the aspects of the test phases that they felt worked well and those that worked poorly. The pilot interview contained ten questions, whereas the experiment interviews reduced this number to seven. This number of questions proved to be adequate given the length of the interviews and the discussion generated. As noted, these questions were used to guide the interview. The questions asked are outlined below:

1. How is it decided what length the testing phases are there?
2. How should it be decided what types of testing should be undertaken?
3. What different types of projects are you asked to test?
4. What are the types of testing under taken, e.g. smoke, formal testing?
5. What would be the best types of testing to undertake on projects?
6. How would the type of project affect the type of testing that should be undertaken?
7. What is the role of the Business Owners and Users during UAT?

5.5.4 Conducting Interviews

Permission was given by the Test Manager and IT Delivery Lead to carry out the interviews, which took place in the main office of the organisation under consideration over a two week period. All interviews were recorded on a Dictaphone for later transcription and review. Each interview began by the interviewer giving some background information on the project and the goals of the research, after which the interview began. Firstly, there some background was collected on the interviewees. There were five interviews conducted, three with Test Leads and two with Test Analysts.

Number Of Interviews	5 Interviews
Average Duration of Interviews	30 minutes
Average Number of Years Test Experience	7 Years

Table 5.2: Number of Interviews, Length and Interviewee Experience

As shown above the team had an average of seven years testing experience, with the most being sixteen years and the least being two years, representing a very experienced team of Test professionals. The next section will outline the responses generated from each of the questions and the follow-up discussions that these questions generated.

5.5.4.1 Question 1: How is it decided what length the testing phases are there?

The initial question gave the interviewees the opportunity to discuss issues around project sizing. The responses indicated that most of the team understood that sizing took place at the start of the project and it was at the Feasibility stage that the sizing was completed. This line of questioning with all participants led to further discussion relating to how projects are currently being sized. There was a unanimous view that the current process isn't working as well as it could and wasn't the best way of sizing projects. Some indicative quotes are:

Test Lead: "we (Test representatives) should be involved from as early on as possible.....maybe then have less involvement in design phase..."

Test Analyst: "the scheduling and sizing done at the moment is poor"

Test Analyst: "sizing seems to be right for some of the smaller projects...but maybe looking at some of the larger more complex ones seems that there sizing on these was off"

Test Lead: "I think we (the Test Leads) have a bit of experience.....I think they should use us a bit more in the sizing process. If you've had a bit of experience on projects with different phases.....it's a good idea for us to get more involved in early sizing workshops. If you're used to be down in the nitty gritty detail from a test perspective you will have a good idea of phases that a project requires rather than someone looking at it from a release level"

5.5.4.2 Question 2: How should it be decided what types of testing should be undertaken?

This question was also related to the first. It was noted that currently when undertaking sizing there is a generic spreadsheet that is completed and based purely on the size of the project. Thus, large projects automatically get 6 weeks of E2E testing. It was noted that this is not the best way to decide the level of testing required on projects. The need to have Test representative involved in early phase project planning was highlighted by all interviewees. It was also noted that there was a need to consider the type of project that is being undertaken when looking at test phases. For example there would be no need for E2E testing on simple tariffing projects (that might be sized as large). Also, on small projects that require significant inter-vendor integration may require E2E test phases.

Test Lead: "there is no point in saying oh you need this phase, that phase just for the sake of it, time to market is important"

E2E was thought to be required on projects that require significant co-ordination with multiple vendors.

Test Lead: "where we are introducing new functionality or loads of different vendors, then there is value in end to end"

A significant new recommendation came from the experiences that the team had in recent projects, which related to carrying out testing in a production environment. It was found that during testing of projects that integrated fixed and mobile initiatives the test environments were not able to accurately mimic traffic and network scenarios found in the Production environment. Thus, the testing in the Test environment was found not to add significant value. A suggestion given by all interviewees was to that there should be concept of "In-life testing". This is testing in the Production environment but before the project is made available to customers. This was a consistent theme that emerged and questions progressed along this line to develop this theme.

Test Analyst: "we need to change the way we do things for sure, we had always built our remit on (testing in) a pre-production phase….that's all well and good on a billing type project….but the way the company is moving towards fixed projects…..with these projects you need to schedule testing post-deployment"

Test Lead:" things that involve fixed elements, highly complex projects or projects that introduce new functionality….those types of projects where we would have said that we would have an E2E test phase, this should be done in Production. Still do SIT as always…but rather than do another E2E test phase in a test environment……cause there are differences between our test environments and production are significant……in order to make sure we can test that everything works properly….we should looking to deploy into Production as soon as possible"

Interviewer: So do you think early on in the planning phase we could look at the type of project it is, the complexity, whether it's a fixed services or another type of project and then based on that we should come out with recommendations that say this type of project should have an "In-Life testing phase"?
Test Lead: "Yes, it is easy to get to. I think you could easily put together a matrix at the start of the project that states all those things and then comes out with a recommendation as to the phases of testing that is undertaken"

Test Analyst: "It would be good to have like a shopping cart, or a matrix that looks at things like if it's a new project, new functionality and that kind of thing…..then that would kind of indicate of regression or end to end was required…or if you can cut out phases"

5.5.4.3 Question 3: What are the different types of projects are you asked to test?

This question was designed to elicit the different information on the types of projects that the team were, and will be, asked to test. It was found in many of the interviews that it was not necessary to ask this question as the interviewees naturally mentioned the types of projects while answering questions 1 and 2. It was found that there was an increased focus in projects that have elements of mobile and fixed technologies and this was felt to be a challenge by the team as they were relatively inexperienced with the fixed propositions.
Test Analyst: "we tend to do more fixed services projects now"

Test Lead: "we have run into difficulty with these (fixed services projects) as the environment we use does not mirror production"

5.5.4.4 Question 4: What are the types of testing under taken, e.g. smoke testing etc, formal testing?

This question was designed to get a view of the types of testing that is undertaken. Do the team rigidly follow test scripts or are there less formal methods undertaken as well. Do the team rigidly follow test scripts or are there less formal methods undertaken. Again there was a unanimous answer here. All participants believe that formal scripted testing is absolutely necessary for reporting purposes but all reported that they use some informal methods such as exploratory testing for the first day or so of the test events.
Test Analyst: "I do a lot of exploratory testing. First day of my project (test phase) I won't go through any main flows I'll check click all the links and stuff just to try and find errors….the smallest things you can get"

Test Analyst: "The first day of my testing usually consists of exploring around the system, exploring around the new functionality and trying to iron out some of the silly niggly defects!"

Test Lead: "you would have to say that 70% (of testing) is formalised, but there is an element of ad-hoc…you know your system go try and break it"

Test Analyst: "I think smoke testing is good, I would be of the opinion that we factor in two days of this. But I think we do need formal testing too, just from a reporting point of view and so you have a good measure of progress and see if you are on track to meet exit criteria"

Test Lead: "smoke testing and exploratory testing are a great way forward but you do need to have some structure behind how the formal, accepted test cases are run. But if you get in there on the first day and run through a couple of the flows, not formally, not structured, just get in there and use your experience of the systems to see how they all are sitting together, that will flush out a lot of major issues on day one."

5.5.4.5 Question 5: What would be the best types of testing to undertake on projects?

As noted it was mentioned on a number of occasions that where there are multiple vendors, fixed services elements or a lot of new vendors on a project then this would benefit from an E2E phase. Also, it was mentioned that certain projects would benefit from an "In Life" testing phase. There seemed to be a consensus that the phases at the moment from CIT to SIT were appropriate:

Test Lead: "If there is a project that is introducing significant change, or with vendors.....if there are new vendors in a complex project....then this could benefit from an E2E phase"

Test Lead: "fixed services, high complexity, new systems being introduced...these types of things should be being tested in production"

Test Analyst: "I think complexity plays a big part in it, also the environment available, whether it makes sense to test in Production"

5.5.4.6 Question 6: Would the type of project effect the types of testing that should be undertaken? How?

It was generally felt that technology being introduced would significantly affect the types of testing that were undertaken. It was felt that the phases before E2E (CIT/SIT) were necessary on all projects and UAT was also thought to be necessary on all projects. However not all projects would require the E2E phase, for example if there were projects that contained mixed elements of Fixed and Mobile technologies then this type of project would require E2E testing However, projects that were more straightforward tariffing changes would not require E2E testing.

Test Lead: "conditions when we should have E2E would be larger, multi vendor project with elements of fixed stuff"

5.5.4.7 Question 7: What is the role of the Business Owners and Users during UAT?

This question generated perhaps the biggest discussion of the interview. There were unanimous feelings that they way the Business were engaging during UAT was not optimal and there were a number of recommendations as to how best use the UAT phase to fully incorporate Business Users into the testing process. The goal of the UAT phase is that it should be owned and run by the Business users; but this generally does not actually turn out to be the case. In the initial concept of the UAT phase it was noted that during this phase the Business Users (i.e. the end users of the new systems, the customer service representatives) would be involved in running tests during this phase. However, it was noted by many interviewees that this does not happen in practice. There were a number of reasons put forward for this but the main outcome of the discussions and the unanimous opinion of all those interviewed was that it would be extremely beneficial to have these Business users assist with Test Execution during the UAT phase. Currently it is seen that the business are involved in the signing off of the Test Case Matrix. It was felt that while this was being undertaken in a lot of cases this was the only involvement the Business had in the project. All interviewees felt that it was worthwhile getting Business end users (such as Call Centre Representatives (CSRs)) to be involved in the execution of test cases. While it was noted that there was some time taken up in training the CSR's in the test methods (e.g. how to run test scripts, how to log defects, etc) it was felt that this time was well spent as when they got testing they would be able to spot issues that Testers might not have seen.

Test Lead: "The way it used to be was the business were more involved it was their job to come with the tests to be done, it was the whole idea of the business saying "Yeah we are happy with this product".....because we (as Technology) for the most part are delivering projects for the business"

Test Lead: "Getting the business more involved I think outweighs any hand holding that may need to be done"

Test Analyst: "I think there is always going to be value into getting business users involved....it would be good to get them more involved earlier"

Test Lead: "Ultimately we need them (the business) to specify what they want tested"

Test Lead: "It is becoming harder and harder to get resources to run test cases"

Test Lead: "There is definite value in the UAT phase....but it may be worth re-evaluating how the phase is run....the go-to people are CSRs...but it doesn't necessarily need to be CSRs it could be someone from the business"

Test Analyst: "the few projects I have been working on that have had Production issues....these would have been caught by an end-user if tested...there are things that the end user will notice that we (as Test Analysts) wouldn't"

5.6 Conclusions

This dissertation is developing a best practice framework for software testing in a telecommunications organisation. Once the background was established, it was necessary to outline how the data that was to be used for the generation of the Grounded Theory. This was done via interviews and the questions asked were outlined in the chapter. Then each question was examined in detail and the answer given by all participants were investigated. This allowed for the further investigation of areas outlined and the development of Grounded Theory.

The next chapter will examine in detail the how the Grounded Theory was developed. It will detail how axial codes were developed, how codes were selected and how they were refined to allow for core categories to be developed and detail how the framework emerged from this.

6. GROUNDED THEORY IN ACTION

6.1 Introduction

The goal of this dissertation is to use Grounded Theory to develop a framework for testing best practices in a Telecommunications organisation. This chapter will detail how the Grounded Theory was created from the data gathered. Firstly, the version of Grounded Theory to be used will be discussed, with reasons given for choosing that specific version. Next, there will be detailed discussion on the creation of Grounded Theory and Framework that emerged. Finally, the Grounded Theory will be detailed.

6.2 Version of Grounded Theory to be Used

As noted earlier since the initial version of Grounded Theory was proposed by Glaser and Strauss (1967) there have been other competing versions of the theory proposed. Most notably there was another competing theory introduced by one of the initial authors, Strauss, now working with Corbin (Strauss and Corbin (1998). The version proposed by Strauss and Corbin (1998) was more prescriptive in terms of how the theory should be used. This was criticised by Glaser (1992) as he believed that this was not conducive to theory emerging from the data. However, a criticism that Strauss and Corbin (1998) had of the original theory was that it was too vague and hard for novice researchers to implement. Also, Glaser and Strauss (1967) original theory differed from Strauss and Corbin (1998) in terms of their approach to the research topic. Glaser thought that the researchers should approach the research with as little background as possible, but Strauss and Corbin (1998) were a little more pragmatic and accepted that the researchers were unlikely to come into research without prior knowledge of the subject area.

In the current study it is necessary to choose one version to follow over the other. It was felt that in this case the version proposed by Strauss and Corbin (1998) was a better version to follow; there were a number of reasons why it was selected. Firstly, as noted above, Strauss and Corbin's theory was developed to be more prescriptive for the novice researcher. As the researcher in this study was not an experienced social science researcher it was felt that this more prescriptive approach as espoused by

Strauss and Corbin would be the better approach. Also, Strauss and Corbin are more realistic in their approach to the researcher's previous experience. As the researcher is an experienced member of the Test Team, with a lot of knowledge in the Testing area, the Strauss and Corbin approach can take this into account. Once again, given this it was felt that Strauss and Corbin's approach to the development of Grounded Theory was best for this study. Therefore the three stages of process that will be undertaken are:

- Open Coding
- Axial Coding
- Selective Coding

6.3 Conducting the Study

The first stage of the study was the gathering of the data which was covered in detail in the previous chapter. All interviews were recorded on a handheld Dictaphone for later analysis. Once the data was gathered then it needed to be transcribed and analysed. This would involve reviewing the interviews in detail and selecting words and phrases that are telling and could contribute to the development of the theory. It was found from a review of the literature that the process of the gathering and coding can be very time-consuming and hard to manage. In these studies the interviews were transcribed by hand, printed out and were coded by hand. This process involved cutting out phrases or words from the interviews and placing them in different coded piles. It was found by these researchers that this process was very time consuming and became nearly unmanageable as the number of codes increased and they looked for an alternative method (Coleman and O'Connor, 2007). Thus, it was felt that for this study it would be best to use a software tool to assist with the coding and recording of interviews. There are a number of software tools that are available to assist with Qualitative Research, an analysis of these tools was undertaken and it was found that the Atlas.ti (Atlas.ti n.d.) software was the easiest to use and most suitable for our study.

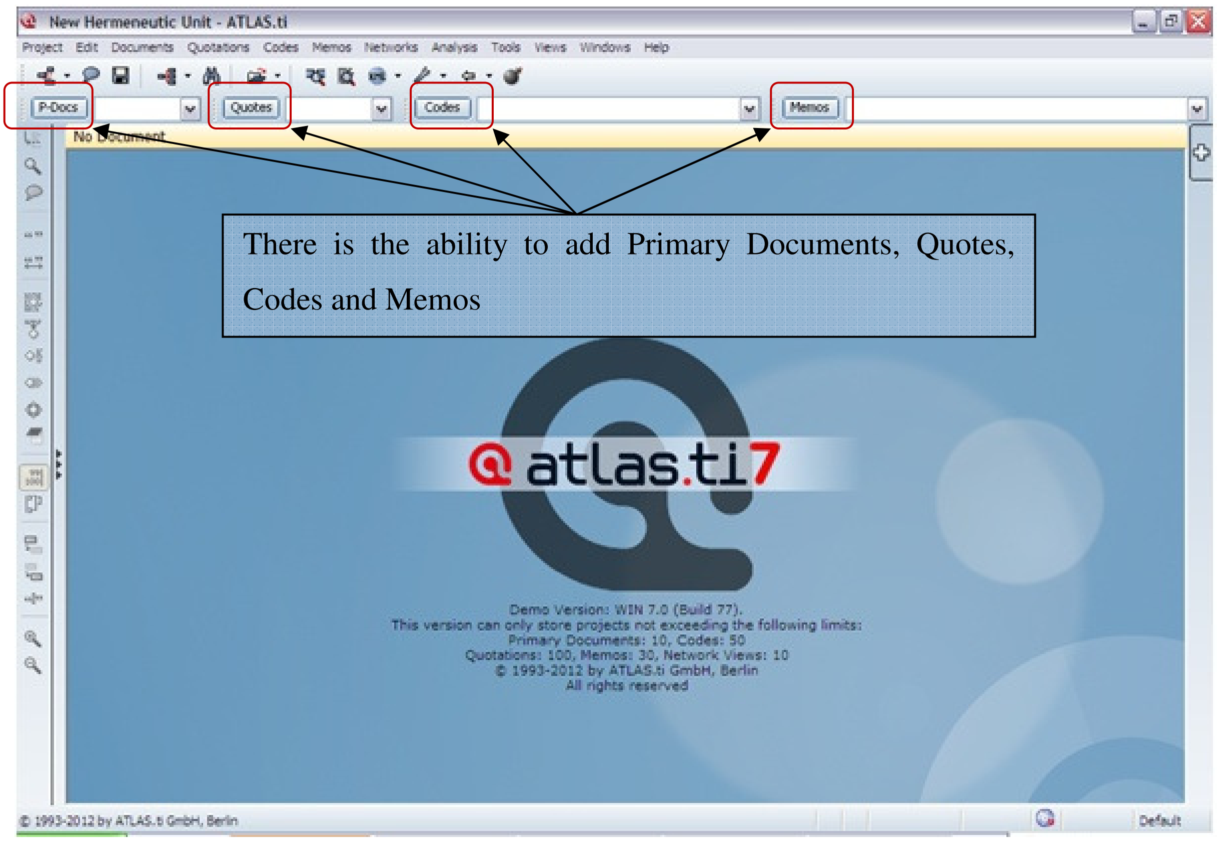
6.3.1 *Atlas ti*

Figure 6.1: Atlas.ti Splash screen

Atlas.ti is an integrated suite of tools that can be used to support analysis of written texts, audio, video, and graphic data used in qualitative research. Atlas.ti provides researchers with tools to manage, extract, compare, explore, and reassemble meaningful segments of large amounts of data in a flexible manner (Atlas.ti n. d.). The first version of the software was delivered in 1993 under the company name Scientific Software Development and it is now used the world over by many blue-chip companies such as Google, Yahoo and other non-governmental organisations such as the World Bank and the United Nations (Atlas.ti n. d.). Given these references it was felt that the software had the necessary rigour and credentials to be used on this research project. Atlas allows researchers to upload text, data and voice recordings and provides an ability to code and categorise this data.

6.3.1.1 New Project Creation and Coding Procedure

Once a new project (called Hermeneutic Units) is created the researcher is presented with a blank screen that they can now add Primary documents to the project. These can be text documents, pictures, videos or audio files. There is also the ability to quote sections and add codes and memos.


New Hermeneutic Unit - ATLAS.ti
Project Edit Documents Quotations Codes Memos Networks Analysis Tools Views Windows Help
P-Docs
Quotes
Codes
Memos
No Document
atlas.ti7
Default

There is the ability to add Primary Documents, Quotes, Codes and Memos

Demo Version: WIN 7.0 (Build 77).
This version can only store projects not exceeding the following limits:
Primary Documents: 10, Codes: 50
Quotations: 100, Memos: 30, Network Views: 10
© 1993-2012 by ATLAS.ti GmbH, Berlin
All rights reserved

© 1993-2012 by ATLAS.ti GmbH, Berlin

Figure 6.2: Home Screen in Atlas.ti

Once a new project has been created then it is necessary to add Primary documentation to the project. Once the documents have been uploaded then the coding can begin.

Below is the screen that the researchers started with. Firstly all the interviews were conducted and uploaded as primary documents (the interviews were named in the format FirstName_LastName_DateofInterview). The First Name and Last name have been removed from the screen shot below to preserve anonymity of participants. They are listed in the section on the left hand side that is called Primary Documents. The first step was to listen back to each interview. In Atlas there is the ability to playback, stop, fast forward and rewind conversations.

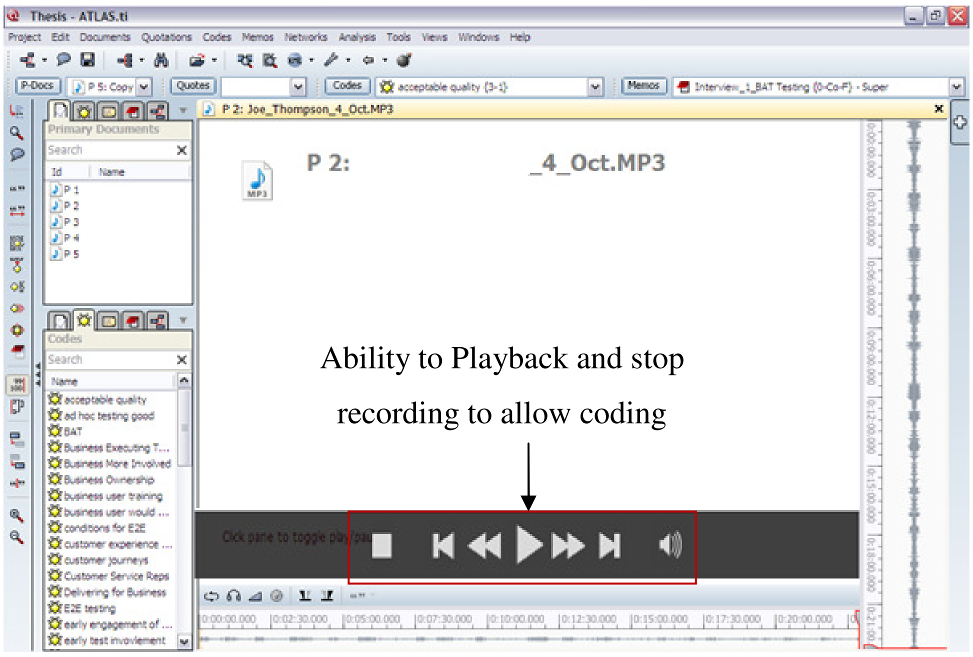


Figure 6.3: Initial screen

Once the interviews were loaded into Primary documents they are ready to be coded. This is done by using the speech wave bar on the right hand side of the GUI. The researcher can quickly and easily select areas of speech and insert codes against these. This is detailed in the screenshot below.

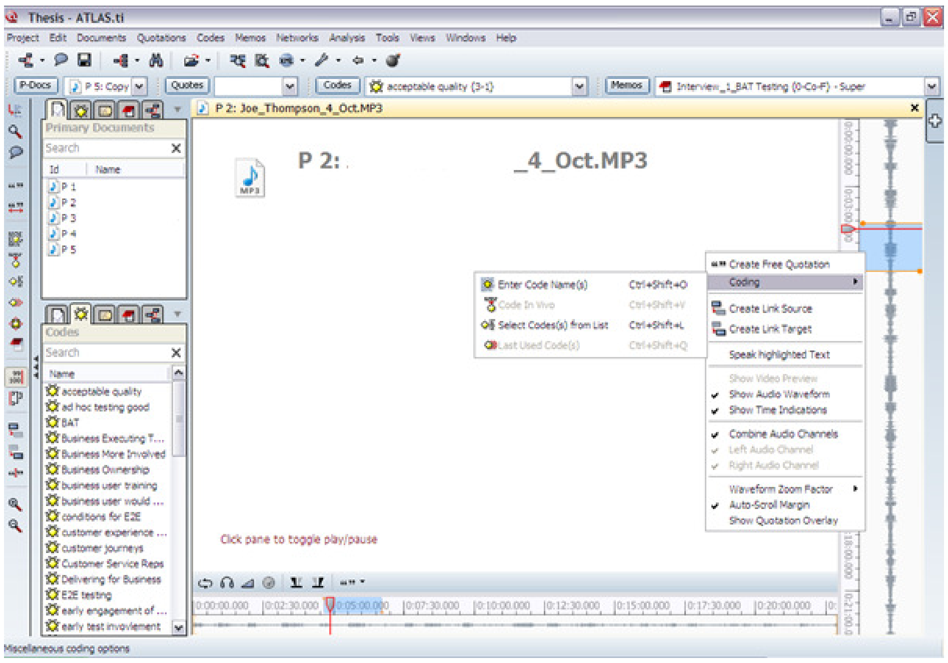


Figure 6.4: Procedure for Coding

6.3.2 Coding Interviews and Emergent Categories

Once the initial project was set up, the interviews conducted and loaded into Atlas.ti the coding could commence. This section will detail the results of the coding and the emergent categories.

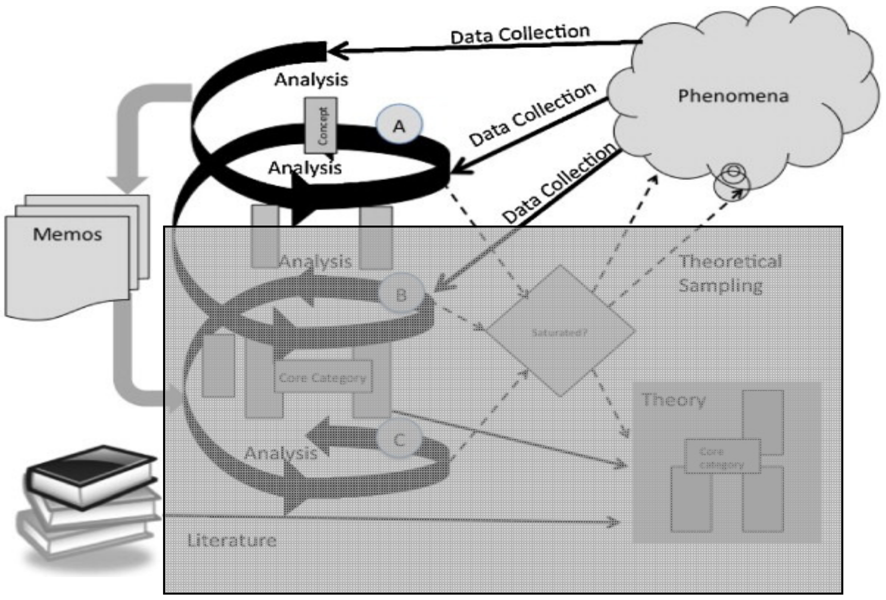


Figure 6.5: Open Coding Maps to Data Collection and Initial Analysis (Adolph *et al* 2012)

6.3.2.1 Open Coding

The first stage in the Strauss and Corbin (1998) model is that of Open Coding. This involves reviewing the data that has been gathered (usually in the form of interviews) line by line, word-by-word, to gain an insight into what participants are saying (the Concurrent Data Generation and Collection step, the crucial point that differentiates Grounded Theory from other research methods (Birks and Mills 2011)). The purpose of this phase is to generate as many codes as possible (which are later refined and linked). During the interviews with participants it was easy to identify codes as they spoke. While listening back to interviews it was possible to pause playback and select portions of the interview to be included as part of a code (see Figure 5.3 above). The length of the interview time that was associated with codes was determined by how relevant what the interviewees were saying. Some were only a couple of seconds while others were close to a minute. As the interviews were semi-structured it was found that interviewees mentioned many of the same areas that were able to be included as codes. The initial set of codes numbered over 100 and a sample of the types of codes that were gathered are outlined below:

Acceptable Quality	Business More Involved	Customer Journeys
Ad-hoc Testing	Business Ownership	E2E Testing
User Acceptance Testing	Business End User Training	Early Test Involvement

Figure 6.6: Sample Codes Generated in Open Coding Phase

While undertaking this phase the researcher was engaged in Constant Comparative Analysis. This involved constant comparison of interviews with each other to understand what was being said and searching for any emergent patterns in what was being said. This involved comparing codes with other codes and seeing similarities as well as comparing the emergent categories. Memos were collected consistently throughout the process of the Initial Coding; usually these were short one-line details that link what was being said by one interviewee with what was being said by others. An example of a memo is below:

There is an emphasis on the use of Business resources in the UAT phase

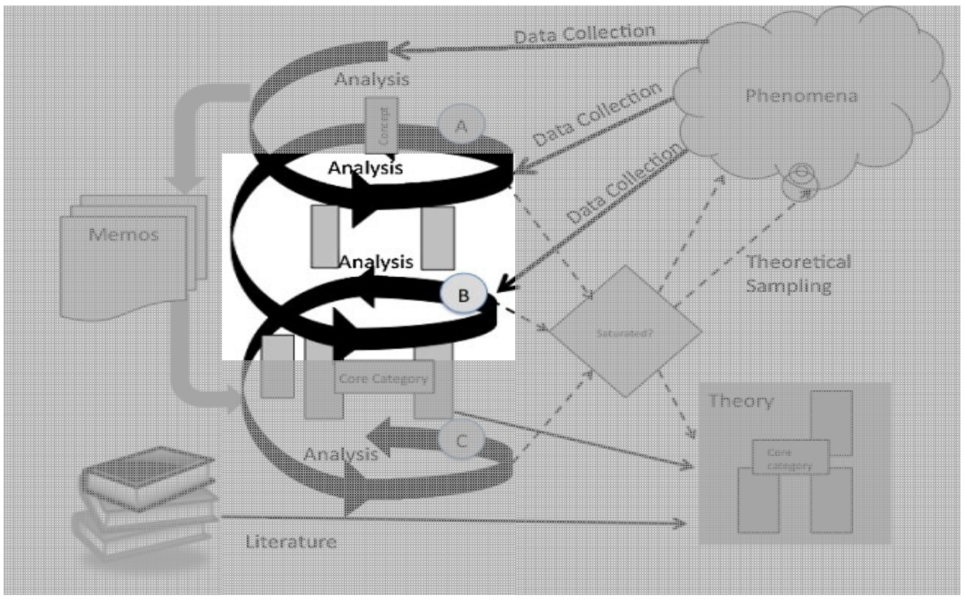


Figure 6.7: Axial Coding Maps to the Further Analysis (Adolph *et al* 2012)

6.3.2.2 Axial Coding

The next stage in the Strauss and Corbin (1998) version of Grounded Theory was that of Axial Coding. While the goal of Initial Coding is to generate codes the goal of Axial Coding is to reduce the number of codes and cluster them based on their themes. This involved looking in detail at all codes that were created. Where possible there was follow-up with interviewees via email and face-to-face communication to clarify issues. This allowed for the reduction of codes from over 100 to just 23. It was also possible to cluster codes and develop categories or themes that began to emerge. It was found that a number of categories emerged that the existing codes could be placed into these categories. There were 2 categories and areas of concern. The first related to Project Sizing. As noted in the previous chapter there was significant concern that Project Sizing that was undertaken at the minute on projects was not optimal and was having negative effect on quality. Also, the fact the business users should be more involved in the development of test cases was a category that emerged. Axial Coding Maps to the Further Analysis Stage in Adolph *et al* (2011) diagram of Grounded Theory as seen in Figure 6.6 above.

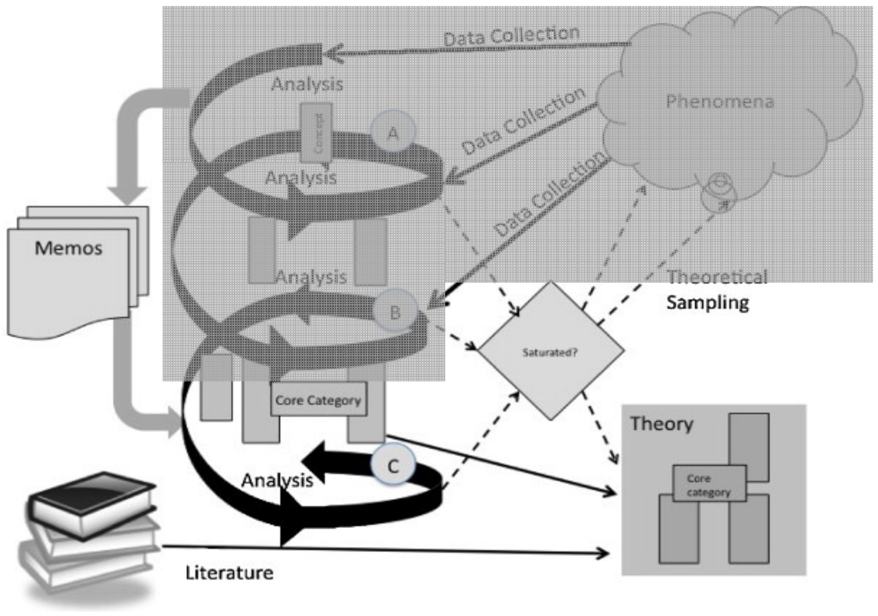


Figure 6.8: Selective Coding and Emergent Categories Mapped (Adolph *et al* 2012)

6.3.2.3 Selective Coding and Emergent Categories

The next stage in the processes is that of Selective Coding. In this stage the core categories that emerged in Axial Coding are grouped and clustered under themes and the framework is created. The key to this phase is identifying key themes and categories that act as anchor or lynch pins of the results. Selective coding can be used to explain any relationships among categories found that can assist in the development of a theoretical picture of what is happening. (Coleman and O'Connor 2007). . Selective Coding and Emergent Categories maps to the Analysis and Saturation of Categories in Adolph *et al* (2011) diagram of Grounded Theory as seen in Figure 6.7 above. In this stage there is seen to be category saturation and any addition of codes does not assist with the development of more explanatory categories. In this phase the emphasis is on tying to gather all the categories and codes under themes. As noted during Axial Coding two areas emerged that were seen to be key themes (that of Project Sizing and User Acceptance Testing). The categories that were developed were seen to link to one of these and each category can be linked back to quotations in the interviews.

Theme	**Category**
Project Sizing	Size and Complexity
	E2E Testing
	In Life Testing
	Fixed Services
	Testing Matrix
Theme	**Category**
UAT	Business More Involved
	Business Ownership
	Business Executing Test Cases

Table 6.1: Themes and Core Categories

The next step was to integrate the core categories and combine them to develop a framework.

For example Network Elements, Test Environment Constraints, Production Test environments were combined to *In Life Testing*. Also, Test Planning, Early Test Involvement and Project Sizing were combined to *Size and Complexity* and Business

User Training, Test Case Execution and Quality Implications were combined to *Business Executing Test Cases*.

The Atlas.ti software provides functionality that allows researchers to graphically display their findings. The theoretical framework that emerged from this research is outlined below:

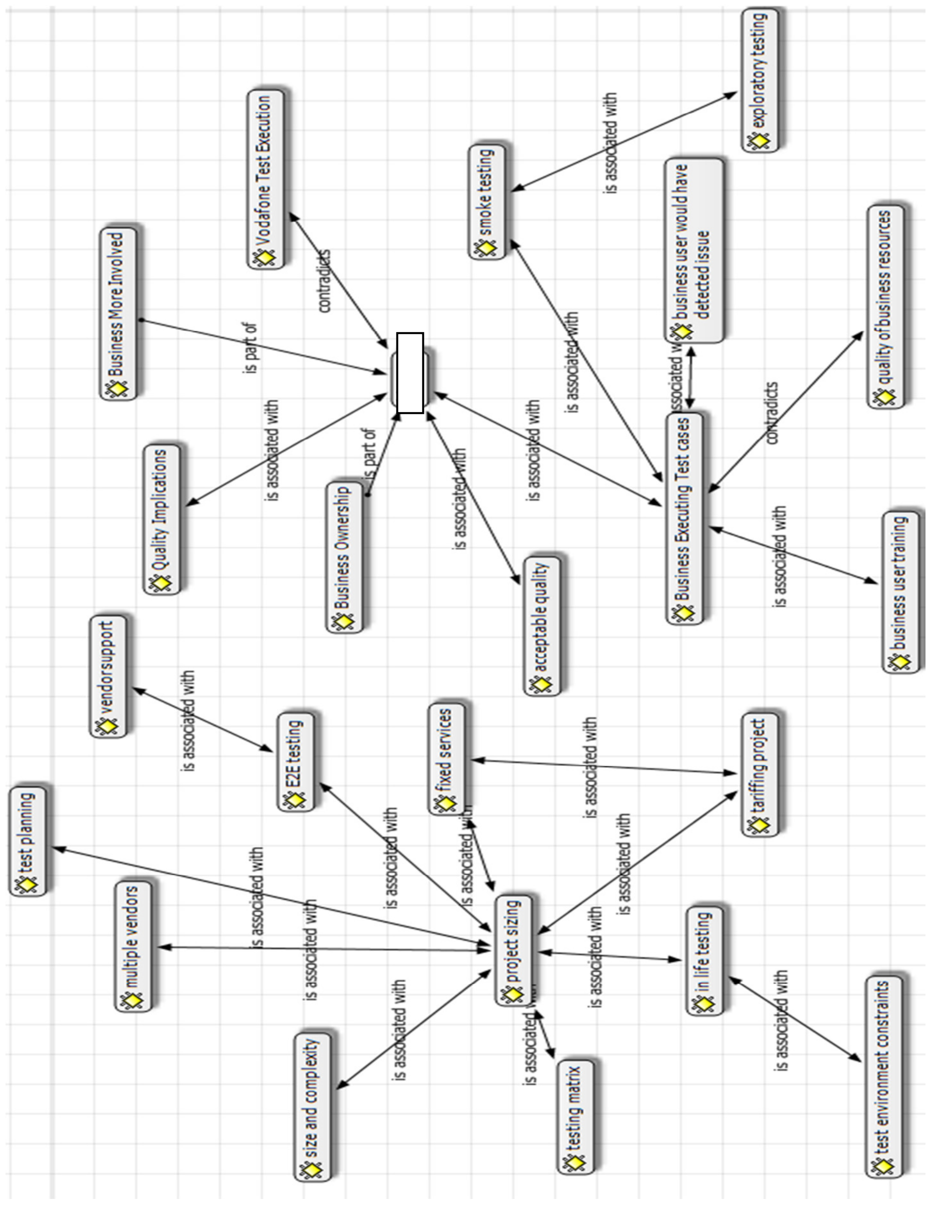


Figure 6.9: Theoretical Framework

There are two core nodes here that relate to the two main themes: Project Sizing and User Acceptance Testing (or UAT) these themes link a number of the core categories together.

The *Project Sizing* is guided by a number of elements that are inter-related. Firstly the *Size and Complexity* of the project need to be considered. Whether the project is a *Tariffing Project* or *Fixed Services* will affect the sizing and if there are *Multiple Vendors* will guide if there should be an *E2E Testing phase* on the project. The UAT theme ties together a number of the related categories relating to Business involvement in the UAT test phase. There is a need to have the *Business More Involved. Exploratory testing* and *Smoke testing* are areas that need to be completed during the UAT phase and relate to *Business Executing Test Cases.*

6.4 Summary of the Process

To recap on the key points in the use of the Strauss and Corbin (1998) model in this experiment to develop a Grounded Theory, the following steps were undertaken:

	Open Coding	**Axial Coding**	**Selective Coding**
Codes	100+ Codes generated	23 Codes	23 Codes
Activities	Constant Comparative Analysis	Developing themes • Project Sizing • UAT	Relationships between codes and categories
Annotations	Memos	Follow-up communications	Identify category saturation

6.5 Findings and Recommendations

The purpose of this project was to develop a framework for testing best practice within a telecommunications industry. The next section will detail the key recommendations that emerged from the Grounded Theory that if implemented would constitute Best Practice from the data gathered. It will also then look at applying this framework to recently completed projects that introduced defects into Production. It would then hope to show that had the best practice recommendations been followed.

6.5.1 Project Sizing

The first finding related to the key theme of Project Sizing. It is recommended that there should be better upfront project sizing with more Test team involvement upfront.

The key recommendations that came from the Grounded Theory are outlined in the following sections:

6.5.1.1 More Up Front Test Involvement in Planning and Sizing

Currently the Test team only become involved in projects after the requirements are signed off. This is considered to be too late in the lifecycle. There is no involvement of Test team at the Sizing and Planning phase. The Test team should be a key stakeholder in this phase as they can recommend different test phases that might be required on a project based on their experience and the type of project. Also, there may be no need for some Test phases on certain projects (for example there would be no need for an E2E Testing phase on Tariffing projects).

6.5.1.2 Need to consider In-Life Testing as Phase

Currently the model of test phases as outlined in Chapter 4 appears to be no longer suitable for the types of projects that they team are engaged in. There is a recommendation to consider, at the planning stage, what the project involves. Based on this review it should be clear the areas (at a high level) that need to be tested. If there are certain elements that can't be tested in the Test environment (due to technical and environmental constraints) then these will need to be completed in the Production environment. This would result in the removal of a Test phase in these types of projects and the earlier release of the project to the Production environment. Then there would be a phase of testing in the Production environment that would result in the discovery of issues that would not have been seen in the Test environment due to the differences in their configuration. It is not always possible to set up a Test environment to accurately mimic a Production environment which was seen in recent projects where issues were only discovered in Production when traffic was being handled by other operators' switches and routers. By having this Production Test phase, it would allow Test Team to uncover problems that would not have been seen in Test, but would have been experienced by customers in Production. Thus, these issues could be eliminated before the project was commercially released, thereby preventing a bad customer experience.

6.5.1.3 Creation of a Test Phase Matrix

There was also a recommendation that there should be some sort of quantitative method to be used by Test Team to inform the phases of testing and length of these phases on projects. This should be completed at the Planning phase. It would involve a spreadsheet that would allow the Test Lead to input data about a project (e.g. number of vendors, type of project, is there Fixed Services element, etc.). This could then be completed and based on inputs the different phases that are required and their length would be recommended.

6.5.2 UAT Phases

The next major finding was in relation to the UAT phases. It was felt that there was a sense that there was a need for a greater involvement by the Business Owners and Users in the UAT phase. There were some specific recommendations here. All of these recommendations would require support from and be driven by IT Delivery Lead and/or the CIO.

6.5.2.1 Greater Business Involvement in Test Script Creation

It was recommended that the business users or business owners should be more involved in the creation of Test Scripts. The team that write the Test Scripts are the Test Analysts and will write scripts based on Requirements documents. However, it was felt that there should be greater engagement from the Business users to help guide the Analysts. This would mean getting a portion of Business Owners time which can be challenging but it was felt that this was worthwhile.

6.5.2.2 Business End Users Executing Test Cases

This was perhaps the finding that gained the most responses. There was a clear consensus among the team that the practice of getting business end-users to assist in the execution of test cases was a very beneficial exercise. It was felt that there was more resistance from the Business and Customer Operations to release end-users (usually CSR's) due to impacts on the call centre staffing levels. This was understood by interviewees but was felt that the benefit of having users who use the systems everyday engaged in a final acceptance test was vital. It was noted by many of those interviewed that having CSR's executing test cases would have resulted in them catching defects that were released into Production on recent projects.

6.5.2.3 Use of Exploratory Testing

Another recommendation was that there should be time taken at the start of the UAT phase where there should be no formal testing taking place. This would allow the experienced Test Analyst to complete some Exploratory Testing without running through test scripts. It was felt that this was a worthwhile exercise as in the past when the UAT environment was released there were a number of environmental issues seen. These often weren't uncovered until a few days into testing as the Test Analyst was following a detailed script and didn't hit the page with the error until the end of the script. This meant that testing was delayed. It was suggested that if the Test Analyst was free to test all the areas that they know in the past caused issues this would highlight these at the start of the testing. This would then allow for more uninterrupted test script running in the later stages of the phase. This would mean that there would be no test report completed for the first day or two and the Test Lead would need to plan run rates for test scripts in a shortened time period.

6.6 Grounded Theory

The previous sections outlined the results these can be summed up, based on the Framework Diagram, recommendations and data gather through interviews as the Grounded Theory for this study which states:

Project Quality can be increased by ensuring that the key stakeholders are permitted to participate in each of the stages that they can meaningfully contribute to.

Two specific instances of this are:

- *Project Quality and Sizing can be improved by involving the Test Team in Wider Sizing*
    - *More Up Front Test Involvement in Planning and Sizing*
    - *Need to consider In-Life Testing as Phase*
    - *Creation of a Test Phase Matrix*

- *Project Quality can be increased by have End User involvement in Test Case Execution*
    - *Greater Business Involvement in Test Script Creation*
    - *Business End Users Executing Test Cases*
    - *Use of Exploratory Testing*

Such a theory may seem obvious and almost tautological in retrospect, but nonetheless if an organisation was to adopt this theory as its guiding principle on projects, and to redevelop all processes and documentation (in practice and on paper) from this principle, the quality of the projects should increase significantly as a result.

6.7 Conclusions

The aim of this research is to develop a Framework for Testing Best Practice in a Telecommunications company. This chapter detailed how the Framework was created using Grounded Theory. The data that was collected was firstly analysed. The tool used to complete this analysis was outlined. How it was used by the researchers to code the data was also highlighted. Following this there was a detailed analysis of how the Grounded Theory was created. This involved detailing what was done at each step of the coding process from Initial Coding, through Axial and then Selective. The Framework that emerged was discussed in detail.

The next chapter will discuss how the Framework and Theory created was validated and tested. It will do this by applying the recommendations to recently completed projects

7. EVALUATION OF SUCCESS FACTORS

7.1 Introduction

The aim of this dissertation was to create a Grounded Theory for best practice in software testing. The previous chapters have detailed how the Grounded Theory was generated through the gathering and analysis of data. However, in order to assess the validity of the theory that has been developed it is necessary to be put it to the test. The aim of this chapter is to evaluate how valid the theory generated is when applied to real world projects. This chapter will firstly consist of a description of the projects (due to commercial and confidentiality reasons elements of the projects not directly relevant to this thesis will be omitted). The recommendations and Framework will then be applied to these projects. The projects were chosen as they have been recently released and it was also found that they introduced defects into Production. The aim of this comparison will be to assess whether by applying the recommendations and Framework these issues could have been avoided. Finally, the results of the findings will be compared with the literature to see if they are in good agreement with other relevant research findings.

7.2 Application of Framework to Completed Projects

The Framework that was developed from the use of Grounded Theory made a number of recommendations. In order to test the validity of these recommendations it is proposed to examine a number of recently completed projects (that would not fully adhered to the recommendations) and retrospectively see if the application of the Framework recommendations would have had an effect on the quality of these projects.

7.2.1 Project 1: Tariffing Project

The first project to be examined was a Tariffing project. It was to be delivered mainly by a single vendor. In terms of complexity this was a relatively straightforward project offering new tariffs to the Consumer base. This was an extension to an existing system completed last year. However, as there were a number of tariffs to be changed it was sized as a Large Project in the initial planning meeting and was given a four week E2E testing phase. The project itself had spanned three months and followed the delivery process as outlined in Chapter 4.

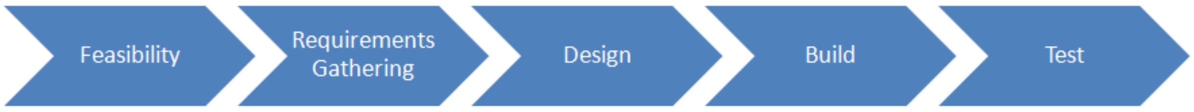


Figure 7.1: Project Phases

There was an initial feasibility study followed by requirements gathering session completed by the organisations Business Analysts. The requirements were then handed over to the vendor to complete the Design, Build and Test elements. As noted the project was completed by a single vendor. Thus, there was very little Test Team involvement in the CIT and SIT phases. The vendor completed all the development and test work offshore and then handed the project over to the Test Team for the organisations test phases. Then the Test Team completed four weeks of E2E and two weeks of UAT. It was found that this was far too much testing on the simple project. As mentioned previously the reason that the E2E stage was brought in was for projects that contained multiple vendors, introduced new products or had elements of Fixed and Mobile combined deliveries. This project did not meet any of these criteria for having an E2E phase and there should not have been any E2E test phase on this project. The Test Team was not involved in the initial project sizing and the rigid model used by the Demand team automatically assigned six weeks of E2E to the project. However, by the time the Test Team were engaged it was too late to change phases as vendors had been paid and the Marketing and Business teams wouldn't have been ready to accept a product any earlier. Thus, the team completed 6 weeks of testing on a project that should only have had 2 weeks of UAT testing.

7.2.1.1 Project Outline

As noted this was a fairly straightforward project in terms of complexity (as it was an extension of an earlier project). However, there was a lot of change and was sized as a

Large project with over 20 people engaged on it over its life time. An outline of the project is provided below:

Project Type	Tariffing	
No. of Vendors	1	
Project Team	**Vendor Team**	**Organisation**
	Project Manger	Project Manager
	Tech Lead	Business Analyst
	Developers X 4	Business Owner
	Test Lead	Solution Architect
	Testers X 4	Tech Lead
		Test Lead
		Test Analyst

Table 7.1: Project 1 Outline

7.2.1.2 Recommendations

Had the recommendations above of More Up Front Test Involvement in Planning and Sizing and Creation of a Test Phase Matrix been followed there could have been significant savings on this project. Had the Test Team been involved earlier they would have been able, at the initial sizing stage to recommend that as this was a simple tariffing project there would be no need for the E2E testing phase. This would have resulted in four weeks saving on timelines which would mean being quicker time to market, and in the current competitive environment of the mobile telecoms industry this could have large implications for revenue for the company. Also, based on recommendations there would have been a reduction in the project size by four weeks. This would have had significant cost saving as the project needed to pay for vendor support in the four weeks of E2E as well as increased cost of the Test Team resources. The table below contains a description of the test phases on the project, what was completed in each and whether or not the phase was required:

	CIT	SIT	End-to-end	UAT
Description	Vendor internal unit testing	Vendor integration testing	Test Team tariff testing completed by Test Analyst	User testing completed by Test Analyst

Test Team Involvement:	No	No	Yes	Yes
Test Phase Required:	Yes	Yes	No, should have be removed at planning stage	Yes

Table 7.2: Test Phases Comparison on Project 1

To complete this review of project 1 the findings of this analysis were presented to a member of the test team who agreed that a *More Up Front Test Involvement in Planning and Sizing* and *Creation of a Test Phase Matrix* would have helped in this project. They said that "*Four weeks of E2E was not needed on a project of that nature and More Up Front planning from Test Team, would eliminate this scheduling blip in future*".

7.2.2 Project 2: New Handset Project

The second project to be examined related to the release of a new tariff option as well as the launch of a new handset related to this tariff. This project was to be delivered by two vendors and in terms of complexity as this were only one tariff and one handset it was sized as a *Small Project*. As it was felt that that this was a small, non-complex project it was decided, at the initial Planning and Sizing, that there was no need for the Test Team to be involved. Although there were two vendors involved they had worked together previously on projects and had a good working relationship. It would only require Vendors to deliver and test their own project and there was to be no E2E or UAT on this project. The project also completed along the lines of delivery projects as noted in Chapter 4 and above. There was an initial feasibility and requirements gathering completed. This was followed by Design, Build and Test by vendors following which the project was released into Production. However, once the project was released to Production a number of serious incidents were discovered which resulted in emergency fixes needing to be deployed in the soon after launch. These were serious issues that caused loss of revenue to the company but if they had been left undetected for a number of weeks would have caused serious revenue loss.

7.2.2.1 Project Outline

As noted this was a fairly straightforward project in terms of complexity (releasing a new tariff and handset is standard practice) and the project was sized as Small. However, as noted below there was significant pressure to launch the tariff in an accelerated manner, this meant more project resources required than usual on a Small project. An outline of the project is provided below:

Project Type	Tariffing	
No. of Vendors	2	
Project Team	**Vendor Teams**	**Organisation**
	Project Manger X 2	Project Manager
	Tech Lead X 2	Business Analyst
	Developers X 3	Business Owner
	Test Lead X 2	Tech Lead
	Testers X 4	

Table 7.3: Project 2 Outline

7.2.2.2 Recommendations

This was chosen as a good project to test the recommendations as it was recently launched and resulted in significant issues being introduced to Production. There were conflicting demands on this project. It was required to be delivered by a certain date to coincide with a new handset launch, had this date not been met it would have resulted in significant loss of market share to the company. Also, as the handset was very high profile any issues with the project would have been very visible and noticeable and would have resulted in reputational damage to the company. It was noted that if the recommendation *More Up Front Test Involvement in Planning and Sizing* had been followed then the Test Team would have been able to recommend that a project like this should include an element of UAT. Also, it was found that the issues, while serious, would have been very easily spotted by Business End Users. Thus, it the recommendation *Business End Users Executing Test Cases* had been followed then the defects would have been resolved before Production. Thus, in this case it was noted that the recommendation to add a Test Phase to the project would have caused an increase in timelines and the use of Business End users would have caused issues in call centre staffing levels. However, had these recommendations been followed then the defects wouldn't have been introduced and the company would have avoided revenue loss as well as the increased cost associated with fixing a defect in Production. The table below contains a description of the test phases on the project, what was completed in each and whether or not the phase was required:

	CIT	SIT	End-to-end	UAT
Description	Vendor internal unit testing	Both vendors completing integration testing	NA	NA
Test Team Involvement:	No	No	No	No
Test Phase Required:	Yes	Yes	No	Yes, there should have been user and Test Team involvement. Would have prevented introduction of Production defects

Table 7.4: Test Phases Comparison on Project 2

To complete this review of project 2 the findings of this analysis were presented to a member of the test team who agreed that a *More Up Front Test Involvement in Planning and Sizing* and *Business End Users Executing Test Cases* would have helped in this project. They said that "*by not having a UAT phase you lose an extra layer of business confidence. If you include a User acceptance phase, you get an internal review and analysis of requirements, an internal test coverage matrix signed-off, a different set of eyes executing the scenarios, and so a greater chance of trapping any bugs*"

7.2.3 Final Findings

Thus, it would appear that the Framework that was developed using Grounded Theory has merit and can be used to guide Best Practice in testing in a telecommunications company. It has been seen that had the Best Practice Framework been followed in recently completed projects these would have avoided the introduction of defects and also seen a reduced time to market and reduced cost. Thus, there would be a recommendation made to senior management that this Framework be adopted and employed by the organisation for all future projects.

7.2.4 Comparison with the Literature

From an analysis of the findings above it would appear that they would lend support to the Grounded Theory developed in Chapter 6:

Project Quality can be increased by ensuring that the key stakeholders are permitted to participate in each of the stages that they can meaningfully contribute to.

In relation to End User involvement in the UAT phase this is seen in the literature as a key feature to ensure the success of the projects (Leung and Wong 1997). It has been noted that one of the key reasons why projects fail is because they don't meet the needs of users (Ricca *et al.* 2009). Thus, the findings of this project that quality can be increased by appropriate and effective end user involvement in the UAT phase is in agreement with findings in the existing research literature.

The finding that project quality can be improved by better planning would also tie in with a number of studies completed in this area. More specifically on the area of project size and quality and the need for better upfront planning. Jiang *et al.* (2007) noted that earlier and accurate project size estimation can increase quality of the delivered project. It has also been found that software quality and effort are related to project size and can effect overall project quality (Agrawal and Chari 2007).

It appears that the two key findings in the Grounded Theory:

1. That project quality can be increased through end user involvement in UAT phase
2. That project sizing and quality can be improved through better upfront planning with Test team

would relate to and be in agreement with findings existing in the literature. From this point of view it would appear that the project findings are valid and would merit further study.

7.2.5 Presentation of Results to CIO

There was significant interest in this research in the organisation. The IT Delivery Lead had spoken about this at monthly meetings with CIO, who expressed a desire to be kept informed of the results of the study. As previously outlined the CIO is senior level executive that is responsible for the IT elements of the Technology group within this company. He is responsible for the long term strategic direction of the IT team as well as overseeing the day to day running of all the organisations IT systems. Once the results were compiled, the recommendations along with the Grounded Theory were presented by the researcher to the CIO. The format was a PowerPoint presentation. The presentation consisted of some background and introduction to Grounded Theory. Next the details of the study and interviews were outlined, including a sample of the questions and answers given. Following this the Grounded Theory was presented along with the findings of the research. The presentation lasted about 20 minutes with about 5 minutes of questions from the CIO. He was very receptive to findings and somewhat dismayed that the team weren't getting the support from the Business Owners in releasing end users for test execution. It was clear to him that increased support from the Business Team could result in higher quality deliveries from the IT team.

7.3 Conclusions

The aim of this project is to develop a Framework for best practice software testing in a telecommunications organisation. This chapter looked at the validity of the Framework by applying the recommendations to recently completed projects and comparing the actual quality outcomes of these projects against the quality had the Framework been applied. It was found that had the Framework been applied to these projects the quality would have been improved on both projects as well significant cost saving in both instances. Thus, it is concluded that Framework is potentially valid and could add value to the organisation if it was to be implemented. The next chapter is the Concluding chapter. It will detail the project, analyse the findings and recommendations for future work.

8. CONCLUSIONS AND FUTURE WORK

8.1 Introduction

The purpose of this project was to use Grounded Theory to develop a framework for best practice in software testing within a Telecommunications organisation. The aim of this chapter is to discuss the findings of the study and how these contribute to the body of knowledge. Following this there will be a discussion on the experimentation phase, which will involve evaluating were the aims of the research met and what the limitations of the current study were. Lastly, recommendations for further study in this area will be discussed.

8.2 Research Objectives

The area of study for this project was in the realm of Software Engineering. Specifically it was related to Software Testing within the Software Development Lifecycle. The aim of the research was to develop a framework for best practice in software testing with in a large telecommunications company. Below there is a list of these objectives with discussion on how each of the objectives was achieved and most importantly why it was important that objectives were met:

- Performed a literature review on the area of Software Development methodologies with a particular focus on Software Testing. This was important because if gave some background on the state of research in the area of software development and testing in particular. It was especially insightful by the fact that it highlighted a significant gap in the research. This was in the area of software testing best practice. There are a number of frameworks for software development but there appears to be a lack of academic research into software testing. This dissertation aims to contribute to this area of study.

- Performed a literature review on Research Methodologies (both Qualitative and Quantitative) with a particular focus on Grounded Theory and the application of this theory to software engineering projects. This was important as from this it was noted that the use of Qualitative Research Methods and Grounded

Theory in particular have been shown to be useful and applicable to research in the Software Engineering domain.

- Development of a framework and theory for best practice in software testing in a telecommunications company. This was necessary and important as it was the basis for the experimentation phase of the project. It was found that the framework and theory developed could be used to improve project quality and sizing.

- Tested the validity of the framework and theory by retrospectively applying it to two recently completed projects in the company and evaluated the results and recommended the adoption and roll out of framework across IT projects. In order to assess whether the model developed was valid it is necessary to test the recommendations. This was completed on two recently closed projects and it was found that had recommendations been followed the project quality could have been increased and time to market could have been reduced. This is important because it gives a basis and justification for further implementations of the framework.

8.3 Experimentation and Evaluations

There were a number of steps taken in the experimentation phase of this project. The first was the gathering of the data. It was then analysed and finally a Grounded Theory framework was created.

The data was gathered in the form of interviews with the Test Team. It was found that a semi-structured approach to the interviews worked best with seven questions used to guide the questioning. Interviewees were very open in their discussion about the way testing was being conducted and what they felt would be the best way of conducting testing. The data that was gathered was considered very relevant, the team had a lot of experience and held very valid opinions on the best ways to conduct testing.

Once the data was gathered it was then analysed. The researcher used a software tool called Atlas.ti which assisted in the coding of the interviews. Once the interviews were coded the researcher then began to apply the rigour of Grounded Theory on the data. This involved undergoing numerous rounds of further coding while continually completing constant comparative analysis on the data and linking it with areas already covered, making memos continually throughout this process. Then, once theoretical saturation of categories was reached the theory emerged. This was represented graphically for ease of understanding using the Atlas.ti network mapping tool.

Once the Grounded Theory Framework was developed it then needed to be tested to ensure that its validity. This was done by evaluating of two recently completed projects that proceeded without implementing the recommendations in the framework. The framework recommendations were applied retrospectively to these projects and it was tested to see whether or not if the framework had been applied to these projects, could the Production issues that were seen on these projects have been prevented. It was found that had the recommendations been implemented then the Production incidents wouldn't have been encountered. This would have had the results of reducing time to market as well as significant cost saving due to the removal of unnecessary phases of test as well as the cost of fixing issues in Production (as opposed to test).

8.4 Contributions to the Body of Knowledge

As mentioned above in recent years there has been an increase in the use of Qualitative Research in the area of Software Engineering. It has also been seen that Grounded Theory can be used to explain and understand practises in software companies. This project aimed to add to the body of knowledge by firstly demonstrating again that Qualitative Research methods are valid methods of investigation in the Software Engineering domain.

This project was successfully implemented and the Framework was seen to be valid. Thus, this project would add to the body of empirical evidence that suggests that both Qualitative Research Methods and in particular Grounded Theory can be used as methods of investigation in Software Engineering.

Also, through the literature review it was found that there was a lot of research conducted into technical and mathematical areas of testing (such as coverage calculation etc and test case selection). However, there appeared to be a lack of practical studies that investigated practical aspects of software testing. Thus, this project contributes significantly to this area on knowledge

8.5 Future Work

This study has shown that there is value in applying the Framework to future projects. Thus, a recommendation for future research would be to fully implement the recommendations and framework on an entire project and check the results in terms of savings in time and increases in quality (compared to if the recommendations weren't followed).

It would be interesting to expand the audience involved in the study. For example management from the IT side could be interviewed and their views on Testing could be gained. There could also be expansion to include Business Owners and even Vendors. Getting this wider perspective from different sources (albeit from people with less "at the coalface" experience) could result in alternative recommendations.

As mentioned, this study used the version of Grounded Theory as described by Stauss and Corbin (1998), it could be interesting to use an alternative version of Grounded Theory (the original Glaser and Strauss (1967) version or the version refined by Glaser (1978)) to check if the outcome would be the same.

It would be beneficial to use another Qualitative method (such as Case Studies or Narrative Research) on the same data sources to see if any different or alternative recommendations emerge.

Grounded Theory could also be used to determine a best practice approach to software testing in another large telecommunications company and compare the results with those found here and see if there are any different results or recommendations found.

There could also be more research completed into the area of software testing in terms of development of best practice in other fields (this study was limited to the telecommunications industry).

Finally, it could be looked to use of Grounded Theory to develop frameworks for best practice in other phases of the software development lifecycle in an Information Technology company (for example Design or Development). The results of this could be compared to the findings of this research to see if there are similarities in the recommendations.

BIBLIOGRAPHY

Abrahamsson, P., Salo, O. and Ronkainen, J., 2002. *Agile Software Development Methods: Review and Analysis*, VTT.

Adolph, S., Kruchten, P. and Hall, W., 2012. Reconciling perspectives: A grounded theory of how people manage the process of software development. *Journal of Systems and Software*, 85(6), pp.1269–1286.

Agrawal, M. and Chari, K., 2007. Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. *IEEE Transactions on Software Engineering*, 33(3), pp.145 –156.

Anon, ATLAS.ti: The Qualitative Data Analysis and Research Software. Available at: http://www.atlasti.com/index.html [Accessed November 14, 2012].

Babbie, E.R., 1990. *Survey Research Methods, Second Edition* 2nd ed., Wadsworth Publishing.

Baresi, L. and Pezze, M., 2006. An Introduction to Software Testing. *Electronic Notes in Theoretical Computer Science*, 148, pp.89–111.

Bartels, A., Holmes, B.J., Lo, H., 2006 uS Slowdown in 2007 will Dampen the $1.6 Trillion Global IT Market. Forrester Research, Retrieved from http://www.forrester.com/Research/Document/Excerpt/0,7211,40451,00.html.

Beck, K., 2000. *Extreme programming explained: embrace change* 1st ed., Boston, MA: Addison-Wesley.

Bertolino, A., 2007. Software Testing Research: Achievements, Challenges, Dreams. In Future of Software Engineering. Minneapolis.

Bertolino, A. and Marchetti, E., 2005. Introducing a Reasonably Complete and Coherent Approach for Model-Based Testing. *Electronic Notes in Theoretical Computer Science*, 116, pp.85–97.

Birks, M. and Mills, J., 2011. *Grounded Theory: A Practical Guide*, SAGE Publications.

Blumer, H., 1986. *Symbolic Interactionism: Perspective and Method*, University of California Press.

Boehm, B, 1986. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes*, 11(4), pp.14–24.

Boehm, B., 2002. Get ready for agile methods, with care. *Computer*, 35(1), pp.64 –69.

Boehm, B. and Turner, R., 2004. *Balancing Agility and Discipline: A Guide to the Perplexed*, Boston, MA.: Addison-Wesley.

Bryant, A. and Charmaz, K. eds., 2010. *The SAGE Handbook of Grounded Theory: Paperback Edition*, Sage Publications Ltd.

Charmaz, K., 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis* 1st ed., Sage Publications Ltd.

Charon, J.M., 1995. *Symbolic interactionism: an introduction, an interpretation, an integration*, Prentice Hall.

Cockburn, A., 2002. Agile software development joins the would-be crowd. *Cutter IT Journal*, 15(1), pp.6–12.

Cockburn, A., 2004. *Crystal clear a human-powered methodology for small teams* First., Addison-Wesley Professional.

Coffin, R. and Lane, D., 2006. A Practical Guide to Seven Agile Methodologies. *DevX.com The know-how behind application development*. Available at: http://www.devx.com/architect/Article/32836/0/page/2 [Accessed October 17, 2012].

Coleman, G. and O'Connor, R., 2007. Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology*, 49(6), pp.654–667.

Creswell, J.W., 2006. *Qualitative Inquiry and Research Design: Choosing among Five Approaches* 2nd ed., Sage Publications, Inc.

Creswell, J.W., 2008. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* 3rd ed., Sage Publications, Inc.

Crotty, M., 1998. *The Foundations of Social Research*, Sage Pubns.

Czarniawska, B., 2004. *Narratives in Social Science Research* 1st ed., Sage Publications Ltd.

Denzin, N.K. and Lincoln, Y.S. eds., 2005. *The SAGE Handbook of Qualitative Research* 3rd ed., Sage Publications, Inc.

Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B., 2012. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6), pp.1213–1221.

Dybå, T. and Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10), pp.833–859.

Edgren, R., 2011. The Little Black Book on Test Design. *thoughts from the test eye*. Available at: http://thetesteye.com/blog/2011/09/the-little-black-book-on-test-design/ [Accessed October 14, 2012].

Engel, A. and Last, M., 2007. Modeling software testing costs and risks using fuzzy logic paradigm. *Journal of Systems and Software*, 80(6), pp.817–835.

Erickson, J., Lyytinen, K. and Siau, K., 2005. Agile Modeling, Agile Software Development, and Extreme Programming. *Journal of Database Management*, 16(4), pp.88–100.

Glaser, B.G. and Strauss, A., 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Chicago: Aldine.

Glaser, B. G., 1992. *Basics of Grounded Theory Analysis: Emergence Vs. Forcing* 1st ed., Sociology Pr.

Glaser, B. G., 1978. *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory* 1st ed., The Sociology Press.

Goede, R. and De Villiers, C., 2003. The Applicability of Grounded Theory as Research Methodology in Studies on the use of Methodologies in IS Practices. In *Proceedings of SAICSIT*. pp. 208–217.

Guba, E.G., 1990 The Alternative Paradigm Dialog In Guba, E.G. ed., 1990. *The Paradigm Dialog*, Sage Publications, Inc.

Hamlet, D., 1995. Software Quality, Software Process, and Software Testing. In Marvin Zelkowitz, ed. *Advances in Computers*. Elsevier, pp. 191–229. Available at: http://www.sciencedirect.com/science/article/pii/S006524580860234X [Accessed October 9, 2012].

Hansen, B., Kautz, K., 2005. Grounded theory applied – Studying Information Systems Development Methodologies in Practice. In: P*roceedings of 38th Annual Hawaiian International Conference on Systems Sciences,* Big Island, HI.

Heath, H. and Cowley, S., 2004. Developing a grounded theory approach: a comparison of Glaser and Strauss. *International Journal of Nursing Studies*, 41(2), pp.141–150.

Hetzel, B., 1993. *The Complete Guide to Software Testing* 2nd ed., Wiley.

Hoda, R., Noble, J. and Marshall, S., 2010. Organizing Self-Organizing Teams. In *Proceeding of the 32nd ACM/IEEEE International Conference on Software Engineering*. Available at: http://ecs.victoria.ac.nz/twiki/pub/Main/RashinaHoda/Hoda_ICSE2010.pdf.

Huang, C.-Y., 2005. Performance analysis of software reliability growth models with testing-effort and change-point. *Journal of Systems and Software*, 76(2), pp.181–194.

ISTQB, 2012. ISTQB Glossary of Terms. *ISTQB*. Available at: http://www.istqb.org/downloads/viewcategory/20.html [Accessed October 14, 2012].

ISTQB, 2011. ISTQB Syllabus. Available at: http://istqb.org/downloads/finish/16/15.html [Accessed September 1, 2012].

Jiang, Z., Naudé, P. and Jiang, B., 2007. The Effects of Software Size on Development Effort and Software Quality. *World Academy of Science, Engineering and Technology*, 34.

Klunklin, A. and Greenwood, J., 2006. Symbolic Interactionism in Grounded Theory Studies: Women Surviving With HIV/AIDS in Rural Northern Thailand. *Journal of the Association of Nurses in AIDS Care*, 17(5), pp.32–41.

Leung, H.K.N. and Wong, P.W.L., 1997. A study of user acceptance tests. *Software Quality Journal*, 6(2), pp.137–149.

Manen, M.V., 1990. *Researching Lived Experience: Human Science for an Action Sensitive Pedagogy*, State University of New York Press.

Manifesto for Agile Software Development, 2001. Available at: http://agilemanifesto.org/ [Accessed October 10, 2012].

McMaster, S. and Memon, A., 2006. Call Stack Coverage for GUI Test-Suite Reduction. In *Proceedings of the 17th IEEE International Symposium on Software Reliability Engineering*.

Mead, G.H. and Morris, C.W., 1934. *Mind, Self and Society: From the Standpoint of a Social Behaviorist*, University of Chicago Press.

Merton, R. (1973). T*he Sociology of Science*. Chicago: The University of Chicago Press

Moe, N.B., Aurum, A. and Dybå, T., 2012. Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology*, 54(8), pp.853–865.

Morse, J.M., Stern, P.N., Corbin, J., Bowers, B., Clarke, A.E., Charmaz, K., 2009. *Developing Grounded Theory: The Second Generation* 1st ed., Left Coast Press.

Moustakas, C., 1994. *Phenomenological Research Methods*, SAGE.

Murchison, J., 2010. *Ethnography Essentials: Designing, Conducting, and Presenting Your Research* 1st ed., Jossey-Bass.

Myers, G.J., 1979. *The Art of Software Testing* 1st ed., Wiley.

Myers, M.D., 1997. Qualitative Research in Information Systems. *Management Information Systems*, 21(2), pp.241–242.

Neuman, W.L., 2011. *Social Research Methods: Qualitative and Quantitative Approaches* 7th ed., Allyn and Bacon.

One Stop Testing, One Stop Testing: Your One Stop Guide to Testing. Available at: http://www.onestoptesting.com/sdlc-models/v-model.asp [Accessed October 16, 2012].

Orlikowski, W.J., 1993. CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly*, 17(3), p.309.

Paulk, M.C., 2001. Extreme programming from a CMM perspective. *IEEE Software*, 18(6), pp.19 –26.

Phillips, D.C. and Burbules, N.C., 2000. *Postpositivism and Educational Research*, Rowman and Littlefield Publishers.

Power, N., 2002. *A Grounded Theory of Requirements Docuementation in the Practice of Software Development*. PhD Thesis. Dublin: Dublin City University.

Ricca, F., Torchiano, M., Di Penta, M., Ceccato, M., Tonella, P., 2009. Using acceptance tests as a support for clarifying requirements: A series of experiments. *Information and Software Technology*, 51(2), pp.270–283.

Rising, L. and Janoff, N.S., 2000. The Scrum software development process for small teams. *IEEE Software*, 17(4), pp.26 –32.

Roper, M., 1994. *Software Testing*, Mcgraw Hill Book Co Ltd.

Rothermel, G., Elbaum, S., Malishevsky, A.G., Kallakuri, P., Xuemei, Q., 2004. On Test Suite Composition and Cost-Effective Regression Testing. *ACM Transactions on Software Engineering and Methodology*, 13(3), pp.277–331.

Royce, W., 1970. Managing the Development of Large Software Systems. In *Proceedings of IEEE WESCON*. pp. 1–9.

Sampeth, S., Ren'ee, C.B., Gokuluanand, V., Vani, K., Gunes, K., 2008. Prioritizing User-session-based Test Cases for Web Applications Testing. In International Conference on Software Testing, Verfication and Validation. Lillehammer, Norway: IEEE Computer Society.

Schmidt, E.P., 1937. *Man and Society ; Substabtive Introduction to the Social Science*, Prentice-Hall, Incorporated.

Schwaber, K, 1995. Scrum Development Process. In *10th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications Addendum to the Proceedings*. OOPSLA '95 Workshop on Business Object Design and Implementation.

Schwaber, K. and Beedle, M., 2001. *Agile Software Development with Scrum* 1st ed., Prentice Hall.

Silva, L. and Backhouse, J., 1997. Becoming part of the furniture: the institutionalization of information systems. In *Proceedings of the IFIP TC8 WG 8.2 international conference on Information systems and qualitative research*. London, UK, UK: Chapman andamp; Hall, Ltd., pp. 389–414. Available at: http://dl.acm.org/citation.cfm?id=278888.278915 [Accessed October 26, 2012].

Srikanth, H. and Williams, L., 2005. On the Economics of Requirements-Based Test Case Prioritization. In EDSER '05. St. Louis, Missouri: ACM.

Srikanth, H., Williams, L. and Osborne, J., 2005. System test case prioritization of new and regression test cases. In International Symposium on Empirical Software Engineering.

Stern, P.N. (2009) In the beginning Glaser and Strauss Created Grounded Theory. In Morse, J.M., Stern, P.N., Corbin, J., Bowers, B., Clarke, A.E., Charmaz, K., 2009. *Developing Grounded Theory: The Second Generation* 1st ed., Left Coast Press

Strauss, A. and Corbin, J.M., 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* 2nd ed., Sage Publications.

Strauss, A.L., 1987. *Qualitative Analysis for Social Scientists*, Cambridge University Press.

Weisert, C., 2003. Waterfall Methodology: There's no such thing! Available at: http://www.idinews.com/waterfall.html [Accessed October 10, 2012].

Whyte, G. and Mulder, D., 2011. Mitigating the Impact of Software Test Constraints on Software Testing Effectiveness. *The Electronic Journal Information Systems Evaluation*, 14(2), pp.254–270.

Williams, L. and Cockburn, A., 2003. Agile Software Development: It's about Feedback and Change. *Computer*, 36, pp.39–43.

Wolcott, H.F., 2008. *Writing Up Qualitative Research* Third Edition., Sage Publications, Inc.

Yin, R.K., 2008. *Case Study Research: Design and Methods* 4th ed., Sage Publications, Inc.

Zhang, X., Xu, B., Chen, Z., Nie, C., Li, L., 2008. An Empirical Evaluation of Test Suite Reduction for Boolean Specification-based Testing. In The Eighth International Conference on Quality Software. IEEE Computer Society.