Towards More Effective Testing of Communications-Critical Large Scale Systems

A thesis submitted for the degree of
Doctor of Philosophy
by

Mohammad Nabulsi

School of Information Systems, Computing and Mathematics
Brunel University

BURA Open Access Version 2: 06 April 2014

Abstract

None of today's large scale systems could function without the reliable availability of a varied range of network communications capabilities. Whilst software, hardware and communications technologies have been advancing throughout the past two decades, the methods commonly used by industry for testing large scale systems which incorporate critical communications interfaces have not kept pace. This thesis argues for the need for a specifically tailored framework to achieve effective testing of communications-critical large scale systems (CCLSS). The thesis initially discusses how generic test approaches are leading to inefficient and costly test activities in industry. The thesis then presents the form and features of an alternative CCLSS domain-specific test framework, develops its ideas further into a detailed and structured test approach for one of its layers, and then provides a detailed example of how this framework can be applied using a real-life case study. The thesis concludes with a qualitative as well a simulation-based evaluation of the framework's benefits observed during the case study and an evaluation by expert external participants considering whether similar benefits can be realised if the framework is adopted for the testing of other comparable systems. Requirements data from a second CCLSS is included in the evaluation by external participants as a second smaller case study.

Table of Contents

Acknowledgements

I would like to thank my supervisor Professor Hierons. The benefit I gained from his supervision was not only to be able to complete this thesis, but I felt I was continuously progressing academically and professionally with his helpful comments and generous advice.

As the work for this thesis was done on part-time basis, it had to be done during weekends, bank holidays, late nights and what otherwise would have been family holidays. I thank my wife Neda and son Reza for their patience with me during my absence on so many occasions. Thank you my dear wife for your encouragement and support which helped me complete this work.

Another factor that helped me complete this work is the positive atmosphere in Brunel University generally and St. John's Building specifically and the comfortable facilities provided to PhD students working in the St. John's Building. Being a part of this community has been a good experience. I would like to thank all members of staff in the Department of Information Systems and Computing for making DISC what it is and wish them success in their careers.

Last but not least, I cannot be grateful to anyone if I am not first of all grateful to my parents to whom I owe an eternal gratitude for everything good and right that I do in my life.

Glossary

ANOVA	Analysis Of Variance
APM	Association for Project Management
AVLS	Automatic Vehicle Location System
BRT	Business Readiness Testing
BURA	Brunel University Research Archive
CCI	Communications Control Interface
CCLSS	Communications-Critical Large Scale System
CMM	Capability Maturity Model
E2E	End to End
eTOM	Enhanced Telecom Operations Map
ETSI	European Telecommunications Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance, and Security management functions (see TMN)
Firelink	The radio system used by the fire services in England, Wales and Scotland
FAB	Fulfilment, Assurance, Billing and Revenue Management
FRS	Fire and Rescue Service
FSM	Finite State Machine
GIS	Graphical Information System
GUI	Graphical User Interface
ICCS	Integrated Communications Control system
IEC	International Engineering Consortium
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
ITU-T	International Telecommunication Union - Telecommunication Standardization Bureau
LAN	Local Area Network
LSCITS	Large-Scale Complex IT Systems
MBT	Model-Based Testing

MDD	Model-Driven Development
MDT	Mobile Data Terminal
MIS	Management Information System
MRMS	Mobilising and Resource Management System
MSC	Message Sequence Charts
NF	Non-Functional
NGN	Next Generation Network
NGOSS	New Generation Operations Systems and Software
OAT	Operational Acceptance Testing
OA&M/OAM	Operations, Administration and Management
OSS	Operations Support Systems
QA	Quality Assurance
QoS	Quality of Service
RCC	Regional Control Centre
RUP	Rational Unified Process
SEI	Software Engineering Institute
SLA	Service Level Agreement
SoR	Statement of Requirements
SPICE	Software Process Improvement and Capability Evaluation
SRB	Secondary Radio Bearer
TIM	Test Improvement Model
TMF	Telecommunications Management Forum
TMN	Telecommunications Management Network
TPI	Test Process Improvement
TTCN	Testing and Test Control Notation
UAT	User Acceptance Testing
ULS	Ultra-Large-Scale Systems
UML	Unified Modelling Language
VBA	Visual Basic for Applications
WAN	Wide Area Network

1. Chapter 1: Introduction and overview

EXPLANATION OF THE PROBLEM AREA, THE MOTIVATIONS AND BENEFITS OF RESEARCHING IT

In the late 1980s and early 1990s, structured software testing was less widespread in industry than it is now. The business case for investing large portions of Information Technology (IT) project budgets on testing was still not clearly defined. The case for allocating project resources to software testing became gradually better established and accepted throughout the 1990s. By the late 1990s, testing became a significant and established part of IT project plans and budgets.

During the 1980s and early 1990s, software development methodologies, terminologies and practices in the public (i.e. Government) sector were a major source for defining accepted test practices in industry. The 1990s saw the emergence of a variety of test methodologies, commonly used terminologies, processes and tools. Despite their variety, they all shared the common aim of making testing efficient, structured and cheaper. This in turn was intended to lead to reduced IT project costs and risks and improved quality of the IT deliverables. However, the detailed definition of test cases and how they are derived and expressed remained largely a subjective process (Bertolino, 2007). The way testing is done and how efficient it will be still relied heavily on the creativity and experience of the tester rather than on the test standard or methodology used.

Current commercial test methodologies, processes and tools can and do help make the test activities better organised and structured, but the design and specification of the tests still rely to a large extent on the tester's interpretation and understanding of the system under test. This creative aspect is a feature inherent in testing and need not be viewed negatively, but reliance on the subjective judgement of testers leads to reduced precision and reduced efficiency of the test activities. Furthermore, prevalent industrial test methodologies, standards and tools are not domain-specific (Bertolino, 2007). This means that individuals or teams involved in testing a system have to adapt the test methodology, standard or tool to the type of the system under test. This inherently incurs further overheads for IT project budgets and timescales. It also means that the experience gained whilst testing one system is not easily transferable to another test activity of another system of the same type.

Inefficient and imprecise testing ultimately results in inadequately tested systems with lower reliability and availability levels than is needed, as well as project delays and higher project costs. The impact of the effectiveness and precision of testing practices is further magnified for large scale IT systems that are prevalent today (The Royal Academy of Engineering, 2004) (Boehm, 2006). These are systems which combine multiple technologies, multiple hardware platforms, multiple software components, multiple internal and external communications interfaces, and are spread over a number of physical locations. Such systems often can only function with the availability of a range of communications networks services. For the purposes of this thesis, such systems will be described as "Communications-Critical Large Scale Systems" (CCLSS).

Such systems are increasingly more prevalent, more complex and critical. In fact, developed societies can no longer continue to function normally without this class of systems. Examples of such systems are numerous and can be found in all sectors of industry and civic life such as:

- Emergency mobilisation applications: e.g. for police, fire and ambulance services.
- Telecommunications network management and operations support systems (NMS/OSS)
- Web based portals and ecommerce sites.
- Distributes banking applications, trading systems
- Supply chain applications
- Fleet management, Automatic Vehicle Location Systems (AVLS)
- E-Health systems.
- Cloud computing systems

Such systems have gradually been evolving since the 1980s and early 1990s. Back then, IT systems were developed using procedural or object-oriented programming languages. The norm was for IT systems to be developed in-house and often be purpose built for specific clients and specific uses. Development work would often be carried out by the same team or the same company. The norms now for developing large scale systems are quite different.

One of today's systems may incorporate a range of technologies and include converged communications and IT technologies. Furthermore, customisable off-the-shelf components and "middleware" components replaced much of the programming activities of the past with configuration and integration activities. Nowadays, one large scale system could be developed by different teams or different companies across different physical locations. All these changes to IT systems have occurred over the past two decades, but the fundamental methods used for testing them did not keep pace.

This thesis focuses on the testing of communications-critical large scale systems in particular because they represent the types of systems which incorporate critical communications related components and services whose testing cannot be done effectively whilst using approaches that were evolved for an earlier (pre-convergence) IT era.

Can the IT industry's vendors, clients and users adopt test approaches that would ultimately lead to more efficient testing of such communications-critical large scale systems?

The remainder of this chapter will mostly outline some key factors that lead to inefficiency in the testing of such systems. It will include a brief discussion of the current state of the art in testing in the IT industry: the theory that shapes it and the practice. This outline will then be used in subsequent chapters as the basis for developing an alternative approach which can lead to more effective and precise testing of communications-critical large scale systems.

1.1 Current Test Theory and Practice

As will be discussed in the literature review (Chapter 2), current published material on software testing methods tends to vary between two ends of a spectrum: precise methods for deterministic systems and for academic experimentation on the one hand, then on the other hand generic methods and process standards to be adopted for industrial and commercial systems testing. Precise industrial methods and standards exist for telecommunications protocols conformance testing, reflecting precisely defined protocols and interface standards. However, there seems to be no theoretical test frameworks that can support precise testing

practices for large scale non-deterministic IT systems (Bertolino, 2007) (Bochmann, Rayner, & West, 2010).

1.1.1 Testing in practice

During commercial IT projects, budgets, deadlines and deliverables are the primary considerations. The objective of the test activity is to support the overall project objectives. Engineering rigour and precision of the test cases are the responsibility of the individual tester who specifies them (Bertolino, 2007) (Reid, 2000) (Smith, et al., 2008). The requirements for large scale systems are usually done at a high level requiring interpretation by the developers (Belgamo, et al., 2005). The active role of testers normally starts when code deliveries are approaching and the project deadlines are becoming nearer (Chernak, 2001). This scenario often applies both to in-house as well as outsourced projects, where testers often have limited time to produce the test design and test cases relying on their experience and understanding of the requirements and the design of the system. This, in turn, often results in the test design and test cases being produced under time and budget pressures and being based on inaccurate understanding of the system (Alicherry, Bhatia, Nagesh, Phadke, & Poosala, 2003) (Davis & Venkatesh, 2004).

Over the past two decades, despite major evolution in the types of communications technologies and systems under test, the role of the tester has not changed significantly. One notable demonstration of this is the V-Model approach (Ammann & Offutt, 2008)(Bertolino, 2007) (IABG, 1993) which is often referred to as a key IT industry approach for structured testing, yet it is only a general framework for test phase organisation and structure rather than an engineering framework for producing precise test design and test cases.

Limitations of the V-Model

The V-Model view of the software lifecycle as illustrated in Figure 1 is a popular approach in the IT industry (Bertolino, 2007). Test design and test specification activities are based on the V-Model test stages: unit and subsystems tests, integration tests, system tests and acceptance tests. Such an approach remains widely adopted for most structured test activities in industry when testing large scale enterprise-wide systems (Bertolino, 2007). Each stage is

then based either on the requirements, the specification, or the design of the system. All tests have to ultimately be traceable back to the requirements.

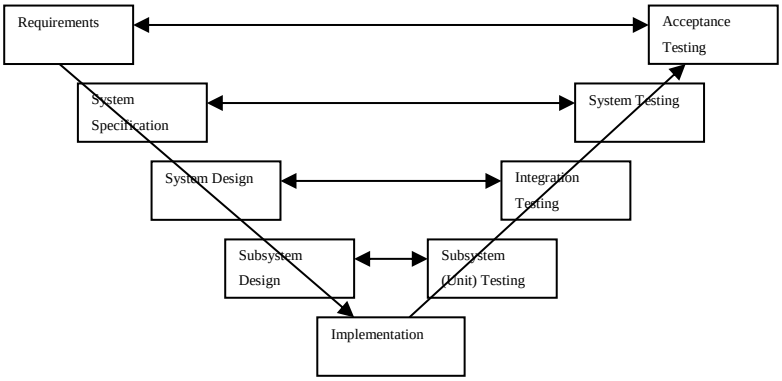


Figure 1: The V-Model test approach

The requirements for a large scale system are usually expressed at a high level and therefore cannot be sufficient as the only basis for precise test cases. However, subsequent specification and design activity done by developers will be based on such requirements. When developers start their specification and design work, they are usually constrained by project timescales and costs. This limits their motivation or time to do further costly analysis and rework of the requirements (Davis, et al., 2004). The developer's productivity is judged by the specification and design they deliver and ultimately the testable code. This stage of interpretation of the requirements into design and specification is the stage of a system's lifecycle where variations between what was intended and what actually is delivered start being introduced. Such variations may remain undetected until a much later stage of the project, e.g. during the V-Model's system or acceptance testing stage. A more effective test approach needs to derive test cases from either more precisely engineered requirements and/or a technical interpretation of them that is independent of the developers' own interpretation.

The V-Model originated during a time when systems were far less complex and more procedural than they are now. IT systems became far more complex during the late 1990s and typically supported more complex and critical services. Industry trends moved towards more use of Customisable (or Modifiable) Off-The-Shelf systems (COTS/MOTS), standardised communications interfaces, and convergence between IT and communications

technologies. Development methodologies moved from being procedural in the 1980s, to object oriented in the 1990s. Nowadays, the trend is for model driven (Bertolino, De Angelis, Di Sandro, & Sabetta, 2011) or service oriented architectures (Choi, Nazareth, & Jain, 2010) (MDA, SOA) and associated development approaches. The V-Model has not been updated to reflect these changes and provides no domain-specific guidance on how test design for communications related services should be done, leading to more costly and less effective testing particularly when the system under test is a communications-critical large scale system. Furthermore, it is not appropriate for dealing with communications technologies and services where distinction between functional and non-functional features is often not clear and where the services cannot be expressed in traditional test case styles of initial condition/input/procedure/output.

Using the V-Model as the fundamental and only test framework when testing communications-critical large scale systems can lead to:

- Reliance on the tester's knowledge and experience of a similar, possibly less complex or less up-to-date, system. This in turns means further cost and less efficiency when testing systems intended for newer technologies;

- Any ambiguity in the requirements that is not resolved during the design phase could remain undetected until a much later stage of the system's lifecycle, such as acceptance testing;

- Testing being based only on what is stated explicitly in the requirements or in the design documentation. This means that complexities such as network services interactions or network events timing issues may not be taken into account sufficiently during test design if they are not explicitly expressed in the requirements and the design documentation;

- Testing of critical communications interfaces to be assigned less importance than is appropriate: the reliability of a communications-critical large scale system is dependent on the reliability of its communications interfaces. The V-Model approach to testing does

not sufficiently address such domain-specific aspect;

- Reliance on the tester's own knowledge and experience to take into account critical but non-technical aspects about the system under test, e.g. what operational benefits/services the system is intended for, how the system fits within the overall service structure and value chain. Understanding aspects such as these is critical to achieve effective testing, yet may not be stated in the requirements or in the design documentation.

- Test approaches based on the V-Model viewing system data purely as part of the pre-requisites for functional test cases rather than one of the sources for test design. The V-Model originated during an era of procedural systems rather than object-oriented or data driven systems. It does not provide guidance that test design should be based, at least in part, on the data model of the system.

Large scale IT systems have been increasingly more critically dependant on their communications interfaces, yet the widely accepted approaches for their testing have not kept up with this convergence between the IT and communications domains. The adoption of widely used standards and methods such as the V-Model, Agile, IEEE 829, BS 7925-2, ISO 12207, CMM, or IEEE 1012, or (the more recent) ISO/IEC/IEEE 29119 would only define the structure of the test process. Such generic standards and methods do not precisely define the coverage and thoroughness of the test cases that would be necessary for testing communications critical large scale IT systems. The IT industry needs a framework that allows for communications-critical large scale systems to be tested more precisely and predictably.

The above arguments do not mean that the V-Model generic test approach is unsuitable for testing a communications-critical large scale system, but that using it effectively requires heavy reliance on the tester's own, often subjective, judgement to creatively adapt it to the system under test.

1.2 Direction and contribution of this research

The issues discussed in this chapter point to the need for a more precise domain-specific test framework to be used as basis for more effective test analysis and test design for communications-critical large scale systems. Therefore, the objective of this research will be to design a new test framework for communications-critical large scale systems (CCLSS) that can point to a way towards more effective testing of such systems.

To provide confidence that the new test framework is usable and can potentially have an impact on software testing practices in industry, it needs to be applied to at least one real-life CCLSS case study and have its benefits evaluated. To ensure reliability of the findings, potential users of the framework who are independent and not part of this research effort should ideally participate in the evaluation.

The research questions that need to be answered will therefore be around whether the new framework is usable and viable for use on real-life CCLSS projects, what efficiency benefits it can bring to the testing activities and whether it can be accepted and applied by external participants who are not part of this research.

The key contribution of this thesis will be the design and application of a new domain-specific test framework for Communications-Critical Large Scale Systems (CCLSS) with its feasibility for real-life industrial projects demonstrated through at least one case study, and its benefits evaluated both by the author and by potential users.

Further explanations about the methodology adopted for this research and the precise research objectives and questions will be presented in the research methodology chapter (Chapter 3). Additional contributions from this research relating to the research methodology, the application and evaluation of the new framework will be outlined in the final chapter (Chapter 9).

The remainder of this thesis will discuss ideas on how such a domain-specific test framework could be formulated.

The overall framework will be presented and discussed in Chapter 4, and then a specific example of one of the framework's components (or layers) will then be detailed further in Chapter 5, and then a case study in Chapter 6 will apply the ideas of Chapters 4 and 5 and evaluate their benefits. Further detailed qualitative and quantitative evaluation of the ideas will be presented in Chapter 8. Chapter 7 will present a user-based evaluation of the framework's ideas carried out by external participants. Chapter 9 will be the conclusion chapter for the thesis.

The next chapter (Chapter 2) will present a review of relevant published literature on software testing.

2. Chapter 2: Literature review

THIS CHAPTER SUMMARISES THE FINDINGS OF THE LITERATURE REVIEW

2.1 Literature Review Introduction

A distinct literature review stage was conducted at the beginning of this research work. Subsequently, and throughout the duration of this work, occasional searches for the latest literature were done to help refine the ideas of the thesis and to ensure they are still up-to-date. The findings were then revisited and finalised during the final stage of this thesis. As well as covering academic journals and conference proceedings, the search also included official commercial IT standards for testing or relevant to testing.

2.2 What is software testing

A classic definition of software testing, which is found in a foundation book on software testing first written by Glenford J. Myers in 1974, is "*Testing is the process of executing a program with the intent of finding errors*" (Myers, Sandler, & Badgett, 2012). The author of this book had been writing on code debugging and software testing (Myers, 1978) since at least the 1970s and it seems that this book is an appropriate introduction to software testing not as a purely technical activity but it also explains the psychology behind it as well explaining testing in a relatively timeless way not reliant on specific technologies, methodologies or standards.

The title is also quite appropriate in that it expresses that software testing remains, to a large extent, an "Art". Basic software testing may not be a complex activity compared to software design and development. However, effective and timely testing of complex systems is a complex activity that requires good experience in a combination of technical, operational, project management disciplines. Additionally, each technical domain and each type of system and each operational environment may have their own considerations that need to be taken into account while preparing for and conducting tests. Therefore, an effective tester needs to combine technical engineering skills with constant creativity and adaptation to new environments and projects. The purely technical aspects of testing can and should be taught,

but other important aspects of testing such as test strategy and test project planning are practice based and can only be gained through actual hands-on test work. That's why describing testing as an "Art" is somewhat appropriate.

2.3 Related terminology

Looking through the testing related literature, old and recent, it is easy to be confused by the numerous overlapping terms used to describe testing or describe test related activities. This section will outline and clarify the few key terms used for testing or testing-like activities, but without attempting to construct the history and evolution of testing and its terminology as this would require a thesis in its own right.

The main terms that can be encountered in the literature that either mean testing or have meanings that overlap with testing are the following:

Debugging:

This is probably the oldest term that referred to testing. One of the earliest computing papers that mention the term "debugging" was a paper titled "MANIAC" (Demuth, Jackson, Klein, Metropolis, Orvedahl, & Richardson, 1952) discussing work on the MANIAC computer project that was carried out in Las Alamos Scientific Laboratory in the late 1940s to early 1950s. Debugging is the activity carried out by the programmer to identify and remove programming errors in order to remove "bugs" from the code. From this paper and other papers from the 1950s (Orden, 1952) (Campbell, 1952), it is clear that debugging was an activity carried out by the "scientist" that developed the code and was not a distinct activity as testing is nowadays. Whereas there may have been an overlap in the past between the terms "testing" and "debugging", these are now two distinct activities.

In the second section of a recent overview paper on software testing (Machado, Vincenzi, & Maldonado, 2010), testing and debugging are mentioned as activities that complement each other: *"Once a fault is detected, the testing activity is usually interrupted and the debugging activity takes place".* This viewpoint is probably due to the paper being oriented towards the

testing of program code rather than large scale systems. During the testing of program code, it is expected that the tester is at least aware of the programming language used and the result of each test failure is likely to be a de-bugging effort by the programmer. This type of vocabulary, whilst still applicable to an extent to the testing of large scale systems, does not represent the commonly used tester vocabulary for test phases other than possibly during unit testing.

Quality Assurance:

Also referred to as "QA", this is a current term that is often used interchangeably with the term "testing". According to ISO standard 9000:2005 3.2.11 (ISO, 2005), Quality Assurance is *"A part of quality management focused on providing confidence that quality requirements will be fulfilled"*. QA activities should be part of an overall standards compliance effort within an organisation, part of which can be software testing. It's a useful term to use because, depending on the context within which it is used, "software testing" alone may not be sufficient to confirm that a new complex system is fit for its purpose. There may be additional inspection, review and demonstration activities that might be more efficient to detect potential problems especially in areas that are not purely software related, e.g. ergonomics, usability, quality of the overall technical design, service management, training, etc.

Verification and Validation:

Validation is the *"Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled"* (ISO 9000:2005 3.8.5) (ISO, 2005). Verification is the *"Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled"* (ISO 9000:2005 3.8.4) (ISO, 2005). More intuitive definitions of the two terms are found in a 1984 paper by Barry Boehm (Boehm B. W., 1984): "Verification" as being *"The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase"*, while "Validation" as being *"The process of evaluating software at the end of the software development process to ensure compliance*

with software requirements". Simplified yet further, Verification means *"Am I building the product right?"* and Validation means *"Am I building the right product?"*

V&V is another term that can often be encountered either to include testing or combined with testing. They are necessary activities to incorporate in an IT project to ensure that the final delivered system is fit for purpose and cannot be substituted by testing alone. However, in terms of widespread practice and acceptance in the IT industry, the term "testing" is far more widely used in the IT industry.

2.4 Goals of software testing

A summary of the goals of "good" testing is provided in (Quadri & Farooq, 2010). The paper lists the goals of testing as: 1) Verification and Validation 2) Priority Coverage 3) Balanced 4) Traceable and 5) Deterministic. In section 2.1, the paper states that *"It would not be right to say that testing is done only to find faults. Faults will be found by everybody using the software"*. This is a viewpoint worth highlighting. Finding faults is a main objective and benefit of testing, but testing and finding faults are not one and the same. Testing is also about providing structured, planned and formal evidence that the system under test is ready for operational use and is, in the case of user acceptance testing, acceptable to the users. It is notable also that the paper seems to treat testing as a sub-activity of Verification and Validation, which is a valid but not a universally acknowledged viewpoint. Prioritisation of testing is also a significant attribute of good effective testing, as testing should not be a random unplanned activity even if it intentionally includes exploratory or ad-hoc testing stages. Likewise, the testing activity should balance between the requirements, user expectations and the operational environment for the system under test. In other words, testing should not be rigid and dogmatic on one hand, on the other hand it should take into account the pragmatic realities and factors surrounding the system under test. Traceability is also an important feature of effective testing. Testing should have a formal basis and should not be an ad-hoc activity, i.e. the test cases should be derived from formal sources. Sources for deriving test cases could be one or a combination of the following: the user requirements, technical design, technical specification, system use cases and user business processes. The last goal, according to (Quadri & Farooq, 2010), is determinism. The test strategy and test

design should provide a view of what types of tests will be conducted during the different stages of testing; therefore this should provide a concept of the types of faults and issues that are likely or not likely to be encountered. For example, during a user acceptance phase immediately preceding the service management phase of the system's lifecycle, it should be expected by that stage that any internal or external interface tests have been completed and that the user acceptance testing should not encounter interface related faults or incompatibilities. User acceptance testing could encounter issues or faults relating to the business process, setting up of real user data, documentation or training issues. If during a user acceptance testing phase too many interface related issues are being uncovered then that should be a cause to review whether the system is in fact ready for user acceptance testing or whether it should undergo further integration testing instead.

A more project-oriented view of the objectives of testing is provided by (Everett & McLeod Jn., 2007) where the primary objectives of testing are explained as: 1) identify the sources and magnitude of development risk reducible by testing 2) perform testing to reduce identified risks 3) know when testing is completed 4) manage testing as a standard project within the development project. These are more an overview of how to do testing rather than purely "objectives" of testing.

The variation between the two references cited in this section is an example of how perceptions of testing can vary from one environment to another and from one author to another. In reality, the objective for testing varies depending on the type of testing being carried out and the type of system under test.

Ultimately, the objective of testing is to contribute to the delivery of fit for purpose software by identifying and resolving faults and weaknesses in the software that crept into the software during the preceding development phases.

2.5 How is software testing done?

According to (Kasurinen, Taipale, & Smolander, 2011, p. 556), the key test documentation of ISO/IEC/IEEE 29119 (ISO/IEC 29119, 2014) are listed as *"test policy, test strategy, test plan*

and test completion report". This also concurs with (IEEE 829, 2008, pp. 6 Fig-2), titled "partial use of the standard", which describes the core test documents as *"test planning"*, *"test design"*, *"Test cases/test procedures and their execution"* and *"Test results reporting"*. ISO/IEC/IEEE 29119 is a relatively recent standard (Kasurinen, Runeson, Riungu, & Smolander, 2011) that attempts to encompass all generic dimensions of test management and testing practices. IEEE 829 is a standard that has been known since the 1980s. Although IEEE 829 is a standard for test documentation, it has been in use in industry as a test standard to define structured test practices. ISO/IEC/IEEE 29119 is a new standard so it is still not known how well it will be adopted by industry. Referring to the core test documentation sets for both standards is intended as a way to distil their apparent complexity. The basic or core documentation sets defined by both standards actually describe the key generic steps for testing that are applicable in all environment and for all types of testing. The test policy in ISO/IEC/IEEE 29119 is meant to be the overarching organisational policy for testing. The remaining three documentation types of ISO/IEC/IEEE 29119 are consistent with the four core documents defined in IEEE 829. Both standards define a strategy or overall planning stage, the term "test strategy" in ISO/IEC/IEEE 29119 refers to "test planning" under IEEE 829. The "test plan" in ISO/IEC/IEEE 29119 is actually equivalent to both the "test design" and the "test cases/procedures" in IEEE 829. The term "Test completion" is the same for both standards.

The above variation in terminology is an example of the variations of terms in testing in general. Beyond the terminology, both standards express a simple overarching concept of how testing is done: firstly, by strategic thinking and definition of how testing will be done, followed by preparations for testing which include test case design and specification, which is then followed by execution of the tests, when the test phase is completed then a test phase completion step is carried out represented by a test completion report.

2.6 The structure and setup of a testing organisation

A minimal test practice framework that is compatible with CMM (Paulk, Curtis, Chrissis, & Weber, 1993) and other process assessment frameworks specific to testing such as TMMi (Gelperin, 1996), TIM (Ericson, Subotic, & Ursing, 1997) and TPI (Koomen & Pol, 1999) is

presented in (Karlstrom, Runeson, & Sara Norden, 2005). According to the paper, the framework *"defines the kind of practices that are needed in small and emerging software companies"*. It is intended to provide small organisations with a formal test structure while avoiding the costs associated with test process frameworks or standards that are disproportionate to their needs or resources. It proposes an overall test structure divided into five categories and three phases. The five categories are: "Problem and experience reporting", "Roles and organisation issues", "Verification and validation", "Test administration" and "Test planning". The three phases define levels of structure and practices corresponding to the size of an organisation, with phase 1 being relevant to organisations with around 10 employees working in development, phase 2 for 20 employees in development and phase 3 for 30+ employees.

This paper offers a useful insight into the type of preparations needed for structuring test teams within small organisations. Using its proposals can potentially save a small organisation considerable effort researching process improvement and process assessment frameworks, and then having to tailor their organisation around such frameworks. The recommendations are likely to be of tangible value to an organisation if it has no test specialists familiar with the relevant test standards and who are able to tailor a test team structure specific to the needs of that organisation. Nonetheless, it should at least be interesting to read by the management of a small organisation regardless of their existing knowledge in testing.

2.7 Testing Standards and methods

Below is a (non-exhaustive) list of a number of well-known testing standards and methodologies:

V-Model: A methodology that is widely accepted and cited in industry although it is not documented explicitly in any well-known international testing standard. It was noticeable that not many recent journal articles advocating the continued use of the V-Model. Examples of recent papers advocating the continued adoption of the V-Model are: (Clark, 2009) and (Mathur & Malik, 2010). However, both propose ideas for how the V-Model can be adapted

and updated to current systems. One of the key factors in the V-Model's absence from recent literature is very likely to be that its development equivalent, the Waterfall model, is considered outdated. For example, in a book titled "How We Test Software at Microsoft" (Page, Johnston, & Rollinson, 2009) the authors state that *"waterfall has become somewhat of a ridiculed process among software engineers, especially among Agile proponents. In many circles of software engineering, waterfall is a term used to describe any engineering system with strict processes"*. Whether the V-Model is outdated or not is debatable, however, for the purpose of this literature review it is useful to note its absence from recent literature despite it being a commonly used term in industry.

Agile: The Agile methodology, which incorporates testing, seems to have become more widespread over the past several years. The methodology is explained and applied in a case study described in (dos Santos, Karlsson, Cavalcante, Correia, & Silva, 2011). Behind the new terminology it introduces, it seems that Agile is an updated approach to development and testing that is keeping up with how systems are more typically developed nowadays, by being evolved iteratively and through continuous interaction between developers, testers, users and business analysts, rather than having rigid fixed phases. It is notable that the paper uses some terms which are borrowed from the V-Model such as unit and integration testing. Agile promotes a dynamic and pro-active way of developing software, however, it also seems based on an implicit acceptance that software development is not a precise engineering discipline. Rather than assume that software is designed precisely at the start of a project, then this precise design is implemented and tested, there seems to be an implied acceptance that in reality software development is creative, fluid and changeable requiring constant interactions by the developers and the users. As long as the development of software continues to require iterative efforts, and where the initial user requirements and technical specification are treated as a starting point that is subject to constant alterations then the Agile approach to testing seems to be appropriate. The paper's description of the experience with Agile was generally positive and the issues it reports were matters that can be resolved through more practice of Agile by the team.

The paper does not mention potential pitfalls or discuss when Agile may be less appropriate. For example, when adherence to strict timescales to deliver precise and complex

requirements for a large scale system to a budget is required or when the testers' technical abilities and training does not prepare them to work closely with specialist developers.

Test Driven Development (TDD): Described in (Kollanus, 2011) as follows: *"The basic idea of TDD is simply to write tests before code in small iterations. First, developer writes a test case that is just enough to define the next functionality. The next step is to write code that is just enough to pass the test. Finally, the code is refactored, if needed. These steps are iterated in short cycles through the whole development process. Originally TDD was introduced as a development, not a testing, method".* The paper's focus seems to be the lack of empirical evidence on TDD. However, how many other methodologies in software development are supported by empirical evidence? Does the lack of empirical evidence indicate any more than the level of interest by the research community in TDD? Ultimately, as a methodology it either gets acceptance and adoption by industry or it does not. Also it can be argued that the level of commitment and support (e.g. tools, training) shown in industry for a methodology are also key factors on whether it proves to be a productive methodology rather than the qualities of the methodologies in its own right. The paper is a notable example of the difference in viewpoints between the research community and industry, i.e. the academic and evidence based vs. the management or commercial viewpoints.

A variation of TDD is "Behaviour Driven Development" (BDD), which is a style of development, often combined with Agile, which has automated unit and integration testing at its core (Code Magazine, 2008) and where automated tests are written in conjunction with, or before, the new code to be tested.

IEEE 829 Software Testing guidelines and terminology: Specified in (IEEE 829, 2008). This standard has been in use in testing since the 1980s and is still current and valid despite the enormous changes and evolution that the IT world has undergone. The reason for this durability could be that this standard seems to have been based on a simple core concept of what software testing is. Although reading through the sections of the full standard document might seem daunting, its core is expressed in page 6 Figure 2 showing its minimum set of documents (partial use of the standard) as *"test planning"*, *"test design"*, *"Test cases/test procedures and their execution"* and *"Test results reporting".* IEEE 829 is a documentation

standard, but because the documents it defines represent the high level tasks of testing it can be used as a test standard as well. Its endurance and wide acceptance are probably due to the simplicity of its core concept that the test process is divided into a planning/strategy/thinking part, then a preparation part, then execution then final reporting part. Such concept is intuitive and easy to remember and follow by test practitioners.

BS 7925-2 for component testing: A paper by one of the original authors of the standard (Reid S. C., 2000) provides an explanation of how it evolved and an outline of its contents. One of the main contributions of BS 7925-2 is that it lists a number of test design and test measurement techniques. The test process it defines does not seem much different from IEEE 829 except that in BS 7925 it is actually described as a process rather than a set of documentation. Both BS 7925 and IEEE 829 have for years been the most precise description of structured testing practices in the IT industry. However, IEEE 829 is a documentation standard and BS 7925 is a "component" test standard and both are not all encompassing for all aspects of testing in all environments. This is probably just a symptom of testing remaining as an "Art" as was discussed earlier in this chapter rather than a precise engineering discipline. Furthermore, BS 7925 usefully defines a set of test design and measurement techniques, but does not specify when each technique should be used. Therefore, even with BS 7925, the decision on how the test cases are selected and specified ultimately remains an experience based decision for the test analyst.

Test process improvement models:

A paper that comes firmly from industry (Steiner, Blaschke, Philipp, & Schweigert, 2010) provides an insight into the perspective of a well-known testing consultancy regarding three test process management, improvement and assessment models and how these map onto ISO/IEC15504 (ISO/IEC 15504), the widely accepted generic (not specific to testing) standard for IT process assessment. The paper considers whether and how three major test process models can be adapted to ISO/IEC 15504. The analysis and discussion contained in the paper is informative of how these models are applied and the efforts needed to apply them.

The paper initially considers the following three test process "models": 1) International Software Testing Qualifications Board (ISTQB-WEB, 2012) 2) Sogeti's Test Process Improvement model (Sogeti, 2012) and 3) The Test Maturity Model Integration (TMMi Foundation, 2012). The paper provides an explanation of each model and whether it can be adapted to ISO/IEC 15504. The paper then concludes that *"None of the current available test process assessment models meet the conformance requirements of ISO/IEC 15504"*. The paper then continues to explain how a "Test SPICE" (A *"Software Process Improvement and Capability Determination"* specific to testing), was developed by the authors to fill the gap of a test specific process assessment and improvement model that can be mapped onto ISO/IEC 15504. A "Test SPICE" simply means an interpretation of ISO/IEC 15504 that is specific to testing.

The paper can be quite difficult to read for someone who is not familiar with software development or lifecycle standards in general. It also uses many acronyms without explaining them. However, the overall content of the paper can be expressed quite easily using far simpler language. Effectively, the paper explains that the authors evaluated three common models for test process improvement and found that they are not feasible for adaptation to ISO/IEC 15504. The authors then created their own test process improvement model and tailored it to map easily onto ISO/IEC 15504. What is not explained in the paper, which is clearly intended for test consultancy professionals, is the significance of ISO/IEC 15504. ISO/IEC 15504 is a generic process assessment and improvement model. It is quite a sizeable standard and will require a lot of adaptation and interpretation effort before an organisation is able to achieve compliance with it. Its use in industry is to provide confidence that an IT brand, such as a testing consultancy, is capable of delivering consistent good quality to its customers. Consultancies need to demonstrate that they are compliant with ISO/IEC 15504 in order to provide confidence in their brand to their customers. What this paper is explaining is how such compliance can be achieved easily and with less cost by a test consultancy such as SQS (SQS, 2012).

ISO/IEC 15504 was originally derived, at least in part, from ISO/IEC 12207 (ISO/IEC 12207, 2008). According to the ISO website (ISO/IEC 12207, 2008) *"ISO/IEC 12207:2008 establishes a common framework for software life cycle processes, with well-defined*

terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are to be applied during the acquisition of a software product or service and during the supply, development, operation, maintenance and disposal of software products".

Both of the above standards are likely to be of interest to organisations building large scale systems for other clients or outside governmental organisations where they have a strong business case to invest in standards compliance as a way of providing formal certifiable evidence to the clients of the quality of their deliverables. The general area of IT standards is a specialised area that requires domain knowledge as well as knowledge and practical experience of how a specific standard is applied in real life. The paper discussed in this section (Steiner, Blaschke, Philipp, & Schweigert, 2010) seems to have been written by specialists in ISO/IEC 15504.

It is worth bearing in mind that standards such as ISO/IEC 15504 or 12207 only in fact address the "process" of software delivery or testing rather than the detailed technical aspects of how testing is done. Adherence to them does indicate a well-controlled and consistent process but does not guarantee that the resulting products are necessary of high "quality" as perceived by the end user.

Verification and Validation standard IEEE 1012: Described in (IEEE 1012, 2012) as a Verification and Validation standard that *"applies to systems, software, and hardware being developed, maintained, or reused [legacy, commercial off-the-shelf (COTS), non developmental items]"*. Searching through academic and research journals did not uncover any recent or significant literature relating to IEEE 1012 so it seems that it does not have a presence (at least an explicit direct presence) in the academic domain. However, it is worth noting that IEEE 829 refers to, or endorses, IEEE 1012 as the standard for Software Verification and Validation. Therefore, as long as IEEE 829 is in use in industry, IEEE 1012 remains (even if indirectly) a current standard relevant to testing practices.

Test standard ISO/IEC/IEEE 29119: An explanation of the new 29119 standard is provided in (Reid S. , 2012). The paper explains the purpose of ISO/IEC/IEEE 29119, what it covers, and its main components. According to the website used to provide updates on the

progress of the 29119 working group (ISO/IEC 29119, 2014) *"The aim of ISO/IEC/IEEE 29119 Software Testing is to provide one definitive standard for software testing that defines vocabulary, processes, documentation, techniques and a process assessment model for software testing that can be used within any software development life cycle"*. Also the website lists its five parts as "*Part 1: Definitions & Vocabulary, Part 2: Test Process, Part 3: Test Documentation, Part 4: Test Techniques*" and the fifth part as *"ISO/IEC 33063 Process Assessment Model for Software testing processes"*. How well this process will be adopted by industry is still too early to tell. It seems well structured and it also seems to address all dimensions related to testing therefore providing a "one-stop shop" of a standard. It does seem to be a strong candidate to become the new de facto generic standard in testing. On the other hand, precisely because it seems to be a comprehensive standard could be a factor that can act against its widespread adoption. The IT industry has many examples of standards or technologies that were adopted because they were simple and easy to apply at the expense of others that were comprehensive but more costly and more complicated. Could ISO/IEC/IEEE 29119 become such as example? This is yet to be decided. Furthermore, the economic climate (at least in the Western world) might also discourage organisations from investing in the adaptation and compliance with new standards and might be more attracted to less structured methodologies that are easier and cheaper to apply. These are the ideas that make this paper (Kasurinen, Runeson, Riungu, & Smolander, 2011) proposing a framework for assessing compliance with ISO/IEC/IEEE 29119 seem less likely to have a long term impact. Although it seems reasonably structured, what it is essentially proposing is an additional assessment framework around ISO/IEC/IEEE 29119. Assuming the paper's ideas have not already been superseded by the fifth part of the standard (ISO/IEC 33063 Process Assessment Model for Software testing processes), it seems to be proposing to ultimately create more complexity in the adoption of ISO/IEC/IEEE 29119 where what is needed is simplification.

The other question regarding ISO/IEC/IEEE 29119 is whether it will be capable, if it becomes widely adopted, of making a significant difference to the effectiveness of testing practices in general. This seems doubtful despite it being a well-structured and comprehensive standard that covers all processes of testing. It seems to be following a similar line to other previous and existing testing standards in the way it defines the process

of how good testing should be conducted. The difference it seems to bring is that it is comprehensive and well structured. However, it does not seem to address the purely technical test case design, selection and specification aspects differently from other standards which leads to the conclusion that its impact will be more noticeable around the test management and organisation rather than on the precision of identifying technical software faults.

The subject of standards in general and test standards in particular can be extended much further. In fact, a whole thesis can be devoted to the subject. So far the few key standards that are of most potential relevance to this thesis have already been outlined in this chapter. However, there are other standards that are of indirect relevance to testing which are worth mentioning, such as: BS 5760 on reliability, BS 61508 and ISO 26262 on electrical and electronic safety, ISO/IEC 9126 on the evaluation of product quality, IEEE 1028 on software review and audit, IEEE 610.12 on software terminology, IEEE 1008 on unit testing, NIST (National Institute of Standards and Technology) 800-42 on network security testing, IEEE 1220 on product and COTS re-use, BS 9241 on Ergonomics of human-system interaction, and ISO/IEC 20000 on Service Management/ITIL.

2.8 Different types and levels of testing

An overview paper on software testing (Machado, Vincenzi, & Maldonado, 2010) provides an easily readable overview of testing and is a good example of how an academic paper can explain practices in industry. It explains the importance of testing, key testing terminology, presents a simplified generic test process based on IEEE 829, the types and levels of testing, domain-specific test approaches, and finally it discusses different approaches to the key activity of test case selection. This paper is a useful introduction to software testing and it manages to simplify the complexities of software testing to the reader.

For example, in the section on "Types of Testing" the paper provides a simple but meaningful classification based on the types of the requirements of a system. It outlines just three types of testing: *"testing for functional properties"*, *"testing for non-functional properties"*, and *"testing for structural properties"*. The paper (through references to other chapters in a book

that was included in this literature review) seems to treat further subtypes as test strategies, e.g. model-based testing or specification-based testing as being functional testing strategies rather than different types of testing in their own right. As an overview, this classification seems appropriate because it explains the concept of different types of testing in an easily comprehensible way without risking confusing the reader with a much larger list of testing terminology.

The paper then explains different levels or phases of testing. In the first paragraph of the relevant section it explains: *"In the context of procedural software, the testing activity can be divided into four incremental phases: unit, integration, system, and acceptance testing"*. It then goes on to discuss the different levels and what testing is done during each level/phase. Again in this section the paper provides a clear and easily readable overview that manages to simplify the concepts while at the same time explaining them well. It also mentioned the context of the four incremental phases as procedural software, therefore explaining a typical structure of the testing activities without implying that it is the only or best structure for every type of testing.

The paper seems to succeed in explaining testing and testing concepts in a limited space, but it is worth noting that its level of detail does not explain to the reader how to practice testing in real-life.

2.9 Testing domains potentially relevant to this thesis

There are multiple "domains" or "classes" of test literature that are potentially relevant to this thesis that, despite using similar and familiar testing terminology, are in fact quite varied.

The list below summarises those "classes" into seven separate categories, citing examples for each:

1. Research work into code testing techniques linked to mathematical theory (Hierons, 2003) (Jia & Harman, September/October 2011) (Mattiello-Francisco, Martins, Cavalli,

& Yano, 2012) (Schatz & Pfaller, 2010);

2. Research work into object and model based testing (MBT) (Pretschner, et al., 2005) (Utting & Legeard, 2007) (Batteram & Romijn, 2003) (Neto, Subramanyan, Vieira, & Travassos, 2007), e.g. using Unified Modelling Language (UML) (Selic, 2006) (Baker, Dai, Grabowski, Haugen, Schieferdecker, & Clay, 2008) or Testing and Test Control Notation Version 3 (TTCN3) (ETSI, 2005);

3. Research work into automation of aspects of the testing process, especially the automated generation of test cases (Hartman, Katara, & Paradkar, 2007) (Boroday, Petrenko, & Groz, 2007) (Flores, Lucas, & Villanueva, 2008) (Lei, Liu, Morisset, & Li, 2010) (Castro, 2011);

4. Commercial and open source test tools, either for automating the running of test cases (Graham & Fewster, Experiences of Test Automation, 2012) (Holmes & Kellogg, 2006), or for test process management (TestLink, 2012) (Mantis, 2012) (Marathon, 2012), or both (Strasser, Mayr, & Naderhirn, 2010) (Collins & Lucena, 2010);

5. Commercial test methodologies and standards: These focus on the organisation and structure of test activities. Examples of this are the V-Model (IABG, 1993) (Clark, 2009) (Mathur & Malik, 2010), Agile test methodology (dos Santos, Karlsson, Cavalcante, Correia, & Silva, 2011),Test Driven Development (TDD) (Paugh, 2011), IEEE 829 Software Testing guidelines and terminology (IEEE 829, 2008), BS 7925-2 for component testing (Reid S. C., 2000), test process improvement models such as the Test Maturity Model (TMM) (TMMi Foundation, 2009) software life cycle standard ISO/IEC 12207 (ISO/IEC 12207, 2008), verification and validation standard IEEE 1012 (IEEE 1012, 2012) and the more recent test standard ISO/IEC/IEEE 29119 (ISO/IEC 29119, 2014) (Reid S. , 2012) (Kasurinen, Runeson, Riungu, & Smolander, 2011);

6. Project management methodologies that overlap with testing in certain areas. These are concerned with project management in general. Examples of this are the Capability Maturity Model (CMM) (SEI, 2012) and PRINCE2 (APM, 2012);

7. Communications protocols standards conformance test methodologies, produced by Telecommunications organisations such as ETSI's group for "Methods for Testing and Specifications" (MTS) (ETSI, 2012) or the ITU-T's Test Specifications Recommendations (ITU-T, 2012).

2.10 Discussion about testing domains relevant to this thesis

The first three categories (as outlined in Section 2.9) represent inherently precise approaches where the creation of test conditions, cases and the meaning of test coverage are according to pre-defined theories and rules. The second and third categories represent the areas where the academic research and the commercial practices overlap, for example by IT companies initiating or funding research projects or adopting methods and tools developed by academic teams.

The fourth and fifth categories are the most relevant to the industrial and commercial testing of large scale systems. These two categories represent the space that is relevant to much of the activities in the commercial IT domain. Some of the test automation tools may be specific to certain technologies, e.g. web enabled applications, but their use by itself cannot guarantee precise and effective testing. The commercially prevalent test methodologies and standards that were identified during the review are generic and aimed at organising and structuring the commercial test activities rather than achieving engineering precision of the test cases. They tend to be based on a concept of how the development or testing processes should be organised rather than on the basis of the structure or use of the systems to be tested.

The sixth category relates to the organisation and management of IT commercial projects. This category relates to the testing activities or testing phases as part of the overall project's lifecycle, i.e. in this category testing is viewed from a project management point of view. Project management standards and methodologies do not seem to have direct relevance to the precision and effectiveness of the detailed test cases.

The seventh category represents conformance and interoperability testing, which is inherently precise and effective because it is based on precisely defined communications protocol specifications. The precision of testing in this domain is an attribute that will be of great benefit if a way can be found to adapt its practices to the testing of large scale systems.

The use of modelling languages such as UML and, to a lesser extent, TTCN3 (second category) have been finding growing acceptance and adoption in industry, but still do not have the support of widely accepted industrial test methodologies and practices commonly deployed for large scale enterprise wide applications, with the possible exception of the Rational Unified Process (RUP) (IBM RUP, 2012). RUP is not widely practiced in industry outside development teams that are trained and equipped to use Rational tools (IBM Rational, 2013).

The fourth and fifth categories outlined in the previous subsection are the most relevant to the area of this thesis. Much of the test activities carried out in industry are primarily dictated by both the commercial test tools available, and the test terminology and standards (as discussed earlier in this chapter) that are accepted and understood across industry. However, these are generic tools, terminology and standards. Adopting them to organise the testing for a large scale IT project mean a degree of structure and formality to the testing carried out during an IT project but the effectiveness and precision of the testing are not guaranteed simply by adopting a particular test standard or a particular test tool.

Furthermore, there seems to be terms accepted and commonly used in industry that do not belong to any test standard, such as the use of terms such as Factory Acceptance Testing (FAT), End-to-End Testing (E2E) or even Smoke Testing. The plethora of terminology and the different meanings that can be attached to the same term creates a background where professional testing practice on large IT projects remains strongly dependent on the experience-based judgement of the test analyst rather than being an engineering discipline with precise rules and criteria. This is also due to a large extent to the way user requirements tend to be defined for large scale IT systems: high level, using natural language with the inherent tendency to having different interpretations, gaps and inconsistencies.

Based on the experience of the author, it appears that much of the test case specification work on large scale IT projects is done with no distinct test analysis and design phase. Instead, the test analysis will often start creating test cases derived straight from the requirements and/or the technical specifications. The availability and wide adoption of test scripts management tools, e.g. HP Quality Center (HP, 2013) encourages this practice, at least partly because it indicates to management that the test analyst is being more productive by producing detailed test cases that can be counted rather than embarking on a time consuming test analysis and design phase that may not have obvious visible "added value" to the overall project.

The recent test standard ISO/IEC/IEEE 29119 (ISO/IEC 29119, 2014) seems an important development for software testing in terms of its potential to unify a number of standards into one generic standard. It also seems to be aimed at covering all dimensions of test activities, from vocabulary, to process, documentation, techniques and assessment. However, it does not seem to propose to address domain-specific features. This, in the opinion of the author, is a gap that needs to be bridged before the IT industry can witness a noticeable evolution in the day-to-day testing practices for large scale projects such as the one used in the case study presented in this thesis.

2.11 Software testing roadmaps

The 2007 software testing research "roadmap" paper (Bertolino, 2007) is a summary and commentary paper on software testing research. Although it is a 2007 paper, further searches did not uncover newer material or other comparable papers with similar wide coverage of software testing that made the paper's ideas seem superseded or out-of-date. It seemed a good primer for new researches in this field. Examples of some of the paper's ideas are below:

<u>Regarding the need for domain-specific test approaches:</u>

In page 12 of the paper it states "*Research should address how domain knowledge can improve the testing process. We need to extend domain-specific approaches to the testing stage*". Although it later goes on to talk about test automation, this section of the paper

clearly promotes the idea that domain-specific testing needs further research work.

Furthermore, in the introduction section of one of the paper's references (Sinha & Smidts, 2006), an explanation for the need for domain-specific testing is provided: "*test models for model-based test automation techniques are created from software artifacts like requirements and document or design specifications of the software, and hence, these techniques overtly rely on the specification for completeness of the test models. These software artifacts are frequently underspecified because the user, who is familiar with the domain and defines the product requirements, may consider certain domain-specific requirements to be too trivial to be specified explicitly in the requirements document. The tester and developer may not have the necessary domain knowledge, thus may never realize that such a requirement is missing*".

In the Protocol Testing section:

The paper explains in page 6: "*software testing research could fruitfully learn from protocol testing the habit of adopting standardized formal specifications*". In the same way there has been increasing convergence between IT and communications technologies, there could be benefits to gain from converging the testing techniques of IT and communications, e.g. by adapting communications protocol testing techniques (Bochmann, Rayner, & West, 2010) to large scale software systems[1] [2].

In the section on Test Effectiveness:

The paper states in page 7: "*In particular, it is now generally agreed that it is always more effective to use a combination of techniques, rather than applying only one, even if judged the most powerful*". Although the discussion refers to the need to detect a variety of fault types, it also (at least indirectly) supports the idea that a number of approaches are needed to

[1] Chapter 4 shows how this might be done.

[2] The work in this thesis in general and the ideas in Chapter 4 in particular were neither prompted by nor relied on Bertolino's paper, but were retrospectively found to be supported by some of its statements.

effectively test a system[3].

In the "Controlling evolution" section:

The paper states in page 13: "*because of the high cost of regression testing, we need effective techniques to reduce the amount of retesting, to prioritize regression test cases".* Testing practitioners need to have objective as well as simple indicators of test effectiveness. Test prioritisation and reduced regression testing costs can be such key indicators of test effectiveness[4].

Finally, the paper explains in page 13 section 5.4 that "*The ultimate goal of software testing research, today as it was in FOSE2000, remains that of cost-effectively engineering "practical testing methods, tools and processes for development of high quality software*". It also states that "*The main obstacle to such a dream, that undermines all research challenges mentioned so far, is the growing complexity of modern systems*". The paper also states in page 15 in the "Education of software testers" section, that "*Research on its side should strive for producing engineered effective solutions that are easily integrated into development and do not require deep technical expertise*". The rest of this thesis will hopefully demonstrate, through a realistic case study, how the complexity of modern large scale systems can be simplified through the adoption of a domain-specific test framework.

Another "roadmap" paper presented in the same Future of Software Engineering (FOSE'07) conference (Lyu, 2007) offers further insight into the importance of developments in testing to other areas of software engineering, namely software reliability. The paper suggests that improvements in software engineering can be achieved by bringing testing research and software reliability research closer together (Lyu, 2007, p. 14): *"Software testing and software reliability have traditionally belonged to two separate communities. Software testers test software without referring to how software will operate in the field, as often the*

[3] This concurs with the idea in this thesis that each layer within the pyramid model (Chapter 4) needs its own test approach.

[4] This concurs with the ideas in the evaluation chapter (Chapter 8) and the simulation based analysis section within it.

environment cannot be fully represented in the laboratory. Consequently they design test cases for exceptional and boundary conditions, and they spend more time trying to break the software than conducting normal operations. Software reliability measurers, on the other hand, insist that software should be tested according to its operational profile in order to allow accurate reliability estimation and prediction. In the future, it will be important to bring the two groups together, so that on the one hand, software testing can be effectively conducted, while on the other hand, software reliability can be accurately measured". It could be that the way to achieve tangible improvements in the research and practices of software testing is to combine it with, and bring it closer to, other related disciplines in software engineering such as the area of software reliability.

2.12 Large Scale Systems Research in the UK

So far in this literature review only literature specific to testing has been cited and discussed. However, there is one particular systems research area that is relevant to this thesis that needs to be mentioned, namely the research in large scale IT systems.

There is a UK national research and training initiative in the science and engineering of Large-Scale Complex IT Systems (LSCITS, 2013). The overall aim of this programme as cited on its overview webpage (LSCITS Initiative Overview, 2013) is "*to improve existing technical approaches to complex systems engineering and to develop new socio-technical approaches that help us understand the complex interactions between organisations, processes and systems*". The description of the LSCITS research programme (LSCITS Research Programme, 2013) starts with the following explanation: *"We consider Large-Scale, Complex IT Systems (LSCITS) to be a mesh of different systems, each of which is a technical or a socio-technical system in its own right. Adding functionality to a LSCITS may involve developing some new software and composing this with newly-procured COTS and with existing systems. LSCITS may therefore be in a state of continual change, with systems and processes added and removed in response to changing organisational needs and ongoing technological advances. This state of continual change, where parts of the system have no control over changes going on, is the primary contributor to system complexity. It also means*

that the notion of there being discrete phases in the life-cycle of such systems is a significant over-simplification. Rather, there is a continuous cycle of procurement, development, deployment and decommissioning with component systems being regularly modified and replaced. As a consequence, there cannot be an over-arching system design process where the overall system is designed and developed top-down. Each phase in the procure-develop-deploy cycle includes a range of overlapping processes which interact with each other and with processes in other phases. Thus, procurement decisions influence system development and evolution, design decisions place constraints on how the system will be deployed, deployment issues affect future procurements, and so on".

The awareness of the LSCITS programme, supported by EPSRC, enforced the original idea of this research that there is a distinct class for large scale IT systems that has, or needs, its own engineering considerations which should also include testing[5].

Additional to the LSCITS in the UK, there is also an equivalent programme run by Carnegie Mellon Software Engineering Institute (SEI ULS System Research , 2012). Having reviewed the contents of both websites (i.e. LSCITS and SEI ULS), there seemed to be no distinct activities (at the time of last reviews) relating to the testing of such types of systems.

The nearest body of current research activities identified for and during this literature review that can be considered to be of potential and indirect relevance to this thesis were the research activities referred to in this paper (Riungu, Taipale, & Smolander, 2010) on Cloud testing. The paper's conclusion that further research is needed in the area of Cloud[6] testing enforced

[5] The term "large-scale systems" used in the title of this thesis was borrowed from LSCITS. The term "communications-critical" was inspired by the by the term "communications-enabled applications" (CEA) (Becchina, Ciccarelli, & Kenny, 2007) that was encountered during the literature search.

[6] Cloud Computing is defined by the NIST (National Institute of Standards and Technology) (NIST, 2011) as *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"*. The document then goes on to explain the essential characteristics, the service models and the deployment models of Cloud Computing.

the potential value of the work presented in this thesis. Although this thesis is in a different but potentially overlapping area to Cloud computing, the ideas it presents regarding the need for conceptual frameworks for testing that are based on the structures of the systems rather than the development processes seem to be similarly relevant to Cloud computing.

2.13 The Zachman Framework

According to the Gartner IT Glossary (Gartner, 2012), Enterprise architecture (EA) *"is the process of translating business vision and strategy into effective enterprise change by creating, communicating and improving the key requirements, principles and models that describe the enterprise's future state and enable its evolution"*. According to the ISO/IEC/IEEE 42010 website (ISO/IEC/IEEE 42010, 2012) , an Architecture Framework is the *"conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders"*. The website also provides a list of over 50 architecture frameworks (Survey of Architecture Frameworks, 2012)[7]. One of the most widely known and longest established Architecture Frameworks is the Zachman Framework (Zachmann, 1987) (Chen & Pooley, 2009).

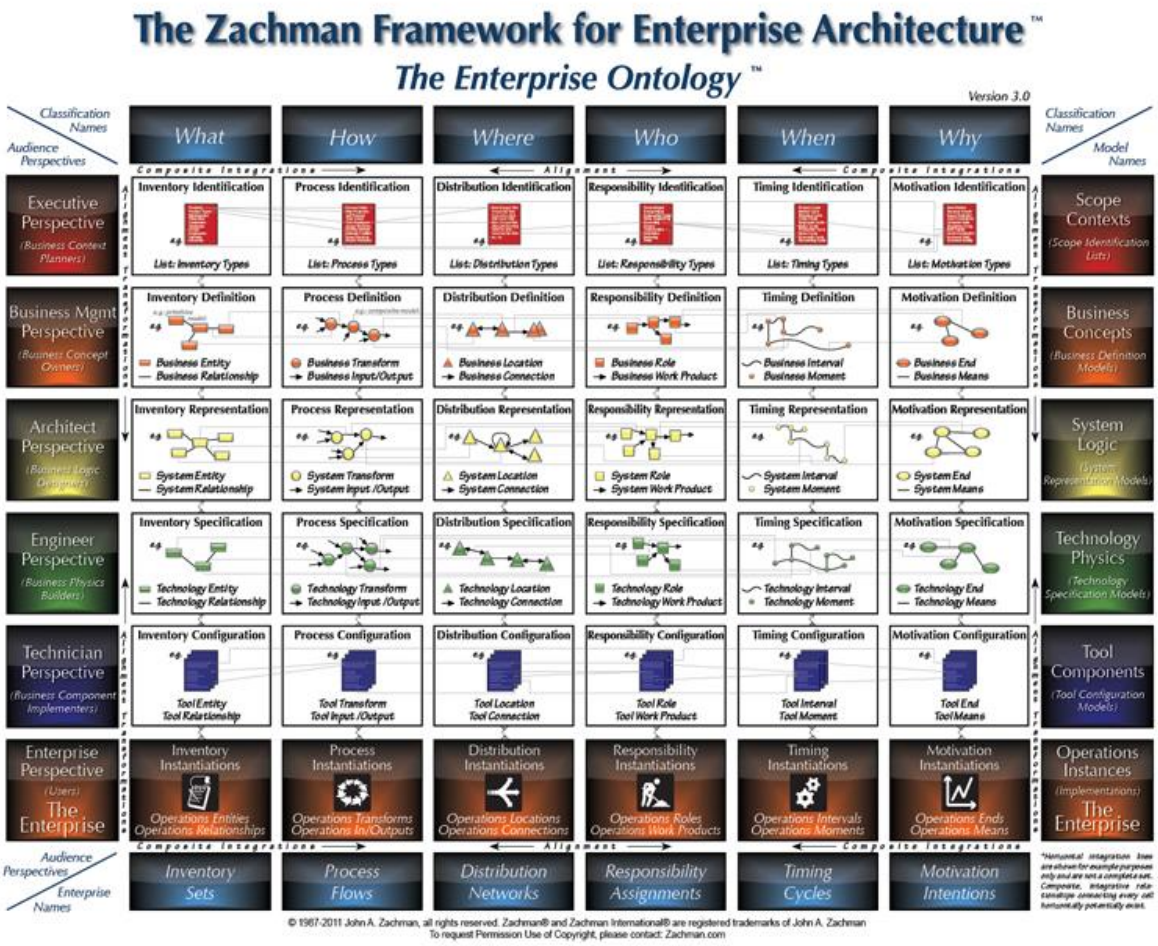
The official Zachman Framework website (Zachman International, 2012) described the Zachman Framework as a "schema" that is *"the intersection between two historical classifications that have been in use for literally thousands of years. The first is the fundamentals of communication found in the primitive interrogatives: What, How, When, Who, Where, and Why. It is the integration of answers to these questions that enables the comprehensive, composite description of complex ideas. The second is derived from reification, the transformation of an abstract idea into an instantiation that was initially postulated by ancient Greek philosophers and is labeled in the Zachman Framework™: Identification, Definition, Representation, Specification, Configuration and Instantiation"*.

According to the same website, the Zachman Framework *"typically is depicted as a bounded 6 x 6 "matrix" with the Communication Interrogatives as Columns and the Reification*

[7] Last updated 19 April 2012

Transformations as Rows. The Framework classifications are represented by the Cells, that is, the intersection between the Interrogatives and the Transformations. This matrix would necessarily constitute the total set of descriptive representations that are relevant for describing something... anything: in particular an enterprise".

The Zachman Framework's matrix (copied from the official website) is presented below:



No equivalent or comparable frameworks for testing were encountered during the literature search, i.e. one that is derived from the system's architecture and purpose rather than from a

concept of how the system is developed (e.g. V-Model). There seemed to be a need for what can be described as a "Test Architecture" framework[8] such as what this thesis proposes[9].

2.14 An example of testing practice in industry

A description of an example of common testing practices in industry is provided in (Martin, Rooksby, Rouncefield, & Sommerville, 2007). The paper shows how testing is done day-to-day in a small company which develops software products for mobile devices. It shows that testing is done somewhat ad-hoc to the outsiders but is based on experience of the testers and their knowledge of the product, the company, and the users of the products as well as the commercial realities for the company. These are all aspects specific to the company and the product that cannot be addressed by generic testing standards or methods. The paper sums up the differences between the academic research world and the day-to-day realities in industry. One of the conclusions of the paper was *"Our studies of W1REsys have convinced us that the agenda for software testing research has to be extended to address the relationship between the organization and the testing processes"*. It is worth bearing in mind that the paper was published in 2007 and, since then, the new testing standard ISO/IEC/IEEE 29119 has been published which covers the Organizational Test Policy, Organizational Test Strategy and the Test Management aspects of testing. However, small companies such as the one described in the paper (W1REsys) are less likely to be able or willing to invest the time and effort complying with a new standard unless such compliance was required (and ultimately paid for) by their customers.

The paper briefly touched on the subject of automated tests and how automation was being used in W1REsys. Automation was used in the company to check the results of acceptance tests, but most of the testing done was manual testing. The section in the paper on testing is concluded with the comment *"Our results …. < a reference to another paper deleted> …confirm that it is unrealistic to expect automated tests to fully replace manual tests"*. The company's testers were using automation in a limited but pragmatic way to reduce the effort needed for some of the repetitive and time consuming tasks.

[8] An example of such a test framework that is specific to communications-critical large scale systems (CCLSS) is presented in Chapter 4

[9] Starting from Chapter 4

Overall, one implicit thread can be deduced from the paper, that the case study described in the paper shows that testing in industry is shaped by pragmatic reasons and the realities of the company, or in other words, there are reasons beyond the purely technical why (as the paper concludes) *"there is a disconnect between software testing research and practice"*.

2.15 Significant areas of testing outside the direct scope of this thesis

To survey completely all the areas and branches of testing research would require a systematic review that would go, in terms of scope and the required effort, beyond the purposes of thesis. So far this literature review provided an overview of samples of recent software testing literature covering general software testing topics. It intentionally did not cover specialised software testing topics that are not of direct relevance to the rest of this thesis.

However, there are two key areas of software testing current literature that are of particular general importance that are worth mentioning briefly in this review, even though that are not of direct relevance to this thesis. These are: test automation and model-based testing.

Test Automation[10]:

An introduction by the guest editors of a special test automation edition of the Software Quality Journal (Budnik, Chan, Kapfhammer, & Zhu, 2011) refers to, by outlining the contents of five papers included in the special edition, the following recent trends in test automation research:

- **Test case generation:** The discussions in the two papers mentioned in the introduction (Papadakis & Malevris, 2011) and (Aichernig, Griesmayer, Johnsen, Schlatte, & Stam, 2009) demonstrate the complexity yet the importance and promise

[10] Providing a summary of the state of the art in test automation research would require its own literature survey and is outside the scope of this thesis. This subsection is intended as a brief reference to the area of test automation rather than an in-depth test automation related literature survey.

of automatic test case generation for furthering the effectiveness and precision of software testing in general. However, the discussions also demonstrate that such techniques are still not ready to apply for real-life industrial large-scale systems due to the lack of widely accepted and easy-to-adopt methodologies and tools.

- **The use of test oracles in the automation of unit and integration testing:** (Just & Schweiggert, 2011) demonstrates the viability of using test oracles, implemented as matrix transformations, to automatically determine whether the output of unit and integration tests for a graphical transformation system is correct. As with automatic test case generation, this is an important research area for improving effectiveness and precision of software testing but it is still a complex experimental area that requires further significant advances before it can be used widely in industry.

- **Deriving test plans from requirements expressed in natural language:** A method for deriving semi-automated tests from requirements (written in natural languages) is presented, along with a supporting tool called TORC, in (Guldali, Funke, Sauer, & Engels, 2011). The paper describes an acceptance test automation method which *"consists of three stages: annotation, clustering, and test plan specification"* and described how this method can be applied to a real-life case study. The paper demonstrates that the use of such a (semi) automated test method in the context described is feasible and can produce efficiency improvements in the testing of real-life systems. However, this is dependent on the adoption of an appropriate methodology, both for the requirements capture and the testing stages of a system's lifecycle, and the availability of appropriate automated tools.

- **Performance test automation:** The feasibility of using five Java GUI "capture and replay" tools for GUI performance test automation is discussed in (Jovic, Adamoli, Zaparanuks, & Hauswirth, 2010). The paper's stated objective is to "*determine whether the different tools are able to capture and replay realistic interactions on real-world applications"*. The overall findings of the papers are that *"most of the tools we study have severe limitations when used for recording and replaying realistic sessions of real-world Java applications, and that all of them suffer from the temporal*

> *synchronization problem"*, also *"that the most reliable test automation tool, Pounder, produces performance measurement results that are close to the performance of manually performed interactions"*. It seems that this area of test automation research, despite its limitations, is capable of delivering benefits to test effectiveness for real-life systems.

Needless to say, each of the four test automation areas outlined above is a research area in its own right. A reasonable general conclusion from the four papers cited is that test automation has realistic potential to deliver real-life test effectiveness improvements for large scale systems; however it is still currently encountering significant limitations and obstacles. In general, it seems difficult to visualise how testing practices in industry for large scale systems can become significantly more effective without improvements in test automation methods and tools contributing to such change.

A 2009 paper presenting an industry-based empirical study "*to shed light on the current situation and improvement needs in software test automation"* (Kasurinen, Taipale, & Smolander, 2010) concludes that *"the applicability of test automation is still limited and its adaptation to testing contains practical difficulties in usability"*. The paper then goes on to explain how the study was conducted using a number of industrial case studies and how the data was gathered and analysed. This paper is not an introductory paper on software test automation, although it provides some explanations about the basics of test automation, e.g. in the introduction section it explains that *"Automation is usually applied to running repetitive tasks such as unit testing or regression testing, where test cases are executed every time changes are made [reference removed]. Typical tasks of test automation systems include development and execution of test scripts and verification of test results. In contrast to manual testing, automated testing is not suitable for tasks in which there is little repetition [reference removed]"*. The value the paper offers to the reader is that it provides observations on the importance of test automation, how test automation is applied in industry, as well as the limitation and challenges associated with test automation. It also provides the reader with a starting point to familiarise with the area of test automation generally.

The paper provides analysis, data, explanations of the issues and challenges relating to test automation and concludes with some general recommendations, but no definite solutions. This is a reasonable outcome considering the complexity and variety of the various sub-areas of test automation that cannot realistically be covered in more detail within just one paper. Finally, it is worth noting that the companies used in the case study seem to all be based in Finland yet the paper does not discuss whether the findings of the study can be generalised to other countries' software testing environments.

Model-Based Testing:

A definition of Model-Based Testing and what it stands for can be found in the abstract of (Naslavsky, Ziv, & Richardson, 2007): *"Practitioners regard software testing as the central means for ensuring that a system behaves as expected. Due to the recent widespread adoption of model-driven development (MDD), code is no longer the single source for selecting test cases. Testing against original expectations can be done with model based testing that adopts high-level models as the basis for test generation"*. Beyond such explanation, MBT has many other sub-areas for research that the paper touches on such as how models are constructed, how test cases are generated and how traceability between the model and the test cases can be maintained ultimately in order to support *"tests result evaluation, coverage analysis and regression testing"*.

Additionally, a Model-Based Testing industrial user survey (Binder, 2012) defined Model-Based Testing (MBT) as *"The use of a software system that represents abstract aspects of a system under test to generate test cases"*. The survey appears to be well-conducted and was communicated to a wide variety of appropriate potential participants. It was also supported by an MBT user conference part-sponsored by ETSI (ETSI/Fraunhofer-Institute, 2011). The report presents a summary of the position and perceptions of MBT amongst specialist organisations and practitioners. A few key indicators are worth citing here are the responses to the questions: *"In your opinion, how do others view the overall effectiveness of MBT?"* and *"Overall, how effective do you think MBT has been?"* The response to the first question indicated that *"to the extent that other stakeholders are aware of MBT, they view it as either effective or neutral"*. Interestingly, amongst test professionals the percentage of respondents

answering *"effective"* was 56%. For the second question the summary explained that *"Three out of four respondents see MBT as either moderately or extremely effective. No respondent rated MBT as ineffective, in any degree"*.

Although this paper was produced by a commercial source, with the inherent possibility of bias towards MBT, it was supported and sponsored by an international standards organisation and the respondents come from an appropriate cross-section of forums interested in MBT. Therefore, its overall results should be worth taking note of as a useful indication of the adoption of MBT in industry. Overall, the paper shows that MBT adoption is likely to continue and to increase in the future especially as solutions and improvements become available to mitigate the limitation it cites relating to areas such as cost, process, tools/technology, modelling methodologies and training.

Another paper which provides further support for such a view is a paper describing how MBT is being used by the Microsoft's Windows Protocol Engineering Team to support its quality assurance processes (Grieskamp, Kicillof, Stobie, & Braberman, 2011).

The paper initially explains the importance and scale of the work of the Protocol Engineering Team. The paper then explains the proprietary methodology used to carry out quality assurance on Windows protocol documentation (for over 250 protocols), called Protocol Documentation Quality Assurance Process (PQAP) and describes how MBT is one of its "cornerstones". Description of PQAP and how MBT is applied within it are provided. The paper also discusses the MBT tool used to support the PQAP processes, called "Spec Explorer" (Microsoft Research, 2012).

In the Conclusion section, the paper states "*There are numerous instances of MBT being successfully used in the industry, but to the authors' knowledge, none has the scale of the one presented here. The magnitude of the current effort is expected to help MBT become main-stream in the software industry. The feasibility and scalability of MBT is evident in the fact that the project has delivered 'model to metal' test suites for over 75 protocols, and this number is growing. At the end of the project, around half of the 250 protocols in scope will have been modelled, reflecting an investment of over 50 person-years in MBT application*

alone. In addition a substantial investment has been made in tool development, based on a continuous feedback loop from the test-suite development process into the Spec Explorer development team. According to a preliminary statistical analysis, the application of MBT resulted in a 42% productivity gain when compared with traditional test suites in a site where similar numbers of requirements have been verified with each approach". Also in the conclusion *"With the aid of slicing, there is no technical reason why MBT should not scale to larger systems"*. The paper's last statement indicates that, subject to increased familiarity with MBT concepts and the wide availability of MBT training, *"MBT can become a mainstream technology in testing"*.

In the Related Work section (Section 6), the paper mentions a number of other MBT tools/toolsets and briefly compares them to Spec Explorer, such as: UniTESK toolset (UniTESK, 2006), TorX (TorX, 2006), TGV (TGV, 2012), Qtronic (Conformiq Qtronic, 2009), XRT (Microsoft Research, 2006), Java Pathfinder (JPF, 2012). It is worth noting that these tools seem to be suited for unit and conformance testing. Therefore their potential use in the context of large scale systems testing is, if any, likely to be limited to specialised areas such as precisely defined APIs and deterministic components of large scale systems.

Overall, the above cited papers seem to indicate that MBT is likely to continue to grow in industry and is likely to become the modern general "best practice" approach to software testing. The obstacles seem to be mainly: tester familiarity, training and better tools support before it can eventually replace the V-Model as a de-facto software testing methodology.

2.16 Conclusion

An early question that initially motivated this research was why, despite the availability of a number of well-established IT test standards and despite the significant investment that companies commit to IT testing, do many IT projects end up with late and poor quality deliverables. Is this phenomenon inherent in the IT industry? and can there be other systematic ways to improve the effectiveness of testing? (Scully, 1998) (Juristo, Moreno, & Strigel, 2006)

The literature review carried out over the duration of this research effort led to the idea that improvements in test effectiveness could be achievable via the adoption of new domain-specific test frameworks. The literature review also led to the general conclusion that, at least for large scale IT systems, the adoption of generic test frameworks and standards alone only provides confidence that the test process is structured, but it does not guarantee the outcome or the efficiency of the testing. Furthermore, the literature review did not uncover the presence of a widely known and widely adopted domain-specific framework for communications-critical large scale systems (as defined in Chapter 4). These were the ideas that initiated the work presented in the rest of this thesis, which is to define, apply and evaluate a test framework tailored specifically for communications-critical large scale systems.

3. Chapter 3: Research Methodology

3.1 Overview

The ideas that initiated the work on this thesis were evolved initially through the experience of the author in software testing[11] in industry generally and, more specifically, in the telecommunications software systems field. It was clear from the start of the work on this thesis that the research effort required to evolve and crystallise these ideas needed to happen within a real-life industrial context because that's where the ideas originated from, and that's where any "solution" is likely to emerge from and be applicable to. The research effort was initially intended to investigate how real-life industrial practices in software testing can be improved. There was also an awareness that some of the "answers" might be found in the testing theory and practices within the academic research, e.g. model-based testing, but that these might need to be adapted and applied to large scale non-deterministic systems. Furthermore, there was an awareness also that some practices, design frameworks and standards in telecommunications seemed to result in precisely engineered telecommunications systems which can, if suitably adapted (i.e. the practices, frameworks and standards), be potentially beneficial to the wider IT industry. Overall, there was a collection of ideas based on experience of real-life software testing that needed to be investigated methodically and developed further using a recognised research approach, with the intended outcome being a "solution" that will be relevant to real-life practices in software testing while at the same time being based on academically sound research principles.

Therefore, the research approach to adopt for this work had to be suitable for researching ideas related to real-life situations, it had to be flexible to allow the course of the research to be adapted according to how the ideas and the real-life circumstances might develop, yet it has to be sufficiently rigorous and formal for academic research purposes and for deriving

[11] One of four factors that support high-quality analysis according to Yin (Yin, 2009, p. 161) is the use of *"own prior expert knowledge"* in the case study.

academically reliable results and conclusions. Additionally, the nature of "data" was not determined at the earliest stages of this thesis, therefore the research method also had to be suitable for working with qualitative as well as quantitative data, as the needs arise. The options for research methodologies that can potentially fulfil such criteria can be narrowed down to the following broad categories (Runeson & Höst, 2009, p. 134):

- Surveys
- Experiments
- Action research
- Case study research

Surveys are appropriate for collecting then analysing raw data, therefore can be effective in researching existing practices, opinions and attitudes of participants. They are less appropriate for adapting ideas from one field, e.g. telecommunications, then applying them to another field, e.g. software testing. The research intended to devise or design a new approach to software testing, for this purpose surveys are not the appropriate research methodology. Similarly, controlled experiments could have been a feasible route for the research had it been known that the research could have been conducted in a controlled environment with a limited number of well understood variables. However, this might be either mutually exclusive with working within a real-life industrial context or at least would have limited the flexibility needed for this research work to proceed. Action research is another option and may well have been the most appropriate route for this research because of its "action" and "change" nature. However, there needed to be an industrial collaboration agreement in place that is designed to be the foundation for the research work and where the role and responsibilities of author are clearly defined from the outset as well the support and participation that will be offered by the industrial sponsor of this research. The absence of such an arrangement would have caused any subsequent action research effort to proceed with too much ambiguity and external dependencies for it to be correctly described as "action research". Whereas this research effort was intended to be improving and flexible, which related well to action research, there were other aspects of action research, particularly a collaborative agreement with an industrial sponsor (Somekh, 2006, p. 7), that were not available.

The above considerations pointed to a case study research approach because it allowed for the necessary flexibility and adaptation as well it being suitable for real-life investigations of the "big picture" of software testing which was closest to what this research was about. According to Yin (2003), a "case" is described as a *"contemporary phenomenon in its real-life context"*, also that a "case study" is *"inquiry"* where *"the boundary between the phenomenon and its context may be unclear"*. Such descriptions correspond well with the early intentions behind this research work, yet do not constrain its options and necessary flexibility. Additionally, adopting a case study research methodology need not exclude all other approaches to research, e.g. surveys, action research, or experiments because (potentially) elements of these approaches can still be incorporated within the case study research work.

Despite the flexibility and adaptability of case study research and its suitability for research within a real-life context, it remains an empirical type of study research that requires structure and definition and needs to follow defined research steps such as: a case study design, a defined data collection process, collecting evidence, data analysis and reporting of the outcome (Runeson & Höst, 2009, p. 137).

The remainder of this chapter is intended to present the case study methodology adopted and the research process decisions that shaped it.

3.2 Introduction

The core of this thesis is a case study based on a real-life IT project called FireControl (FiReControl, 2013). The initial ideas of the case study are independent of, the FireControl project. However, the FireControl project presented an opportunity to crystallise, expand, apply then evaluate what otherwise would have been fragments of conceptual abstract ideas.

The author of the thesis was a participant in the FireControl project and had detailed knowledge of all its functional and non-functional requirements[12]/[13]. This knowledge was acquired through several months of detailed analysis work in preparation for the project's acceptance testing phase. Through this knowledge and participation, the data collection and initial data analysis were possible, and the ideas presented in chapters 4 and 5 could be applied to a real-life CCLSS project. Further analysis and evaluation work continued well after the end of the author's involvement in the FireControl project.

The participatory aspect of the research work conducted on the FireControl project may point to it being action based research. However, the "data analysis"[14] activities were conducted by the author and continued well after his involvement in the project ended. The project was the source of the case study data and the author's participation in the project made access to, and understanding of, the data possible. Direct participation in the project, although helpful, was not an essential part of, nor a prerequisite for, the research activity.

The remainder of this chapter will be structured to reflect the case study research nature of this work. The following subsections explain the case study design adopted as well as other relevant aspects of the case study methodology. The chapter will use a notable foundation book on case study research (Yin, 2009) as a source of information on how to conduct and present good case study research. It will also use as a second source a more recent book discussing case study research in software engineering (Runeson, Höst, Rainer, & Regnell, 2012) which explains case study research in the context of software engineering and provides relevant example case studies.

[12] Permission was obtained by the author in 2008 from the senior management of the project to use the FireControl project's requirements for this thesis purposes.

[13] One of four factors that support high-quality analysis according to Yin (Yin, 2009, p. 161) is the use of *"own prior expert knowledge"* in the case study.

[14] The requirements of the project are treated as the "data" for the purposes of this chapter.

3.3 The research objective

As already discussed in Chapter 2 and summarised in the chapter's conclusion (subsection 0), there is a generally acknowledged problem in the IT industry with the quality and effectiveness of the testing practices on IT projects, which becomes more noticeable for the more complex projects. This thesis is intended to present new ideas that can lead to improvements in industrial testing practices, crystallise these ideas into a new test framework, provide one or more examples of how such ideas can be applied in practice to a real-life IT project, then analyse the evidence for whether "effective testing" has been achieved or can be achieved by the new test framework.

Therefore, the research objective can be expressed as the following:

"Design a new conceptual test framework for communications-critical large scale systems (CCLSS) that is capable of producing effective testing of such systems"

3.4 The case study questions

The ideas of the new test framework and the communications layer (as explained in Chapters 4 and 5) existed before the work on the case study started. The case study was used to further refine, apply and evaluate the ideas. The questions that were intended to be answered by the case study relate to whether the new framework's is applicable in real life, its benefits, whether it represents what was initially meant by a "test framework" and whether it can be used by other potential users for testing other CCLSS. Therefore, the case study questions can be expressed as follows[15]:

- Can the new test framework be applied (and if so how)?
- Are there benefits from the framework?
- Can the benefits be generalised beyond the case study?

[15] Should there be any variation between the list in this section and the evidence table in section 8.2, then the latter has precedence.

- Can the benefits be estimated numerically?
- Can the framework be compared to a rival?
- Can the framework fulfil the initial intended criteria?
- Can the framework be applied by other potential users/participants?

3.5 Purpose of the case study

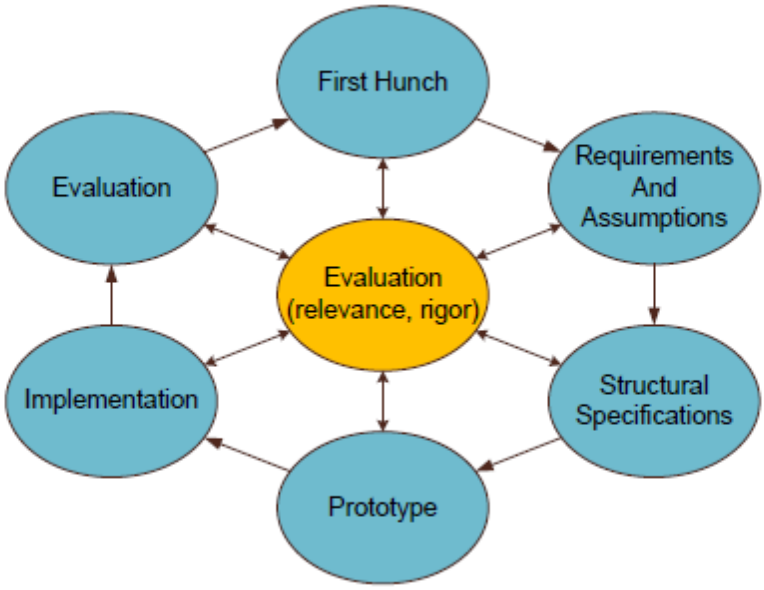
The FireControl case study was intended to evaluate how and whether 1) the new test framework (as explained in chapters 4 and 5) can be applied to a real-life IT project or system and 2) to provide evidence to determine whether the new test framework offers benefits and can lead to more "efficient" testing using a real-life CCLSS IT project. A key success criterion for the new framework is whether a set of coherent and usable outline test cases can be produced following the adoption of the new framework as basis for the test analysis and test design for FireControl.

3.6 Case study type

The work was prompted by ideas about what a new "efficient" test framework for CCLSS might be like; these ideas are presented in Chapter 4. These ideas then needed to be crystallised and expanded further, which was done in Chapter 5 for a key component of the overall framework (the communications layer). Finally, the ideas of the new test framework needed to be refined and applied in a real-life setting then evaluated (through a real-life case study) in order to determine whether the framework is feasible and useful for the original purpose it was intended, which is to deliver more effective testing of CCLSS.

Therefore, the thesis starts by developing a new conceptual test framework then uses the FireControl case study to apply then evaluate this framework. Due to this specific context, the case study can be considered an exploratory case study because it is used to explore the validity, applicability and benefits of adopting the new test framework. However, the overall intention was to design a new "effective" test framework. Therefore, with this context in mind, the case study type can also be considered as an improving case study.

Furthermore, whereas the case study is the central part of this thesis, the overall research process is also comparable to the design cycle discussed in (Verschuren & Hartog, 2005) and summarised by the diagram below as presented in (Boucharas, van Steenbergen, Jansen, & Brinkkemper, 2010, p. 20):



The thesis starts by identifying a need in software testing practices and theory for an efficient new test framework for CCLSS[16] in particular. The ideas about the new test framework were then developed and expanded further into a conceptual framework (the layered model). The conceptual framework was then developed further for one of its layers, the communications layer, as an example (or prototype) for how the conceptual framework can be used in practice. This is when the need for the case study became clearly established, which is to evaluate whether the framework's ideas, and in particular the communications layer, can be applied in practice.

The case study was used primarily to evaluate whether the framework's ideas were feasible in practice when conducting test analysis and test design for a real-life CCLSS project. The

[16] According to design cycle terminology, the new test framework can be thought of as the requirement for the design.

secondary purpose of the case study was to generate data and evidence to allow for the evaluation of the benefits of the framework.

The design research paper by Verschuren & Hartog (Verschuren & Hartog, 2005) was not written specifically for software engineering practice nor for designing conceptual methodology frameworks, but it can serve as a guide for how new ideas in software engineering can be evolved, applied and evaluated methodically and can be used as a link between software engineering research practices and other engineering disciplines. This discussion (within this subsection) is not intended to imply that this thesis was intended to adhere strictly to all the design research guidelines discussed in (Verschuren & Hartog, 2005), or that the research methodology of this thesis is pure design research with no theoretical basis or support. Rather, the structure and flow of this thesis follows a pattern comparable to that depicted in the diagram above where the conduct of the case study in itself is part of the evaluation efforts. In other words, rather than the research methodology of this thesis being pure "theory-based case study research" which starts with a theory then a case study is used to collect and analyse data which in turn will prove or disprove the theory, it is more accurately described as "design-based and practice-inspired case study research" which follows a pattern comparable, or akin to, that depicted in (Boucharas, van Steenbergen, Jansen, & Brinkkemper, 2010, p. 20).

Having clarified the research process, the remaining subsections within this chapter will explain the case study methodology used.

3.7 The FireControl case study

The case study part is based on the Statement of Requirements (SoR) of a real-life IT project, FireControl (see introduction in section 6.1). Although it is a single case study, the ideas of were applied to the requirements of five different communications interfaces of the (then planned) FireControl system. This was done to increase the reliability of the observations and to increase the likelihood of being able to derive meaningful generalisations.

The new test framework presented in the thesis is a conceptual framework which needs to be applied in real-life in order for it to be tested and evaluated. The FireControl project was selected as the case study "real-life IT project" because it fitted well with the definition of a communications-critical large scale system and because its requirements, although sizeable and complex, were readily available for use with the new test framework as the "case study data". A subset of the requirements (the technical communications requirements) was used as the "input data" into a test analysis and design activity that instantiates the ideas of the new test framework[17].

Any approach other than the case study approach would not have been suitable to evaluate the applicability of the new test framework to real-life IT projects. A quantitative approach in the absence of a real-life case study could not have been sufficiently meaningful because the ideas of the framework are conceptual, hence would not be appropriate to analyse or evidence using a precise numerical approach. On the other hand, a purely qualitative approach would be highly subjective because the new test framework was initially developed from an expert's view of the CCLSS structure and components. The case study approach allows the flexibility of combining both qualitative as well as quantitative approaches as suits the circumstances and the needs of the research effort.

To strengthen the reliability of the findings, "input data"[18] from five different FireControl communication interfaces, instead of just one, was used. The ideas of the new framework are applied to five different but comparable sets of data (explained in chapter 6) to improve the reliability of the case study outcome and conclusions. Additionally, a subset of communications requirements from another CCLSS project is also used as part of the evaluation of the framework by external participants, presented in Chapter 7.

The use of the FireControl project as the basis for the case study was intended to translate the ideas (or more accurately, a subset of the ideas as is done in chapter 6) of the new CCLSS test framework to a tangible form and, if possible, compared to a "rival theory" in testing. The context of this work is outlined in chapter 1, which is: testing theory and practices common in

[17] Specifically for one its layers: the communications layer.

[18] i.e. the requirements

the IT industry in general and more specifically, for the purpose of this thesis, for communications-critical large scale systems, have not kept up with the complexity and criticality for such class of systems.

3.8 The case study design

As explained by Yin (Yin, 2009), a well-defined case study design should help the researcher avoid a common weakness often encountered in case study research, which is failing to maintain a clear chain of evidence between the main elements of the case study research, namely: the case study questions, the data collection, the analysis, the evaluation, conclusions and generalisations.

This section and the remaining subsections are intended to explain the case study design applied. It will discuss the context and perspective of the case study, explain further why the particular case study was selected, what is being studied by the case study, the hypothesis being examined, and other aspects that together form the overall case study design[19].

As already explained, this thesis starts with a number of ideas regarding a new test framework that is specific to communications-critical large scale systems. The need for such a framework and the source of the ideas on how to formulate it were based on the understanding that testing practices in the IT industry for such a class of system tend to be inefficient and that a domain-specific approach is more likely to result in improved efficiency than generic methodologies and approaches often adopted in industry, a key example of which is what is referred to as the "V-Model" (see Chapter 2). The thesis then uses a case study based on a real-life IT project (FireControl) to apply and evaluate the ideas regarding the new test framework.

[19] No formal case study protocol (Yin, 2009, p. 80) (Runeson & Höst, 2009, p. 37) was maintained, however the research progress and direction was considered carefully throughout with the intention to follow good research practice, as the subsections in this chapter will explain.

3.8.1 How the case study was conducted?

A more detailed explanation about the use of FireControl requirements for case study purposes and the initial preparation of the requirements "data"[20] is provided in Chapter 6, subsections 6.1, 6.2, and 6.3.

However, as a summary for the purpose of this chapter, the case study was conducted as follows:

- Identification of all the FireControl requirements that represent the "communications layer" according to the new test framework.

  The new test framework proposes a conceptual model which consists of six layers (i.e. categories) of requirements. One of these six layers was used to apply the ideas of the framework in more granular detail and to take the test analysis and design work based on the test framework to a stage when the outline test cases are defined for the communications layer of FireControl.

- Once the communications requirements were identified as a separate group within the overall Statement of Requirements of the FireControl project (FireControl project, 29 March 2007), they were treated from then on as the FireControl communications layer.

- The communications requirements were then categorised further according to the communications interfaces of FireControl, these are: Firelink interface, Secondary Radio Bearer interface, Telephony interface, WAN interface and LAN interface[21]. Another group was used to group together requirements that relate to the

[20] In the context of this thesis, the preparation of data means the classification of the FireControl requirements into categories and subcategories to align them to the new test framework. See diagrams B.1-B.5 in sections 6.4 to 6.8.

[21] These five communications interfaces can be thought of as the sub-units of analysis of the case study

communications layer but do not have a specific category, e.g. GPS data[22].

- The FireControl communications requirements representing the communications layer became a subset of the case study data which was used to apply, explore and evaluate the framework's ideas. The individual requirements were therefore "the units of data" used in the case study.

- The requirements categorised under each of the five communications interfaces to the FireControl system were then further sub-categorised and ordered according to the communications layer's proposed sub-categories (as described in chapter 5).

- These subcategories, and the subset of requirements that each of them contains, were then be used as the basis for the test analysis and test design of the "outline test cases" needed to test the interface. The output of this exercise and of the case study in general, is presented in one of the appendices (Appendix 2).

- The requirements of five interfaces were mapped onto a logical (graphical) template of the subcategories, using one diagram for each interface (sections 6.4 to 6.8), to assess whether the subcategories were applicable consistently to the communications subcategories for the five different interfaces[23].

- Further evaluation work, primarily using randomly generated sets of requirements where appropriate, was then carried out (Chapter 8) to consider whether the adoption of the framework's ideas for the test design and analysis for the communications layer produced some or all of the expected benefits and whether the outline test cases that resulted after applying the framework's ideas represent efficient testing.

[22] The case study will proceed with further analysis and evaluation work based on the five main interfaces and not used the sixth "none of the above" group because it contains requirements that are standalone and do not belong in the main communications layer categories.

[23] According to (Runeson, Höst, Rainer, & Regnell, 2012, p. 68) this step can be thought of as a form of "pattern matching" type of data analysis.

- Additional "summative"[24] evaluation involving a small sample of external participants was also conducted. The external participants were IT professionals with an understanding of testing and who have experience working on large scale IT systems and projects. The external participants were asked to complete an evaluation form in their capacity as "potential users" of the new test framework. The aim of the evaluation was to deliver feedback from potential users of the framework on 1) whether it is feasible to use as a test framework 2) whether it can deliver benefits to the test effort 3) whether its ideas, such as the 19 test subcategories, can be applied to other communications-critical large scale systems other than the one used for the case study part of this thesis.

- One of the participants in the evaluation mentioned above also mapped two sets of communications requirements to the nineteen test subcategories. One set was the FireControl Firelink requirements; the other set was a sample set of requirements from another CCLSS project referred to as "CS2". This was done to observe how the external participant was able to map requirements to the nineteen test subcategory, to compare the participant's mapping with the mapping already carried out as part of the FireControl case study and also to evaluate the applicability of nineteen subcategories to requirements from another CCLSS (i.e. another case study). See Chapter 7 and Section 8.9.

3.8.2 The broader theory behind the thesis

The thesis starts with an idea regarding a new test framework for communications-critical large scale systems, which is presented in Chapter 4. The thesis then focuses on one layer[25] of

[24] An alternative type of evaluation is the "formative" evaluation. The difference between summative and formative evaluation is explained in (Verschuren & Hartog, 2005, p. 741) as follows: *"If the client tastes the soup this is summative, and if the cook tastes the soup this is formative evaluation"*. In effect, according to (Verschuren & Hartog, 2005) all other evaluation provided in the thesis constitutes "formative" because the evaluation was carried out by the author.

[25] Which can be thought of as the "unit of analysis" of the case study.

the framework and provides further definition to how the test analysis and design effort for this layer can adopt the ideas of the framework. This is done in chapter 5. The ideas of the new framework are subsequently applied to five "subunits of analysis" of the communications layer of FireControl as is presented in the main case study Chapter 6.

In general terms, the broader theory behind the thesis is that a simplified conceptual model of the logical components of a large scale system can be an appropriate and efficient basis for its test analysis and design. This is opposed to generic test methodologies and standards that are based on a concept of the development cycle of systems (e.g. waterfall). This will be discussed in more detail in Chapter 4.

3.8.3 What evidence is needed?

It is important that a "chain of evidence" (Yin, 2009, pp. 123, 127) is maintained in a case study to ensure that the fundamental elements of the case study, i.e. the research question, the data collection and analysis, the evaluation of the outcome and the conclusions, remain aligned and do not drift away from the original research question.

The fundamental evidence needed to support this thesis is that the proposed new test framework is feasible and can be adopted as an alternative framework for testing a communications critical large scale system. In other words, can it be shown through the application of the thesis ideas to the case study that a meaningful and adequate set of outline test cases can be derived? This will represent the first and fundamental level of evidence needed to support the initial theory regarding a new conceptual test framework for communications-critical large scale systems.

However, because the evidence will be derived from a real-life IT project case study, it will inherently have a degree of subjectivity (actual or perceived) because of the nature of case study research using real-life scenario as well as the inherent creative nature of test analysis and design for large scale non-deterministic systems. In other words, there will have to be a trade-off between the "realism" benefits gained from using a real-life case study of a large

scale IT systems against the reality that the research, analysis and the resulting evidence cannot be as precise and rigorous as it would be in, for example, quantitative research in experimental sciences. Rather than attempting to establish a single piece of evidence that will prove or disprove the initial theory behind the research, mounting and incremental evidence needs to be combined and considered together to evaluate the new test framework and, where possible, try to combine qualitative as well as quantitative methods to obtain the evidence.

Therefore, should the case study and the resulting outline test cases be able to establish that the adoption of the new test framework is feasible in real-life, further evidence is needed to strengthen the reliability of the ideas of this thesis in order to offset any perceived or possible weakness due to the evidence emerging from a single case study (Runeson, Höst, Rainer, & Regnell, 2012, pp. xiii,xiv) due to the imperfections which a real-life case study might entail.

Therefore, further "levels "of evidence will be investigated in this thesis to further strengthen the reliability of its propositions, namely: evidence that the framework can be applied to other equivalent systems and not just usable for the specific case study, and evidence that the adoption of the framework can be shown to lead towards more efficient testing.

To achieve the intended additional evaluation, a comparison between the framework and a "rival theory" will be made. This thesis is not intended to be critical of other rival theories, hence the intention of seeking this type of evidence is merely to evaluate whether the framework can lead to "effective testing" even when compared to another widely adopted "rival" theory. This will be done in Chapter 8. The idea of numerically measuring effectiveness relates to the concepts of "effect measurement" and "effectiveness assessment" as discussed in (Verschuren & Hartog, 2005, p. 741). The idea of comparing a theory behind a case study to a rival theory is discussed by Yin (Yin, 2009) as a way to improve the reliability of the findings of case study research.

Lastly, an evaluation exercise involving participants external to this research work was conducted in the form of a "goal-free", "ex post", "summative evaluation"[26] by potential users of the new test framework. The participants were asked to complete an evaluation form asking them to comment on the feasibility and benefits of adopting the new test framework for the testing of communications-critical large scale systems/projects they are familiar with or have worked on in the past.

In conclusion, the evidence that will be evaluated for this research can be summarised, in general terms[27]:

- Evidence that the framework's ideas can be applied to a real-life IT project to carry out initial test analysis design activity and to generate a meaningful set of outline test cases.
- Evidence that it is possible to apply the ideas to other communications-critical large scale systems and not just to one specific system.
- Evidence of test efficiency benefits derived from using the framework in the case study, both qualitative and, ideally, quantitative as well using (for example) randomised simulation where possible.
- Evidence that the framework's ideas can lead to improved test efficiency when compared to a "rival theory".
- Evaluation and feedback carried out by potential users of the new test framework that were not otherwise involved in this research work and commenting on its potential applicability to other communications-critical large scale systems.

[26] "Goal free", "ex post" and "Summative" are terms adapted from (Verschuren & Hartog, 2005). The terms describe the research methodology context of this additional evaluation work. "Goal free" refers to nature of the evaluation because the participants will be primarily asked to provide their general feedback and opinions to open ended questions. "Ex post" refers to the fact that the evaluation was carried out after the new framework was formulated (for the purposes of this thesis only, as there remains potential for a lot more further work). "Summative" refers to the fact that the evaluation is being carried out by participants that were not otherwise involved in this research work, representing potential users of the framework.

[27] This list will be specified more precisely in the evaluation Chapter 8, specifically in Section 8.2. Should there be any variation between the list in this section and the evidence table in Section 8.2, then the latter has precedence.

3.8.4 The case study data

The "data" for the case study is the set of requirements for the FireControl project[28], specifically the communications requirements. The case study applies and refines the ideas of the new test framework using these requirements then generates its own "output" data of the outline test cases (Appendix 2).

One of the activities needed to make this FireControl case study usable was to identify the requirements that represented "the communications layer" according to the layered model of the new test framework. Doing so involved reviewing all requirements one-by-one to identify the set of requirements that were communications requirements rather than IT requirements. The fact that this activity was possible and a set of communications requirements could be identified from the overall set of requirements (over 2000) was by itself a positive initial indication that the framework's ideas could work with the FireControl project. Subsequently, these requirements were classified according to the five major communications interfaces of FireControl, thereby five "subunits" or "sub case studies" of the main FireControl case study were defined to allow for the new framework's ideas to be applied to more than one set of data, thereby allowing for better reliability of the observations and conclusions of the case study.

3.8.5 The case study data analysis

As was discussed earlier in Section 3.6, the case study was carried out to evaluate, in its own right, (as well as refine and apply) the feasibility of the new test framework's ideas.

According to case study terminology, the analysis carried out on the case study data consists of applying the new test framework's ideas to the communications requirements during the test analysis and design phase for FireControl. The raw or "input data" for the case study was the FireControl requirements, then a communications set of requirements was identified to

[28] According to (Runeson, Höst, Rainer, & Regnell, 2012, p. 48) the FireControl data source can be thought of as "third degree" data source type "*where the researcher independently analyses the work artifacts*".

represent the "communications layer" of FireControl to prepare the data to be used for investigating the theory proposed by the thesis. Once the "communications layer" was identified, it was subdivided according the communications interfaces of FireControl resulting in five "subunits" of the case study. For each of the subunits, the requirements were ordered according to 19 subcategories of the communications specific test approach (presented in Chapter 5). For each of the five interfaces, the ordering of the requirements according to the nineteen subcategories was used as the foundation that informed how the outline test cases were defined for each of the interfaces. The outline test cases were then reviewed to evaluate whether they represent an acceptable test analysis and design effort and to evaluate whether the adoption of the framework's ideas were feasible and productive. Most of this work will be presented in the main case study chapter (Chapter 6). In Chapter 6, the initial focus will be on the first communications interface of FireControl before it is developed further into the remaining four. The first communications interface of FireControl is in effect treated as a "pilot" (Yin, 2009, p. 92) for the case study before the analysis is continued for the four other interfaces to identify whether the new test framework's ideas could be applied consistently across the five different interfaces.

Further qualitative and quantitative analysis on the requirements, their ordering according to the nineteen subcategories and the resulting outline test cases, will be presented in the overall evaluation chapter (Chapter 8) to investigate further evidence regarding the benefits of the framework.

It is worth noting that the initial and more fundamental evidence for the validity of the new test framework's ideas will be whether it was feasible and productive to apply its ideas to the FireControl requirements to generate an "efficient" set of outline test cases. This evidence will be integral to the main case study chapter (Chapter 6), which will then be developed further in the evaluation chapter (Chapter8) by evaluating in more detail specific benefits of applying the framework's ideas, specifically the communications layer test approach, to the communications requirements of FireControl. The first part of the evaluation will be in the form of qualitative analysis which will then be further developed using simulation based

analysis in the second part of the evaluation chapter[29]. The simulation based analysis will initially compare a key test "efficiency" factor of the framework against randomly generated "data" sets, and then it will be developed further to compare the same efficiency factor of the framework against a "rival" test theory.

3.8.6 Case study outcome

The FireControl case study was expected to provide an answer regarding the viability or otherwise of the proposed new test framework. This answer will primarily be in the form of a set of outline test cases derived from a test analysis and design effort based on the new test framework. As discussed in in the earlier subsections of this chapter, to avoid relying solely on one type of qualitative evaluation (based on expert opinion) of this outcome, further evidence was sought in the form of identifying specific examples of the benefits derived from adopting the framework, checking the consistency of the observations across more than one communications interface from the main case study, and also using numeric simulation-based comparisons of the "efficiency" of the framework when compared with randomly generated "data" (sets of requirements) as well as a rival theory. The simulation will be presented and explained further in the evaluation chapter (Chapter 8). Finally, an additional user-based evaluation of the framework by external participants was carried out, presented in Chapter 7 and cited where relevant in Chapter 8.

3.8.7 Evaluation

Ultimately, the evaluation should be traceable back to the case study questions explained earlier in this chapter (Section 3.4) which are in turn traceable to the research question/objective (Section 3.3).

[29] An explanation of the simulation analysis, the test efficiency factor that will be the basis of the analysis and the rival theory will be included in the evaluation chapter (Chapter 8).

Therefore, in broad terms, the evaluation approach will be determined by seeking answers to the following questions[30] presented in Section 3.3:

1- Can the new test framework be applied (and if so how)?
2- Are there benefits from the framework?
3- Can the benefits be generalised beyond the case study?
4- Can the benefits be estimated numerically?
5- Can the framework be compared to a rival?
6- Can the framework fulfil the initial intended criteria?
7- Can the framework be applied by other potential users/participants?

The answer to question 1 will be determined by the success (or otherwise) of the case study in demonstrating that the framework's ideas are viable for at least one of its layers. Viability in this context means whether the mapping of the requirement to the framework's categories and subcategories was feasible and whether it led to (a) a meaningful categorisation of the case study requirements (i.e. the case study data), (b) a successful test analysis and design effort which resulted in useable sets of outline test cases.

The answer to question 2 will be determined through a review of the outcome of the case study effort in general and the "output data" (the outline test cases in Appendix 2) in particular to identify if and where expected benefits may have materialised.

The answer to questions 4 and 5 will be determined through the use of simulation based analysis to compare the performace of the framework against randomly generated orders of data (i.e. a set of requirements) as well as simulated data based on a "rival theory". The comparison will be based on a key test "efficiency" indicator of "re-testing" overhead, as will be explained in more detail in Chapter 8.

[30] Also see the discussions in the early sections of Chapter 8, particularly Section 8.1, which will discuss in more detail the rationale behind the evaluation approach for this thesis.

The answer to question 6 will be determined by responses from external particpants regarding a list of initial criteria relating to the form of a test framework as presented in Chapter 4. The question, the list of criteria and the responses will be presented in Chapter 7.

The answer to questions 3 and 7 will be determined by feedback provided via a user-based evaluation form from external participants who represent "potential users" of the new test framework as will be presented in Chapter 7. Question 3 will be further supported by evidence from applying the new framework's ideas to multiple communications interfaces of FireControl.

The explanation within this subsection is intended to provide a snapshot of the overall evaluation approach within this thesis. The details and explanations regarding the overall evaluation of this thesis are presented in Chapter 8, with some of the evidence presented in Chapter 7.

3.9 Validity

This section summarises construct validity, internal validity, external validity and reliability (Yin p.41) considerations for this thesis.

Construct Validity

Three tactics are mentioned by Yin (Yin, 2009, p. 41) as useful for improving the construct validity of case study research, these are: (1) using multiple sources of evidence, (2) establishing a chain of evidence, and (3) having key informants review the draft case study report. All of these three tactics have been included in this work. For the first tactic, five different communications interfaces of FireControl were used to apply the ideas as presented in Chapter 6, then requirements from a second CCLSS case study were used in the evaluation carried out by an external participant, as presented in Chapter 7. For the second tactic, a chain of evidence is established linking the research objective to the case study questions in this chapter, which are then linked to the evidence sought and the overall evaluation carried

out in Chapter 8. For the third tactic, key "informants"[31] reviewed a draft paper[32] describing the ideas of this thesis then carried out an evaluation of the new test framework, as presented in Chapter 7.

Internal Validity and Reliability

The work carried out for the case study was defined very specifically and narrowly (i.e. requirements-based testing, domain-specific framework for the analysis and design of the outline test cases), the "data" sets (i.e. the FireControl requirements) that are within the scope of the work were identified from the outset. The main threat to the internal validity and reliability of this work could arise if key sets of communications requirements that were fundamental to the test analysis work were missed during the early searches that identified the "communications layer" of FireControl. Another threat is if selection of the "dependency sets" used in the simulation based analysis had significant errors or omissions that could lead to incorrect conclusions and inferences. Appendix 3 lists the requirements included in each of the communications interfaces and lists the dependency sets for each of the interfaces, therefore this work is visible and possible to review again if necessary. Overall, such internal threats to validity are an inherent feature of real-life case study research[33]. These threats were mitigated by: (a) using different types of evidence, both qualitative and quantitative, (b) by using and comparing[34] the outcome of five different communications interfaces (i.e. the five sub-units of analysis), (c) by comparing the new test framework's "efficiency" to a "rival theory" as will be presented in Chapter 8, and (d) by involving external expert participants in the evaluation as will be presented in the summative evaluation chapter (Chapter 7).

External Validity

A discussion about the applicability of the ideas presented in this thesis to other systems is presented in the final chapter (Chapter 9). The fact that the communications layer's test

[31] Described in Chapter 7 as "expert external participants".

[32] Included in CD supplement No.2.

[33] Especially for explanatory case studies (Yin, 2009, p. 42).

[34] Including using what can be described as pattern matching via diagrams B.1 – B.5 in Chapter 6.

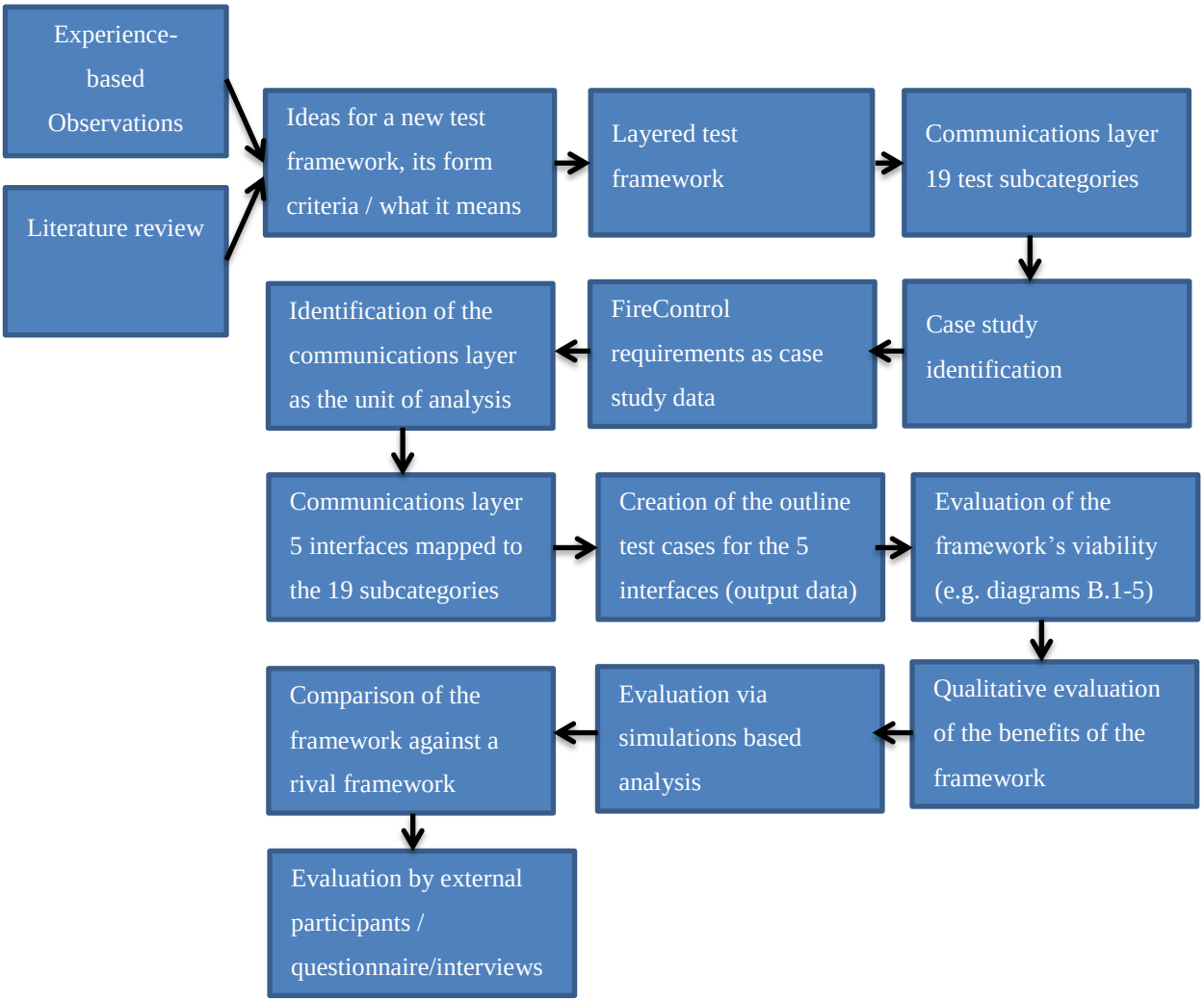
approach was applicable, without modifications, to the five different communications interfaces provides a degree of confidence that the ideas may be applied to other systems. Additionally, the applicability of the communications layer's test approach was tested further by having one of the external participants apply the test approach to a sample of communications requirements from a second CCLSS, as a second (smaller) case study presented in Sections 8.9.2 and 7.5, as well as replicating the Firelink requirements work as presented in Section 8.9.3. There is an implicit assumption that the applicability of the framework is dependent to a large extent on the way the requirements are defined. The closer the style of the requirements of another system to FireControl's requirements, and the similarities in the structure between the two systems, the more likely the framework and the ideas presented in this thesis can still be valid.

3.10 Further discussions regarding research methodology

Further discussions and reflections on the use of case study methodology for the type of research presented in this thesis, as well references to other comperable research will be included in other chapters where most relevant, e.g. Chapter 8 in general, and in particular sections 8.1.

3.11 Summary diagram

To provide a condensed visual view of the contents of this chapter, the diagram below depicts the main logical stages of this thesis work and how the FireControl case study fits within the overall thesis structure.



4. Chapter 4: The proposed new test framework

DISCUSSION OF IDEAS ON HOW A SOLUTION COULD BE FORMULATED, AND WHAT THE TERM "EFFECTIVE TEST FRAMEWORK" MEANS WITHIN THE CONTEXT OF THIS THESIS

4.1 Introduction

Chapters 1 and 2 of this thesis already discussed the current theory and practices relating to testing generally, and specifically to testing communications-critical large scale systems. The two chapters pointed to a need for increased precision in the way tests for a large scale communications-critical system are defined. Generic test methods that are widely understood and practiced, e.g. V-Model (Ammann & Offutt, 2008), leave the important decisions about what should be tested and how it should be tested to individual testers working on the project during a relatively late stage. This is because the test case specification work is often done towards the end of the test preparation cycle. This feature need not be a problem if the system under test is precisely defined from the outset. However, for a large scale system that is not precisely defined from the outset this means that a complete view of what will be tested will only be available in the fragments of the detailed test case specifications. This effect is compounded further for large scale complex systems when detailed design decisions are not, or cannot be, defined early in the project's lifecycle, hence also leaving important decisions about the system's detailed functionality to the individual developer during the implementation phase. Such practice is the IT project equivalent to allowing the detailed design of a building to be decided whilst bricklaying is being done. This clearly needs to be avoided if a project is to succeed in delivering a predictable well-tested system on-time and within budget.

For large scale IT projects to have more predictable and precise outcomes (The Royal Academy of Engineering, 2004), improved precision needs to be incorporated as much as possible during the earlier phases of the projects with the aim of resolving the more complex technical issues or uncertainties earlier in the project's lifecycle and making the later phases

more “mechanical” and more predictable. By “precision” what is meant here is any or all of the following: a more detailed and complete set of requirements (Davis & Venkatesh, 2004), a design phase that involves the end users and detects gaps and inconsistencies in the requirements at an early phase (Davis & Venkatesh, 2004), and finally (the subject of this thesis) a test phase that is more precise about what should be tested, how it should be tested and the optimal prioritisation of the tests. A test approach is needed that can help deliver better prioritised and more objectively defined test stages, i.e. where the test effort and outcome are less dependent on the individual tester (Smith & Thompson, 2008) (Bertolino, 2007). How can an IT project team ensure, objectively, that the more complex fundamental tests are not left too late in the project’s lifecycle where remedial action would be too expensive or impractical?

So far this thesis argued that there is a need for a widely understood and practiced framework to help achieve this objective for the specific domain of communications-critical large scale systems. The rest of this chapter will discuss what such a framework might be like then provide an example.

4.2 Domain-specific test framework

How can the effectiveness of testing of communications-critical large scale systems be improved? Can a different framework, alternative to generic methodologies such as the V-Model, facilitate such improvement? What attributes and features should such a framework have?

An effective test framework for such type of systems needs to be meaningful and transparent to non-specialist stakeholders (ISTQB, 2012, p. 18) who need to have visibility of a system’s quality but may not be involved in its detailed design and implementation effort. It needs to minimise the reliance on the tester’s own subjective judgement and intuition for determining where test efforts should be concentrated and how functional and non-functional features of a system are to be tested (ISTQB, 2012, p. 11). It needs to also reduce the differences that occur between the various levels of abstraction of the system’s functionality (Belgamo, Fabbri, & Maldonado, 2005) (Smith & Thompson, 2008) (Davis & Venkatesh, 2004).

Any such framework needs to provide the tester with the foundations to carry out the test analysis and design (ISTQB, 2012, pp. 11-13) as objectively and precisely as possible. With the variety of complex systems and technologies that exist today and are likely to evolve in the future, it is difficult to envisage a single general approach that can be used to test all types of systems without requiring considerable adaptation and interpretation. This is why this thesis is concerned with defining a new domain-specific test framework for communications-critical large scale systems[35].

The term "framework" can have a variety of meanings. Therefore, for clarity, a list of criteria and attributes is provided below to explain what is meant by a domain-specific test framework and what features and attributes are needed to define its form and potential uses. The list is not intended to be either exhaustive or exact, but is intended to better specify what is meant by the term "domain-specific test framework" and to clarify some of the thinking and theory[36] behind the ideas[37] that later on were developed into the layered model presented in Section 4.4:

1. Be already tailored and aimed for the specific class of systems it is intended for, hence it should contain communications specific features, i.e. be domain-specific to communications-critical large scale systems (Bertolino, 2007, p. 12);

2. Enable testing to remain linked to system requirements and ultimately provide evidence that the requirements have been fulfilled (Barmi, Ebrahimi, & Feldt, 2011);

[35] Although adapting any such framework to other domains might be feasible, this is outside the intended scope of this thesis.

[36] In terms of the design evaluation ideas presented in (Verschuren & Hartog, 2005), this list along with another list in Section 4.3 resemble the "assumptions" and "structural specifications" for the design.

[37] The lists in this section and in section 4.3 represent the ideas that followed on from the initial awareness of a gap in testing and before devising the new layered model. The individual items in the lists in this section and in Section 4.3 are details of the thoughts that explain the progress from the initial awareness to the development of the layered model. They are not intended to be a final set of formal "requirements". The overarching "requirement" can be thought of as the design of the new test framework.

3. Be applicable from the requirements capture stage up until rollout and implementation of the system, but allowing for finding errors early during an IT project's lifecycle where the cost of fixing is lower (Boehm, Basili, & Rombach, 2005, p. 426).

4. Support close cooperation between the test team and the rest of the project team and the project management, and thereby contributing to the overall project success (Yanga, Huang, & Wua, 2011, p. 265).

5. Be suitable as a conceptual basis that can be adopted for defining the test strategy (ISTQB, 2012, p. 32) for real-life IT projects, e.g. by providing guidance on some or all of the following aspects of testing for an IT project:

    - Test approach;
    - Test coverage;
    - How test design will be done, what it will be based on;
    - How the test phases or stages will be defined and prioritised;
    - How traceability to requirements will be confirmed;
    - How compliance to the framework can be evaluated.

4.2.1 Why a domain-specific approach is appropriate?

Before this thesis moves into discussing how ideas of the test framework can be applied to a real-life communications-critical large scale system, consider initially this hypothetical minimal example. The example is intended to demonstrate potential differences between a widely known and widely adopted generic test approach such as the V-Model approach (Ammann & Offutt, 2008) when testing a communications-critical system and, on the other hand, a domain-specific test approach.

Consider a project for an enterprise-wide system for, say, a courier company with a requirement to provide a solution for the remote management of mobile user terminals used by the company's delivery staff to keep track of their status and locations. The mobile user terminals contain AVLS (Automatic Vehicle Location Service) functionality as well as user GUI forms and data for communicating between the staff while out on the road and the company's distribution and control centre. See Figure 2 below:

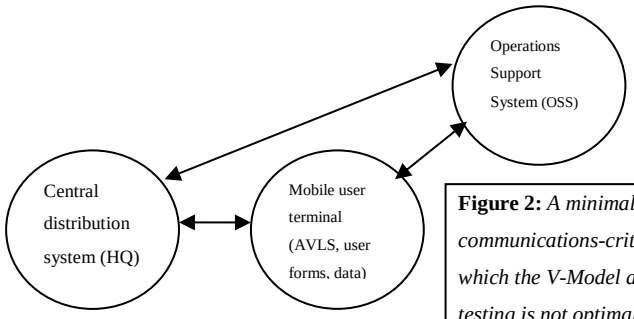


Figure 2: *A minimal example of a communications-critical system for which the V-Model approach to testing is not optimal*

Using the V-Model approach when testing the solution for this requirement would lead to the following.

- The testers being "consumers" or "users" of the technical design as decided by the developers (Smith & Thompson, 2008);
- The tester not being provided with precise guidance on what information should be expected from the developers about how the solution will be implemented, hence will not be in a position to objectively detect design weaknesses or omissions (Ponte, Rossi, & Zamarian, 2009);
- The test design being organised along the traditional V-Model phases of: unit testing, integration testing, system testing and end-to-end (E2E)/user acceptance testing (UAT). These four[38] test phases would be further divided along the lines of functional vs. non-

[38] An initial stage can be added for reviewing the requirements and the design before the implementation work commences

functional tests, and should include both positive and negative tests (Machado, Vincenzi, & Maldonado, 2010);
- The test cases being derived from what is stated explicitly in the technical design plus further additions based on the tester's own knowledge and experience. The tester's ability to specify detailed and effective test cases will be dependent on the completeness and quality of the requirements and the technical specifications (Smith & Thompson, 2008);

If the V-Model approach is adopted, the following test phases are likely to be applied:

- Individual **unit tests** for each of the three main components of the solution, for example by using test harnesses, to verify that each component functions as defined.
- **Integration tests** to check that each of the components is able to send/receive messages and data from the other two components.
- **System tests** to check that the overall solution functionality and its non-functional attributes are as described in the requirements and the technical specifications.

Such test phases, even when conducted in accordance with good V-Model practices, can still leave weaknesses and faults in the system undetected. Below are examples of such weaknesses and faults.

- Faults due to the developers' incorrect or incomplete interpretation of the requirements;
- Timing and synchronisation issues between the mobile terminals and OSS, unless these are explicitly mentioned in the requirements and the technical design;
- Data integrity issues following failures of interfaces, unless these are explicitly mentioned in the requirements and the technical design;
- Potential contradictions/feature interactions between different remote management and configuration functions carried out by the OSS.

Furthermore, the V-Model approach risks leaving the more important and complex features of the system to be tested towards the end of an IT project's lifecycle where costs of correcting faults are high and the impact on project delivery deadlines are likely to be

significant (Boehm & Basili, 2001) (Machado, Vincenzi, & Maldonado, 2010). Additionally, according to this approach the tester has no independent and objective source to evaluate the completeness and adequacy of the requirements, identifying potential gaps and risks. Instead, the effectiveness of the testing ultimately relies on the skills, experience and subjective judgement of the tester (Offutt, 2008).

With a domain-specific test framework, the design and organisation of the test activities could be more closely related to the structure of the system (Bertolino, 2007, p. 8)[39]. This in turn can potentially bring benefits to the testing analysis being simpler and requiring less adaptation to the system under test. Below are a few examples of how the domain-specific testing for the sample system (Figure 2) could be different from testing based on the V-Model approach:

- Test phases being organised according to the architecture and purpose of the system rather than the V-Model defined development phases: Instead of having unit, integration and system tests there could be tests for the setup of the communications hardware and software for the user terminals; tests for messages exchanged between the user terminals and the remote management system; tests for the integrity of application and user data residing on the terminals; tests for the detailed functionality used to remotely manage the user terminals; then finally tests for the operational processes used to remotely manage the user terminals. Some of the tests could be based on relevant domain-specific industry standards or guidelines, such as eTOM (Kelly, 2003), TMN and FCAPS (ITU-T, 2000)(ITU-T, 1997) (ITU-T, 1996) (Parker, 2005) or ITIL (Adams, 2009)(ITU-T, 2004);

- The testers potentially being guided by the domain-specific framework to include test cases that otherwise only a domain expert would be able to identify (Bertolino, De Angelis, Di Sandro, & Sabetta, 2011) unless explicitly mentioned in the technical design, e.g. timing synchronisation between the OSS and the mobile terminals or

[39] Compositional testing ideas referred to in (Bertolino, 2007, p. 8) might coincide with some of the ideas in this thesis if the framework's layers are thought of as, potentially, actual physical components of a CCLSS rather than a conceptual representation.

potential conflicts or interactions between different remote functions carried out by the OSS.

- The testers having an objective basis with which to evaluate the contents of the design documentation produced by the developers;

- The testers being expected to take an active role and to add significant value to the work at an early stage of the project before actual "hands-on" testing starts.

The requirements and the technical design of a system do not always explicitly state all aspects that could be important for the acceptable operation of a system, hence the need for domain expertise during construction of that system, and equally during testing. A domain-specific test framework should reduce the need for domain expertise during testing, and hence, could increase the objectivity and repeatability of the tests.

In summary, the following are examples of possible significant types of faults in a system such as that described in Figure 2 that could be missed by the V-Model approach but are more likely to be detected by a domain-specific test approach:

- Timing problems between the three components;
- Inability of the OSS to correctly synchronise the state of the two other components following a failure;
- Data integrity issues between the three systems, e.g. if the OSS is not able to identify and resolve data synchronisation issues;
- Messages or commands exchanged between the OSS and the two other components being capable of interfering with each other in a way not considered or defined in the technical specification. This phenomenon is sometimes referred to in telecommunications as "feature interaction", a term which relates to voice call services in the telecommunications world but is also relevant to IT system interfaces in today's communications-critical systems.

A general test method such as V-Model will not guide the tester to cover the above examples of domain-specific features unless they are explicitly stated in the requirements or the design of the system (Smith & Thompson, 2008).

This section is a general discussion of one of the ideas behind this thesis. Further explanations and details will be presented in the remainder of this chapter then in Chapters 5 and 6 on how a domain-specific test framework can be defined, detailed and applied for testing a communications-critical system.

4.3 What would the proposed test framework be like?

To be able to analyse a complex issue, any type of complex issue, i.e. not necessarily IT or testing related, requires a conceptual analysis framework that will: (a) make the complexity manageable, and (b) allow objective evaluation of the details of the complexity, e.g. provide basis for determining whether a detail is correct, appropriate, sufficiently defined or if there are contradictions or inconsistencies between the details. This is why the idea of modelling is so well established in, and integral to, many non-IT engineering disciplines such as, say, civil engineering or mechanical engineering. The idea of creating high level models as a first stage of constructing a new "system" is now becoming well established in IT engineering practices as well (Schmidt, 2006) (Bertolino, De Angelis, Di Sandro, & Sabetta, 2011) (OMG, 2011). However, IT differs from other engineering disciplines by exhibiting a relatively high degree of change in technology and in the tools that are used, also in that the forms of the "systems" that it constructs do not intuitively and naturally relate to their intended function, i.e. not in the same way that the idea of a house or a car are clearly and intuitively understood, making it relatively easy to comprehend them conceptually without necessarily being a specialist engineer and without delving into the technical details. Therefore, when the process of constructing a house or a car starts, it often starts from a well understood knowledge "framework" about what that "system" might look like or is meant to be used for. The engineer's role would then be to apply specialist knowledge and skills to detail and implement the well understood concept of that "system". IT engineering does not have the same benefit of well understood and agreed upon "frameworks" for the "systems" it

creates. Furthermore, the intended uses of IT systems, the materials and tools used to construct them tend to change too rapidly to allow for the evolution of clearly understood IT systems knowledge "frameworks" compared to, say, houses or cars. This is a feature of IT systems that needs to be addressed in order to be able to construct IT systems precisely and predictably, and equally (which leads back to the subject of this thesis), in order also to be able to carry out more predictable and precise testing. The ideas that are discussed further in the rest of this chapter are derived from the thought that a "framework" derived from the tester's view of a large scale communications-critical system is needed as a starting point for analysing the requirements, then defining and planning the tests for such system.

The remaining chapters within this thesis will discuss how such a framework, representing and simplifying the concept of the "form" (i.e. structure) of the type of systems it is aimed at, is capable of leading to more efficient testing than a generic methodology or standard that is based on a process of testing that should be applied to all or any type of system.

As discussed earlier in Chapter 2, such a framework could be thought of as the testing equivalent to Zachman's Architecture Framework (Zachmann, 1987) (Chen & Pooley, 2009) in the sense that it is intended to organise the concept of a large scale communications-critical system from a test specific point of view. Having such a view would then help with managing complexity and improving objectivity during the test analysis, design, specification and prioritisation (Khan, Rehman, & Malik, 2009) (Yoo, Harman, Tonella, & Susi, 2009) stages. Therefore, such a framework could be a step towards more precise and effective testing for such a type of system.

Conceptually, such a framework should provide a representation of how the system should be viewed from a test viewpoint. It should represent a domain expert's view on how such systems are structured, and what components or groups of components should be tested first. The preliminary test analysis and test design work for a specific system would then initially be organised on the basis of this framework, rather than simply derive the test design and individual test cases from the individual requirements or the technical specifications. This way the test activity would be based on, and be organised according to, this conceptual "framework" of what a large scale communications critical system is and how it should be

tested. Such a framework should capture a domain expert's knowledge into conceptual "chunks" as discussed in (Gobet, et al., 2001) and therefore can be expected to facilitate more efficient and precise testing by reducing reliance on the individual tester's knowledge and judgement while specifying the detailed test cases, and by moving the more fundamental decisions about what features should be tested and when to earlier phases of the IT project. Instead of expecting the testers to develop their own knowledge of the system, the conceptual testing view of the system is made beforehand so that the tester can do more testing and less "learning", or at least the learning will be done within a predefined framework.

To clarify some of the early thoughts[40] behind the ideas of a new test framework, below is a further list of features and attributes, which compliments the list provided in Section 4.2, regarding the form and potential uses of such a framework[41]:

1. Provide a domain-specific conceptual overview, or model, of the system's structure to aid the test analysis and design efforts (Henderson-Sellers, 2011);

2. Be used as the basis for review and verification of both the system requirements and the system specification and design to help improve the link between the different phases of a system's development lifecycle and to reduce the differences and faults that can be introduced between phases (Belgamo, Fabbri, & Maldonado, 2005);

3. Allow the testers to start their verification and validation work from an early stage, e.g. just after the requirements capture work has been done (ISTQB, 2012, pp. 14, 48).

[40] The lists in this section and in section 4.2 represent the ideas that followed on from the initial awareness of a gap in testing and before devising the new layered model. The individual items in the lists in this Section and in Section 4.3 are details of the thoughts that explain the progress from the initial awareness to the development of the layered model. They are not intended to be a final set of formal "requirements".

[41] In terms of the design evaluation ideas presented in (Verschuren & Hartog, 2005), this list along with another list in section 4.2 resemble the "assumptions" and "structural specifications" for the design.

4. Facilitate test traceability and coverage analysis (ISTQB, 2012, pp. 18, 40).

5. Have further use in an IT project outside purely testing and allow better linkage between the testing and the original requirements (Barmi, Ebrahimi, & Feldt, 2011), e.g. allow the business analysts, developers, testers and users to have shared conceptual view of the system, which in turn could improve the outcome of an IT project by reducing the variations of ideas between the different groups involved in the project (Belgamo, Fabbri, & Maldonado, 2005).

How can such a test framework, intended specifically for communications-critical large scale systems, be formulated?

4.4 Explanation of the domain-specific test framework

The structure of any communications-critical large scale system can be viewed as a layered pyramid structure, with the infrastructure (including the communications infrastructure) at its base and the end-user services at its apex, a communications layer, data layer and a functionality layer between them. These layers represent groups of components, logical as well as physical, of the system and associated functions that need their own specific testing approaches. The same test approach adopted for high-level transactional functionality of a system cannot be effectively applied to validating its data, its communications links or its business processes.

A domain-specific test framework for communications-critical large scale systems could be based on such a conceptual view of the system under test, and aim to organise the test design according to it. The framework's starting point will be to categorise the system's components and functionality in a way that will facilitate objective analysis and test design at a later stage. Specifically for communications-critical systems, it should guide the test design to isolate the communications functionality of the system and treat it as a sub layer of the system. As discussed earlier in this chapter, such a domain-specific framework can be used

to improve precision and objectivity in test design, and simplify the effort for defining the test stages.

The idea of layers is a fundamental and well established feature of communications protocol design and testing in the world of telecommunications (Bochmann, Rayner, & West, 2010) (ETSI, 2011) (ITU-T, 1994). Protocol testing is a good example of how precise and effective testing can be achieved. A comparable approach to testing large scale systems can be expected to lead to significant benefits - if it can be adapted to testing large scale systems with significant communications layers. Motivated by this idea and the ideas discussed earlier in this chapter, the proposed test framework could be based on the following view of a communications-critical large scale system:

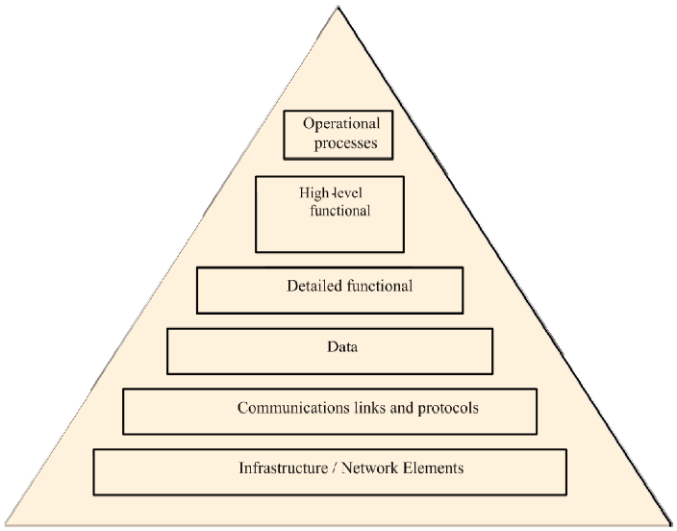


Figure 3: A communications-critical large scale system test framework

A communications-critical large scale system's test subgroups or phases could therefore be categorised according to one of the following categories:

- Non-IT commercial: non-IT features that are outside the scope of software testing;
- Infrastructure: communications hardware, IT hardware and software packages, configuration and setup needed for the infrastructure;
- Communications links and communications features;
- Data;

- Detailed functional: i.e. functional features that are intended to facilitate other higher level functional features but are not in themselves what the system is intended for;
- High level functional: i.e. the functional features that describe how the system is meant to achieve the intended business and operational processes;
- Business and operational processes: these represent the purpose of the system, i.e. what the system is meant to achieve.

Note: Non-functional characteristics and constraints of the system and how the system will be operated and managed are outside the intended scope of this research work. These are intentionally excluded to maintain clarity of the concepts discussed.

Furthermore, the functional features can potentially be subdivided into further categories, each viewed as a logical subgroup, such as:

- Static functional: describing how the system is structured;
- Dynamic functional: describing what the system does.

For the purposes of this discussion, functional features will be referred to without a distinction between the static and dynamic.

The V-Model approach was an appropriate test approach when the design and development activities of systems were organised according to clear categories of unit/integration/system, but this is no longer the case for large scale systems with a significant communications layer. This chapter (and thesis generally) are not intended to argue that the V-Model cannot or should not be used. In fact, the V-Model or aspects of it could still be an appropriate approach for one of the layers within this framework. This is a subject outside the intended scope of this thesis. The main proposition discussed by this chapter specifically, and this thesis generally, is that large scale communications-critical IT systems can tangibly benefit from a conceptual framework based on their structure to be the foundation of the testing carried out on such type of systems.

This framework can be used as the first step for creating a precise interpretation of the system's requirements and technical specifications that can be used as the foundation for subsequent test design work, where each layer can be dealt with via a specifically tailored test approach.

The details and the notational form of these specifically tailored test approaches is outside the scope of this research work, which is concerned with defining a more effective test framework for communications-critical large scale systems.

4.5 How specific test approaches for each of the categories can be derived

As discussed earlier in this chapter, each of the six layers that form the proposed test framework needs to have its specific test approach. How can such specific test approaches be defined?

On one hand, each layer is different and needs an approach that is specifically tailored for it, derived from a domain expert's viewpoint of what should be tested in that layer and how the tests should be prioritised and ordered for test design and scheduling purposes. On the other hand, this thesis proposes a generic theme for defining these approaches. This theme will become clearer through the example in the next chapter (Chapter 5) when the communications layered test approach is explained. However, this theme is outlined below for this section's completeness.

Each of the six layers within the test framework can, conceptually, be thought of as a layer that deliver "value" to the overall system in order to allow it to fulfil its requirements and overall purposes. Within each layer, there are a number of logical stages or "subcategories" that enable the "value" expected from each of the six layers to emerge and to contribute its part to the overall system. Such subcategories should be possible to classify in the order of how fundamental each of them are and in order of their dependence on each other, i.e. the first subcategory will be the most fundamental of all others and upon which the others are dependent. The next subcategory will be one that most closely relates to the first but is less

fundamental than it, and upon which the rest of the layer is dependent but to a lesser extent than the first subcategory. This pattern of decreased fundamentality and increased interdependency can then be progressively applied to define the "test subcategories" within each layer until the last subcategory is arrived at. The later or final subcategories should inversely be more complex and are closer logically to the required "value" that the layer delivers.

The term "value" as used in this chapter is borrowed from a business management concept established since the 1980s, of the "value chain" (Rayport & Sviokla, 1995). The "value chain" represents the tangible activities or products that, when combined together in a specific order, deliver the required "customer value". It can be considered that a large scale IT system is part of the overall value chain of the organisation using that system. Therefore, the system's conceptual layers are also subcomponents of the organisation's overall value chain. Based on this consideration, extending and adapting the ideas of the value chain for the purposes of the domain-specific test framework could be more of a viable proposition than it may appear at first. The theme for extending such business management ideas with software testing could be extended further by the idea that software faults and design weakness can be viewed as obstacles to the system contributing its required "value" to the business. The similarities between what is proposed by this thesis and the original 1980s business management concept probably end here, at the core idea.

It can even be considered that this approach has some relation to the V-Model approach to software testing. Both the test framework proposed by this thesis (i.e. the layered model) and the V-Model can be thought of as implicitly related to the concept of "value chain" but in different ways. The layered model is based on a conceptual test view of a system's ***layers*** value chain, whilst the V-Model is based on a conceptual test view of a system's ***development process*** value chain. Whereas the V-Model's test stages implicitly represent the value chain of the Waterfall development stages, the layered test framework presented in this chapter implicitly represents the value chain of the layers of a large scale communications-critical large scale system. Therefore, the conceptual difference between the layered test framework and the V-Model could be explained by the difference between the views of

business and project management of software testing vs. a technical developer's view of software testing.

In conclusion to this subsection, the generic "theme" for deriving specific test approaches for each of the six layers in the proposed test framework is as follows: the components in the "value chain" for each of the framework's six layers are to be identified based on a domain expert's viewpoint of how the value of each specific layer is delivered and subcategorised. An example of this will be provided in the next chapter (Chapter 5).

4.6 Theoretical basis for the ideas behind this thesis

As was discussed in the research methodology chapter (Chapter 3) section 3.6, the work presented in this thesis is primarily design-based and experience-inspired case study research. However, it is by no means without theoretical basis as has already been presented in the literature review chapter (Chapter 2) in general and more specifically in section 0, as well the introduction chapter (Chapter 1). The discussions within these three chapters, when combined, represent the overall basis (including the theoretical basis) behind this thesis.

As an addition to the material referred to above, a list of specific theoretical foundations that together helped motivate, or at least support, the ideas of this thesis is presented below.

- The need for a theoretical test framework for large scale systems is apparent from papers discussing the state of art of testing in industry such as (Bertolino, 2007). Specific quotes from the paper are presented in section 2.11.

- The absence of equivalents to Enterprise Architecture Frameworks such as Zachman's (Zachmann, 1987) in the field of testing yet the concept seemed to be potentially beneficial for leading to consistent and better defined testing.

- The presence of a UK research and training programme (LSCITS, 2013) treating Large-Scale Complex IT Systems as a distinct class of systems motivated the focus on

communications-critical large scale systems.

- The idea of the layered protocols architecture from communications engineering (Bochmann, Rayner, & West, 2010) provided precision in communications systems design, development and testing that could also lead to more precisely engineered large scale software systems if it can be adapted for their use.

- Other Telecommunications concepts also contributed indirectly to the ideas in this thesis, such as The ITU FCAPS model (ITU-T, 1996) used in the 1990s, the more recent eTOM (ITU-T, 2004) (TMF, 2013) and the FAB model (TMF, 2011). Such models or processes provide simplified conceptual views of what otherwise are complex telecommunications networks and systems. A common theme between them seems to be[42] the use of the ideas of layering (as mentioned above) as well as an interpretation of a "value chain" for the network management services to construct a model for network management systems.

- The notion of clustering of test cases as is present in test literature such as (Yoo, Harman, Tonella, & Susi, 2009). Although clustering is often referred to in the context of code or unit testing of deterministic systems, the concept seemed potentially of benefit if it can be adapted to larger complex non-deterministic systems.

- The need to link requirements to test cases as discussed in papers such as (Barmi, Ebrahimi, & Feldt, 2011) and (Belgamo, Fabbri, & Maldonado, 2005).

- The need for, and the potential benefits of, optimal test case prioritisation as present in test research literature in the context of code/unit testing of deterministic systems, e.g. (Yoo & Harman, 2007), (Elbaum, Malishevsky, & Rothermel, 2001), (Elbaum, Malishevsky, & Rothermel, 2002) or for larger systems, e.g. (Svensson, et al., 2011).

[42] This is an author's interpretation based on experience

- Model-Based Testing (MBT) ideas (as well the concept of modelling in general) as already referred to in section 2.9 of the literature review. Although MBT is usually mentioned in the contexts of test case generation for small systems (Hemmati, Acruri, & Briand, 2010), it seemed to be a potentially beneficial source of ideas if it can be adapted to large scale systems.

- The concept of the Value Chain from business management theory, e.g. (Rayport & Sviokla, 1995) (Value Chain Group, 2007) also seemed as a source of potentially useful ideas for organising the requirements and the testing for large scale systems[43].

- Chunking theory from Psychology (Gobet, et al., 2001) and the use of mental representations of design problems (Bjorklund, 2013) also point to potential benefits to software engineering practices of capturing expert knowledge and sharing it, e.g. via conceptual models. Accordingly, the author's thinking is that using a simplified conceptual model of CCLSS as basis for testing could help lead to better testing of such systems because it makes it simpler for testers to work with CCLSS as well as captures and re-uses expert knowledge of such systems.

4.7 Chapter summary

This chapter discussed the domain-specific test framework proposed by this thesis and explained how and why it was defined. The chapter explained how the framework provides a way of conceptually viewing a communications-critical large scale system from a test viewpoint. In order to be able to apply this framework during the testing of a real-life system or IT project, a further level of detail of the test approach specific to each of the layers needs to be defined. However, detailing the test approaches specific to each of the six layers of the framework requires an amount of work disproportionate to this thesis. Therefore, the communications layer was selected for further detail in this thesis for a variety of reasons, but mainly because: 1) it is what makes the type of system discussed in this thesis distinctive, i.e. the communications-critical aspect, 2) it is where convergence between IT and

[43] Already discussed in this chapter in section 4.5

communications is most evident, which makes it more interesting for research purposes, 3) it is likely to provide a good example of where widely practiced ideas about IT testing need to be re-examined and possibly challenged, and finally 4) a suitable real-life industrial case study was possible.

The next chapter (Chapter 5) will develop further the ideas of the domain-specific test framework specifically for the communications layer. It will then be followed by a relevant real-life industrial case study (Chapter 6) applying and examining the benefits of the ideas presented both in this chapter and in Chapter 5.

5. Chapter 5: The communications layer

ONE OF THE LAYERS OF THE FRAMEWORK, AS PRESENTED IN CHAPTER 4, DETAILED

5.1 Introduction

As discussed at the end of the previous chapter, in order to begin to apply the new test framework proposed by this thesis, one further level of detail is needed for one of its layers. For this purpose, the communications layer was selected to be detailed further and to be included in a case study involving a communications-critical large scale system.

This chapter will describe a test approach specific to the framework's communications layer. It will be followed by Chapter 6 describing a case study.

The following text is an extract from Chapter 4 discussing the domain-specific test framework: *"Conceptually, such a framework should provide a representation of how the system should be viewed from a test viewpoint. It should represent a domain expert's view on how such systems are structured, and what components or groups of components should be tested first. The preliminary test analysis and test design work for a specific system would then initially be organised on the basis of this framework, rather than simply derive the test design and individual test cases from the individual requirements or the technical specifications. This way the test activity would be based on, and be organised according to, this conceptual "framework" of what a large scale communications critical system is and how it should be tested. Such a framework should capture a domain expert's knowledge into conceptual "chunks" as discussed in* (Gobet, et al., 2001)*".*

A conceptual representation (from a test viewpoint) of the communications layer of the framework discussed in Chapter 4 needs follow the same logical approach as outlined above, but at a more granular level than the layered framework. The layered framework is a higher level representation, whereas the communications layer is a lower level, more detailed elaboration of the framework, which brings the framework closer to being applied to real-life systems.

5.2 A domain-specific test approach for the communications layer

Any communications-critical large scale system (CCLSS) will have a number of critical communications interfaces. These interfaces will naturally vary in the networking protocols they use, the data they carry, their architecture, their senders/receivers, the purpose they serve, etc. However, it is possible to identify from a test viewpoint a common set of "testable features or subcategories" that form the dimensions of any communications interface that needs to be tested and the order in which they need to be tested to provide confidence that the communications part of a system is ready to support the rest of that system in live operation.

Such test subcategories, as will be discussed later in this chapter, can be thought of as a "value chain" (Value Chain Group, 2007) representing the functionality and the services provided by the interface to the overall CCLSS.

Before presenting the proposed communications layer test subcategories, an explanation regarding the theoretical basis behind the idea of theses subcategories will be provided in the next subsection.

5.2.1 Discussion regarding the theoretical basis for the ideas in this chapter

The ideas presented in this chapter are an extension to, and represent further detail of, the ideas presented already in Chapter 4 and are motivated by the same theoretical as well as experience basis that are behind the ideas in Chapter 4. These bases have already been explained in Chapter 4 in general, and concluded in section 4.6. More specifically, the idea of the nineteen test subcategories presented later in this chapter was motivated, in part[44], by communications concepts such as the OSI layers[45] (Zimmermann, 1980) and TMN layers[46] (ITU-T, 2000) and eTOM/FAB process models (Kelly, 2003) where abstract layers are common logical processes and functions are used as the basis for unifying and standardising the approaches to managing complex and technically varied telecommunications networks,

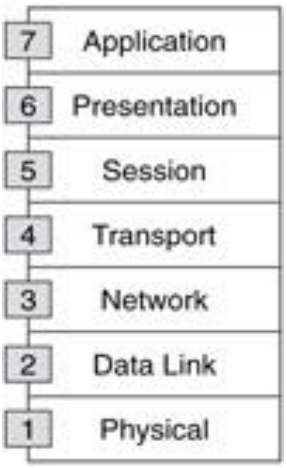
[44] As well as by value chain ideas as explained later in the subsection

[45] Diagram was copied from (Network World, 2008)

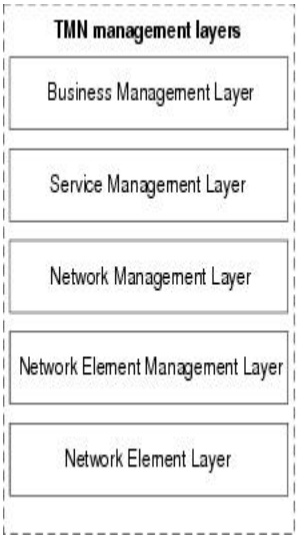
[46] Diagram was adapted from Figure 1-1 from (Cisco, 2013)

e.g. core networks, access networks, wireless networks, voice services, data services, etc. Furthermore, the layers and functions are organised logically to progress from the fundamental to the more complex, from the physical to the logical, from the layers that the end users of the services are not in direct contact with, to the features and services that are delivered to the end users. One intended potential benefit from this approach is to achieve fewer interdependencies between the subcategories and relatively optimal ordering of the resulting tests, as will be examined in the case study chapter (Chapter 6) and the overall evaluation chapter (Chapter 8).

Additionally, there seemed to be, at least implicitly, a common pattern to such abstractions akin to the concept of the value chain (Value Chain Group, 2007) in how these layers are derived and sequenced. Such concepts and ideas are well established in telecommunications systems design and implementation. They can help make the test analysis and design work for such systems more uniform and less subjective than if they did not exist because they offer a common simplified "framework" for the tester of a network management system (NMS) or a telecommunications operational support system (OSS) to use as basis for testing.



OSI Layers



TMN Layers

The new test framework ideas presented in this chapter and in the thesis generally were motivated by such concepts and the concepts presented in the previous chapter (Chapter 4) especially in section 4.6. In general, this thesis can be viewed as an attempt to take such concepts from the field of telecommunications and networking and adapt them for testing communications-critical large systems. Due to convergence between IT and telecommunications that started occurring in the 1990s, there are sufficient commonality and similarities between the two fields that could help make such an objective achievable. For example, can there be a single domain-specific approach, inspired by telecommunications concepts, and specifically aimed at testing communications interfaces of a CCLSS, that could be applicable to a variety of types of communications interfaces and communications technologies? This is the thought that the rest of this chapter is intended to address by proposing a list of test subcategories for the communications layer of the new test framework that has already been presented in Chapter 4.

5.2.2 The test subcategories

Below is a list of "testable subcategories" proposed for the purpose of applying the ideas of the new domain-specific test framework to a real life case study.

1. The structure/architecture of the network interface, its components and layout, hardware, and wiring

2. The communications protocols used
3. The user terminals
4. The data and messages that are transmitted over the interface/network
5. All possible types of senders and receivers over that interface
6. The different possible modes of transmission used
7. How the transmissions are acknowledged by the receivers
8. How the performance characteristics of the network can affect systems and processes
9. The services provided by the network to the system
10. Performance and volume limits of the services provided by the network
11. How QoS and SLAs are guaranteed, maintained and reported
12. The on-going operation, maintenance and administration of the interface
13. Fault handling processes of the interface, from detection to resolution
14. Certification/compliance requirements
15. Resilience features (of the interface)
16. Business continuity features (of the interface)
17. Documentation: user and technical documentation
18. Risks
19. Operational readiness of the interface for the system's go-live

The above subcategories list is intended as a "recommended example" of instantiation of one of the layers of the new test framework. The exact list, in the exact order it is presented in this thesis, is not intended nor claimed to be the only or the best list to base testing on. The list's primary purpose is to be feasible for use as basis for the test analysis and design of the communications layer of a CCLSS, and the use of its nineteen subcategories is intended to lead to "efficient" testing of the communications layer of a CCLSS as will be explored in the case study in Chapter 6 and the overall evaluation in Chapter 8.

5.2.3 How were the subcategories derived?

The nineteen "testable subcategories" are intended to provide a relatively precise, but at the same time adaptable, description of a communications interface for any communications-critical large scale system. They represent components that, when combined, form the

overall "value chain" of the interface as mentioned earlier in this chapter. They also represent information which an experienced communications systems testing subject matter expert might be looking for when trying to understand and design the tests for any communications interface relating to any communications technology.

They were derived by identifying all the communications specific features that can be used to describe any communications interface to any large scale communications-critical system that has functionality and user processes that rely on transmitting data across external or internal communications interfaces. In other words, they are a conceptual representation of the communications layer (not necessarily the only or the best) of a domain expert's knowledge. They were organised according to the most basic and fundamental aspects first then developing into the more complex. One of the purposes of this sub categorisation is to allow for as much separation as possible of these features into independently testable groups of features where the lower numbered ones can be tested first then progressing to the higher numbered ones. This is to allow for a more optimal prioritisation of the testing and to minimise as much as feasible the interdependency between the subcategories[47]. In reality, further analysis work is likely to be needed to group and prioritise the tests, but starting with the proposed nineteen subcategories is intended to reduce the complexity of such effort and improve its objectivity and precision.

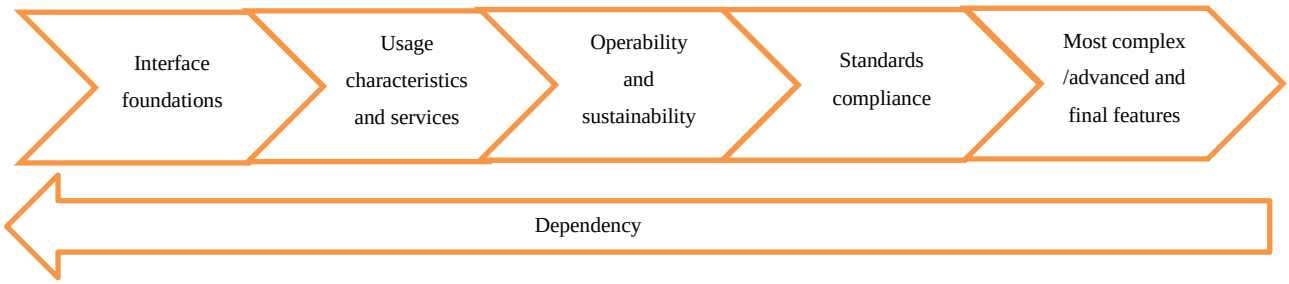
5.2.4 Value chain and dependencies

This subsection provides further explanation regarding the rationale that led to the nineteen test subcategories.

As discussed earlier in Subsection 5.2.1, and based on the premise that there is an overall "value chain" that a communications interface provides to a CCLSS, one way[48] that the high level conceptual components of such a value chain can be thought of is according to the following diagram.

[47] As will be examined in the case study chapter (Chapter 6) and the evaluation chapter (Chapter 8)

[48] Not necessarily the only



The “interface foundations” block refers to core aspects of the interface, such as its structure/architecture, the communications protocol used, the user terminal, the data transmitted across the interface as well as the users of the interface. “Usage characteristics” refer to the more dynamic and changeable aspects of the interface that depend on the interface foundations being in place, such as the different modes of transmission, how acknowledgements are processed (which in turn contributes to the reliability of transmissions), and performance considerations and how the performance of the interface impacts the functionality of the CCLSS. All such components of the value chain deliver the overall “services” that are provided by the interface to the CCLSS and define its performance. The next part, “operability and sustainability” refers to aspects such as the monitoring and reporting of the quality of service (QoS) and the service level agreements (SLAs) that are applicable to the interface, as well as the operation, maintenance and fault handling features for the interface. Such aspects are not meaningful nor have value of their own without the availability of a functioning communications interface, i.e. they depend on the preceding components in the diagram. Next is the “standards compliance” component which, from a test point of view, is more appropriate to test at a later stage when all other more fundamental aspects of the interface are known. The last part is for the “most complex, advanced and final features” that would, from a test view point, be generally dependent on all other aspects of the network to be in place and functioning before they are tested. The block arrow titled “dependency” indicates that, in broad terms, dependency between the components is likely or expected to flow in the opposite direction to the value chain. In other words, the later components are expected to generally be dependent on the preceding components. This aspect will be examined further in the case study chapter (Chapter 6).

5.2.5 Further explanation of each individual subcategory

Following on from the rationale presented earlier in Subsection 5.2.4, the proposed subcategories are defined as follows and explained further one-by-one:

The structure/architecture of the network interface, its components and layout, hardware, and wiring

This subcategory is a conceptual approximation for the OSI Model layer 1 (Physical layer) for CCLSS test purposes. It represents the most fundamental aspect that needs to be understood when investigating the features of a network interface, regardless of what that network interface is. Therefore, this should be the first feature subjected to verification and test activities.

The communications protocols used

This subcategory is a conceptual approximation for, or is equivalent to, OSI Model layers 2 and 3 (Datalink and Network layers). It represents another generic aspect of any network interface which needs to be tested or verified early on in the development cycle of a new system. The reason this is listed as second to the architecture is because the functionality provided by a protocol may vary according to the architecture and structure of the network.

User terminals

This subcategory compliments the previous subcategory to define the CCLSS test equivalent of the OSI Model layer 3 (Network layer). The different types of user terminals and their impact on system behaviour need to be understood and verified next. User terminals vary according to the communications protocol used, hence this aspect of a communications interface needs to be verified or tested after the communications protocol.

The data and messages that are transmitted over the interface/network

After considering the network interface fundamentals such as the architecture, structure, protocol and user terminals, next to consider is the different types of

messages that will be transmitted over that interface, whether general types determined by the protocol or tailored specifically for the purposes of the application under test. Logically, this subcategory and the next subcategory are conceptually closest to OSI Model layer 6[49] (Presentation layer).

All possible types of senders and receivers

The types of users sending and receiving the transmitted data represents the next aspect of a communications interface that needs to be considered during test design of a communications interface.

The different possible modes of transmission used

This is another possible source of system behaviour variation, secondary to data types and receivers and senders of that data. An example of this is the different modes that a 3G network can use to transmit user data, e.g. 3G, 3.5G, or GPRS. This subcategory and the next subcategory are conceptually closest to OSI Model layers 4 and 5 (Transport and Session layers).

How the transmissions are acknowledged by the receivers

Test design needs to take into account not just the message types transmitted over the interface, but also how (or if) they are acknowledged because such considerations need to be incorporated in the test design.

How the performance characteristics of the network can affect systems and processes

A system utilising a network interface to communicate with the other systems or networks may exhibit different behaviour depending on the performance or QoS of the network, e.g. delayed responses may trigger the system to prompt the users to use manual processes to communicate with other users/systems if the performance of a communications interface dropped below a usable threshold or if acknowledgements for sent data were not being received within an acceptable time frame. Such an aspect

[49] The order of the subcategories was determined by prioritisation considerations for testing purposes, based on value chain ideas. Therefore, it may not necessarily, nor was it intended to, match the sequence of the OSI Model layers.

of communications interface needs to be taken into account during test design, even if it is not explicitly stated within the requirements or the technical specifications of a system. This subcategory and the next two subcategories are conceptually closest to the top OSI Model layers 7 (Application layer).

The services provided by the network to the system

All the previous subcategories are individual aspects of a network interface that, when combined, deliver the overall user-visible services provided to the system.

Performance and volume limits of the services provided by the network

The next subcategory represents the acceptable performance and volume limits of the network interface. After a network interface service is provided correctly from a functional viewpoint, test design needs to also cover its non-functional characteristics (ISTQB, 2012, p. 22).

How QoS and SLAs are guaranteed, maintained and reported

A reporting element can be expected in any system that utilises a network interface to ensure that there is sufficient evidence that the required QoS and SLA requirements have been, or are being, fulfilled on an on-going basis (eTOM, 2011).

The on-going operation, maintenance and administration of the interface

So far the subcategories dealt with how a network interface is engineered, what services it offers and that the non-functional characteristics of that interface are as required. This subcategory is concerned with the on-going OA&M (IETF, 2011) of that interface (Operation, Administration and Maintenance).

Fault handling processes of the interface, from detection to resolution

This is a key aspect of OA&M (IETF, 2010, p. Section 3.2.4) which is likely to include more than just a technical solution, but a set of processes and an organisation that is geared to implementing the processes which ensures that faults with the network interface are adequately monitored, analysed and resolved.

Certification/compliance requirements

This subcategory is concerned with any applicable standards or certification that the interface needs to achieve. This subcategory's relative position within the list of subcategories can be amended to be earlier or later but for now it is placed after the basic structural/architecture, functional and non-functional subcategories, but before the most complex subcategories such as resilience and business continuity.

Resilience features (of the interface)

This subcategory (IBM, 2014) comes towards the end of the list because it represents the most complex aspect of an interface. All building blocks of an interface have to be finalised and verified before this aspect is checked. If resilience is checked at an earlier stage of the test and verification cycle then there is a risk that the results of the testing will be invalid if, for example, faults were later found with more basic features such as the types of data being transmitted or with how that interface is monitored and maintained.

Business continuity features (of the interface)

Following the resilience features, business continuity (including fallback and recovery) features and processes (BC Associates, 2012) represent the most complex aspect of an interface which needs to be checked last. Resilience features have to be sufficient then business continuity features take the interface one step further for being ready for deployment and live use.

Documentation: user and technical documentation

Work on documentation needs to start very early within the development cycle, but for test and verification purposes documentation cannot be complete while the system is still undergoing changes. Therefore it is likely to be more efficient if they are checked last.

Risks

"Test risks" (ISTQB, 2012, p. 23) are included at the end of the test cycle as a separate subcategory for purposes of clarity rather than prioritisation or test

scheduling. In practice, timing for actual testing for risks related to an interface is likely to be spread over more than one subcategory, depending on the individual risk. A "test risk" could be a concern expressed by end users about the system's features, or a failure mode that has been identified through a process of Failure Mode Effect Analysis (FMEA) (Stamatis, 2003)

Operational readiness of the interface for the system's go-live
This subcategory will be the final checklist before the go-live of the interface and the system under test (Microsoft, 2014).

5.3 Expected benefits of the proposed approach

Adopting the communications specific test approach outlined in this chapter can be expected to offer a number of advantages over a traditional general approach such as the V-Model. For example, the V-Model is concerned with the organisation of the test phases, major decisions about detailed test design are left to the subjective judgement of the tester. According to the V-Model, the testing effort is primarily about creating test cases and scripts in the form of input/transaction/output for functional and non-functional areas that are explicitly defined in the requirements and the technical specification. This approach is less relevant to IT now than it was in the past, where less outsourcing and COTS were used in IT project.

In contrast, adopting a communications test specific approach tailored for the specific purposes of a project for a communications-critical large scale IT system can be potentially facilitate better understanding of the system's requirements and design by the tester. The nineteen subcategories can be viewed as a type of a visualisation model (Pacione, Roper, & Wood, 2004) which can lead to better identification of overlapping or related requirements, gaps or inconsistencies in the requirements or the design. As well as these general benefits resulting from improved understanding of the requirements and the design, there are more specific and more tangible benefits to the efficiency of the testing that can result from the proposed domain-specific approach:

5.3.1 Prioritisation of the testing

As discussed earlier in this chapter, the nineteen test subcategories represent a "conceptual model" of what communications specific features need to be available first before other more complex features can be ready for testing, with subcategory 1 representing the most fundamental and must be assured first, while 19 represents the most complex that can only be assured last. Ordering the communications related requirements of a system accordingly should help in determining priorities and phases of the assurance activity, and determining when testing and assurance effort needs to be carried out for particular communications related features or components of the system.

One of the benefits of using such subcategories will be to optimise the test activity. By using the subcategories to highlight the dependencies between the communications related features of a system, the test analyst will be able to reduce the possibility of testing features too early or too late. Testing a feature too early during a test cycle means less confidence in how meaningful the test results are. On the other hand, testing a feature too late can potentially mean the IT project having to face delays while fundamental faults that are found late during the test cycle are corrected then retested. Other features of the system that can be affected by the change have to be regression tested as well, leading to additional cost and delay. Adopting a domain-specific test approach that can minimise such risks to the project's costs and timescales should bring significant benefits (Elbaum, Malishevsky, & Rothermel, 2001) (Elbaum, Malishevsky, & Rothermel, 2002).

5.3.2 Resolution of gaps and inconsistencies

As was mentioned earlier in this section, the structuring of communications requirements according to the nineteen subcategories can facilitate easier identification of potential gaps or inconsistencies in the communications requirements. This is because it reduces the complexity of the test analysis effort by turning a larger set of requirements into smaller pre-defined subcategories.

Should gaps or inconsistencies (ISTQB, 2012, p. 30) be identified at the requirements review or test analysis phase, the tester can then ensure these are not reflected in the final delivered solution by raising a "test risk" or by concentrating review efforts on specific areas in the technical specification. In such a case, the adoption of the framework is enforcing the tester's role as a positive participant in the project from the early stages rather than a passive recipient of requirements whose contribution to the project only commences at the start of the testing phase.

5.3.3 Improved synergy with the overall project plans

Test cases should not be the only or the core outcome resulting from the adoption of the communications layer test approach. The outcome of the communications specific test approach will be a range of actionable requirements-based Quality Assurance[50] (QA) activities such as design documentation reviews, inspections and demonstrations of system components while development is in progress. Grouping the requirements according to logical and domain-specific subcategories makes it easier to define what combination of QA methods are appropriate for each group of requirements, and at what stage of the project the assurance work should be carried out. Such an approach will result in better actionable QA activities that can be integrated within the overall project plans (ISTQB, 2012, p. 20). The adoption of the framework can therefore be used to structure not just the testing but also other QA activities throughout the lifecycle of the project.

5.3.4 Improved confidence in the results of other tests

By identifying all communications related requirements and treating them as a logical component of the overall IT project, the complexity of designing tests for the rest of the system's layers can be reduced when compared with categorising the communications requirements simply as functional or non-functional. Doing this can optimise the time and

[50] "Quality Assurance" in this context means Software Quality Assurance, a term often used in industry to either mean or include software testing.

resources available for the final acceptance stage of an IT project and can reduce the risk of major communications related problems being uncovered too late during final acceptance test activities, i.e. potentially improving confidence in the overall results and progress (ISTQB, 2012, p. 39) of the testing. Realising this benefit depends to a large extent on how well the communications requirements can be "standalone" and less interdependent on other non-communications requirements.

5.4 Chapter summary

The ideas of the domain-specific test framework presented in Chapter 4 need one further level of detail in order to be able to apply and evaluate them. Therefore, a domain-specific test approach for the communications layer was derived following a thought process comparable to the thought process that was followed to derive the overall layered test framework.

The outcome is a "conceptual view" of how the testing for any communications interface to communications-critical large scale system should be organised. This conceptual view is represented by the following list of "test subcategories":

1. The structure/architecture of the network interface, its components and layout, hardware, and wiring;
2. The communications protocols used;
3. The user terminals;
4. The data and messages that are transmitted by system over the network interface;
5. All possible types of senders and receivers over the network interface;
6. The different possible modes of transmission used;
7. How the transmissions are acknowledged by the receivers;
8. How the performance characteristics of the network can affect the system and it processes;
9. The services provided by the network interface to the system;
10. Performance and volume limits of the services provided by the network;

11. How the interface's QoS and SLA are guaranteed, maintained and reported;
12. The on-going operation, maintenance and administration of the network interface;
13. Fault handling processes of the interface, from detection to resolution;
14. Certification/compliance requirements that apply to the interface;
15. Resilience features of the interface;
16. Business continuity features of the interface;
17. Documentation: user and technical documentation;
18. Risks (technical and business);
19. Operational readiness of the interface.

The following chapter (Chapter 6) will present a case study that will be used to demonstrate how this approach can be applied in practice, and to evaluate its benefits.

6. Chapter 6: Applying the ideas of the test framework to a real-life CCLSS project

A CASE STUDY APPLYING THE IDEAS PRESENTED IN CHAPTERS 4 AND 5 ONTO AN INDUSTRIAL PROJECT FOR A COMMUNICATIONS-CRITICAL LARGE SCALE SYSTEM

6.1 FireControl introduction

The case study presented in this chapter was used to apply and further crystalize the ideas for a domain-specific test framework as already explained in Chapters 4 and 5, and then to evaluate the outcome. For this purpose, the communications related requirements of the *FireControl* project (FireControl, 2011) (National Audit Office, 2011) were used as the basis of the case study for this thesis.

FireControl is a project, started in 2004 and cancelled at the end of 2010, which was intended to modernise and improve the resilience of the call centres used by the Fire and Rescue Services (FRSs) in England. The overall requirements set of FireControl contained communications related requirements that are relevant for use as the basis for the case study of this thesis.

FireControl, as intended before the project's cancellation, was identified as an appropriate example of a communications-critical large scale system (as already defined in Chapter 1) because it was intended to be large scale covering nine control centres across the nine regional areas of England, it was meant to employ a range of technologies to deliver its services including wireless and wired communications, GIS location technologies and high availability distributed databases. The FireControl subsystems were intended to reliably and quickly handle all emergency incoming calls to all English FRSs and manage the mobilisation of the resources of these FRSs to incidents. It critically relied on the availability of communications networks services, both wired and wireless, to deliver its functionality and to communicate in real-time with real-life resources and users based on their geographic locations, identity, real-life attributes and availability status. The project is also an example of

how IT and communications technologies have converged to a degree where it is not easy to draw a clear boundary between what is IT and what is communications.

If the ideas of the layered test framework outlined in the Chapter 4 then expanded in Chapter 5 are to be applied easily to FireControl, the requirements should ideally have been written in a structure that can directly map onto the layers of the framework, which was not the case. FireControl requirements were not structured according to such a framework. Therefore, for the purposes of this thesis, its communications related requirements had to be identified first and listed in a spread sheet as a first step before test analysis work based on the framework could be done. The communications related requirements represented a subset of approximately 260 requirements out of a total of 2000+ FireControl requirements. Out of these, a subset of 90+ high level requirements were then identified as representing the technical communications features required for the project which are mainly not visible at user or business process level.

This subset of FireControl technical communications requirements was subsequently used to explore further the domain-specific test framework ideas described in Chapters 4 and 5. To avoid ambiguity, the "testing" discussed in this chapter is intended to mean requirements-based testing.

6.1.1 Discussion: FireControl's communications requirements

As many of the communications related features of FireControl belonged to the network communications domain more than the IT software domain, the testing of these features did not fit easily with the traditional IT definitions of functional and non-functional testing. Communications requirements of FireControl were fundamental requirements affecting both functional and non-functional features of FireControl as well defining the design and the configuration characteristics of its communications links. Therefore, their quality assurance and testing efforts could not be deferred until a relatively late acceptance testing stage when it is likely to be too late for correcting any faults found without causing significant delays and added costs. Additionally, the communications requirements were not listed under specific

communications sections in the FireControl requirements document but were included in a number of different sections.

The above were factors that made the testing of the communications features of FireControl according to generic test methodologies (e.g. V-Model) particularly subjective and dependant on the individual experience of the tester. This made the communications interfaces of FireControl good case study material for applying the ideas of this thesis and evaluating the benefits of these ideas.

6.2 Mapping FireControl's requirements to the test framework: identifying a communications layer

The first step to exploring the use of the domain-specific communications test approach described in Chapter 5 for the test analysis and design for FireControl was to identify FireControl's "communications layer" (Figure 3 Chapter 4). This was done by identifying the technical communications requirements which were spread over multiple sections of the FireControl requirements document.

Through a number of manual and automated searches and reviews of the requirements, a spread sheet was generated containing all communications related requirements defined within the main requirements document of FireControl (FireControl project, 29 March 2007).

Out of 2000+ requirements, approx. 260 requirements were originally identified as relating to the communications interfaces of FireControl. Further analysis on this subset of requirements was carried out to identify the communications related requirements that are technical and are not feasible to test at business process or user level. The resulting subset was then treated as the "communications layer" according to the overall test framework defined in Chapter 4. Further analysis of the remaining FireControl requirements and the main IT contract agreement also identified that elements of "Schedule 11" Service Level Agreement (SLA) parameters of the FireControl contract agreement also should be treated as part of the communications layer, as well as a number of overarching requirements that are

not specifically aimed at communications interfaces but are applicable to all systems and interfaces (e.g. documentation, service management and others).

Thus far, the outcome of the analysis effort was to identify what amounted to the "communications layer" of FireControl. This was represented by 90+ requirements, a number of Schedule 11 SLA parameters as well as a number of general requirements that are applicable to the communications interfaces.

This communications layer was then subdivided according to the communications interfaces of FireControl. For each communications interface, a table of requirements belonging to the interface was generated. The communications interfaces are: Firelink, Secondary Radio Bearer, Telephony, LAN and WAN. This resulted in five "communications test tables", included in Appendix 2. A sixth table was created to include communications related requirements that did not fit easily into one of the other five tables, but at the same time did not fit outside the communications layer. Because the sixth table was generated less systematically as a general repository for requirements that did not fit elsewhere, it will not be used for the purposes of this thesis and is not included in Appendix 2.

The requirements within each of the first five communications tables were then grouped according to the nineteen communications test sub-categories as explained in Chapter 5. Finally, test risks that were already identified for the project were reviewed to identify communications related risks. These were then included in the most relevant tables.

Following the above adaptation of FireControl requirements to the test framework's communications layer, it became relatively easy and "mechanical" to decide on the appropriate quality assurance method (mentioned in Section 5.3.3) and testing needed to verify each sub-category (see test tables in Appendix 2), as well as the appropriate timing relative to the project's phases. Also, the identification of any gaps in the requirements became easier and more objective once the communications requirements were organised according to the network interface and the communications test subcategory they fitted best.

6.3 Applying the communications test approach to FireControl's communications layer

To apply the communications test approach to FireControl's "communications layer" (as discussed in the earlier sections of this chapter), the communications requirements were initially categorised according to the following five communications interfaces of FireControl:

Firelink[51]
Secondary Radio Bearer (SRB)
Telephony
WAN
RCC LAN

For each of the above "categories", requirements were then further organised according to the following nineteen "sub-categories" as has been already explained in Chapter 5:

1. The structure/architecture of the network interface with FireControl, its components and layout, hardware, and wiring
2. The communications protocols used
3. FireControl user terminals
4. The data and messages that are transmitted by FireControl over that network
5. All possible types of FireControl senders and receivers
6. The different possible modes of transmission used
7. How the transmissions are acknowledged by the receivers
8. How the performance characteristics of the network can affect FireControl systems and processes
9. The services provided by the network to FireControl
10. Performance and volume limits of the services provided by the network
11. How FireControl QoS requirements and SLAs (Schedule 11) are guaranteed, maintained and reported

[51] Firelink was used as a "pilot" sub-case study before work commenced in the other four interfaces

12. The on-going operation, maintenance and administration of the network/its FireControl interface
13. Fault handling processes of the FireControl interface, from detection to resolution
14. FireControl certification/compliance requirements
15. Resilience features (of the interface)
16. Business continuity features (of the interface)
17. Documentation provided for FireControl, user and technical documentation
18. Risks
19. Operational readiness of the interface for FireControl go-live

The following five sections will represent, in five diagrams, the outcome of the re-organisation of FireControl's communications requirements and provide a visual overview of the contents of the five test tables in Appendix 2. The five diagrams (B.1-B.5) represent how the FireControl communications requirements and the subsequent test analysis and test design work were organised according to the domain-specific communications test approach.

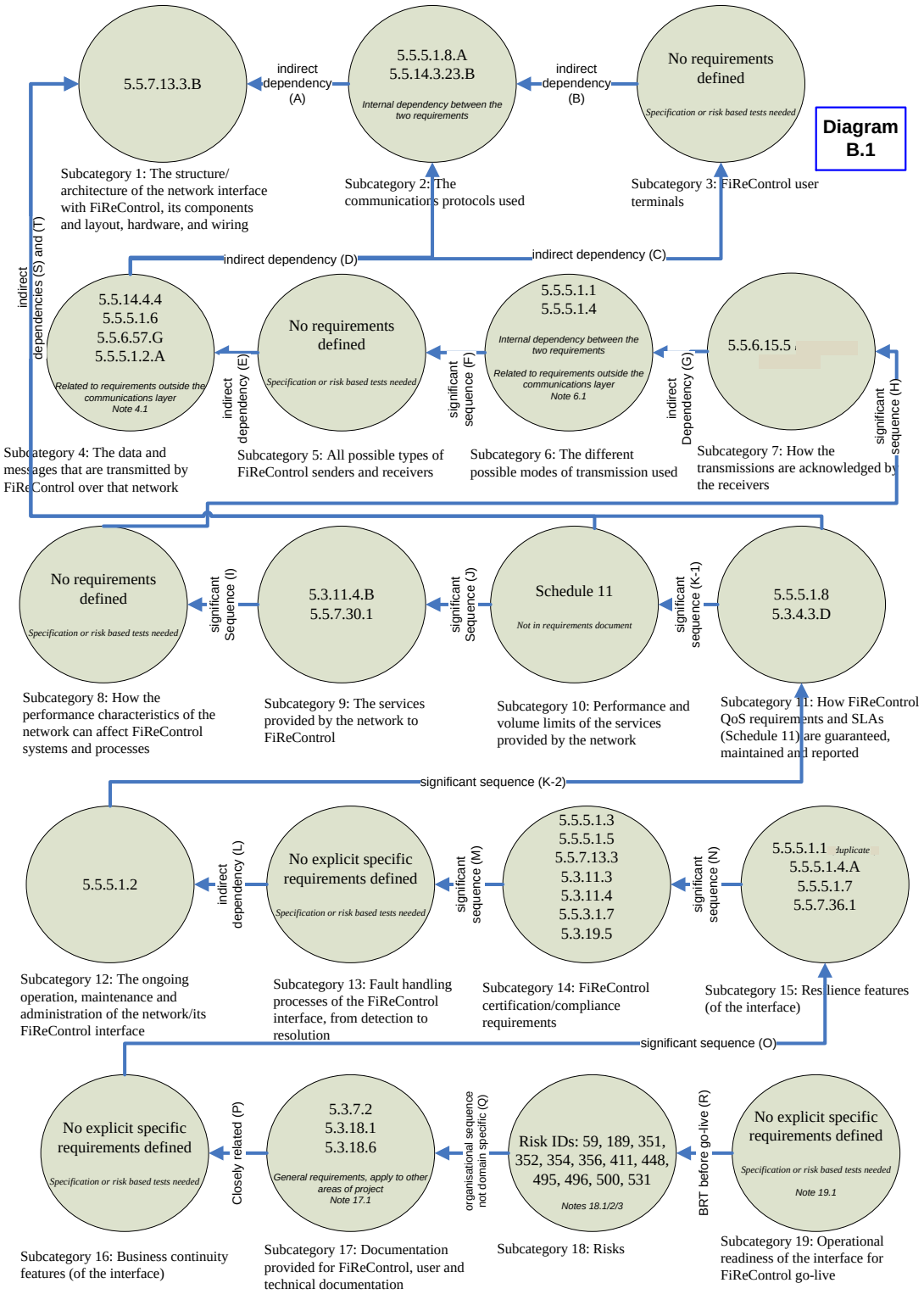
Each diagram shows how the (technical) requirements related to a network interface map onto the nineteen test subcategories, and then it highlighted relationships of precedence or priority between the subcategories[52]. The diagrams contain only the reference numbers of the requirements. The requirements together with the associated outline test cases are available in the five test tables in Appendix 2.

The purpose of the five diagrams (B.1-B.5) is to provide visual supporting evidence that the ideas discussed in this thesis of a domain-specific framework can realistically be applied to a real-life communications-critical large scale system, not just to one interface but consistently across five different interfaces.

[52] The Firelink diagram (B.1) was first created based on the nineteen subcategories (as presented in section 6.3). The relationships and dependencies between the subcategories were then reviewed and the arrows between the diagrams were created to represent such relationships and dependencies (see sections 6.4.1 to 6.4.4 for an explanation of the meanings of the arrows). Subsequently, the same diagram template, including the arrows, was used to map the requirements of the remaining four interfaces to create diagrams B.2, B.3, B.4 and B.5 to evaluate if the same relationships remained valid for the other four interfaces.

The next subsection contains the diagram for the Firelink interface requirements (B.1). It contains an explanation of the criteria used to decide which relationships exist between the subcategories. It then provides comments and notes on specific aspects of the diagram, followed by general comments. The subsequent subsections contain the diagrams for the remaining network interfaces (diagram B.2 to diagram B.5), and contain additional specific comments for each diagram where there is a notable variation from diagram B.1.

6.4 Firelink interface requirements organised according to the nineteen domain-specific subcategories



6.4.1 Explanation of the diagram: Why are the arrows there and what do they mean?

The purpose of diagram B.1 is to provide a single graphical view of how the communications test approach discussed in Chapter 5 was used to organise the testing for the Firelink interface requirements of FireControl. By providing such a view it will be easier to evaluate the benefits of the domain-specific approach, which was derived from proposed new test framework for communications-critical large scale systems.

The diagram shows the nineteen test subcategories with the requirements allocated to each of the subcategories for the Firelink interface. Where no requirements are identified then this is also indicated. The diagram is intended to show how, and if, the sequence of the subcategories from 1 to 19 remained valid and useful when the testing for the Firelink interface requirements is organised according to the domain-specific approach discussed in Chapter 5.

The diagram uses arrows to indicate the relationship(s) between subcategories based on the following criteria:

Direct dependencies:

- Whether that subcategory contains requirements that are explicitly dependent on other Firelink interface requirements, e.g. the dependency is mentioned within the text of the requirement[53].

Indirect dependencies:

- Whether the overall functionality or features described by the subcategory (i.e. as a whole group) depends on the functionality or features of other subcategories being

[53] No direct dependencies were identified between the Firelink interface requirements

available before it can be implemented and delivered for testing.

- Whether the overall functionality or features described by the subcategory (i.e. as a whole group) depends on the functionality or features of other subcategories being available before it can be meaningfully tested.

Significant sequences:

- Whether the sequence of the testing of two or more adjoining subcategories is significant and needs to be organised in a deliberate order[54] for more meaningful test results and to reduce the re-testing effort needed should faults be found.

- Whether the sequence of the testing of two or more adjoining subcategories is significant and needs to be organised in a deliberate sequence for non-technical reasons, e.g. project management or commercial reasons.

For example, dependency K-2 is defined as a significant sequence because it would be more efficient, but not essential, to carry out the operation, administration and maintenance (OA&M) tests relating to Subcategory 12 after the performance and SLA related tests to ensure that the reporting functionality for the interface is sound before attempting to test its OA&M functionality.

6.4.2 Diagram assumptions

- Communications layer selection from the total FireControl requirements is complete and accurate.
- Firelink requirements selection from the communications layer is complete and accurate

[54] Which may not necessarily be mandatory or the only viable order

- Interdependencies external to the communications layer are outside the scope of this discussion; although samples of them are explained with additional notes (e.g. note 4.1 below).

6.4.3 Specific comments on individual arrows (relationships) in the diagram

The relationships between the subcategories that are represented by arrows on diagrams B.1 and B.2 were derived as part of the case study evaluation exercise. By checking that the same relationships that were derived for the Firelink interface remained valid for the other four interfaces represents evidence of the potential for re-usability of the test approach for other CCLSS comparable to FireControl**.**

Below are further comments specific to each arrow on the diagram, indicated by labels (A) through to (T) as shown on the diagram.

(A) Subcategories 1 and 2 are fundamental to the functionality of the network interface, in the case of Firelink requirements their precedence can be interchanged, i.e. protocol first before the architecture. From a testing and project scheduling point of view, checking of the architecture, structure and the fixed aspects of the network interface ought to be done first before any other testing because such aspects are likely to be more difficult to correct during later stages of the project should a fault be found. In general terms, communications protocol configuration may vary depending on the chosen fixed structure of the network interface (e.g. locations, distances, physical medium) hence it is placed second in the list of subcategories. This sequence also reduces the need for retesting Subcategory 2 should a problem be found with Subcategory 1 features.

(B) User terminals vary according to the protocol/networking technology used. In the case study, the adoption of the proposed domain-specific organisation highlighted the fact that the requirements do not explicitly mention what user terminals should be used. This would guide review effort of the design documentation and allows the generation of further test

cases/conditions which could have otherwise been missed out if the test design was only based on the stated requirements.

(C)/(D) Testing of message and data types transmitted over the network interface comes after the checking of both the protocol (i.e. the networking technology deployed) and the user terminals used to transmit these messages. This organisation is well suited to the case study and shows the structure works to organise and schedule the test activity and reduce the likelihood of earlier testing becoming invalid if problems are found with Subcategory 4.

(E) From a test scheduling point of view, confidence that the network interface is capable of transmitting all types of messages needs to be established first before checking that different user types are able to use the system.

(Note 4.1) Requirement 5.5.6.57.G in Subcategory 4 is further detailed by 5.5.13.3.9.A, 5.5.14.2.G, 5.5.14.3.5, 5.5.14.4.13.D, and others outside the communications layer. Requirement 5.5.14.4.4 is related to and further detailed by 5.5.6.25.3/6, 5.5.6.32.3 outside the communications layer because they were not considered as technical communications requirement, but rather it relates more closely to the "functionality layer" of the framework (Figure 3, Chapter 4)

(F) Sequence of testing for Subcategory 6 and the preceding subcategories is significant and needs to be explicitly scheduled. Other sequences may be valid but need to be based on domain-specific knowledge.

(Note 6.1) Requirement 5.5.5.1.1 in Subcategory 6 is related to 5.5.14.3.23.C

(G) Indirect dependency of Subcategory 7 on Subcategory 6. The acknowledgement protocol is secondary to the transmission mode from a functionality and test viewpoints.

(H) Impact of the performance characteristics of a communications interface on the systems and processes that utilise it (Subcategory 8) need to be assessed after more fundamental features of subcategories 1-7 are confirmed.

(I) /(J) The sequence of the testing for subcategories 8/9/10 needs to be explicitly scheduled and defined according to domain-specific considerations. Not having such domain-specific sequence is likely to lead to the need for more regression testing should problems be found.

(K-1)/(K-2) Order of subcategories 10/11 is less critical. Other options based on domain-specific knowledge may be possible, but such grouping should lead to better test efficiency compared to a non-domain-specific approach.

(L) A gap is highlighted by the lack of explicit requirements under Subcategory 13. Fault handling is a subset of overall OA&M, hence needs to be tested after a more general evaluation/test of the overall OA&M functionality provided for each interface.

(M) After all functionality and SLA features are confirmed, test activity can then be focused on checking standards compliance, e.g. ITIL certification.

(N) Resilience features of the interface represent the most complex and advanced features to test, more basic subcategories need to have been tested to ensure that the resilience features of the interface are ready for testing.

(O) Business Continuity features and plans to be tested in more detail (e.g. including non-technical aspects) after the resilience of the interface has been tested.

(P) Subcategories 16/17 are closely related and need to be tested together, although they are logically organised as two different groups.

(Note 17.1) These are general requirements that cover the whole system. They could be easily overlooked when testing a particular communications interface. The domain-specific communications test approach can help highlight where and when they should be tested, and ensure adequate coverage for them when testing the communications layer.

(Q) Risks are left till last for project organisation, rather than technical, reason. They can alternatively be treated as additional requirements and placed within the most relevant subcategories.

(Note 18.1) Subcategory 18 is an example of how the domain-specific communications test approach does not preclude adapting other industry practices and methodologies, e.g. ideas from risk-based testing.

(Note 18.2) Gaps in subcategories highlighted by this approach can be used as a source for generating risks in a structured way.

(Note 18.3) Risks can be further identified within each individual Subcategory by reviewing its content and determining if the Subcategory has been sufficiently and clearly defined.

(R) Business Readiness Testing (BRT) Subcategory 19 for each communications interface is placed at the end after all testing is done and just before go-live of the system.

(Note 19.1) This domain-specific approach helps highlight how much was covered or not covered and gives a useful (domain-specific) indication of test coverage. This is one further potential benefit of the domain-specific approach.

(S)/(T) Performance and SLA features (subcategories 10/11) are dependent on Subcategory 1. A change to the features defined in 1 could impact many other subcategories, but is particularly relevant to subcategories 10/11.

6.4.4 General comments on diagram

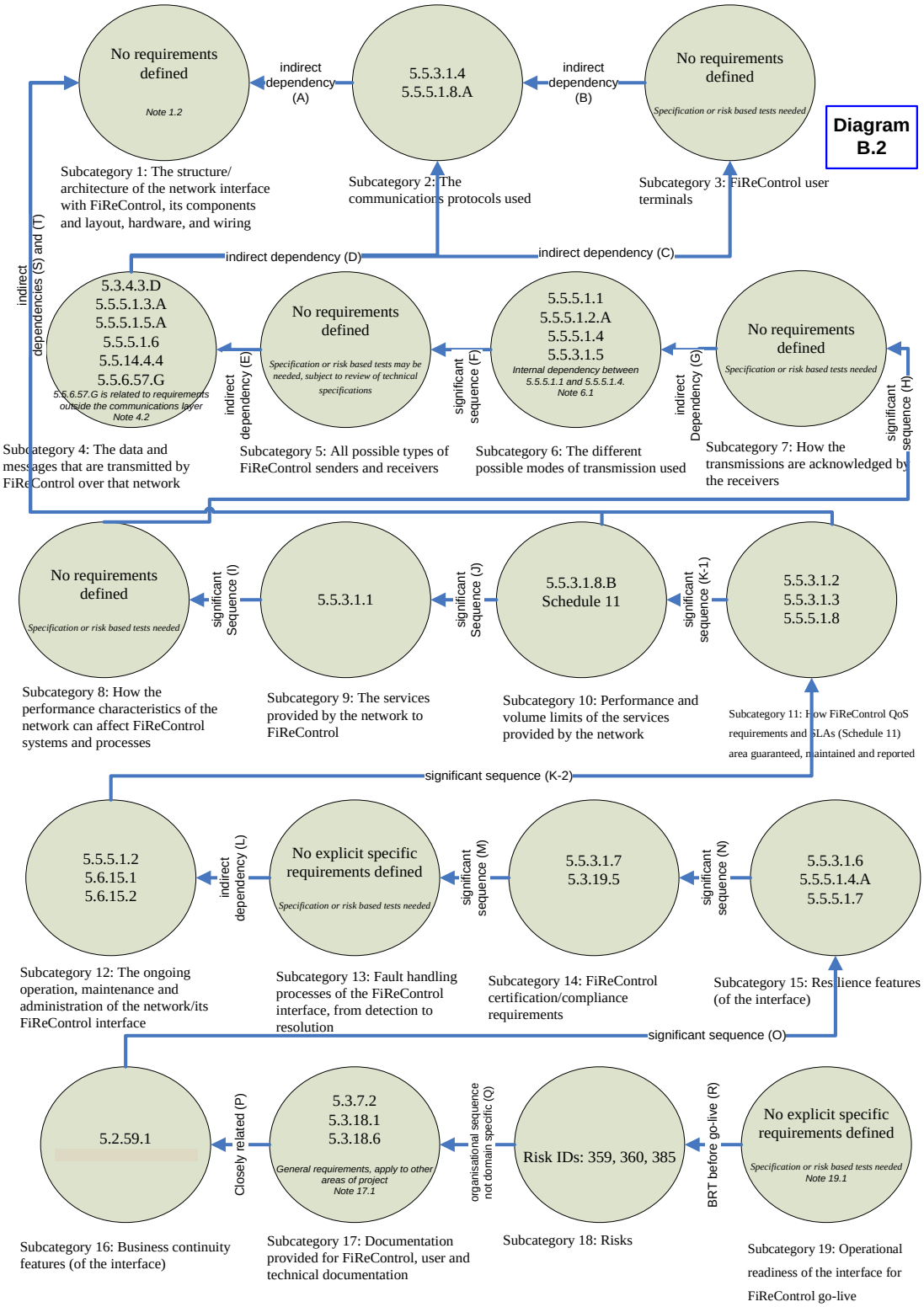
Further general comments on the diagram:

- Dependencies between the subcategories are not always obvious without domain-specific knowledge.

- Gaps in the requirements are highlighted using the proposed approach, allowing more effective and pro-active review of design and specifications and also structured test risk generation.
- Sometimes the sequencing between the subcategories is not critical or can have other valid alternatives, but the fact that it is explicitly defined is a benefit compared with a generic test approach.
- Such a diagram would be a useful basis for defining requirements from the start if adopted early in a project's lifecycle. Doing so would represent a useful convergence between requirements capture, review and test activities and could potentially help streamline a project's activities and make its planning more viable.
- The diagram helps highlights gaps in requirements where a subcategory does not have any requirements defined. The contents of each individual subcategory can be further analysed to assess their completeness.
- The exact organisation of the nineteen subcategories is not necessarily the only valid one or the best possible one. However, what is intended is that it offers an improvement over a non-domain-specific approach or standard (e.g. V-Model) that conceptually organises the testing according to assumptions about the processes used to develop an IT system. Needless to say, such assumptions that evolved in the past cannot necessarily remain correct or accurate for all IT systems now or in the future, especially for the type of systems this thesis is intended for.
- Review effort of the technical specifications can be better planned when having such a diagram as an "aide memoir" because it prompts the reviewer to consider all relevant sub topics rather than be guided mostly by the organisation of the technical design documentation.

6.5 Secondary Radio Bearer interface requirements organised according to the nineteen subcategories

The diagram below maps the SRB interface requirements to the nineteen test subcategories:



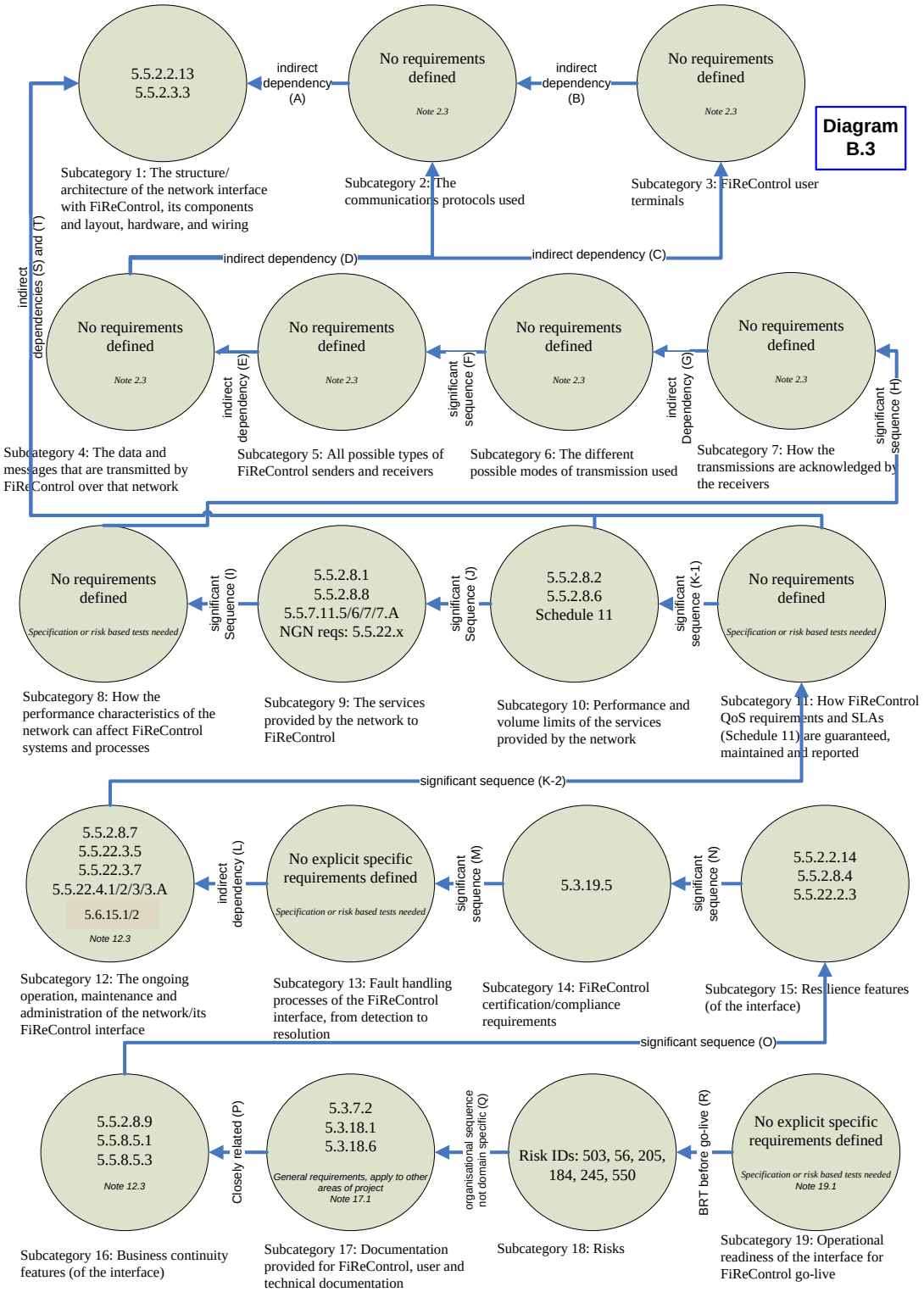
The diagram shows that the template used for the Firelink interface (in diagram B.1) remained valid for the SRB interface. Below are general comments on diagram B.2 (SRB) variations from diagram B.1 that are notable for the purpose of this thesis, else see the notes associated with B.1 in sections 6.4.1 to 6.4.4:

Note 1.2: For SRB no need to specify structure of the network because it is using one of the commercial cellular networks in the UK.

Note 4.2: For SRB, more requirements are focused on services offered by the network.

6.6 Telephony requirements organised according to the nineteen subcategories

The diagram below maps the Telephony requirements to the nineteen test subcategories:



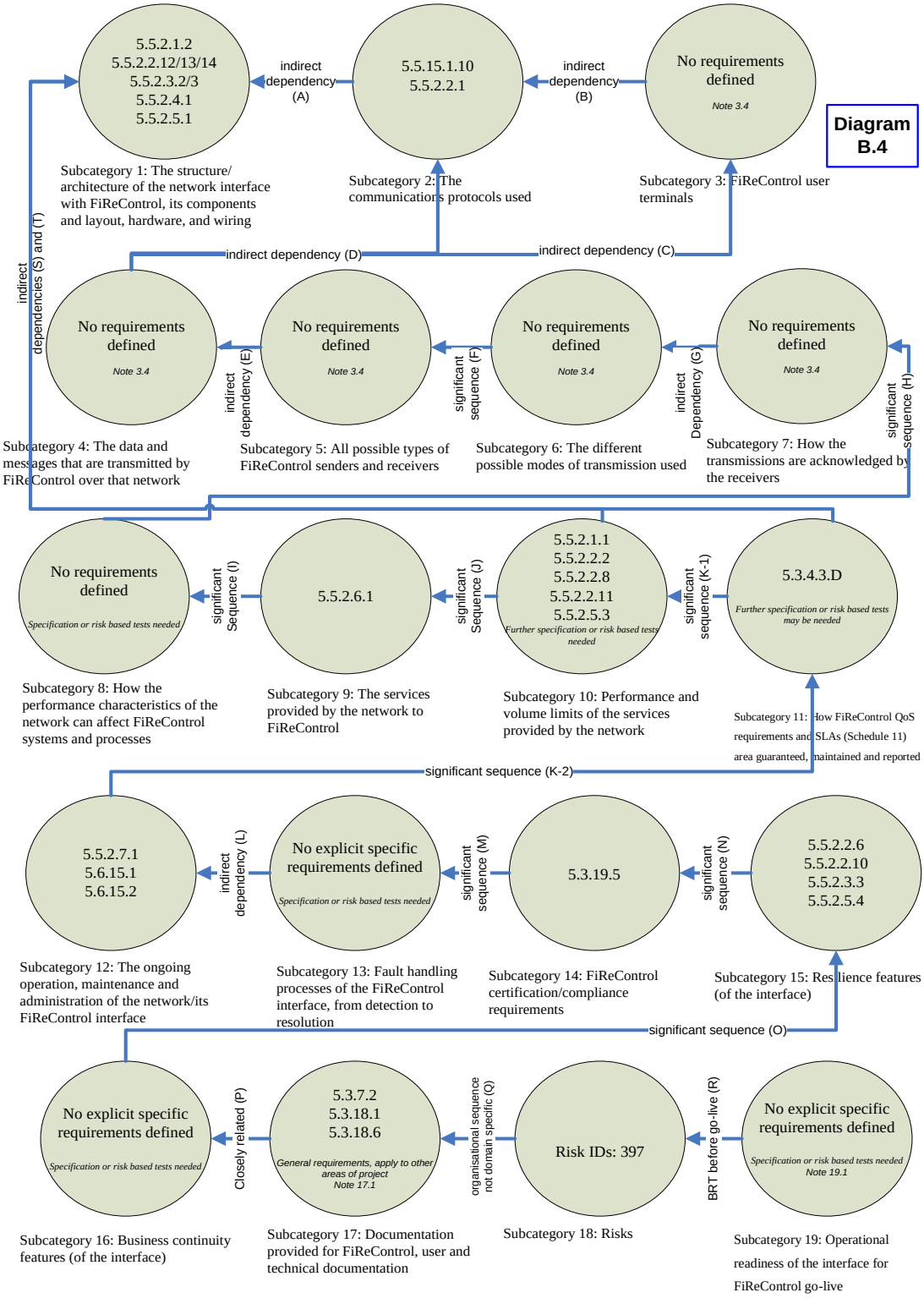
The template used for the Firelink and SRB interfaces (in diagrams B.1 and B.2) remained valid for the Telephony interface. Below are general comments on diagram B.3 (Telephony) variations from diagram B.1 that are notable for the purpose of this thesis, else see the notes associated with B.1 in sections 6.4.1 to 6.4.4:

Note 2.3: Telephony networks use established widespread technology, hence there is no need to specify protocol, user terminals, types of traffic, or other detailed technical aspects of the technology of the network.

Note 12.3: Subcategory 12 for telephony is relatively more significant than for other communications interfaces because routing and configuring of emergency telephony is central to the correct operation of the network of the RCCs.

6.7 WAN interface requirements organised according to the nineteen subcategories

The diagram below maps the WAN interface requirements to the nineteen test subcategories:

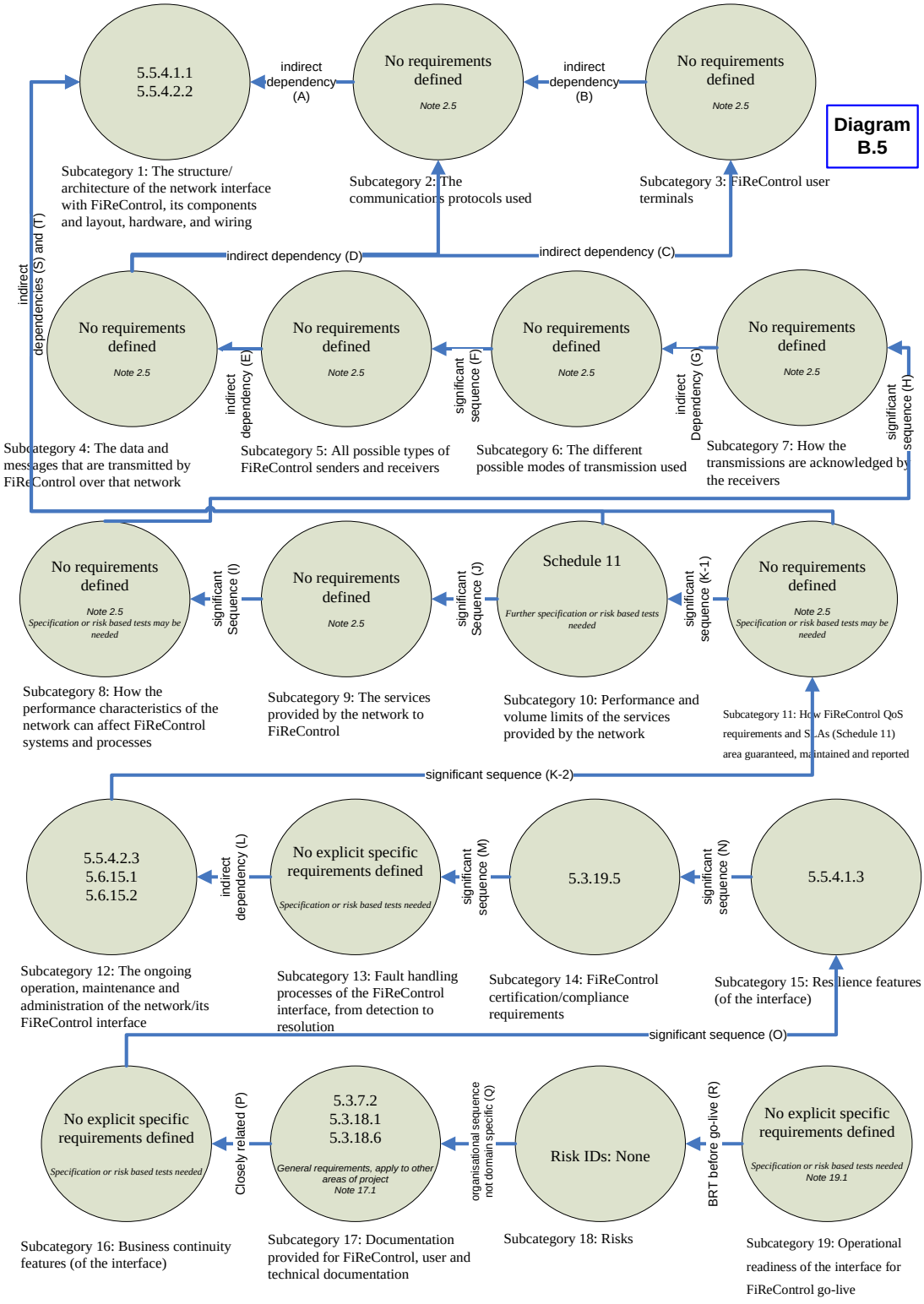


The diagram shows that the template used for the Firelink, SRB and Telephony interfaces (in diagrams B.1, B.2 and B.3) remained valid for the WAN interface. Below are general comments on diagram B.4 (WAN) variations from diagram B.1 that are notable for the purpose of this thesis, else see the notes associated with B.1 in sections 6.4.1 to 6.4.4:

Note 3.4: The WAN was treated in the FireControl requirements document as an outsourced service, hence no requirements in subcategories 3-8.

6.8 RCC LAN requirements organised according to the nineteen subcategories

The diagram below maps the RCC LAN interface requirements to the nineteen test subcategories:



The diagram shows that the template used the Firelink, SRB, Telephony and WAN interfaces (in diagrams B.1, B.2, B.3 and B.4) remained valid for the LAN interface. Below are general comments on diagram B.5 (RCC LAN) variations from diagram B.1 that are notable for the purpose of this thesis, else see the notes associated with B.1 in sections 6.4.1 to 6.4.4:

Note 2.5: RCC LAN was treated in the FireControl requirements document as an RCC infrastructure and building services matter, hence not much is stated in the requirements about its technology.

6.9 Additional ideas relating the new test framework

6.9.1 Organisation and management of the test activities

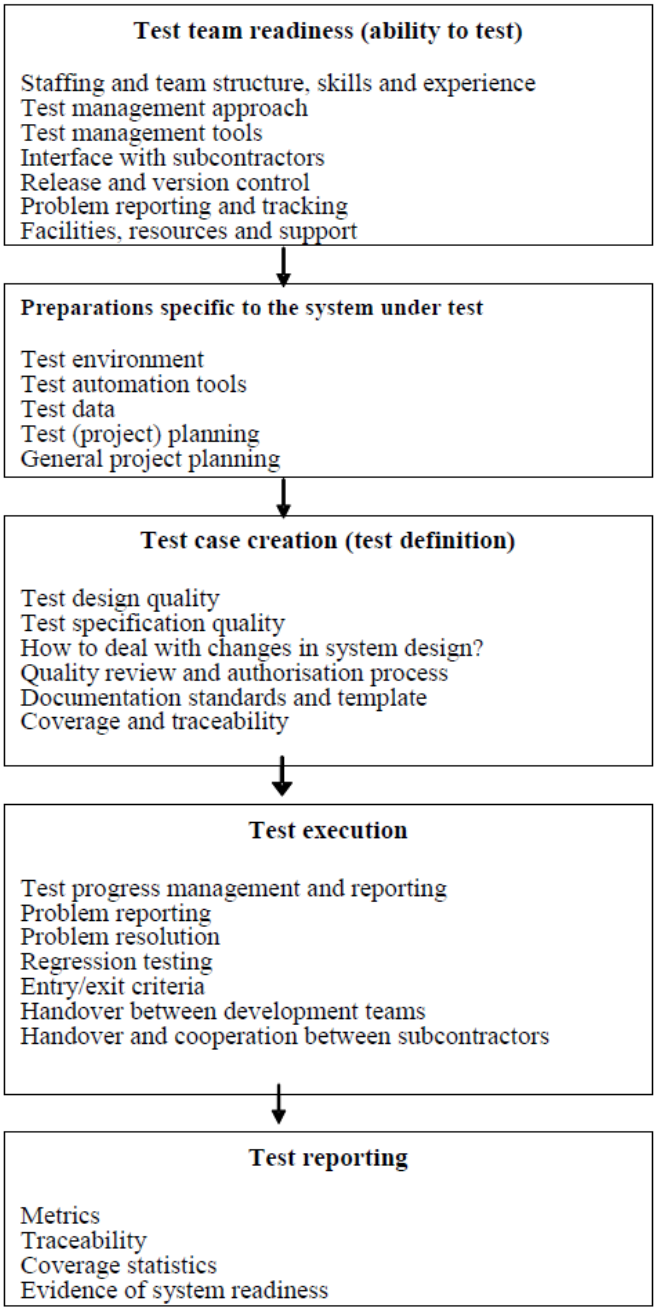
The new test framework requires the support of a widely used test management tool to become feasible commercially and widely practiced. Below is an example of how this might be achieved with a widely used test management tool such as HP Quality Centre (QC) (HP, 2013). This subsection is intended to provide an example of how an existing test management tool might be adapted to support the proposed test framework. It does not rule out other tools, e.g. IBM DOORS (IBM Rational, 2013) and Rational Quality Manager (IBM Rational, 2013), nor does it indicate a recommendation of QC as a commercial product.

If QC is to be used to support such an approach, then the communications requirements should be labelled with their own test categories and subcategories. Each requirement should also be tagged with the applicable assurance and test methods (i.e. review, demonstrate, inspect, test), and be allocated a phase of the project when each assurance or test activity should be carried out.

A user of QC should be able to search for communications requirements belonging to a specific category and subcategory, and be able to view what assurance methods should be applied to that requirement and during which stage of a project. Where a requirement is tagged for testing, test cases should be generated for it as normal in QC's Test Plan section. However, a way is needed to also use QC to manage and track the progress of other non-test assurance activities. QC is not intended for such usage but it can be adapted, e.g. it is likely to be feasible to configure further distinct types of "test cases" to manage review, demonstration and inspection activities. Also, QC is not intended to support graphical representations of test cases, such as FSMs or MSCs but it can be adapted by making use of document attachments to its test designs or test steps.

As well as using a test management tool such as QC as a repository for creating, managing and tracking the test cases, an overarching test management plan is needed for any large scale IT project.

The following diagram represents the testing activities that need to be planned and managed to ensure an effectively tested system.



6.9.2 Non-functional testing brief discussion

Although non-functional[55] testing is outside the intended scope of this thesis, especially in terms of the level of detail of the new test framework, this section provides a brief discussion on how non-functional testing might be incorporated in the new test framework. This discussion is intended to highlight further possibilities for adapting the framework's ideas to cover non-functional testing areas such as performance, stress, volume, or others for a project such as FireControl.

FireControl's non-functional requirements primarily defined Service Level Agreement (SLA) parameters and thresholds for FireControl's systems. These requirements would not be sufficient as the only source for requirements-based non-functional test design that is comprehensive, objective and independent from the vendor's interpretation of the requirements.

How can FireControl's requirements-based non-functional test design effort be comprehensive, objective and independent? How can the test analysis activity detect gaps and risks not explicitly stated in the requirements or in the technical specifications?

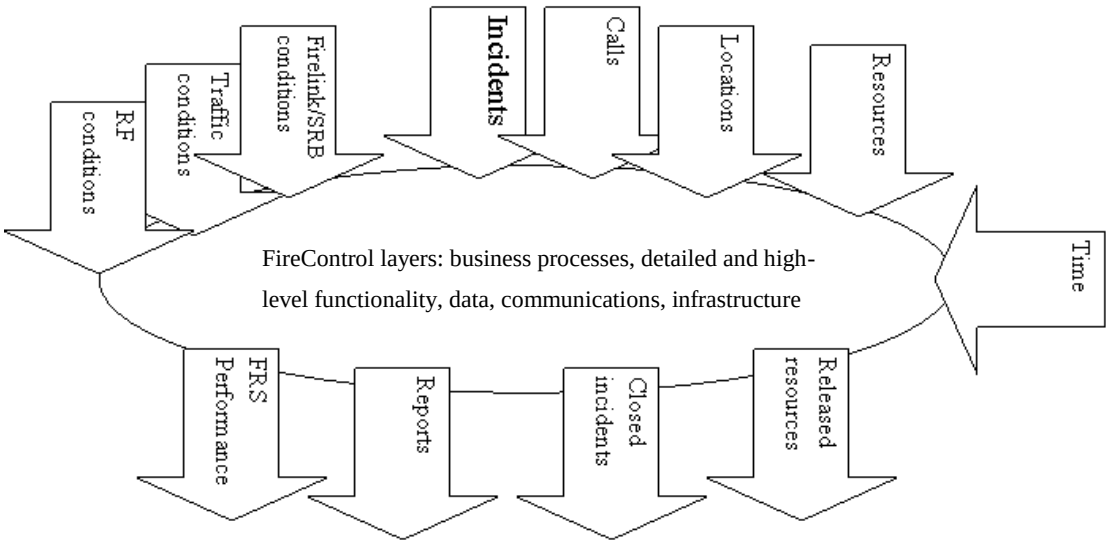
A conceptual non-functional framework can be defined as an additional dimension or extension to the proposed test framework. This framework can be based on the Model-Based Testing (MBT) principle of testing a system according not only to how it is built but according to its external inputs and outputs (Siegl, Hielscher, & German, 2010).

A non-functional test framework can be constructed based on its real-life inputs that are derived from the requirements, i.e. these are the real-life external variables that FireControl subsystems need to process (see diagram below). This (non-functional) test framework can

[55] Defined in (Graham, Veenendaal, Evans, & Black, 2008, p. 47) as *"the testing of the quality characteristics, or non-functional attributes of the system"*

then be used to identify the potential non-functional weakness areas by identifying worst case input scenarios that are capable of stressing the FireControl "layers", i.e. business processes, functionality, data, communications interfaces and infrastructure.

The diagram below represents what a non-functional test framework, as an extension of the overall framework presented in this thesis, could look like. It can be used as conceptual basis for defining FireControl's non-functional requirements-based tests for each of the test six layers. This can be achieved by defining worst case combinations of inputs into FireControl and then analysing their impact on each of the six test layers of the system under test (e.g. FireControl).



The non-functional test design can be detailed further using additional diagrams representing each specific layer and derived from key business processes/operational scenarios. For example, there can be further diagrams representing each of the communications interfaces and modelling how extremes of external inputs can affect each of the interfaces. Another example is a diagram modelling GIS data volumes and their effect on the datasets of FireControl. Such diagram could be used to identify the tables in the data schema where testing efforts (using large data volumes) need to be directed. A third example of a non-functional test diagram can be a diagram modelling AVLS location updating activities, which

can be used to identify test risks of inaccurate (e.g. delayed, out of sequence, missing) location data of fire appliances affecting the behaviour of the system.

Another possibility for incorporating non-functional testing specifically into the communications test approach would be to treat the non-functional areas as additional communications test sub-categories (e.g. performance, volume, stress, disaster recover, and service).

The testing for each of the additional "non-functional" subcategories can then be detailed using one or both of the following:

- Include the input/output scenarios derived using the sample non-functional test framework.

- For each requirement, identify whether that requirement contains or implies information relevant to any of the non-functional subcategories. An example of this is the requirement for plume modelling data to be transmitted over Firelink or SRB to MDTs (5.5.14.4.4). Although this requirement would traditionally be classified as a functional requirement, it can also be used as basis for defining performance, stress and volume tests for both Firelink and SRB interfaces because plume modelling is likely to be data and bandwidth intensive functionality.

6.10 Chapter summary

The case study presented in this chapter demonstrated how the ideas of the new test framework and, specifically, its communications layer test approach can be applied to a real-life project for a communications-critical large scale system such as FireControl. As well as demonstrating that the concepts presented in Chapters 4 and 5 can actually be applied across five different network interfaces, it also discussed the benefits that can be observed from applying them.

The fact that the proposed nineteen test subcategories could be uniformly and consistently used for five different communications interfaces of FireControl (as shown in diagrams B.1

to B.5 in sections 6.4 to 6.8) is a notable outcome of this exercise, especially since the FireControl requirements were not written to fit the test framework proposed by this thesis.

Lastly, this chapter included discussions and samples on how the proposed framework can be adapted further for wider adoption in industry, and discussed ideas for how the framework can be adapted and refined to also incorporate non-functional testing.

The next two chapters (Chapters 7 and 8) will provide a more detailed evaluation of the benefits of the applying the ideas of the test framework and the communications test approach to FireControl as well as other CCLSS.

7. Chapter 7: User-based evaluation

7.1 Introduction

This chapter presents a user-based evaluation of the new test framework by expert external participants. This evaluation is intended to form part of the evidence presented in the next overall evaluation chapter (Chapter 8). This user-based evaluation was conducted in the form of a "goal-free", "ex post", "summative evaluation"[56] by potential users of the new test framework. The participants agreed to complete an evaluation form asking them to comment on the feasibility and benefits of adopting the new test framework for the testing of communications-critical large scale systems/projects they are familiar with or have worked on in the past.

The intention behind this user-based evaluation is to improve the overall reliability of the research by involving multiple participants and including more than one case study in the evaluation.

7.2 How the evaluation was conducted

Four external participants agreed to take part in an evaluation exercise of the new test framework. The purpose of this part of the overall evaluation of the new test framework was to provide an additional dimension to the evaluation by involving other external participants (other than the author) in the evaluation who can evaluate the ideas of the thesis and comment

[56] "Goal free", "ex post" and "Summative" are terms adapted from (Verschuren & Hartog, 2005). The terms describe the research methodology context of this additional evaluation work. "Goal free" refers to nature of the evaluation because the participants will be primarily asked to provide their general feedback and opinions to open ended questions. "Ex post" refers to the fact that the evaluation was carried out after the new framework was formulated (for the purposes of this thesis only, as there remains potential for further work). "Summative" refers to the fact that the evaluation is being carried out by participants that were not otherwise involved in this research work, representing potential users of the framework.

on whether they believe it can be feasible as well beneficial to adopt for other CCLSS projects that they worked on in the past. Their role in the evaluation would be as potential users of the new test framework, and they would be contributing their experience of CCLSS testing on IT projects other than the FireControl project used for the case study in this thesis. Additionally, one of the participants[57] agreed, and is suitably qualified, to map a sample set of communications requirements[58] taken from a CCLSS project[59] to the 19 test subcategories for the communications layer (Chapter 5).

The participants were selected according to the following criteria:

1- Long and varied IT expertise with good understanding of testing
2- Past experience working on a communications-critical large scale system/project/programme
3- Experience and visibility of how testing for that CCLSS was conducted
4- Testing specialism was intentionally not mandatory, but understanding of testing and past involvement was needed. The rationale is that involving only test practitioners with long experience of testing may introduce bias as they are more likely to have well established preferences, opinions and practices related to testing and may be more resistant to new ideas. Only one of the participants was what could be described as a professional test specialist but his background was in communications systems testing rather than large scale software testing.

The participants' job titles in their place of employment are: communications test consultant, enterprise architect, business analyst and a project manager.

[57] The same participant also mapped the Firelink requirements to the nineteen subcategories. See Section 8.9.3.

[58] This set of requirements was three "sanitised" fragments taken from the overall requirements set of a communications-critical large scale system used by an emergency service. The project is comparable in scale and complexity to FireControl. Appropriate approval to use this material was obtained from the relevant owning authority within the organisation. A Brunel University ethical approval was obtained prior to the commencement of this part of the evaluation involving external participants. Appendix 4 contains material and artefacts relating to this evaluation effort.

[59] Other than FireControl but comparable to it in scale and complexity

The participants were provided with a summary of the ideas of the thesis, and then they were given the opportunity to discuss it with the author before completing the evaluation form as well as the Brunel University consent form.

The replies were provided by the participants using a combination of email replies, hand-written form and, on two occasions, the author met the participant and filled the answers to the questions in the form while the participant was dictating the answer. All relevant responses material is provided in Appendix 4.

A pilot evaluation with the first participant was carried out to assess the suitability and clarity of the questions before the remaining three participants were provided with the evaluation form.

7.3 The summative evaluation form

Screenshots of the seven pages of the evaluation form are included in Appendix 4 (Section 13.3).

7.4 Responses to the evaluation questions

The responses provided by the four participants are shown below. The participants will be referred to as IA (first participant who did the pilot evaluation), IM, RJ and KK.

Question 1:

Please provide a brief description of the "communications-critical large scale" IT project/system that you were involved in which you will use as basis for evaluating the ideas presented in the attached paper titled "A new test framework for communications-critical large scale systems". This project/system will be referred to as "CS2" in the next five questions.

IA: *"The system I was involved with consisted of switching sites containing WAN & LAN switching equipment, Routers, Servers, data storage, network management entities and inter-site communication network circuits linking radio base station sites to these switches"*

IM: *"Banking programme involving a lot of applications and network infrastructure, servers. Applications covered: branch applications, general ledger, retail banking, debt collection and others"*

<This section has been removed from the Open Access version of this thesis to preserve confidentiality>

Question 2:

Please provide an outline of your involvement in the CS2 project/system and your familiarity with how the testing was conducted on CS2 before it was delivered into live service

The purpose of this question is to capture information about the evaluator's technical and professional background and knowledge of CS2, which can be treated as the context for his/her feedback and to assess their suitability to conduct the evaluation and their potential preferences/bias

IA: *"I have been involved with this system on several projects and for a variety of clients. On one project this was in the capacity of an engineering consultant working for one of the preeminent UK consultancy companies. The role involved ensuring that all the requirements of the end user were proved to have been met through test & assurance activities. This involved reviews of the design architecture, test strategy and test script reviews produced by the supplier and test witnessing activities. On another project I was the Test Manager ensuring that the enhancements to the system were fit for purpose and that they were ready for operational use. Again this involved reviews of supplier technical documentation and test witnessing activities. In addition I designed User test scripts and managed the User testing phase of this project. I am a member of I.E.T. and an Incorporated Engineer with 30 years experience of installing and testing large scale complex communications systems"*

IM: *"I built the capacity management and analysis software for the network and tested the network and all its functions. Familiarity with the testing: yes, I guess so, had some knowledge of what the testing was intended to achieve"*

RJ: *"I was initially responsible for the requirements/requirements management of the system. I was also involved in system design and testing. Later on I became fully responsible for the system testing in the later part of the project's lifecycle. I worked closely with the system's suppliers to agree client acceptance of the system across two main sites"*

<This section has been removed from the Open Access version of this thesis to preserve confidentiality>

Question 3:

Based on the "new test framework" paper you read, and according to your professional opinion, could the ideas about the new test framework have been applicable and feasible for use with the CS2 project/system as basis for designing and conducting the testing?

Key evidence sought from this evaluation is whether the ideas of the new test framework are possible (or otherwise) to apply to real life projects. The purpose of this question is to determine whether the evaluator believes the ideas can also be applicable to another real life project comparable in scale and complexity to CCLSS as described in the paper

IA: *"I believe the ideas set out in this paper could be applicable and feasible to use with the CS2 project / system. The theoretical approach / process has merit in that it provides a logical approach to testing and requirements coverage. It is also useful in highlighting 'critical paths' in the testing. As to adoption of the process in a commercial environment, I believe this would be dependent on proving that there are tangible benefits in terms of time, cost and quality"*

IM: *"Probably, as a framework. We had domain experts who tested their own stuff. We had network domain experts, applications experts testing the applications, server experts testing the servers. We had to do a lot of work to test the data/transactions going across the network.*

Test subcategories 1, 2, 3, 4, 5 would have been relevant. I had to understand all these areas, plus subcategories up to 10 (subcategories 1-10 relevant). <a diagram was drawn to explain>"

RJ: *"In one point it would have definitely supported more consistent testing and unified the overall test approach across the whole project/programme. There was no common/shared template to unify all testing activities and prioritise them due to the complexity of the project"*

KK: *"Yes, the ideas can be applied to comparable real life projects provided adequate Security and Service management functionalities are also built into the test framework. This test framework backed up by a robust Test Strategy and scripts would help achieve a successful outcome. A good test script needs to be developed for each component and end to end testing. The test strategy will address issues surrounding the what, when, how and why etc. for each domain in your pyramid model"*

Question 4:

What advantages and disadvantages do you think could have resulted if the new test framework was adopted as the conceptual basis for test analysis and test design for CS2?

The purpose of this question is to provide additional general feedback and assessment of the potential benefits/pitfalls as expected by the evaluator when considering the possibility of the new test framework being used as the basis for CS2 testing

IA: *"The advantages are that it provides a logical approach to the testing enabling key elements to be focused on in terms of test design and execution and can aid a risk based approach to testing. Adopting this approach may have highlighted earlier on in the project some of the riskier areas of the system design and functionality. Testing these earlier on and resolving any issues found could lead to reduced project costs and improved completion dates. Disadvantages could be that it may appear complex on the face of it with the number of categories involved in the mapping of requirements. However, I believe that once experience was gained on this process these perceptions would be unfounded. There could be some difficulties for some individuals, mapping the requirements against the categories"*

IM: *"We did not have a framework to explain interdependencies and reduce them because we are domain experts, we knew what we need to eliminate first and in what order. Now if people working on such programme are not networking experts then such a framework – people need frameworks such as this now because they understand such aspects less well. People doing applications testing knew they did not have to worry about the network. The dependency framework is quite good, forces people to think of the hierarchy of requirements and design and re-introduces some of the rigour of the early 90s when people were more specialised"*

RJ: *"The project's testing had a high reliance on domain experts which caused a variety of approaches to testing and <around?> completeness and continuity of test activities. I think the new test framework could have helped with this aspect. One other benefit is that the model (framework) is re-usable and easy to adopt for the type of system it is aimed at. Its benefits will be potentially more noticeable for the complex systems"*

KK: *"It is difficult to see how successful it will be until it is applied to a real project. A robust Test Strategy and Test Script backed up by an experienced team of testers will be required for all large Communication projects to succeed. In a large Communications system, successful delivery of voice and data will depend on the quality and reliability of the infrastructure and hundreds of different components within the overall test life cycle. Proof of pudding will be in the satisfaction of the end users"*

Question 5:

Please indicate your opinion on whether the new test framework is likely to fulfil the following criteria:

The purpose of this question is to provide additional specific feedback and assessment on whether the new test framework is capable of fulfilling a number of specific criteria if used as basis for CS2 test analysis and design

The criteria list (in the criteria table below) presented to the users for evaluation was derived from the lists of criteria in Sections 4.2 and 4.3 which summarise what is meant by the term "test framework" for the purpose of this thesis. The list crystallises the ideas that link the early identification of the need for a domain-specific test framework CCLSS, presented in

Chapter 1, to the development of the layered model as presented in Chapter 4. It represents the intended form, usage and attributes of the new framework to make it appropriate for use as basis for CCLSS test analysis and design. It was incorporated in the summative evaluation form for the purpose of having the external participants comment independently on whether (and the extent which) the originally intended criteria was fulfilled by the new test framework. Below is an explanation of the motivation for the items in the list one-by-one:

1. **Suitability for testing communications-critical large scale systems**

   This criterion reflects the initial objective of this research (Section 1.2). Its inclusion in the evaluation form is intended to link the user-based evaluation with the initial objective of the research, i.e. to evaluate whether the outcome of the research is as was originally intended.

2. **Enabling testing to be linked to the system's requirements, and being the basis for evidence on whether the requirements have been fulfilled**

   As the new test framework is intended for use with large scale systems, the requirements for such a type of systems are likely to be complex and numerous, which makes traceability of a new system's design, implementation and testing a potentially complex task. Therefore, it is important that the new test framework should support traceability analysis to make it possible to verify that the system under test does accurately and completely fulfil the original requirements and that no divergence has crept in between the requirements capture phase and the final testing phase.

3. **Enabling early detection of faults in a new system**

   Enabling the early detection of faults is an indicator of "efficiency" of a test framework because the cost of faults lowers the earlier they are identified and fixed (Boehm, Basili, & Rombach, 2005, p. 426).

4. **Applicable to the full lifecycle of a new system**

   This is closely related to the previous criterion. A test framework needs to be applicable and relevant to all stages of development of a new system and not aimed only at detecting faults during the latter stages when the cost of rectification is higher and potentially more complex. In other word, the test framework needs to incorporate quality assurance activities that are not limited only to dynamic testing of functionality, e.g. design reviews, inspections or demonstrations of prototypes prior to the system being ready for the final stages of testing.

5. **Supporting close cooperation between the test team and the rest of an IT project team**

   Team cooperation is an important factor in the success and efficiency of a project (Yanga, Huang, & Wua, 2011, p. 265) and the test team is a significant part of an IT project, therefore it is a positive indicator of efficiency if the test framework promotes cooperation and interworking with the rest of an IT project's team.

6. **Useful as basis for defining a test strategy for a new system**

   This criterion is included because the test framework needs to provide comprehensive basis for the testing of a CCLSS, and the test strategy is at the core of any structured testing conducted within an organisation or on a project (ISTQB, 2012, p. 32).

7. **Provides a simplified conceptual view of a communications-critical large scale system's structure which can be used as an aid for the test analysis and design efforts**

   Conceptual modelling is an area of software engineering research where the use of modelling to achieve further advances in software engineering is investigated (Henderson-Sellers, 2011). The use of conceptual modelling has the potential of helping to simplify the complexity of CCLSS for testing purposes and to reduce its subjectivity by a providing a common basis for the test analysis and design work, i.e.

potentially lead to more precision in testing, which in turn can potentially lead to better efficiency.

8. **(A) Can be used as basis for review and verification activities of the system requirements, technical design and technical specification**
   (B) If used as basis for testing a communications-critical large scale system, allows the testers to start their verification and validation work from an early stage of the project

   Both these criteria are related to the earlier criteria 2, 3 and 4. For any new system, faults can start to be introduced from the earliest stages (Belgamo, Fabbri, & Maldonado, 2005), e.g. requirements capture, of the project then further variations and deviations can be introduced during the intermediate phases, e.g. high-level design, detailed design, implementation, integration, all through to the final testing stage and before operational deployment. Therefore, an effective test framework needs to be suitable for use as basis for review and verification activities that span the complete development lifecycle, including design and technical specification, and not be limited to detecting faults during the final phase of an IT project.

9. **Can facilitate test traceability and coverage analysis**

   Test traceability and coverage analysis are important aspects of structured formal testing (ISTQB, 2012, pp. 18, 40), therefore a comprehensive test framework needs to support such test management activities. For the purpose of this thesis, this criterion is intended to obtain the professional opinion of the external participants on whether the new test framework has the potential to support such test management activities.

10. **Have potential uses in an IT project outside purely testing, e.g. providing a shared view of a system for business analysts, developers, testers and suppliers/vendors**

Further to the earlier explanations for criteria no. 7 and 8, bringing closer the key activities of an IT project, e.g. requirements engineering and testing *"could benefit both disciplines"* and *"making a strong link between them will improve the outcome of the software development process"* according to (Barmi, Ebrahimi, & Feldt, 2011, p. 1). If a new test framework for CCLSS is to introduce a simplified conceptual view of the system for use by testers, then it would be more beneficial if such a view has the potential of becoming a shared view amongst other participants in an IT project, not just the testers, which in turn could contribute to reducing inconsistencies or gaps in the system build activities and to encourage collaborative working between the project's participants (criterion no.5).

Criteria table of Question 5 including responses of the participants

Criteria	Reply (tick one box)		Comments
Suitability for testing communications-critical large scale systems	Likely ***IA, IM, RJ, KK***	Not likely	RJ: *None exists (refer to literature)*
	Neutral	Don't know	
Enabling testing to be linked to the system's requirements, and being the basis for evidence on whether the requirements have been fulfilled	Likely ***IA, IM, KK***	Not likely	RJ: *From BA's (IAP) point of view (and existing methodologies) this could be achieved without additional framework modelling*
	Neutral ***RJ***	Don't know	
Enabling early detection of faults in a new system	Likely ***IA, IM, RJ, KK***	Not likely	RJ: *Reference to a recent review of importance of improving ICT testing via instruments to support cyber security – "Achievement and Next Steps: towards global cyber-security" 2010-2012*
	Neutral	Don't know	
Applicable to the full lifecycle of a new system	Likely ***IA, IM***	Not likely	RJ: *It seems we try to replicate the existing phases (by subject, e.g. Requirements Engineering and so on)* KK: *Service management and security requirements may not be fulfilled*
	Neutral ***RJ, KK***	Don't know	
Supporting close cooperation between the test team and the rest of an IT project team	Likely ***IM, RJ, KK***	Not likely	RJ: *Definitely!!! (Have a bad experience, very cost consuming)*
	Neutral ***IA***	Don't know	

Criteria	Reply (tick one box)		Comments
Useful as basis for defining a test strategy for a new system	Likely ***IM, KK***	Not likely	RJ: *Perhaps due to my background, not sufficiently qualified to confirm this*
	Neutral ***IA***	Don't know ***RJ***	
Provides a simplified conceptual view of a communications-critical large scale system's structure which can be used as an aid for the test analysis and design efforts	Likely ***IA, IM, KK***	Not likely	RJ: *As above*
	Neutral	Don't know ***RJ***	
Can be used as basis for review and verification activities of the system requirements, technical design and technical specification	Likely ***IA, IM, RJ, KK***	Not likely	RJ: *However, more evidence would be needed if some complex projects actually try it*
	Neutral	Don't know	
If used as basis for testing a communications-critical large scale system, allows the testers to start their verification and validation work from an early stage of the project	Likely ***IM, RJ, KK***	Not likely	RJ: *Even though the "V"-model implies the same*
	Neutral ***IA***	Don't know	
Can facilitate test traceability and coverage analysis	Likely ***IA, IM, RJ, KK***	Not likely	RJ: *As per assumptions in the paper*
	Neutral	Don't know	
Have potential uses in an IT project outside purely testing, e.g. providing a shared view of a system for business analysts, developers, testers and suppliers/vendors	Likely ***IA, RJ, KK***	Not likely	RJ: *With a lack of other unified framework perhaps agree*
	Neutral	Don't know ***IM***	

Question 6:

Please map the set of communications interface requirements provided in Appendix B to the nineteen communications subcategories in the list below.

The purpose of this question is to evaluate the viability and potential for generalisation of the 19 test subcategories by asking the participant to map to them an additional set of communications requirements from a real-life CCLSS project/system. A secondary objective is to also identify potentially ambiguous aspects of the subcategories by comparing the participant's mapping to the expectation of the author for how the mapping is intended to be done

This question and the related Question 7 were answered by one of the participants, IA. The set of requirements used for this mapping is included in Appendix 4.

Subcategory	CS2 Requirement IDs
Subcategory1: The structure/architecture of the network interface with CCLSS, its components and layout, hardware, and wiring	1.1.1.3, 1.1.1.4, 1.1.1.5, 1.1.1.6, 1.1.1.10, 1.1.1.11, 1.1.1.12, 1.1.1.14
Subcategory2: The communications protocols used	
Subcategory3: CCLSS user terminals	4.2.1.1, 4.2.1.2, 4.2.1.3, 4.2.1.4, 4.2.1.5, 4.2.1.6, 4.2.1.7, 4.2.1.8, 4.2.1.9, 4.2.1.10, 4.2.1.11, 4.2.1.12
Subcategory4: The data and messages that are transmitted by CCLSS over that network	
Subcategory5: All possible types of CCLSS senders and receivers	
Subcategory6: The different possible modes of transmission used	
Subcategory7: How the transmissions are acknowledged by the receivers	
Subcategory8: How the performance characteristics of the network can affect CCLSS subsystems and processes	
Subcategory9: The services provided by the network to CCLSS	
Subcategory10: Performance and volume limits of the services provided by the network	3.1.1.1, 3.1.1.2, 3.1.1.3, 3.2.1.1, 3.2.1.2, 3.2.1.3, 4.2.1.13, 5.2.1.1, 5.2.1.2, 5.2.1.3, 5.2.1.4, 5.2.1.5, 5.2.1.6, 5.2.1.7
Subcategory11: How CCLSS QoS requirements and SLAs are guaranteed, maintained and reported	
Subcategory12: The ongoing operation, maintenance and administration of the network/its CCLSS interface	3.5.1.3,
Subcategory13: Fault handling processes of the CCLSS interface, from detection to resolution	
Subcategory14: CCLSS certification/compliance requirements	3.5.1.1, 3.5.1.2, 3.5.1.4, 3.5.1.5, 3.6.1.2, 3.6.1.3, 3.6.1.4, 4.1.1.1, 5.1.1.1
Subcategory15: Resilience features (of the interface)	3.3.1.1, 3.3.1.2, 1.1.1.9, 1.1.1.13, 5.3.1.1
Subcategory16: Business continuity features (of the interface)	1.1.1.8,

Subcategory17: Documentation provided for CCLSS, user and technical documentation	
Subcategory18: Risks	
Subcategory19: Operational readiness of the interface for CCLSS go-live	
No suitable subcategory	3.5.1.6, 3.6.1.1
Not a technical communications requirement – not relevant	

Question 7:

Please comment on how the categorisation of the list of communications interface requirements sample according to the nineteen test subcategories (from question 5) might affect the test analysis and test design for these requirements relative to other generic test methodologies you are familiar with, e.g. the v-model?

The purpose of this question is to try obtain an assessment from the evaluator on the potential benefits he/she thinks can be gained from applying the 19 test subcategories (of the communications layer) when compared to a generic test methodology, i.e. one that does not specify the basis of the test analysis and design for communications interfaces

This question was answered by one of the participants, "IA".

IA: *"Provides clarity of key areas for testing, improved requirements verification traceability, high risk areas are more likely to be highlighted for early testing. Could save on test tool costs / hire by grouping and phasing the tests in a more efficient manner"*

7.5 Mapping requirements from a second CCLSS to the 19 subcategories by an external participant

The sample of communications requirements sample from a real-life CCLSS (Appendix 4) was selected because it was deemed appropriate as second case study. It consisted of three fragments from different subsections of a much larger document. The size of the sample of requirements of 50+ was selected to be proportionate but larger than the subsets of requirements used in the FireControl case study which ranged from 11-29 requirements per subset.

The mapping (presented in the previous section) was reviewed by the author of this thesis. Below are the resulting observations:

Of the fifty one CS2 requirements mapped by IA, there were two which could be described as being mapped incorrectly, namely 1.1.1.4 and 1.1.1.5, because they describe services and features offered by the ICCS to the overall CCLSS. Therefore, the appropriate subcategory for these two requirements is Subcategory 9. IA seems to have mapped them to Subcategory 1 presumably because both requirements described features and services that will influence the type of components, hardware and wiring required for the interface. However, Subcategory 9 is the more appropriate subcategory. In this instance, IA seemed to have carried out the mapping based on more in-depth understanding and analysis of the implications of the requirement. A similar observation can also be made regarding the mapping of 1.1.1.14, which is related to the interfaces for the ICCS. IA correctly mapped it to Subcategory 1; however, it could also be mapped to Subcategory 8 because it can be used as the basis for testing the impact of the ICCS performance on other CCLSS subsystems. There are two more requirements where IA's mapping might vary from the mapping of the author due to valid variations in interpretation, these are 3.5.1.3 and 3.6.1.1. IA mapped 3.5.1.3, which is related to the functionality for date manipulation and settings, to subategory12. This requirements could also be appropriate for Subcategory 11 regarding QoS/SLA functionality, although IA's mapping seems more appropriate. Requirement 3.5.1.3 is an example of where mappings varied because the wording of the requirement has some ambiguity. IA mapped 3.6.1.1 as "No suitable subcategory", although it could be mapped to Subcategory 14 related to standards compliance. However, the requirement is not an IT related requirement; therefore IA's mapping is also valid.

Overall, there seems to be two main differences between IA's mapping and the opinion of the author, and three more possible variations in interpretations. This ratio of 5:51 is low when compared to the variation between IA and the author in the mapping of Firelink requirements presented in Chapter 8 Section 8.9. There is also correlation on most "empty" subcategories for CS2, nine agreements even if the three potential requirement differences are included.

The variation between the outcomes of IA's mapping to Firelink vs. CS2 can be explained by noticeable differences between the two sets of requirements.

The CS2 set of requirements is noticeably more literal and more clearly classified and contiguous than the Firelink set which was identified from a large set of requirements and from numerous sections within a larger SoR. The CS2 continuity is apparent even from the requirement ID/numbering ranges[60].

The less ambiguous wording of the CS2 requirements also meant that there is less room for variations in the interpretation and analysis with the CS2 set. In fact, they could demonstrate how the mapping can be simpler and more precise when the requirements are originally written in a way that correlates with the subcategories. Where there was ambiguity, e.g. 3.5.1.3 as mentioned earlier, the likelihood of differences increased accordingly due to the possibility of variations in interpretations being increased.

The original purpose of the exercise by IA to carry out the mapping of communications requirements using both FireControl and CS2 requirements was to evaluate whether an external participant was able to apply the ideas of the new test framework's communications layer test subcategories. The outcomes of both exercises presented in this chapter and in the overall evaluation chapter (Chapter 8 section 8.9.3) provide positive indications that this was possible not only with FireControl requirements but also with the requirements of a second case study.

7.6 Conclusions

The feedback from the participants regarding the new test framework was generally positive and their comments represented interesting variations of relating the new test framework depending on each participant's background. The comments provided by RJ in particular

[60] The CS2 sample is made up of three logical fragments, each with its own section number and heading, which meant there is more logical continuity in the numbering of the CS2 requirements than is the case of the Firelink requirements.

regarding the criteria in Question 5 reflected the viewpoint of an experienced BA who is aware of requirements engineering recommended practices, e.g. through professional practice and membership of the Institution of Analysts and Programmers (IAP, 2013). The author concurs that, should there be a consistent universal requirements framework that is adopted in real-life for CCLSS projects, then that would help test design and analysis work to become more precise and less subjective. KK's comments regarding service management also reflected the participant's background of an experienced project manager who is interested in methods and processes that are fully matured and trialled and are proven to cover all aspects of testing that a project manager would be concerned with. The new test framework, although has detailed and applied to a real-life CCLSS project, requires further work beyond the scope of a thesis before it becomes a fully evolved methodology or an industry standard. IM's feedback indirectly pointed out how, even in the 1990s, there was a type of "layering" being adopted to organise the testing for a large scale banking IT programme which is comparable to the new test framework presented in this thesis. Where the new test framework attempts to simplify CCLSS test analysis and design effort and reduce the reliance on the experience and knowledge of the individual tester, IM's answers implied the banking programme did the opposite and used domain experts to carry out the testing within their areas of expertise. To an extent, the new test framework is intended to, and promotes, the capturing of such domain experts' knowledge for each of its six layers and including it in a unified re-usable test framework. IA's comments reflected the ease with which a communications expert as well as an experienced test manager was able to relate to the ideas of the framework. IA's comments mentioned the need for tangible benefits in terms of cost, time and quality as well as the need for familiarity with the test subcategories. The author concurs with such points and agrees that for such a new test framework to gain adoption in industry it needs to be developed further beyond the scope of an academic thesis and be supported by appropriate training, processes and test tools.

To conclude, the user-based evaluation presented in this chapter provides the following evidence regarding the viability and benefits of the new test framework[61]:

[61] The evidence material produced via the user-based evaluation is relevant to evidence ID's EV.1, EV.2, EV.3, EV.6 and EV.7 as will be listed in the evidence table in Section 8.2

1- Four external expert participants indicated that they believe the new test framework is feasible for adoption outside the confines of the specific case study included in this thesis and that it is likely to be usable for other comparable CCLSS that they were familiar with and have personally worked on.

2- They also broadly agreed that the new framework is likely to offer benefits and advantages to the test effort of such CCLSS projects.

3- The responses to Question 5 provided confirmation from the four external participants that the criteria[62] defined in section 4.2 and 4.3 were to a large extent (with a few "neutral" and "don't know" responses) achieved, or at least achievable, by the new test framework.

4- A sample list of communications requirements from a second CCLSS project comparable to FireControl was possible to map to the nineteen test subcategories by one of the four participants. This represents evidence that the nineteen test subcategories are usable for a CCLSS other than FireControl, and by a potential user who had no prior involvement in the FireControl case study.

Further comments on this part of the evaluation are included in the final chapter of the thesis (Chapter 9), in section 9.3.

[62] The term "criteria" is used to also encompass "attributes" and "features".

8. Chapter 8: Evaluation of the framework

8.1 Evaluation considerations

This section discusses some ideas that influenced the way the evaluation was conducted. Further discussions and explanations regarding the research methodology are presented in Chapter 3.

8.1.1 The case study process as evidence

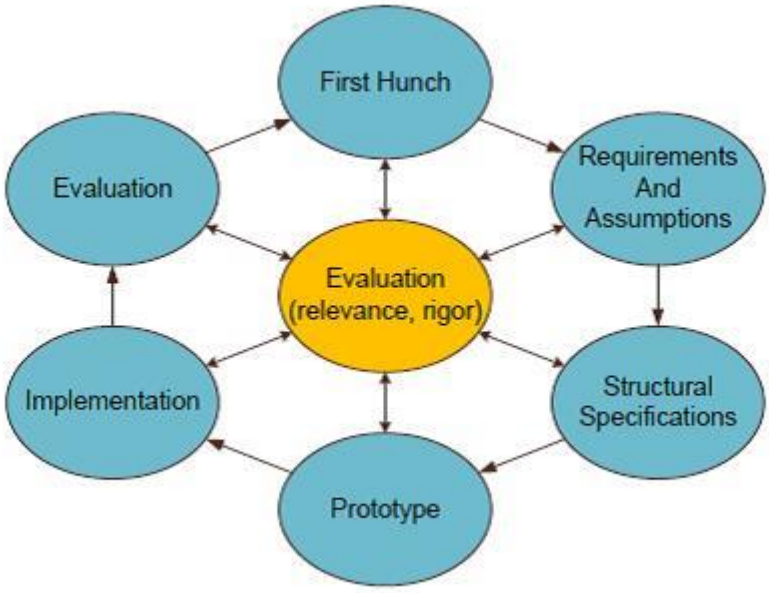
In a working paper on single case study research (Easton, 2010); Professor Easton argues that the process of conducting the case study could be used as evidence, and not just the "output" of the case study. In the context of this thesis, the "process" of conducting the case study work itself can therefore be treated as evidence, or at least part of the evidence. Accordingly, the fact that it was feasible to conduct the FireControl case study (Chapter 6) and generate the five sets of outline test cases (Appendix 2) was evidence that the new framework is a usable framework.

Furthermore, the fact that it was possible to apply the communications test approach to the test analysis and design of five different communications interfaces of FireControl (presented in the five test tables in Appendix 2) is further evidence that the nineteen subcategories are applicable to more than just one specific instance or set of requirements.

This leaves this chapter to provide further[63] evidence relating more specifically to the benefits of the framework, having already demonstrated that the framework's ideas provide a feasible basis for test design and analysis as shown with the FireControl case study.

[63] The primary evidence is provided in the case study chapter and in appendix 2

Rather than follow a theory-based research cycle such as: initial theory, literature review, research question, which in turn is followed by a case study design, case study data collection, case study data analysis and conclusions, the work included in this thesis has followed a design-oriented case study research approach where the case study in itself is used to apply and test the feasibility of a design idea. The process followed is more akin to the following diagram provided in (Boucharas, van Steenbergen, Jansen, & Brinkkemper, 2010, p. 20)[64]/[65] depicting the design cycle according to (Verschuren & Hartog, 2005).



According to such a research cycle, the case study part of this thesis is equivalent to the implementation of the initial design ideas and is a key part of the evaluation cycle.

The rest of this chapter will consider further evidence about the benefits of the new test framework and whether the intended "requirements"[66], "assumptions" and "structural

[64] Adopted from (v Steenbergen, Bos, & Brinkkemper, 2008)

[65] Worth noting that this same technical report demonstrates the scale and complexity of the research effort needed to evaluate benefits of Enterprise Architecture (EA) frameworks, which could be worth taking into consideration (as potentially containing useful ideas) for evaluating test frameworks

[66] In this design cycle context, the primary requirement is to design a new test framework.

specifications"[67] have been adhered to. To continue the analogy with the design cycle diagram above, the "prototype" is represented by the nineteen subcategories described in the communications layer chapter (Chapter 5). The "first hunch" is therefore the initial idea that a new test framework is needed to test communications-critical large scale systems which is presented primarily in Chapter 1.

8.1.2 Deciding on a suitable evaluation approach

In a keynote presentation titled "Useful Software Engineering Research: Leading a Double-Agent Life" to the IEEE International Conference on Software Maintenance in 2011 (Briand, 2011, p. 28), the "complexity of industrial systems and technology", the "Need to be studied in real settings" and "substantial domain knowledge is required" are listed as aspects of industrial challenges that need to be addressed by Software Engineering researchers.

Case study research methodology is clearly appropriate for software engineering research that seeks to link the academic with the industrial and to address the issues outlined in (Briand, 2011). On the other hand, this presents a challenge to researchers regarding how to balance academic rigor with industrial relevance. How to develop new and useful ideas for application to real-life practice, yet at the same time be supported with sound academic basis and evidence?

The type of evidence that is most likely to emerge from the case study is qualitative. Because this thesis uses one industrial IT project, the evaluation should preferably look for multiple sources of evidence[68] to strengthen the reliability of the findings and to offset the risk of researcher bias. However, quantitative evidence should also be sought where possible to strengthen the evaluation and to support the qualitative evaluation.

[67] These are equivalent to the attributes and features of the test framework as outlined in Chapter 4 which outlines the form and potential uses of the new test framework.

[68] A concept in case study research called "triangulation" (Yin, 2009)

Ultimately, conclusive evidence of the suitability and benefits of a new test framework can only be achieved through industrial application and adoption as will be discussed in the next subsection.

8.1.3 Examples from other software testing research

The nature of this work, being case study research regarding a new conceptual framework, means conclusive evidence is not likely to be achievable without further industrial trialling. Examples can be found of comparable research work in software testing that cannot have definitive generalised conclusions without trialling within an industrial context over a number of years or a number of projects, e.g. (Briand & Labiche, 2002, p. 27) where it states "*our methodology needs to be carefully experimented with, within control settings and through industrial case studies"*. Also in (Hemmati, Arcuri, & Briand, 2011, pp. 327, 335) where it states: *"as for all empirical studies, our results might not generalize to other case studies and only replications will help build confidence*". Another example from (Arcuri, Iqbal, & Briand, 2012, p. 275) where it states: "*overall the reported results have not been consistent. As a result, the jury is still out on whether and when random testing is a good option and more, better designed and reported studies are needed"*. This pattern can be detected even with research examples in the inherently more precise code testing such as (Yoo & Harman, 2007, p. 148) where it states: *"The primary concern for this paper is the representativeness of the subjects that were studied. This threat can be addressed only by additional research using a wider range of software artifacts and optimisation techniques"*.

Other than software testing research, examples of non-conclusive evidence of benefits can be found in other notable software engineering areas such as Agile (Dingsøyr, Nerurc, Balijepally, & Moe, 2012) and Enterprise Architecture (Boucharas, van Steenbergen, Jansen, & Brinkkemper, 2010, p. 10).

Therefore, the evaluation approach for this thesis needs to provide an acceptable trade-off between the real-life practicality of case study research against the need for academic rigour

and for reliable evidence, while at the same time not aim for a conclusive level of certainty that may be unachievable for with this type of research.

8.2 List of evidence sought EV.1 – EV.7

Below is a list of the evidence items needed to evaluate the ideas of this thesis. The list reflects the case study questions as outlined in section 3.4.

Evidence ID	The case study question
EV.1	Can the new test framework be applied (and if so how)?
EV.2	Are there benefits from the framework?
EV.3	Can the benefits be generalised beyond the case study?
EV.4	Can the benefits be estimated numerically?
EV.5	Can the framework be compared to a rival?
EV.6	Can the framework fulfil the initial intended criteria?
EV.7	Can the framework be applied by other potential users/participants?

The following seven subsections will present the available evidence material corresponding to each of the above evidence items EV.1 through to EV.7.

8.3 EV.1: Can the new test framework be applied?

As discussed in Chapter 3, the primary evidence that is sought to support the ideas in this thesis is whether it was feasible and useful to apply the new test framework to a real-life case study. This evidence has already been established in the form of the main case study (Chapter 6). The FireControl case study demonstrated that the new test framework can be applied to a real-life project. It demonstrated how one of the "layers" of the framework can be instantiated further and in more detail and then applied to the communications

requirements of five different communications interfaces[69] of FireControl to generate a meaningful and coherent set of outline test cases as provided in Appendix 2.

Furthermore, the feasibility of applying the new framework's ideas, specifically for the communications layer, to five different communications interfaces consistently as shown by the five diagrams B.1 – B.5 in (section 6.4 to 6.8) is further support to the possibility that the nineteen subcategories are capable of being applied to more than just one set of requirements for more than just one particular communications interface.

Additionally, Chapter 7 already presented evaluations by expert external participants, as potential users of the framework, regarding the applicability of the framework to the testing of other real-life CCLSS projects they previously worked on. This evaluation material is already presented in Chapter 7 therefore is not repeated in this section.

8.4 EV.2: Are there benefits from the framework?

This section includes qualitative discussions and presents qualitative evidence regarding the benefits of the new test framework as observed through the FireControl case study.

8.4.1 Overarching benefits of the framework

There are three overarching points that need to be highlighted when evaluating the feasibility and benefits of the new test framework as observed through the FireControl case study.

Firstly, it was possible to apply the framework's layered test model to a large set of requirements (2000+) that were not originally intended to map to the framework. A distinct "communications layer" was possible to identify and to use as an example of how the new test framework can be used to carry out test analysis and design. The new test framework's ideas did not emerge from the case study, but were applied to it. This constitutes evidence,

[69] i.e. five different sources of data

or at least provides an indication, that the framework is capable of being applied to other comparable systems[70].

Secondly, it was possible to apply the more detailed communications test approach to five different communications interfaces, each with its own requirements that were not originally intended to map to the nineteen test subcategories. Furthermore, the dependency and priority relationships between the nineteen subcategories remained valid for the five different interfaces (as shown in diagrams B.1 to B.5 in sections 6.4 to 6.8). It was then possible to define a set of outline test cases for the five interfaces that were independent from the technical design, i.e. independent from the developer's interpretation of the requirements. Without the use of the new test framework, generating a similar outcome would have been more complex due to the fragmentation of the communications related requirements in the original Statement of Requirements (SoR) as shown in Appendix 1 Table A. The viability of applying the nineteen communications test subcategories to five different interfaces represents a form of "triangulation" (Runeson, Höst, Rainer, & Regnell, 2012, p. 72) which was then confirmed by the consistency of interdependencies between the subcategories[71] of diagrams B.1 to B.5. This also represents further evidence that the communications test approach is capable of being usefully adopted for other communications interfaces in other comparable systems[72].

Thirdly, the use of the framework allowed for a systematic transition of a large list of communications requirements, collected from all sections of the complete Statement of Requirements for the project and listed in Appendix 1[73], to be identified as communications related then transformed into a simpler and more manageable structure for testing purposes as shown in Appendix 2. The list of requirements provided in Appendix 1, especially before

[70] Corresponds to evidence reference EV.1: "Can the new test framework be applied?" and to a lesser extent also to EV.3: "Can the benefits be generalised beyond the case study?"

[71] The consistency between diagrams B.1 – B.5 can be described as "pattern matching" according to case study terminology (Runeson, Höst, Rainer, & Regnell, 2012, p. 68).

[72] See also contents of Chapter 7 which present further evaluation of the likely applicability of the framework by external expert participants

[73] Appendix 1 is provided to demonstrate how the communications related requirements featured in the original Statement of Requirements (SoR), to save the reader the need to look through the whole of the SoR.

separating from the total set of 2000+ requirements, is clearly more difficult to use as the basis for meaningful test design without subject matter expertise and without the availability of the detailed design. On the other hand, the tables in Appendix 2 show simplified structure and provide more objective basis for generating test design and for prioritising the tests.

8.4.2 Examples of the benefits of the framework

This section is intended to provide specific examples of the benefits observed during the FireControl case study (Chapter 5).

8.4.2.1 Examples derived from the categorisation of FireControl communications requirements

This subsection is structured according to the expected benefits discussed in Section 5.3. It cites specific examples from Appendix 2 and provides discussions on how the use of the nineteen test subcategories helped the test analysis and design for the cited examples.

Prioritisation of the testing

Examples from the communications test tables in Appendix 2:

Firelink table examples

Testing for requirement 5.5.7.13.3.B needs to be conducted ahead of any other testing because the configuration of the CCI ports is fundamental to the overall operation of the interface and can have significant impact on the running costs of the interface. Checking that the utilisation of the CCI ports is acceptable needs to be done ahead of any other testing. Otherwise it could mean that all other testing of the functionality of the interface can potentially require repeating if at a later stage the configuration of the CCI ports is deemed to be unacceptable.

Furthermore, requirement 5.5.5.1.2 classed as Subcategory 12 needs to be tested before the requirements classed as resilience requirements under Subcategory 15. This is because the configuration functionality that are described in 5.5.5.1.2 can influence the resilience feature described in 5.5.5.1.4.A, i.e. the ability of the Communications Gateway to switch between bearers could be influenced by how the Communications Gateway has been configured.

Telephony table examples

Similar to the earlier Firelink example, checking the fulfilment of requirements 5.5.2.3.3 and 5.5.2.2.13 needs to be done ahead of any other testing for the telephony interface. The realism of all other testing for the interface will be uncertain until these two requirements have been accepted.

WAN table example

Requirement 5.5.2.7.1 needs to be fulfilled before any meaningful testing of fault handling (Subcategory 13) or business continuity (Subcategory 16) can be done.

Resolution of gaps and inconsistencies

The use of the nineteen subcategories highlighted the absence in the requirements of a description for Subcategory 13 - Fault handling processes of the FireControl interface, from detection to resolution, also that no performance thresholds are specified in Schedule 11 for WAN and RCC LAN. This highlighted a potential need to raise test risks or define additional test cases based on the technical specifications.

Improved synergy with the overall project plans

The outcome of using the nineteen test subcategories was a structured list of requirements-based test cases and related assurance activities that were usable for early project assurance

and test effectiveness monitoring activities that are not limited only to testing: e.g. review, demonstration, and inspection actions. See the test tables in Appendix 2.

Improved confidence in the results of other tests

By producing the five communications test tables, the intention was to list all communications requirements that are purely technical and have no direct visibility at business process or user level. Where a requirement is visible and testable at a user level (e.g. telephony features, ICCS functions), it was classified as a functional requirement rather than communications requirement. Most of the assurance activities and the outline test cases defined in the test tables were identified as pre-final acceptance activities. If the test tables were to be developed further into action lists and detailed test cases, then implemented as defined, a significant and complex critical part of FireControl would have been verified during the earlier stages of the project before reaching the final acceptance stage. This approach could have significantly reduced the risk to the project of fundamental communications related problems being uncovered too late.

In contrast to the benefits discussed in this subsection, see the comments[74] in Table A (Appendix 1) discussing how the test analysis and design for the communications requirements could have been conducted in the absence of a domain-specific test framework. The test analysis and design, and subsequently the resulting test cases, would have relied heavily on the judgement of the test analyst and the effectiveness of the testing was likely to vary significantly depending on the individual tester's experience and skills.

8.4.2.2 Examples derived from the outline test cases

The material presented so far in this section (Section 8.4) centred on the benefits related to analysing, understanding, structuring and identifying gaps in the communications requirements of FireControl.

[74] inserted by the author in the rightmost column

The following subsections five present examples of benefits resulting from the outline test cases that were generated as an output from the FireControl case study. For the full list of outline test cases, see column titled "Test Case Descriptions" in the tables in Appendix 2.

Firelink

The outline test cases for 12 of the requirements may not have been correctly prioritised if the original ordering in the statement of requirements (see Appendix 1) was adhered to: these are 5.5.7.13.3.B, 5.5.14.4.3/4, 5.5.5.1.1/2/2.A/3/5/4.A/6/7. Additionally, a further 6 of the requirements that were not directly associated with Firelink have been identified and incorporated, these were mostly MDT, MRMS or ICCS requirements, these are 5.5.14.3.23.B, 5.5.14.4.4, 5.5.6.57.G, 5.5.7.30.1, 5.3.4.3.D, 5.5.7.36.1. 5 potential gaps or test risks, for subcategories 3/5/8/13/16, were also highlighted. Lastly, 8 requirements relating to documentation and certification and standards compliance were appropriately grouped together from three different subsections not necessarily related directly to the Firelink interface.

Other specific examples from the Firelink test table includes, under Subcategory 1, "CClimit1" outline test case which relates to requirement 5.5.7.13.3.B whose fulfilment should be checked during the interface design stage. If this requirement is not fulfilled correctly, it could invalidate much of the testing carried out subsequently, especially for performance and resilience related tests. The allocation of its testing to Subcategory 1 according to the new framework is therefore appropriate. The requirement relates to the efficient allocation of costly network interface resources. Subsequent testing relating to performance or resilience of the network interface could be invalidated if carried out with too many or too few CCIs. Additionally, under Subcategory 12 (OA&M), the test cases outlined in the table were correctly classified to be run before meaningful testing for Subcategory 15 (Resilience) requirements commences because a system's resilience is partly dependant on its OA&M functionality.

Secondary Radio Bearer

Subcategories 4 and 6 contain the more complex test cases for this interface, relating to what data types will be carried and the modes of transmissions to be used. Such functionality needs to be validated before performance or resilience (Subcategories 10 and 15) testing can meaningfully commence, therefore the new framework's prioritisation is appropriate. Subcategories 5 and 13 contain no requirements, and are highlighted potential risks subject to review of the technical design.

Telephony

The fulfilment of the two requirements under Subcategory 1, 5.5.2.3.3 and 5.5.2.2.13, needs to be confirmed ahead of any other testing for the telephony interface. The value of all other testing for the interface will be doubtful until the above two requirements have been validated. Although no specific testing is required for the two requirements, early inspection is needed before further testing should commence. Without the use of the framework, such requirements may not be checked until the start of testing or even just before live operational use of a new system, when it could be relatively late and expensive to make changes to the structure of the telephony interface. Subcategory 13 (fault handling) for this network interface also does not contain any requirements, which is highlighted as a potential gap and a test risk.

WAN

Most requirements for the WAN interface are concentrated in Subcategory 1, which is appropriate because what is required is a WAN managed service, i.e. the WAN is not setup specifically for the project but is a commercially leased service. The new test framework is therefore correctly organising the requirements in a way that reflects the nature of the network interface under test.

Another example of the prioritisation benefit is Subcategory 12's requirement 5.5.2.7.1. This requirement needs to be checked before any meaningful testing of fault handling (Subcategory 13) or business continuity (Subcategory 16) can be carried out without the possibility of costly rework relating to 5.5.2.7.1.

LAN

Subcategory 12 includes requirements 5.5.4.2.3, 5.6.15.1 and 5.6.15.2, which are not LAN specific and were not mentioned in the LAN subsection in the SoR. The three requirements related to the OA&M (Operations, Administration and Management) of the interface, which is an aspect of the interface which can easily be overlooked if test cases were devised according to a generic functional and non-functional classification.

Another example of the framework resulting in appropriate prioritisation of the testing is that it leaves the most complex testing for the LAN interface, relating to Subcategory 15 (Resilience) to be carried out after other more fundamental review or inspection work relating to earlier subcategories has been completed.

8.5 EV.3: Can the benefits be generalised?

A common characteristic and a potential weakness of case study research is that it is often difficult to generalise the findings based on one case study. Whereas case study research can be a helpful research tool for researching real-life phenomena, deriving generalised conclusions from it can be constrained by the boundaries of the specific case study that was used to conduct the research. This does not mean that deriving any generalised conclusions or new understanding is impossible, but is not likely to be conclusive or definite (as discussed earlier in the introduction of this chapter). The remaining material in this subsection is presented within the context of this awareness about case study research.

For the purpose of countering the potential weakness of case study research referred to above, two measures were incorporated within the FireControl case study:

1- The new test framework's approach to testing the communications layer was applied to the five communications interfaces of FireControl. These are technically different interfaces, expressed differently in the FireControl Statement of Requirements (SoR)

(FiReControl project, 29 March 2007) and appeared in different sections of it. The first interface, the Firelink interface, was treated as the pilot interface before the framework's nineteen communications subcategories were applied to the remaining four interfaces. Having this variety of "data" was intended to improve the reliability of the outcome of the case study. As the case study chapter shows, the nineteen subcategories remained applicable for the five different interfaces despite their variety.

2- Diagrams B.1 – B.5 in sections 6.4 to 6.8 were used to check the similarity of interdependencies between the subcategories when applied to the five different communications interfaces. They showed that the interdependencies remained consistently applicable to the five interfaces.

3- An additional evaluation by external expert participants, presented in Chapter 7, resulted in positive feedback from the participants regarding the applicability of the new test framework generally and specifically the nineteen test subcategories to the testing of other CCLSS projects they (the external participants) had direct involvement in. See Chapter 7 for further details of this evaluation.

4- As part of the Chapter 7 evaluation by external participants, one of the participants (IA) mapped a sample set of communications requirements, from a CCLSS project comparable to FireControl, to the nineteen test subcategories and to comment on the effort and clarity of carrying out the exercise. This exercise used a sample of requirements from what is essentially a second case study and demonstrated that these requirements can be mapped to the nineteen test subcategories by an external participant who is also a potential user of the new test framework. This additional evaluation provided further assurance that the new test framework (specifically the communications test approach): (a) can be applied to systems/projects other than FireControl, and (b) it can be used by external participants who had no involvement in designing the new test framework.

5- The same external participant (IA) also mapped the Firelink interface requirements to the 19 test subcategories as is shown in section 8.9.1. This was done to provide an

example for comparison between the mapping carried of an external participant and how it might vary from the mapping presented as part of the FireControl case study. See section 8.9.1 for further details.

In summary, when the list of evaluation measures presented above are taken into consideration together, they represent evidence that the ideas of the framework, specifically the nineteen test subcategories of the communications layer, can be applied beyond the confines of the FireControl case to at least one further set of communications requirements from another CCLSS.

8.6 EV.4: Can the benefits be estimated numerically?

Prioritisation of tests is an indicator of test effectiveness (Elbaum, Malishevsky, & Rothermel, 2002) (Svensson, et al., 2011) (Yoo & Harman, 2007), therefore an effective test framework should lead to optimally prioritised tests. A simulation-based method was devised to numerically compare the prioritisation efficiency of the new test framework with: (a) the original ordering of the communications requirements according to the FireControl Statement of Requirements (SoR), and (b) according to randomly ordered sets of requirements of the five FireControl communications interfaces presented in the case study chapter (Chapter 6). This effectiveness estimation method is presented in the next section (Section 8.6.1).

8.6.1 Effectiveness estimation simulation-based method

This method is based on the premise that well-prioritised requirements should lead to less interdependencies which in turn should lead to reduced testing and re-testing effort. It is used to derive an estimate of the "re-test overhead" total for each of the five FireControl interfaces as discussed in Chapter 6. The re-test overhead was then compared with the re-test overhead of the same set of requirements if they were ordered according to the sequence found in the original statement of requirements (see Appendix 1). Furthermore, a large set of random orders was generated using an Excel macro, then the re-test overhead was calculated for each

of these randomly generated sets. Following such calculations and random simulations of sets of requirements, re-test overhead figures for the following can then be compared: (a) the framework's ordering of the set of requirements for each of the interfaces, (b) the order originally provided by the statement of requirements, and (c) the randomly generated orders. The calculations were carried out using Microsoft Excel VBA Macro[75].

To conduct this simulation work, a "dependency set" for each of the five FireControl communications interfaces was created through analysis of the requirements' interdependencies, carried out by reviewing the requirements one-by-one and analysing their interdependencies with the rest of the requirements set for each of the five communications interfaces. See Appendix 3 for the list of dependency sets and explanations of how they were derived.

A dependency set for a requirement (X) is the set of other requirements within the interface that depend on that requirement. The rationale behind the idea of the dependency set is that, during the testing relating to requirement X, if a fault is found, then the testing related to the requirements within its dependency set will have to be repeated (once the fault is fixed) if these requirements were tested before X. If no other requirement is dependent on X and a fault was found during the testing specific to X, then only the testing relating to X needs to be repeated once that fault is fixed. For the purposes of deriving numeric estimates, this is then assigned a re-test overhead of 1. If on the other hand there were four requirements that depend on X, and all of the four were tested before X was tested, then a fault found with X would mean (for the purpose of the re-test overhead estimates) that all four need to be re-tested and X also needs to be re-tested. This would then lead to the re-test overhead (which can also be called the dependency count) for X being assigned a value of 5.

The re-test overhead (to be called dependency count for the rest of this section) for X can vary according to the way the set of requirements for an interface are ordered. If for example, X was the first requirement in a particular order, then its overhead for that particular order

[75] The coding of the VBA Macro was purchased as a service. The design of how the VBA code should work and how it should do the calculations as well as the "dependency sets" for each of the requirements in each of the five interfaces were specified by the author of this thesis.

will be just one. Similarly, if only two of the requirements within the dependency set of X appeared before it then its dependency count would be 3.

Based on the method described above, each order for each set of requirements for the five interfaces has its own total dependency count. The lower the total dependency count, the better the ordering is for test efficiency purposes.

How would the proposed framework compare with the order of the original statement of requirements and furthermore against 1000[76] randomly generated sets? If the expected benefits of better prioritisation and more effective testing are correct then the dependency count for the test framework would be lower than the other orders.

Before presenting the results of the simulations, it is important to explain key assumptions and adjustments that were necessary to make the simulations possible and meaningful.

The simulations did not include or take into account the following:

- Schedule11 items (overarching SLA and Performance requirements)
- Test risks
- Potential gaps in the requirements identified during the analysis effort
- Duplicates across the interfaces, i.e. where a requirement is not unique to one interface. Most such duplicates related to documentation or to standards compliance.
- There were two examples when the same requirement featured twice within the same interface, for the purpose of the calculations the second occurrence of the same requirement was deleted from the list.

Assumptions:

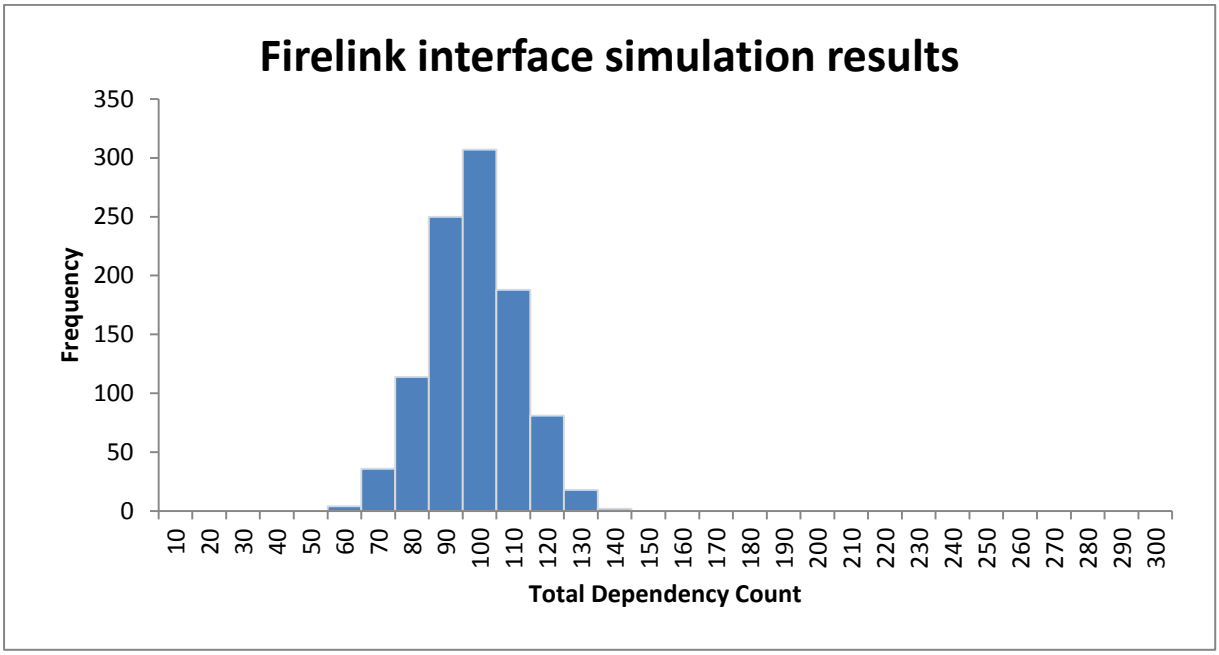
[76] The suitability of this number of sets was checked via a web statistics calculator (Australian Bureau of Statistics, 2013) using the following parameters: 95% confidence level, 0.03 and 0.035 confidence intervals, and a population size of 1,000,000. Sample Size results were 784 and 1,066 respectively.

- The requirements are well defined and complete
- When the testing relating to one requirement is completed, then this requirement is fulfilled and no faults are left undetected.
- Decisions about whether retesting is needed or not were derived for each individual requirement and how it related to other individual requirements within the interface.
- All requirements are of the same complexity and weighting
- The retesting effort required for all requirements is the same.
- Where duplicates occur within a list or between two different lists, they will be counted as other non-duplicates.
- Re-test effort is calculated according to the initial set of requirements that need to be re-tested, i.e. not including further requirements that need re-testing based indirectly on the initial set.

The following five subsections (Sections 8.6.2 to 8.6.6) contain the simulation results for the five interfaces: The distribution graphs[77] of simulations as well as the associated calculations, followed by discussions of the results. The complete log of the simulation results is available on CD supplement No.1 to this thesis which contains the five Excel spreadsheets used to carry out the simulations for the five interfaces.

[77] The graphs show the ranges of dependency counts (e.g. 51-60, 61-70, 71-80, etc.) achieved by the random sets vs. the number of random sets that had a dependency count falling within each range.

8.6.2 Firelink interface



Dependency count when the requirements are ordered according to the framework	47
Dependency count when the requirements are ordered according to the order of the Statement of Requirements	128
Random orders minimum dependency count	57
Random orders maximum dependency count	135
Random orders most common range of dependency count / percentage of occurrence	90-100 range, 307 random sets out of 1000
Statistics of dependencies between individual requirements for the interface (see Firelink interface dependency sets in Appendix 3):	

Number of dependencies that agree[78] with the framework from the viewpoint of the simulation; vs. Number of dependencies that are not enforced by the framework from the viewpoint of the simulation, either because they contradict[79] the framework or because they represent interdependencies between requirements within the same subcategory, i.e. a lower level of granularity than the subcategories; vs. Number of requirements with no dependencies, i.e. neutral[80] with no other requirements depending on them.	99 agree – or approximately 71% of the total 35 dependencies – or approximately 25% of the total. 4 out of the 35 variations are actual contradictions[81] between the framework's ordering and the dependency analysis carried out for the purposes of the simulation. 6 neutral

[78] "Agree" is used in this section to indicate that the dependency is forced to be correctly ordered by the framework.

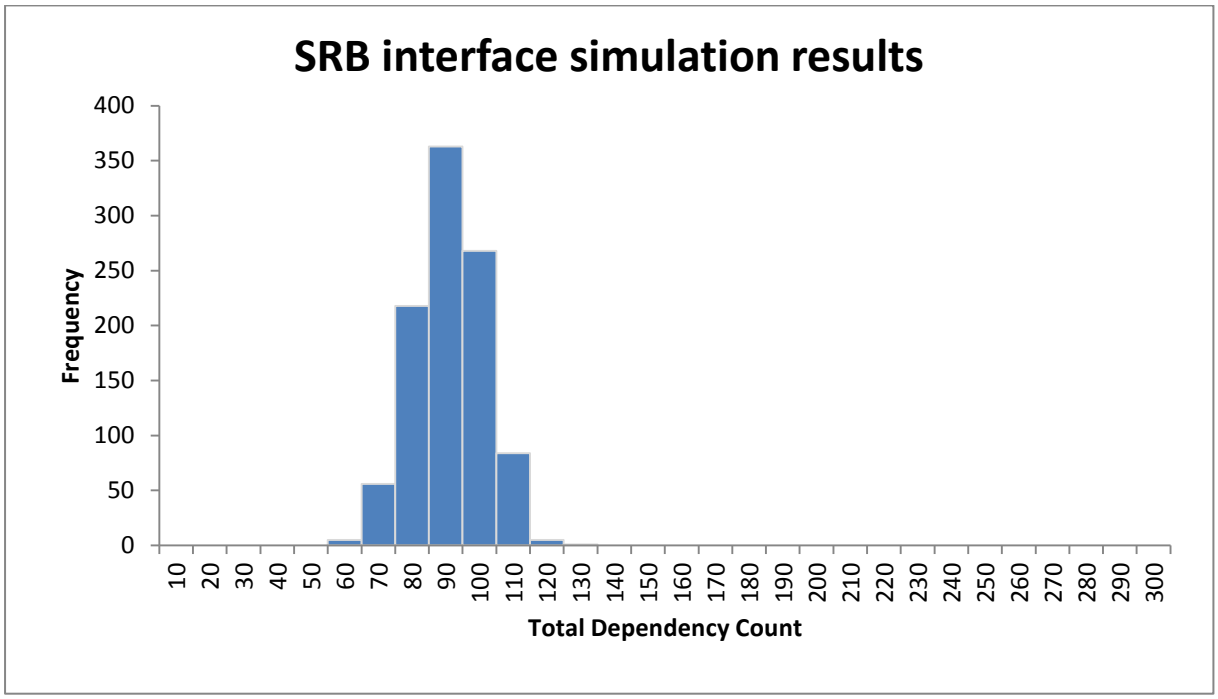
[79] "Contradict" is used in this section to indicate that the dependency contradicts the framework's organisation of the requirements because it contradicts the sequence of a pair of requirements according to the framework.

[80] "Neutral" is used in this section to indicate where no other requirements depend on an individual requirement, which makes it neutral for the purposes of the simulated analysis.

[81] The dependency sets derived via an analysis of the interdependencies between individual requirements had the following contradictions from the framework for the Firelink interface: 5.5.5.1.1 in the dependency set of 5.5.5.1.2, 5.5.5.1.4 in the dependency sets of both 5.5.5.1.3 and 5.5.5.1.5, lastly 5.5.7.30.1 in the dependency set of 5.5.7.13.3. See the table in section 12.2 for the Firelink interface dependency sets for an explanation of the variations.

The relatively wide difference (compared to other interfaces) between the dependency count for the communications test framework and the order of the Statement of Requirements (SoR) reflects the fragmentation of the Firelink related requirements in the Statement of Requirements. The order in the Statement of Requirements had a higher dependency total than most of the randomly generated sets. Compared to the most common range for the randomly generated sets the framework's dependency count was approximately half at 47 compared to the most common range of 90-100 dependency count.

8.6.3 Secondary Radio Bearer interface



Dependency count when the requirements are ordered according to the framework	46
Dependency count when the requirements are ordered according to the order of the Statement of Requirements	110
Random orders minimum dependency count	56
Random orders maximum dependency count	123
Random orders most common range of	80-90 range, 363 random sets out of 1000

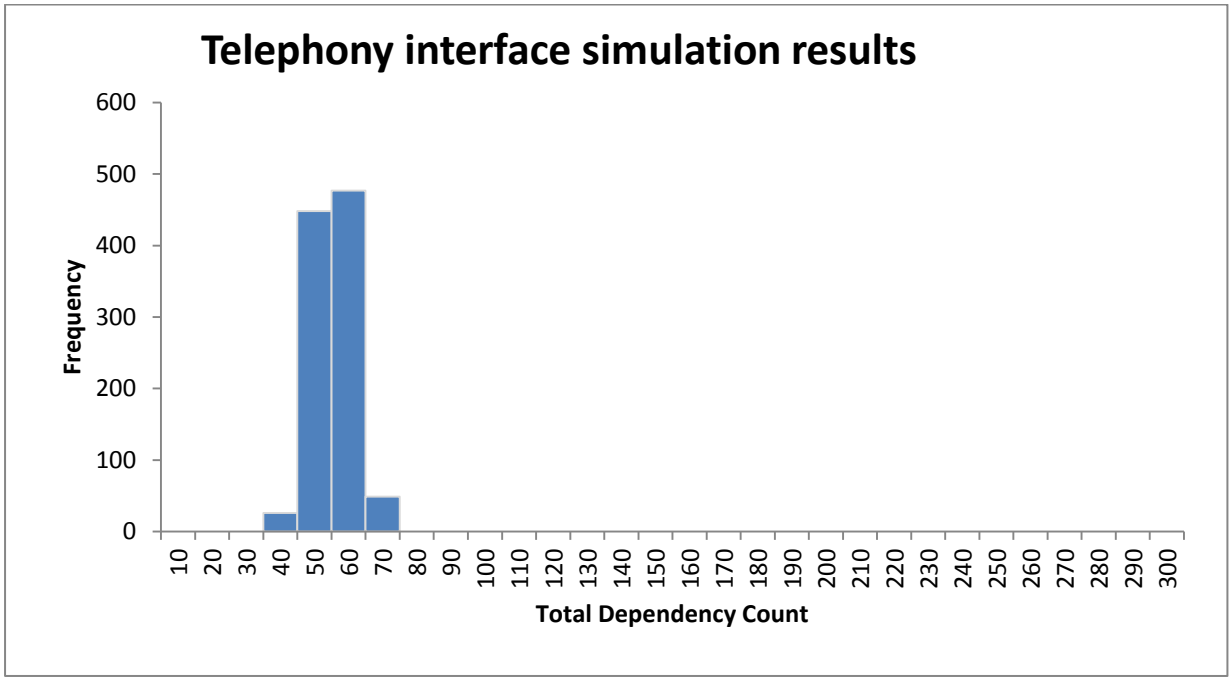
dependency count / percentage of occurrence	
Statistics of dependencies between individual requirements for the interface (see SRB interface dependency sets in Appendix 3): Number of dependencies that agree with the framework from the viewpoint of the simulation; vs. Number of dependencies that are not enforced by the framework from the viewpoint of the simulation, either because they contradict the framework or because they represent interdependencies between requirements within the same subcategory, i.e. a lower level of granularity than the subcategories; vs. Number of requirements with no dependencies, i.e. neutral with no other requirements depending on them.	89 agree – or approximately 68% of the total 35 dependencies – or approximately 27% of the total. 4 out of the 35 variations are actual contradictions[82] between the framework's ordering and the dependency analysis carried out for the purposes of the simulation. 7 neutral

[82] The dependency sets derived via an analysis of the interdependencies between individual requirements had the following contradictions from the framework for the SRB interface: 5.5.14.4.4, 5.5.5.6.57.G and 5.3.4.3.D in the dependency set of 5.5.5.1.2.A, 5.5.5.1.1 in the dependency sets of 5.5.5.1.2. See the table in section 12.4 for the SRB interface dependency sets for an explanation of the variations.

The difference between the dependency count for the communications test framework and the order of the Statement of Requirements is slightly smaller than the difference for the Firelink interface. Compared to the most common range for the randomly generated sets the framework's dependency count was approximately half at 46 compared to the most common range of 90-100 dependency count.

There were similarities in the Statement of Requirements between the structure of the SRB requirements and Firelink requirements, also some requirements related to both interfaces. On the other hand, the SRB requirements were less fragmeneted than those of the Firelink interface. Therefore, the similar but slightly better (when compared to the random sets) SRB dependency count figures of the framework's and the Statement of Requirements (SoR) orders are within what was expected.

8.6.4 Telephony interface



Dependency count when the requirements are ordered according to the framework	38
Dependency count when the requirements are ordered according to the order of the Statement of Requirements	45

Random orders minimum dependency count	36
Random orders maximum dependency count	65
Random orders most common range of dependency count / percentage of occurrence	50-60 range, 477 random sets out of 1000
Statistics of dependencies between individual requirements for the interface (see Telephony interface dependency sets in Appendix 3): Number of dependencies that agree with the framework from the viewpoint of the simulation; vs. Number of dependencies that are not enforced by the framework from the viewpoint of the simulation, either because they contradict the framework or because they represent interdependencies between requirements within the same subcategory, i.e. a lower level of granularity than the subcategories; vs.	28 agree – or approximately 46% of the total 17 dependencies – or approximately 28% of the total. 2 out of the 17 variations are actual contradictions[83] between the framework's ordering and the dependency analysis carried out for the purposes of the simulation.

[83] The dependency sets derived via an analysis of the interdependencies between individual requirements had the following contradictions from the framework for the Telephony interface: 5.5.2.8.2 and 5.5.2.8.6 in the dependency set of 5.3.19.5. See the table in section 12.6 for the Telephony interface dependency sets for an explanation of the variations.

Number of requirements with no dependencies, i.e. neutral with no other requirements depending on them.	16 neutral

The telephony interface figures are markedly different from the other interfaces, especially when compared with the Firelink and SRB interface figures. The Telephony interface requirements are more logically grouped and ordered in SoR than the Firelink and SRB requirements, also many of them were for specific technical features of the telephony interface rather than, say, about the technologies deployed in the telephony network. As well as being less fragmented, the Telephony requirements were also less interdependent.

This explains why the framework's organisation, which was based on whole groups of requirements (the nineteen subcategories) rather than individual requirements, was relatively less efficient than when it was applied to the Firelink and the SRB requirements. Two randomly generated sets out of a thousand had better dependency count (36 and 37) than the framework's dependency count, three random sets had the same dependency count, and eighteen random sets had very close count figures (39 and 40). The distribution of the count for the randomly generated sets also had less spread.

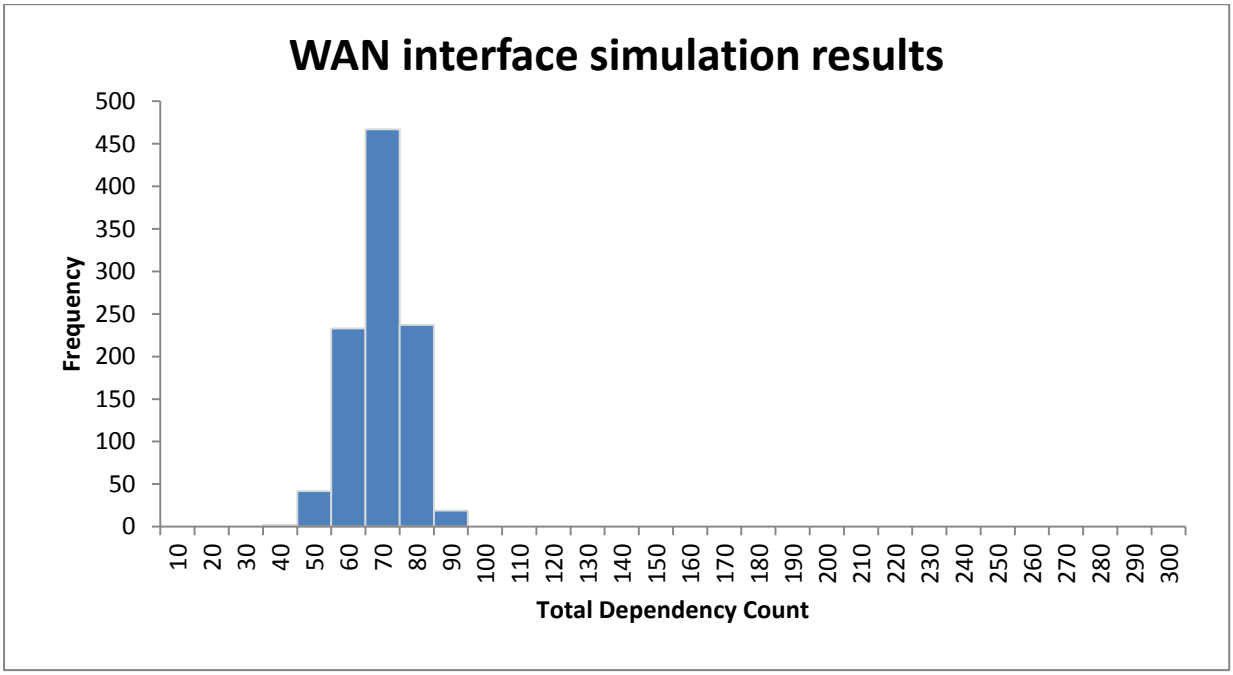
Therefore, although the use of the new test framework for the Telephony interface also did produce an optimal order from a test point of view, the benefit of the framework is less for the Telephony interface when compared to the two previous interfaces.

The difference in the simulated statistics between Telephony and the other interfaces can be understood further when the statistics of dependencies between individual requirements are compared (see summary table). Approximately 46% of the dependencies between individual requirements "agreed" with the framework's ordering of the requirements, and 28% were not enforced[84] by the framework. Also the number and ratio of the neutral requirements were

[84] Either because they contradict the framework's ordering or because they represent interdependencies between requirements within the same subcategory that the framework does not enforce, i.e. due to the lower level of granularity.

higher for the Telephony interface than for the other interfaces. These ratios further explain why the benefit of the framework was less evident through the simulated calculation when compared with the other interfaces. The simulated calculation also highlighted the presence in the Telephony interface requirements of relatively more inter-dependencies between requirements within the same subcategories, which is a level of granularity that is not covered by the framework's conceptual organisation. For further details on how these numbers were derived please refer to the Telephony interface dependency sets in Appendix 3.

8.6.5 WAN interface



Dependency count when the requirements are ordered according to the framework	40
Dependency count when the requirements are ordered according to the order of the Statement of Requirements	76
Random orders minimum dependency count	39
Random orders maximum dependency count	88
Random orders most common range of	60-70 range, 467 random sets out of 1000

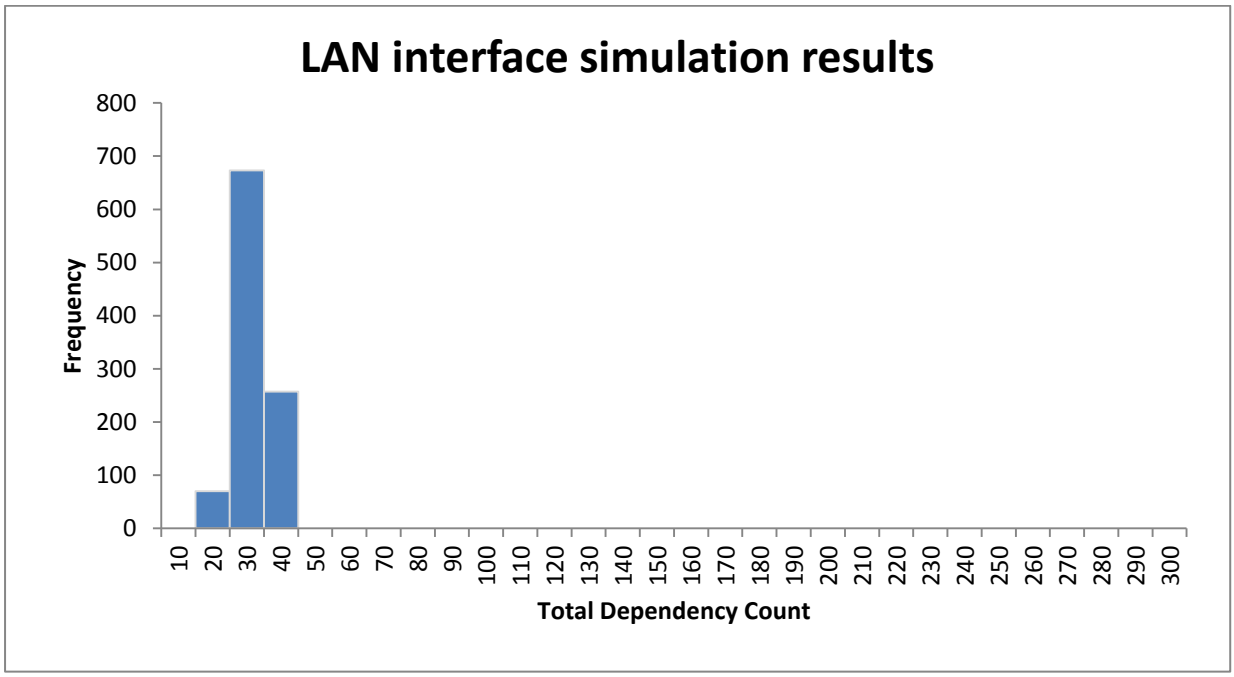
dependency count / percentage of occurrence	
Statistics of dependencies between individual requirements for the interface (see WAN interface dependency sets in Appendix 3): Number of dependencies that agree with the framework from the viewpoint of the simulation; vs. Number of dependencies that are not enforced by the framework from the viewpoint of the simulation, either because they contradict the framework or because they represent interdependencies between requirements within the same subcategory, i.e. a lower level of granularity than the subcategories; vs. Number of requirements with no dependencies, i.e. neutral with no other requirements depending on them.	54 agree – or approximately 61% of the total 27 dependencies – or approximately 30% of the total. None of the variations were actual contradictions between the framework's ordering and the dependency analysis carried out for the purposes of the simulation. 8 neutral

The dependency count results for WAN interface are comparable to the Telephony interface results and also markedly different from the Firelink interface and SRB results. This should be viewed in light of the fact that Firelink and SRB requirements are more complex and more fragmented than the Telephony and WAN requirements. Whereas many of the Telephony

requirements were for specific technical features of the Telephony service, many of the WAN requirements were for supplier contractual obligations rather than describing complex functionality.

However, two random sets out of a thousand had the same or better dependency count as the set ordered according to the communications framework. This is due to the relatively lower difference between the ratio of individual requirement's interdependencies that "agree" with the framework (61%) and the individual requirement's interdependencies that were not enforced by the framework (30%) of approximately 31%. This difference leads to a higher probability for the simulated sets to have a better dependency count than the framework's ordering. The equivalent ratio difference for the Firelink interface is 46%, for the SRB the difference is 41% and for the Telephony interfaces the difference of approximately 18% is lowest of all five interfaces. For further details on how these numbers were derived please refer to the dependency tables in Appendix 3.

8.6.6 LAN interface



Dependency count when the requirements are ordered according to the framework	13
Dependency count when the requirements	37

are ordered according to the order of the Statement of Requirements	
Random orders minimum dependency count	14
Random orders maximum dependency count	40
Random orders most common range of dependency count / percentage of occurrence	20-30 range, 673 random sets out of 1000
Statistics of dependencies between individual requirements for the interface (see LAN interface dependency sets in Appendix 3): Number of dependencies that agree with the framework from the viewpoint of the simulation; vs. Number of dependencies that are not enforced by the framework from the viewpoint of the simulation, either because they contradict the framework or because they represent interdependencies between requirements within the same subcategory, i.e. a lower level of granularity than the subcategories; vs. Number of requirements with no	 27 agree – or 75% of the total 8 dependencies – or approximately 22% of the total. None of the variations were actual contradictions between the framework's ordering and the dependency analysis carried out for the purposes of the simulation. 1 neutral

dependencies, i.e. neutral with no other requirements depending on them.	

The random sets calculations for the LAN interface show an approximate 50% advantage of the framework's dependency count when compared to the most common range for the random sets. This advantage is mostly due to two key architecture related requirements (5.5.4.1.1 and 5.5.4.2.2) that had large (and identical) dependency sets. Despite the small number of LAN requirements, these two requirements with relatively large dependency sets reduced the probability of the random sets having better dependency count than the framework's order.

Furthermore, the LAN interface had the highest difference between the ratio of individual requirement's interdependencies that "agree" with the framework (75%) and the individual requirement's interdependencies that were not enforced by the framework (22%) of approximately 53%. This variation leads to a lower probability of the simulated sets having better dependency totals than the framework. For further details on how these numbers were derived please refer to the LAN interface dependency sets in Appendix 3.

8.6.7 Analysis of the results

8.6.7.1 What the simulation-based results mean?

An overall analytical[85] value that can be derived from the simulation-based results presented in the earlier subsections (Subsection 8.6.2 to 8.6.6) is the ratio of framework's dependency count for each of the five interfaces vs. the median number of the most common dependency count range that featured in the random simulation.

For example, for the Firelink interface, this ratio is calculated as follows:

[85] Due to their qualitative origins, the results are suitable for analytical (Yin, 2009, p. 38) and comparison purposes only and cannot be expected to precisely describe real-life phenomena

The dependency count for the set of requirements according to the framework: 47
Median of the most common dependency count range in the simulation: 95

Therefore:

- The "effectiveness improvement" ratio for the Firelink interface is (95-47)/95 = 50.5%

Using the same calculation with the remaining simulation results, the effectiveness improvement ratios for the other four interfaces are as follows:

- Secondary Radio Bearer Interface: (85-46)/85 = 45.9%
- Telephony interface : (55-38)/55 = 30.9%
- WAN Interface: (65-40)/65 = 38.5%
- LAN Interface: (25-13)/25 = 48%

The framework's ordering of the requirements has shown most benefit for the Firelink and Secondary Radio bearer interfaces. The SRB interface requirements tended to describe general services and features as well as performance characteristics, but were less fragmented and with less ambiguity than the Firelink requirements, which explains the lower ratio compared to the Firelink interface. The requirements of the other three interfaces were relatively better ordered and simpler. The Telephony interface requirements tended to list a number of specific features and services, with many standalone ones, which explains the lowest ratio between the five interfaces. The WAN interface requirements were describing the overall services required rather than specific features, but had a number of complex architecture and resilience requirements that explain why the ratio was higher than the Telephony interface ratio. The LAN interface requirements were a smaller set and were also simple. However, the high ratio was due to two key requirements that featured with large dependency sets that would have skewed the results against the random orders achieving low dependency counts.

One further observation that also resulted from the simulation based analysis was the significance of interdependencies between the individual requirements, especially for the

Telephony interface figures. The benefit of adopting the framework for the Telephony interface, although generally supported by the dependency figures for the random sets, was less evident and in some cases the random sets achieved slightly better dependency counts than the framework. This was due to a relatively higher number of dependencies between individual requirements that were not enforced by the framework when compared to the "neutral" or "agree" dependencies.

8.6.7.2 Variations between the framework and the dependency sets

Because the framework is a conceptual framework based on the nineteen subcategories of requirements, there were instances (indicated within the dependency sets tables of Appendix 3) when the interdependencies between individual requirements were either not enforced by or varied from the interdependencies between the subcategories of the framework. In total there are 120 such dependencies across the five communications interfaces. 110 of the 120 are due to interdependencies identified between requirements within the same test subcategory. Such variations are to be expected as the framework's granularity does not extend beyond the nineteen test subcategories.

However, 10 of these variations were actual contradictions to the flow of dependencies according to the framework. Closer analysis of the 10 contradictions identified a pattern: all 10 contradictions seemed to relate to standards, system configuration rules or routing rules. The three diagrams below highlight the 10 contradictions using dotted arrows placed on the communications interface diagrams B.1, B.2 and B.3 from Chapter 6.

Further details about the 10 contradictions shown on diagrams B.1, B.2 and B.3 are provided in the explanatory footnotes in the dependency tables in Appendix 3: Section 12.2 for B.1 contradictions, Section 12.4 for B.2 contradictions and Section 12.6 for B.3 contradictions.

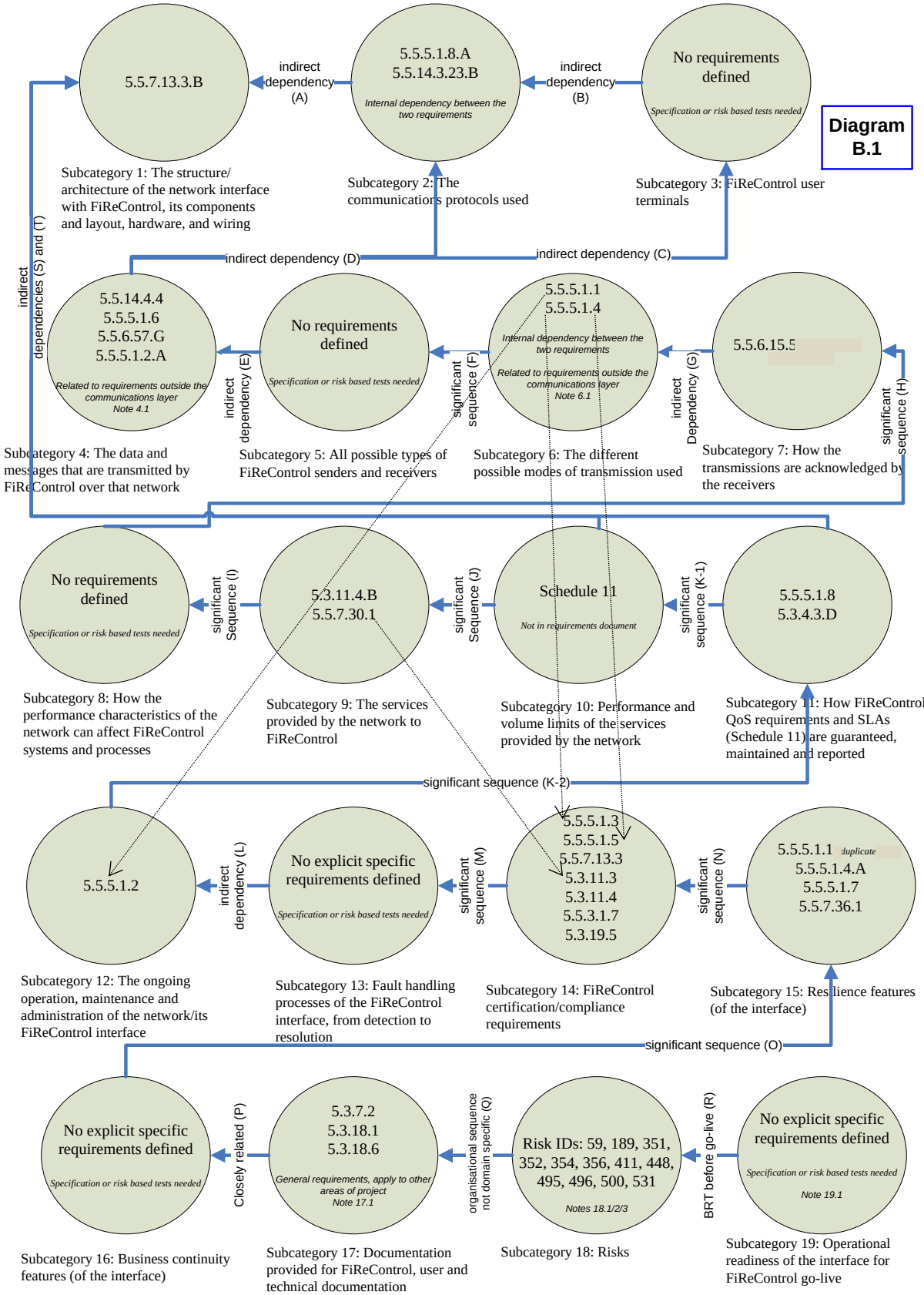


Diagram B.1
5.5.7.13.3.B
indirect dependency (A)
5.5.5.1.8.A
5.5.14.3.23.B
Internal dependency between the two requirements
indirect dependency (B)
No requirements defined
Specification or risk based tests needed
Subcategory 1: The structure/ architecture of the network interface with FiReControl, its components and layout, hardware, and wiring
Subcategory 2: The communications protocols used
Subcategory 3: FiReControl user terminals
indirect dependency (D)
indirect dependency (C)
indirect dependencies (S) and (T)
5.5.14.4.4
5.5.5.1.6
5.5.6.57.G
5.5.5.1.2.A
Related to requirements outside the communications layer
Note 4.1
indirect dependency (E)
No requirements defined
Specification or risk based tests needed
significant sequence (F)
5.5.5.1.1
5.5.5.1.4
Internal dependency between the two requirements
Related to requirements outside the communications layer
Note 6.1
indirect Dependency (G)
5.5.6.15.5
significant sequence (H)
Subcategory 4: The data and messages that are transmitted by FiReControl over that network
Subcategory 5: All possible types of FiReControl senders and receivers
Subcategory 6: The different possible modes of transmission used
Subcategory 7: How the transmissions are acknowledged by the receivers
No requirements defined
Specification or risk based tests needed
significant Sequence (I)
5.3.11.4.B
5.5.7.30.1
significant Sequence (J)
Schedule 11
Not in requirements document
significant sequence (K-1)
5.5.5.1.8
5.3.4.3.D
Subcategory 8: How the performance characteristics of the network can affect FiReControl systems and processes
Subcategory 9: The services provided by the network to FiReControl
Subcategory 10: Performance and volume limits of the services provided by the network
Subcategory 11: How FiReControl QoS requirements and SLAs (Schedule 11) are guaranteed, maintained and reported
significant sequence (K-2)
5.5.5.1.2
indirect dependency (L)
No explicit specific requirements defined
Specification or risk based tests needed
significant sequence (M)
5.5.5.1.3
5.5.5.1.5
5.5.7.13.3
5.3.11.3
5.3.11.4
5.5.3.1.7
5.3.19.5
significant sequence (N)
5.5.5.1.1 duplicate
5.5.5.1.4.A
5.5.5.1.7
5.5.7.36.1
Subcategory 12: The ongoing operation, maintenance and administration of the network/its FiReControl interface
Subcategory 13: Fault handling processes of the FiReControl interface, from detection to resolution
Subcategory 14: FiReControl certification/compliance requirements
Subcategory 15: Resilience features (of the interface)
significant sequence (O)
No explicit specific requirements defined
Specification or risk based tests needed
Closely related (P)
5.3.7.2
5.3.18.1
5.3.18.6
General requirements, apply to other areas of project
Note 17.1
organisational sequence not domain specific (Q)
Risk IDs: 59, 189, 351, 352, 354, 356, 411, 448, 495, 496, 500, 531
Notes 18.1/2/3
BRT before go-live (R)
No explicit specific requirements defined
Specification or risk based tests needed
Note 19.1
Subcategory 16: Business continuity features (of the interface)
Subcategory 17: Documentation provided for FiReControl, user and technical documentation
Subcategory 18: Risks
Subcategory 19: Operational readiness of the interface for FiReControl go-live

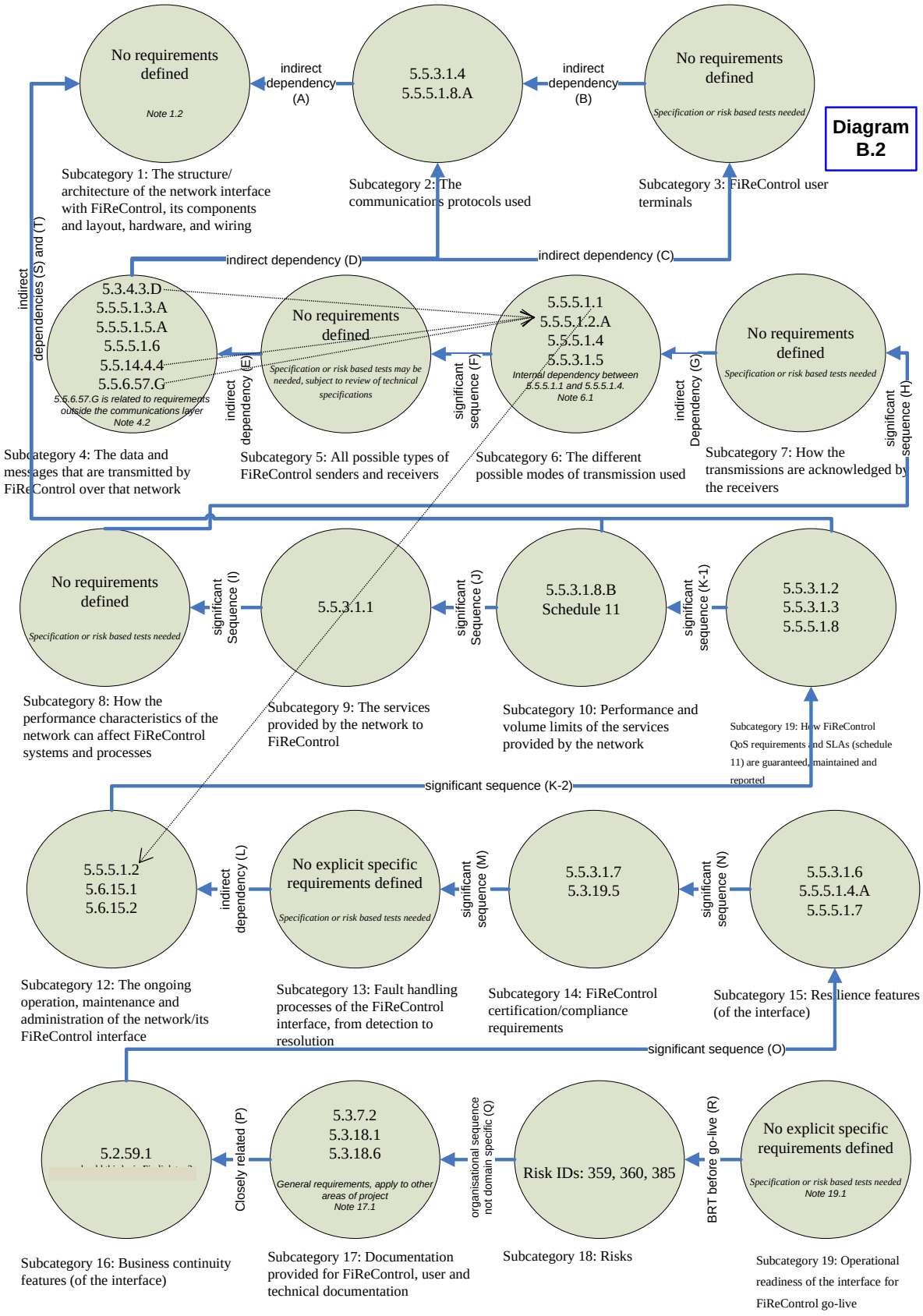


Diagram B.2
No requirements defined
Note 1.2
indirect dependency (A)
5.5.3.1.4
5.5.5.1.8.A
indirect dependency (B)
No requirements defined
Specification or risk based tests needed
Subcategory 1: The structure/ architecture of the network interface with FiReControl, its components and layout, hardware, and wiring
Subcategory 2: The communications protocols used
Subcategory 3: FiReControl user terminals
indirect dependency (D)
indirect dependency (C)
indirect dependencies (S) and (T)
5.3.4.3.D
5.5.5.1.3.A
5.5.5.1.5.A
5.5.5.1.6
5.5.14.4.4
5.5.6.57.G
5.5.6.57.G is related to requirements outside the communications layer
Note 4.2
indirect dependency (E)
No requirements defined
Specification or risk based tests may be needed, subject to review of technical specifications
significant sequence (F)
5.5.5.1.1
5.5.5.1.2.A
5.5.5.1.4
5.5.3.1.5
Internal dependency between 5.5.5.1.1 and 5.5.5.1.4.
Note 6.1
indirect Dependency (G)
No requirements defined
Specification or risk based tests needed
significant sequence (H)
Subcategory 4: The data and messages that are transmitted by FiReControl over that network
Subcategory 5: All possible types of FiReControl senders and receivers
Subcategory 6: The different possible modes of transmission used
Subcategory 7: How the transmissions are acknowledged by the receivers
No requirements defined
Specification or risk based tests needed
significant Sequence (I)
5.5.3.1.1
significant Sequence (J)
5.5.3.1.8.B
Schedule 11
significant sequence (K-1)
5.5.3.1.2
5.5.3.1.3
5.5.5.1.8
Subcategory 8: How the performance characteristics of the network can affect FiReControl systems and processes
Subcategory 9: The services provided by the network to FiReControl
Subcategory 10: Performance and volume limits of the services provided by the network
Subcategory 19: How FiReControl QoS requirements and SLAs (schedule 11) are guaranteed, maintained and reported
significant sequence (K-2)
5.5.5.1.2
5.6.15.1
5.6.15.2
indirect dependency (L)
No explicit specific requirements defined
Specification or risk based tests needed
significant sequence (M)
5.5.3.1.7
5.3.19.5
significant sequence (N)
5.5.3.1.6
5.5.5.1.4.A
5.5.5.1.7
Subcategory 12: The ongoing operation, maintenance and administration of the network/its FiReControl interface
Subcategory 13: Fault handling processes of the FiReControl interface, from detection to resolution
Subcategory 14: FiReControl certification/compliance requirements
Subcategory 15: Resilience features (of the interface)
significant sequence (O)
5.2.59.1
Closely related (P)
5.3.7.2
5.3.18.1
5.3.18.6
General requirements, apply to other areas of project
Note 17.1
organisational sequence not domain specific (Q)
Risk IDs: 359, 360, 385
BRT before go-live (R)
No explicit specific requirements defined
Specification or risk based tests needed
Note 19.1
Subcategory 16: Business continuity features (of the interface)
Subcategory 17: Documentation provided for FiReControl, user and technical documentation
Subcategory 18: Risks
Subcategory 19: Operational readiness of the interface for FiReControl go-live

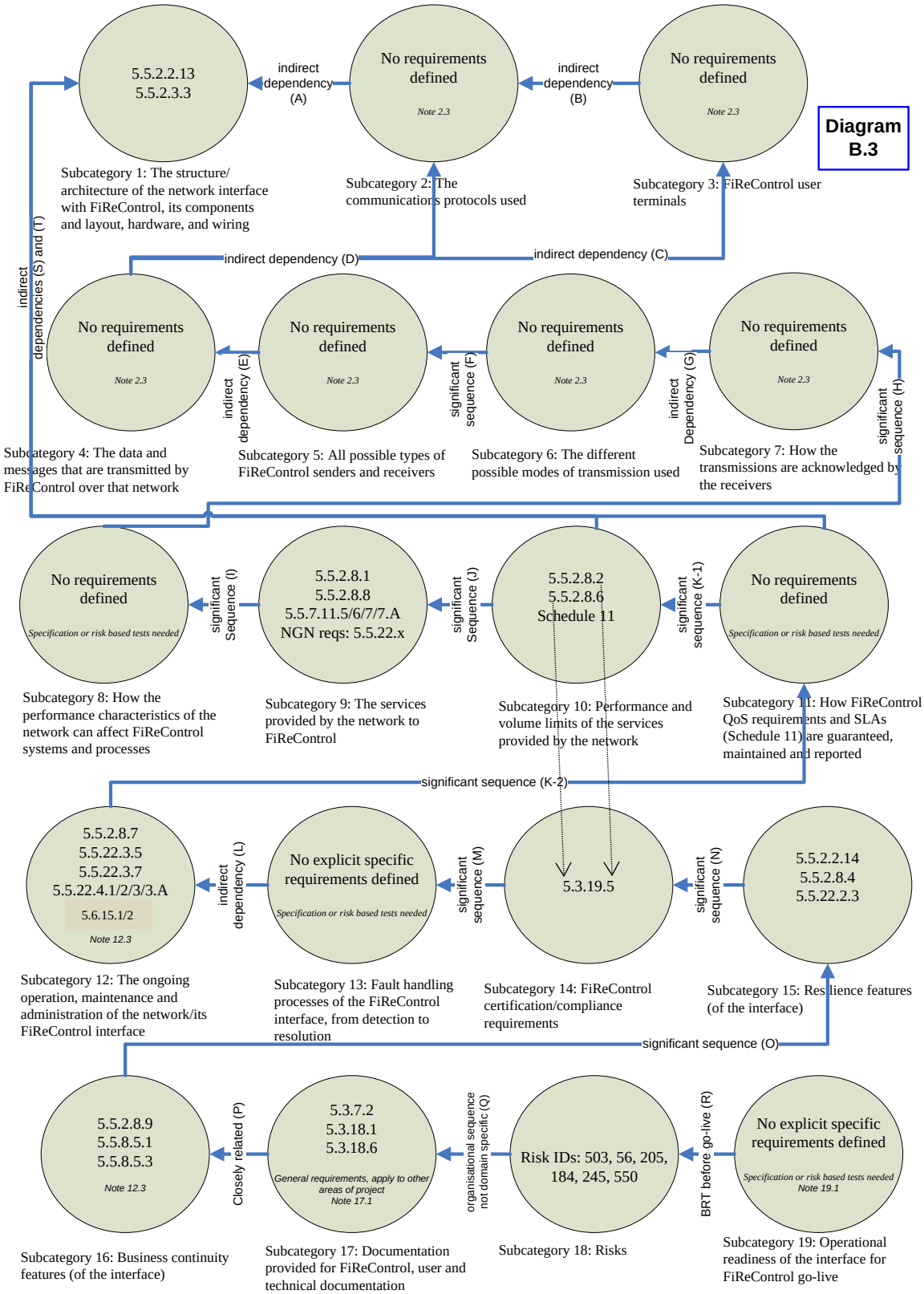


Diagram B.3
5.5.2.2.13
5.5.2.3.3
indirect dependency (A)
No requirements defined
Note 2.3
indirect dependency (B)
No requirements defined
Note 2.3
Subcategory 1: The structure/ architecture of the network interface with FiReControl, its components and layout, hardware, and wiring
Subcategory 2: The communications protocols used
Subcategory 3: FiReControl user terminals
indirect dependencies (S) and (T)
indirect dependency (D)
indirect dependency (C)
No requirements defined
Note 2.3
indirect dependency (E)
No requirements defined
Note 2.3
significant sequence (F)
No requirements defined
Note 2.3
indirect Dependency (G)
No requirements defined
Note 2.3
significant sequence (H)
Subcategory 4: The data and messages that are transmitted by FiReControl over that network
Subcategory 5: All possible types of FiReControl senders and receivers
Subcategory 6: The different possible modes of transmission used
Subcategory 7: How the transmissions are acknowledged by the receivers
No requirements defined
Specification or risk based tests needed
significant Sequence (I)
5.5.2.8.1
5.5.2.8.8
5.5.7.11.5/6/7/7.A
NGN reqs: 5.5.22.x
significant Sequence (J)
5.5.2.8.2
5.5.2.8.6
Schedule 11
significant sequence (K-1)
No requirements defined
Specification or risk based tests needed
Subcategory 8: How the performance characteristics of the network can affect FiReControl systems and processes
Subcategory 9: The services provided by the network to FiReControl
Subcategory 10: Performance and volume limits of the services provided by the network
Subcategory 11: How FiReControl QoS requirements and SLAs (Schedule 11) are guaranteed, maintained and reported
significant sequence (K-2)
5.5.2.8.7
5.5.22.3.5
5.5.22.3.7
5.5.22.4.1/2/3/3.A
5.6.15.1/2
Note 12.3
indirect dependency (L)
No explicit specific requirements defined
Specification or risk based tests needed
significant sequence (M)
5.3.19.5
significant sequence (N)
5.5.2.2.14
5.5.2.8.4
5.5.22.2.3
Subcategory 12: The ongoing operation, maintenance and administration of the network/its FiReControl interface
Subcategory 13: Fault handling processes of the FiReControl interface, from detection to resolution
Subcategory 14: FiReControl certification/compliance requirements
Subcategory 15: Resilience features (of the interface)
significant sequence (O)
5.5.2.8.9
5.5.8.5.1
5.5.8.5.3
Note 12.3
Closely related (P)
5.3.7.2
5.3.18.1
5.3.18.6
General requirements, apply to other areas of project
Note 17.1
organisational sequence not domain specific (Q)
Risk IDs: 503, 56, 205, 184, 245, 550
BRT before go-live (R)
No explicit specific requirements defined
Specification or risk based tests needed
Note 19.1
Subcategory 16: Business continuity features (of the interface)
Subcategory 17: Documentation provided for FiReControl, user and technical documentation
Subcategory 18: Risks
Subcategory 19: Operational readiness of the interface for FiReControl go-live

The new test framework's categorisation considered that requirements relating to the adherence of a system to IT or other standards[86], system configuration[87] or message routing rules[88], i.e. adherence to some type of specification or an external standard, to be less fundamental from a test prioritisation viewpoint than requirements for the way the system is built and operated[89]. This was based on the thinking that "compliance" aspects such as these are more appropriately tested after the functional and non-functional features of the system have been proven. However, when the requirements were reviewed individually for inter-dependencies, the "compliance" type of requirement was treated as more fundamental and hence needs to be tested earlier, also that the detailed functionality requirements actually depended on them. This change of ordering between the framework and the dependency sets seems to have occurred because of a difference between a technically informed vs. project management views of testing.

The higher the ratio of "variations"[90] between an interface's dependency sets and the framework, the more likely that the random sets are able to achieve a similar or even better dependency count total than the framework. This effect was most noticeable in the Telephony interface figures, and to a lesser extent in the WAN interface figures. The percentage difference between individual interdependencies that "agreed" with the framework vs. ones that "varied" from it was approximately 18% for the Telephony dependency sets and 31% for the WAN dependency sets. As for the remainder of the interfaces, the difference was approximately 46% for the Firelink interface, 41% for the SRB interface and 53% for the LAN interface.

8.6.7.3 An overall "effectiveness improvement" ratio

According to the simulation based figures, the average "effectiveness improvement" for the

[86] such as 5.5.5.1.3, 5.5.5.1.5, 5.5.7.13.3, 5.3.19.5

[87] such as 5.5.5.1.2

[88] such as 5.5.5.1.2.A

[89] such as 5.5.5.1.1, 5.5.5.1.4, 5.5.7.30.1, 5.5.14.4.4, 5.5.5.6.57.G, 5.3.4.3.D, 5.5.2.8.2, 5.5.2.8.6

[90] Variations due to inter-dependencies within the same subcategory that are not enforced by the framework as well as actual contradictions

five interfaces from adopting the communications specific test framework is 42.8%. This exact ratio by itself is an analytical (Yin, 2009, p. 38) indicator specific only to the FireControl case study. It represents simulation-based evidence that the new test framework, when applied to FireControl's communications layer, was capable of producing efficiency benefit to the requirements-based test analysis and design work for the communications layer of FireControl.

8.7 EV.5: Can the framework be compared to a rival?

One further approach to increasing the reliability of case study research is to carry out a comparison of the "case study theory" with a "rival theory" (Yin, 2009). The research work presented in this thesis is primarily design-based rather than theory-based[91] and is practice-inspired relevant to industrial real-life practices in testing. Therefore, to adapt Yin's idea of comparison with a rival theory to this thesis, the "new test framework" needs to substitute the "theory" (of this thesis) and the "rival theory" needs to be a comparable or potentially competing framework. For this purpose, an appropriate "rival theory" is identified as the V-Model test methodology (Mathur & Malik, 2010). As stated previously in this thesis, it is not one of the objectives for this thesis to "disprove" the V-Model but because of its wide scale adoption for testing in industry, it is useful as a "baseline" when evaluating the benefits of the new test framework.

8.7.1 Simulation-based method for comparing the new test framework with a rival

The simulation-based evaluation work that was presented in Section 8.6 was developed further to provide a comparison between the new test framework, the V-Model as well as

[91] That's not to say that it has no theoretical basis behind its ideas, as is discussed in chapter 1, section 0, Section 4.6 and Section 5.2.1

fully randomised[92] sets of requirements.

Before presenting the results of the simulation, below is an explanation of the additional work that was carried out to extend the initial simulation work, as well as the assumptions made.

- Considered how the comparison between the new framework and the V-Model can be carried out meaningfully. A number of considerations had to be taken into account, namely: the V-Model is a generic methodology for testing whole systems, it would not normally be applied to the "communications layer" separately, nor would it typically be applied to individual communications interfaces in isolation. However, carrying out such comparison could produce interesting results that can improve the validity of the ideas presented in this thesis, as long as it is acknowledged that they can only be considered as one component of the overall supporting evidence.

- Therefore, a key assumption had to be made in order to reduce the number of variables when comparing between the new framework and the V-Model. This assumption is that the V-Model is applicable to the "communications layer" on its own.

- The five "dependency sets" (Appendix 3) that were used in the initial simulation of Section 8.6 were joined together into one large dependency set for all interfaces. This was done to enable a single simulation including all communications requirements combined.

- The Excel macro's functionality was enhanced[93]/[94] as follows:
  1- Instead of reading just one fixed set of requirements ordered according to the new

[92] The randomisation is applied to the ordering of the requirements within the same set, while the requirements themselves remain unchanged.

[93] The enhanced functionality was implemented using the same service which was used for the initial simulation in section 8.6. The definition of the additional simulation functionality, how the statistical data should be presented, and the testing of the spreadsheet template was done by the author of this thesis.

framework, it was amended to take input from two worksheets, one containing a set of requirements ordered according to the V-Model's five phases and another containing the same requirements ordered according to the new test framework's nineteen subcategories.

2- The dependency sets data was amended to contain the new combined dependency set which was created by joining all five dependency sets into one and removing any duplicated entries.

3- The macro is then amended to generate three separate simulations: one "stratified" simulation for V-Model data, meaning the order of the five V-Model phases always remaining the same but the order of the subsets of the requirements within each of the phases being randomly generated. The other simulation was also stratified, using the data ordered according to the new framework's nineteen subcategories. The third simulation was a fully random non-stratified simulation.

4- For each randomly generated sequence of IDs in each simulation, the macro calculates its "dependency total". The macro then shows the spread of all dependency totals in a simulation in both tabular as well as distribution graph form.

5- The distribution graphs are displayed separately for each simulation as well as together on the same X/Y axis to help with the comparison.

The above changes were done with the intention of allowing a comparison of "efficiency" between the V-Model, the new test framework and fully random sets of requirements. The simulations for both the V-Model and the new test framework used stratified randomisation of the requirements according to the five phases of the V-Model and the nineteen subcategories of the new test framework.

[94] As well as testing the functionality of the enhanced spreadsheet, tests were also carried out to check that it is processing the subcategories and phases data correctly. One particular test was devised to check that the ANOVA calculations are able to detect similarity between two different sets of new framework and V-Model data. The test spreadsheet is named "BiasTest" and is available on CD Supplement No.2. It shows the effect size indicated as "comparable" for the new framework vs. V-Model comparison when presented with similarly sequenced lists of test data.

- Each requirement was classified according to where it best fits under the V-Model's five test phases. The five interfaces tables in Appendix 3 show how the communications requirements were classified under the five V-Model test phases, together with the new combined dependency set.

  Simulations were then run for the five communications interfaces with their respective data to compare 1000[95] randomly generated sets stratified according to the five V-Model phases as well as the nineteen subcategories of the new test framework. Fully random simulations, not ordered according to any of the two methods, were also included to give an indication for how both frameworks compare to a fully random set.

- One further, and probably most significant, simulation using a single set of all communications requirements was also carried out. This was done because a single set is closer to how the V-Model is likely to be applied in real-life.

- The simulation results presented in this section are more complex than the simulations presented in section 8.6, and involve the comparison between three sets of results. Therefore, a statistical test was deemed more appropriate than simply relying on visually observing the difference between the distribution graphs, especially when they overlap. Because the simulations are intended as a comparison between three sets of data (i.e. new framework, V-Model and fully random) to determine which one is more "efficient", (i.e. has the lowest dependency counts), the ANOVA statistical test was used.

- Lastly, and as a way to further improve the reliability of these simulation results, an external participant was asked to map the Firelink requirements to the 19 test subcategories. This mapping was then used in a further simulation to show the difference between the mapping of an external participant, as a potential user of the

[95] The suitability of this number of sets was checked via a web statistics calculator (Australian Bureau of Statistics, 2013) using the following parameters: 95% confidence level, 0.03 and 0.035 confidence intervals, and a population size of 1,000,000. Sample Size results were 784 and 1,066 respectively.

framework) and the mapping done during the FireControl case study. This is presented in section 8.9.1.

The data of the simulations[96] is available in the spreadsheets on CD supplement No.2 as well as the dependecy sets subsections in Appendix 3. The same mapping for the requirements onto the nineteen test subcategories as was presented in Sections 6.4 to 6.8 was used for these simulations. The requirements also needed to be mapped to the V-Model as mentioned earlier. Therefore, to carry out the simulations for the V-Model mapping, each requirement was assigned to a V-Model phase as follows: phase 1 for design review, phase 2 for unit+subsystem testing, phase 3 for integration testing, phase 4 for system testing and phase 5 for UAT+OAT[97]. These mappings are included as part of the tables in the dependency sets sections of Appendix 3.

The following six subsections (Sections 8.7.2 to 8.7.7) contain the simulation results for the five separate interfaces as well as the sixth combined set of all communications requirements. These comprise of: The distribution graphs[98] of the simulations as well as the associated ANOVA[99] data, followed by discussions of the results. The complete log of the simulation results is available on CD supplement No.2 to this thesis which contains the Excel spreadsheets used to carry out the calculations.

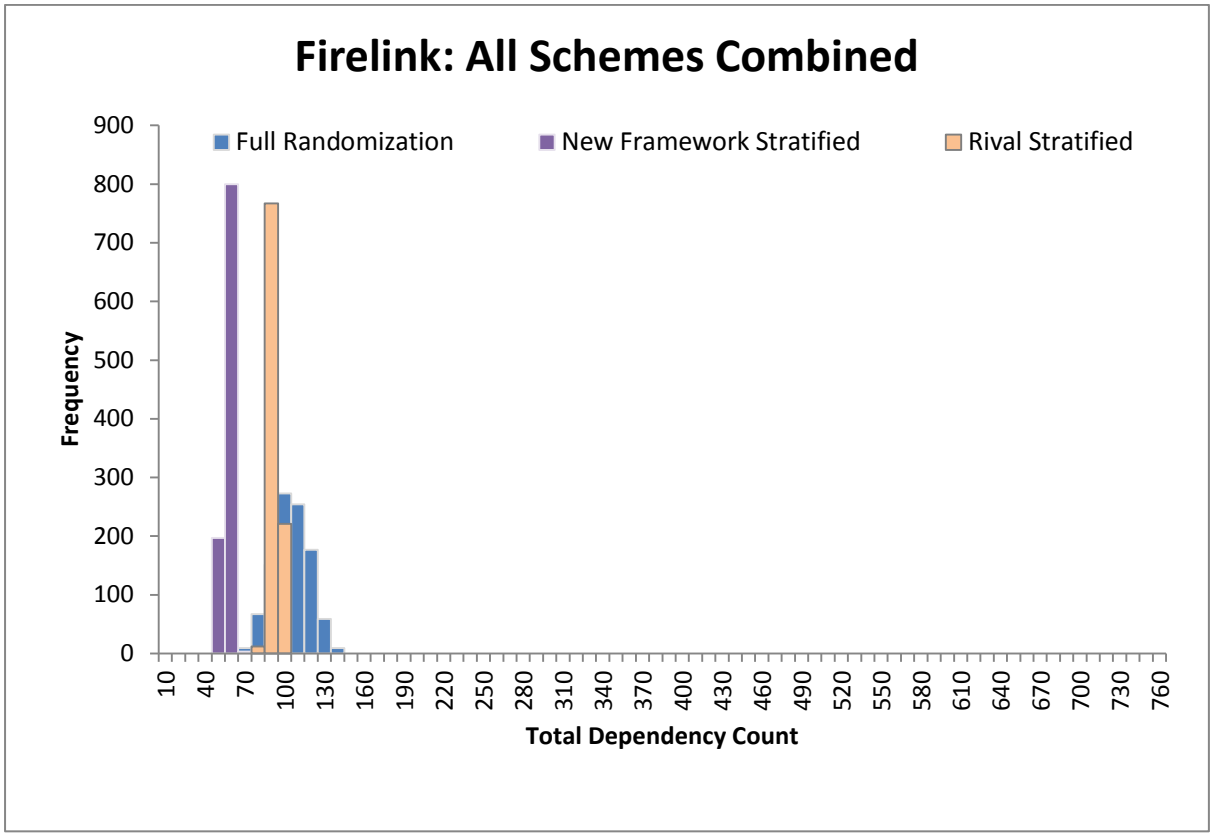
[96] The combined single dependency set, the mapping of the requirements of each of the interfaces to the V-Model's 5 phases, the combined requirements set for all communications requirements and its mapping to the new test framework and to the V-Model phases.

[97] For the purposes of CCLSS testing and with a view to the level of granularity of the FireControl requirements, it was deemed more realistic and meaningful to combine unit and subsystem (i.e. component testing) as well as UAT+OAT (acceptance testing) together.

[98] The graphs show the ranges of dependency counts achieved by the random sets vs. the number of random sets that had a dependency count falling within each range.

[99] The ANOVA data is auto-generated by the enhanced functionality of the Excel spreadsheet.

8.7.2 Firelink interface



	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	52.99	87.91	100.57
St Dev	2.83	3.49	13.42
Var	7.98	12.18	180.21
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	3.17	-11.00	Different
New Framework vs Fully Random	9.70	-4.91	Different
Rival Model vs Fully Random	9.80	-1.29	Different

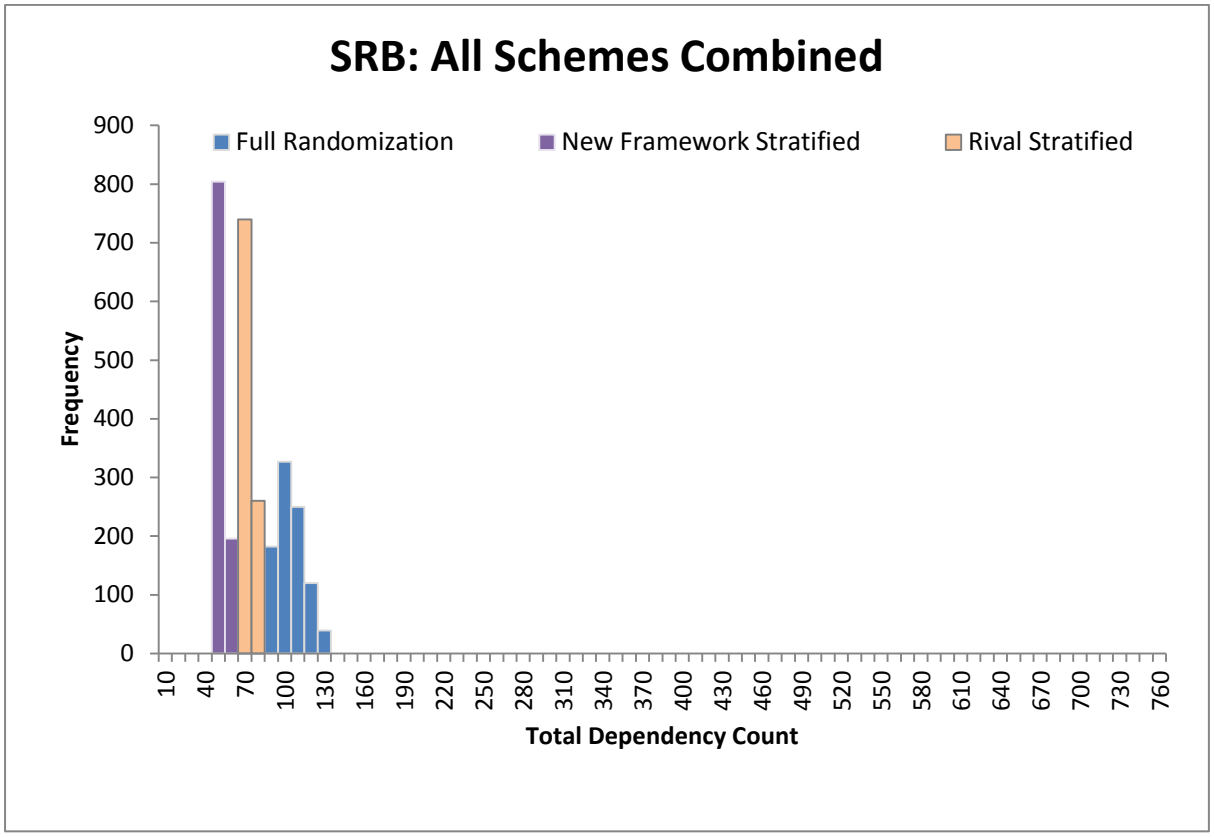
The results above show that, both by visual inspection as well as from the ANOVA data, the new test framework is more efficient than the V-Model and both are more efficient than the

fully randomised sets. The minus values are because the graphs for the new framework and the V-Model are both to the left of the fully random sets and have lower values. The most important values are the "effect size" values which show the sizes of the statistically significant differences for the three comparisons shown: (a) the new framework vs. the rival (V-) model (b) new framework vs. fully random, and (c) the rival model vs. fully random.

The effect size for comparison (a) shows a bigger value[100] than for (b) which might seem counter-intuitive when visually inspecting the distribution graph because the rival model values seems mostly lower than the full random values. This is due to the larger "s_pooled" value for the second comparison relative to the first (9.70 compared to 3.17). This value relates to the width of the distribution graph and a larger value would cause the effect size to be smaller. Furthermore, the effect size for comparison (c), although statistically significant, is smaller than the effect size for the other comparisons. This means that the rival model is only slightly more efficient than the full random set. In contrast, the advantage of the new framework over both the rival model and the fully random sets is larger.

[100] Ignoring the minus sign, because the absolute size of the value is the important indicator

8.7.3 SRB interface



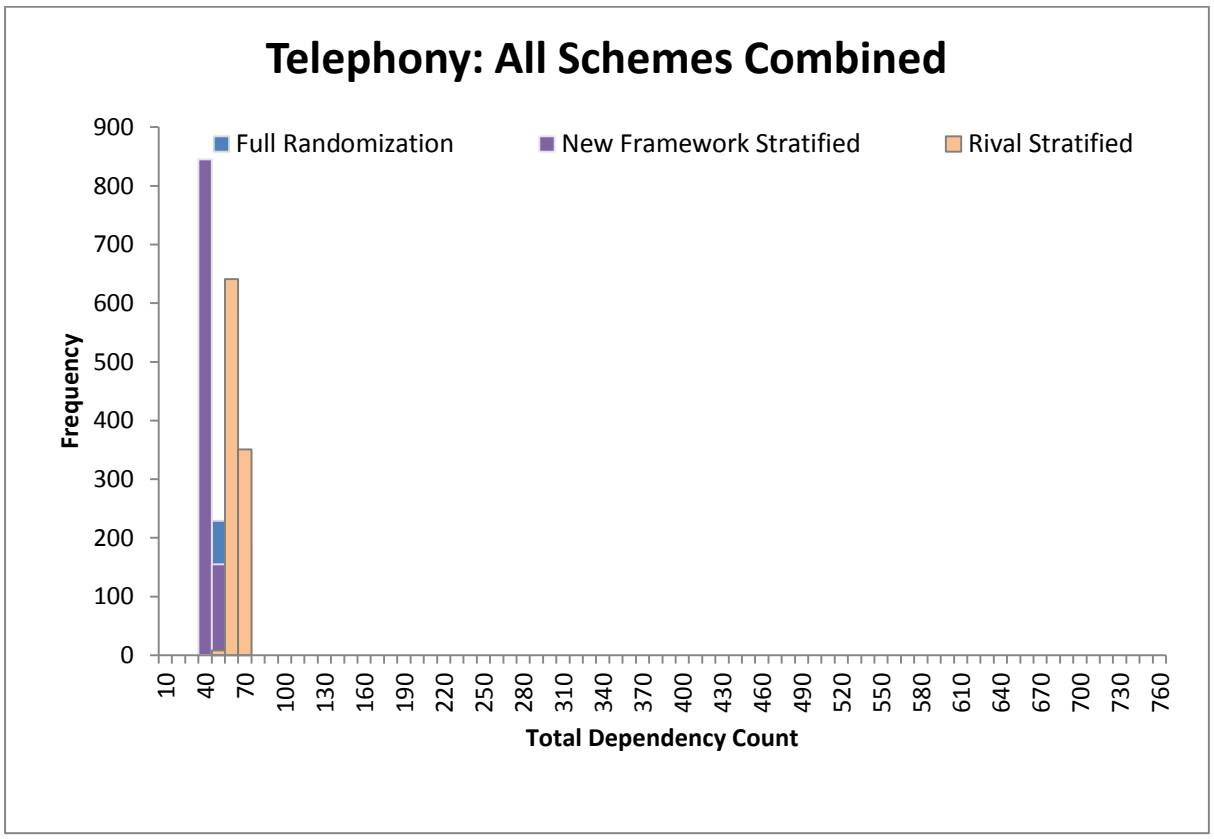
	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	48.99	69.00	98.02
St Dev	1.69	2.32	12.33
Var	2.85	5.40	151.99
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	2.03	-9.86	Different
New Framework vs Fully Random	8.79	-5.58	Different
Rival Model vs Fully Random	8.87	-3.27	Different

The results for SRB continue the pattern of the results for Firelink in the previous section, although with a smaller effect size value for the first comparison of the new framework vs.

the V-Model. Also for SRB the advantage of the V-Model over the fully random sets is larger than is the case for the Firelink interface.

The SRB requirements were less fragmented and better structured than the Firelink requirements, yet generally comparable in complexity. This might explain why the distribution graphs and the ANOVA data are relatively close in size and distribution to those for Firelink but with the advantage of the new framework with the Firelink interface being slightly larger than its advantage with the SRB interface (-11 for Firelink, -9.89 for SRB). Also, the spread of values for the new framework and the rival model is smaller than for the fully random sets. This is due to the randomisations for both the new framework and the rival model being "stratified", i.e. limited to within the subcategories/phases, which limits the amount of variations possible compared to a fully random simulation.

8.7.4 Telephony interface



	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	39.47	58.94	55.00
St Dev	0.95	3.86	5.96
Var	0.90	14.90	35.51
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	2.81	-6.93	Different
New Framework vs Fully Random	4.26	-3.64	Different
Rival Model vs Fully Random	5.02	0.79	Different

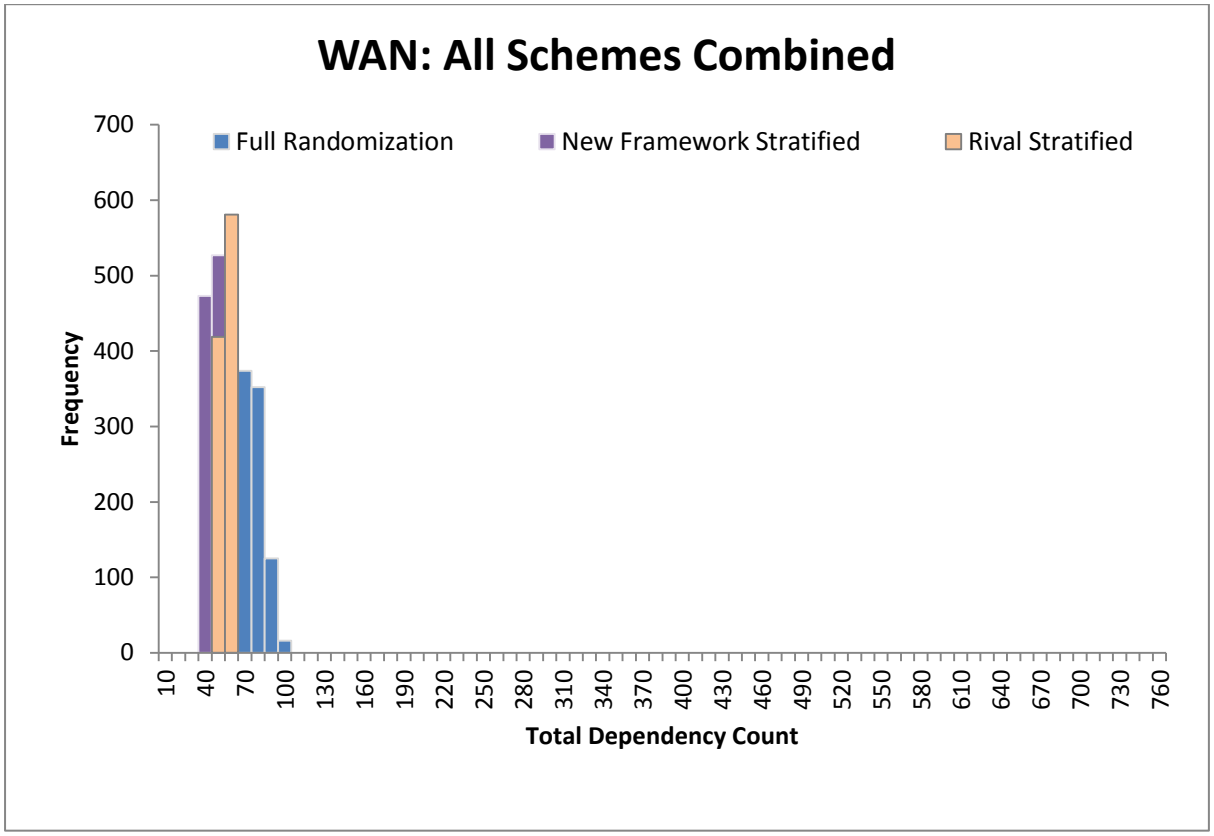
The telephony results above concur with the results of an earlier simulation for the Telephony interface presented in Subsection 8.6.4, especially when compared to the results for Firelink and SRB. Although statistically significant, the effect sizes for the differences between the new framework and the rival model (V-Model) are smaller at -6.93[101] than the equivalent values presented earlier in this section (Subsections 8.7.2and 8.7.3) for Firelink and SRB. What is even more noticeable is that the rival model was slightly less efficient than the fully random sets, indicated by the positive small value of effect size of 0.79.

The results for the new framework's comparisons are consistent with the results in Subsection 8.6.4 for the reasons given there. The Telephony interface requirements were relatively well ordered in the original statement of requirements (SoR) and they tended to describe stand-alone telephony features. Therefore, the 19 subcategories of the new test framework did lead to a better structured set of requirements with fewer interdependencies, but its advantage over both the V-Model and the full random sets is smaller when compared to either Firelink or SRB interfaces which were more complex and more fragmented in the SoR.

[101] In absolute terms, ignoring the minus sign

The V-Model showed a slight disadvantage to the full random sets. This is because all its requirements except two, due their feature/service oriented nature, belonged to the last two V-Model phases of system and acceptance testing. This meant it was not much different from the fully random sets in terms of requirements structure yet its randomisation was stratified, which restricted the variance of dependency totals it achieved during the simulation[102].

8.7.5 WAN interface



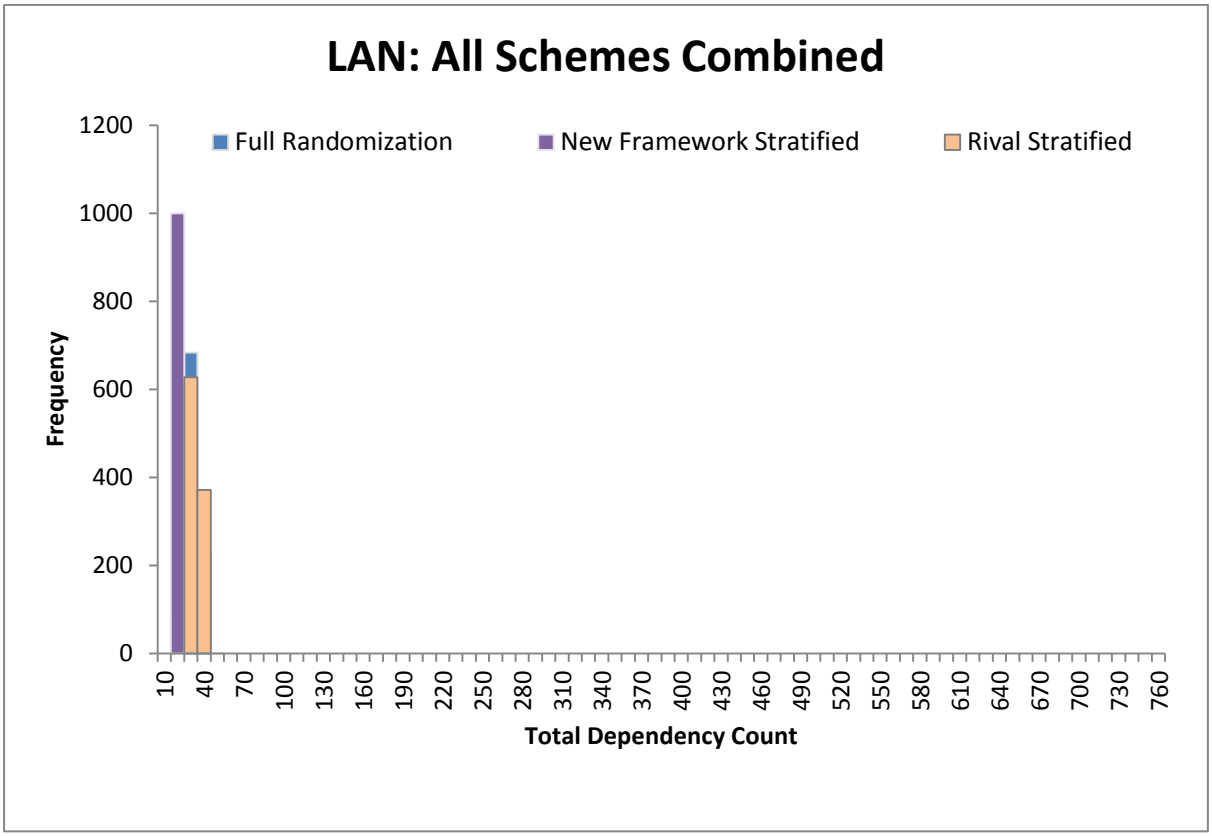
[102] This discussion is not expanded further because the purpose of this section is to compare the new framework to the rival V-Model rather than compare the V-Model to the fully random sets

	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	40.54	50.97	70.47
St Dev	2.16	2.24	9.12
Var	4.69	5.00	83.25
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	2.20	-4.74	Different
New Framework vs Fully Random	6.63	-4.52	Different
Rival Model vs Fully Random	6.64	-2.94	Different

The WAN results also continue to show an advantage for the new test framework that is smaller, in terms of effect size value of -4.74[103], than that shown for the previous three interfaces: Firelink, SRB and Telephony. The results for WAN also visually show more overlap between the three distribution graphs, which is an example where the ANOVA data is of particular help for understanding and analysing the results. The V-Model for WAN requirements did not have the same disadvantage it had with Telephony when most of the requirements belonged to the last two phases. The WAN requirements were better spread over four of the five V-Model phases and were less stand-alone (i.e. had more interdependencies) than the Telephony requirements. These are the factors which explain why the effect size results for the comparisons between the new framework, the rival (V-Model) model and the fully random sets were smaller when compared to the Telephony simulation. However, the overall result still showed an advantage for the new test framework, albeit with more overlap and closeness to the V-Model results.

[103] In absolute terms, ignoring the minus sign

8.7.6 LAN interface



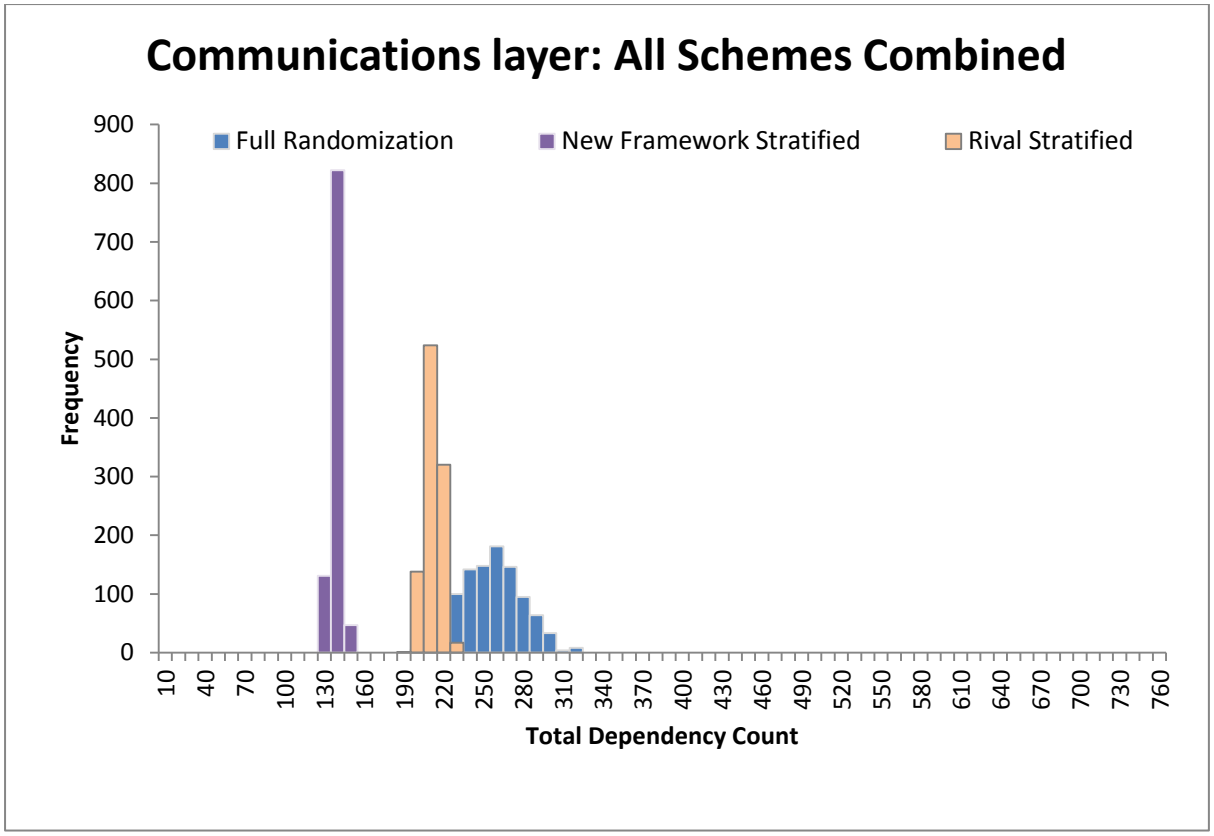
	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	13.99	29.41	27.02
St Dev	0.82	3.40	4.65
Var	0.67	11.56	21.63
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	2.47	-6.24	Different
New Framework vs Fully Random	3.34	-3.90	Different
Rival Model vs Fully Random	4.07	0.59	Different

The LAN simulation results follow a pattern closer to the results of the Telephony simulation presented earlier in Subsection 8.7.4. The LAN requirements set is the smallest set amongst

the five interfaces with only 10 requirements, with eight of them are classed as acceptance testing under the V-Model phases. This is the main factor explaining why the full random sets performed better with 0.59 effect size. However, the comparison between the new test framework and the V-Model for the LAN simulations confirms the pattern of the other four interfaces which shows an advantage to the new test framework represented by effect sizes ranging from (-)11 to (-)4.74.

8.7.7 All communications requirements combined

This simulation combined all communications interfaces requirements into one set. As well as combining the dependency sets for the five interfaces into one, conducting this simulation meant also having one set of requirements classified according to the 19 test subcategories as well as according to the five phases of the V-Model. The reason this was done is because keeping the interfaces separate is more meaningful for the new test framework, but for the V-Model it is more meaningful to have all communications requirements combined into one set. Therefore it was thought that adding this combined simulation of all communications requirements could improve the reliability and usefulness of the results for analysis purposes.



	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	134.48	207.52	251.93
St Dev	3.62	6.49	22.47
Var	13.13	42.17	505.00
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	5.26	-13.90	Different
New Framework vs Fully Random	16.09	-7.30	Different
Rival Model vs Fully Random	16.53	-2.69	Different

The outcome of this simulation was of particular significance because the results were unpredictably more definitive and clearer than the results of the individual interfaces presented in Subsections 8.7.2 to 8.7.6 earlier. The distribution graphs are visually more

separated showing a clear advantage to the new test framework over the rival model (V-Model) and the fully random sets, with relatively small variance and deviation values (13.13 and 3.62). The small variance is a desirable indicator in a test framework because it points to the framework being relatively more precise and less subjective. The idea of precision in testing was discussed in Chapter 1 when identifying the gap in software testing for CCLSS which motivated the early ideas of this thesis.

The effect size[104] for the first comparison between the new test framework and the rival model (V-Model) is larger than the effect size for the second comparison between the new test framework and the fully random sets. This might be counter-intuitive when the graphs are viewed visually. The V-Model looks more efficient than the fully random sets, so why is the new test framework "closer" by effect size to the V-Model? This discrepancy can be explained by the difference in the "s_pooled" values of 5.26 and 16.09 respectively. The larger s_pooled value results in a smaller effect size and is a result of the width of the fully random distribution graph[105].

The outcome of this simulation is that it confirms, with a larger effect size of -13.90[106] than the earlier Subsections 8.7.2 to 8.7.6, the efficiency advantage of the new test framework over the V-Model.

8.7.8 Analysis of the results

The results presented in this section (Section 8.7), including their variations, were internally consistent with each other and also supported the results of the earlier set of simulations presented in Section 8.6. The way they were derived used a defined repeatable process explained at the beginning of the section and using data that is preserved both in Appendix 3 as well as the CD Supplement No.2 that accompanies this thesis. Due to the qualitative

[104] The absolute value is what is significant for this discussion, as the minus sign is only a reflection that the smaller values are "better" and that a graph displayed more to the left is more efficient

[105] This discussion is not expanded further because the purpose of this section is to compare the new framework to the rival V-Model rather than compare the V-Model to the fully random sets

[106] In absolute terms, ignoring the minus sign

origin of the data that was used for these simulations, the data presented in this section could only be used for general analytical purposes[107].

Referring back to the original question for EV.5 which is the purpose of this section, visual as well as ANOVA comparisons between new framework and V-Model presented in this section consistently showed a tangible advantage to the new framework over the V-Model in all of the six simulations.

Also consistent with the results of the earlier and simpler set of simulations presented in Section 8.6, the new framework exhibited most "efficiency" when applied to the Firelink and SRB interfaces. For Telephony, WAN and LAN interfaces it still outperformed both the randomly generated sets as well as the V-Model sets but to a lesser extent than Firelink and SRB. This is visible from the new framework vs. V-Model effect size values of: -11 for Firelink, -9.89 for SRB, -6.93 for Telephony, -4.74 for WAN and -6.24 for LAN[108].

The most important simulation for EV.5 purposes is probably the combined "all communications requirements" simulation which had an effect size of -13.9 for the comparison between the simulation data of the new framework vs. the V-Model. It may not be appropriate to draw generalised conclusions based on the results of each individual simulation in its own right, but when combined they represent tangible evidence regarding the efficiency of the new test framework compared to the V-Model specifically, at least, for the FireControl communications layer.

Another possible observation from the results is that they might be used to point to circumstances when the new test framework can provide more efficiency benefit. This is by comparing the results of the simulations and observing when the effect size was larger, which is when the requirements sets were more interdependent, more complex and more fragmented

[107] Because the data has qualitative origins, any values produced via the analysis are not likely to precisely represent real phenomena.

[108] The minus values for the effect size figures are because the new framework graphs are on the left of the other graphs i.e. has lower dependency totals than the V-Model and fully random sets. For the purpose of comparison of effect sizes, it is the absolute value that is considered.

as was the case for the Firelink and the SRB interfaces. However, this can only be described as an observation which could be subject to further investigation and evaluation, rather than definite evidence.

Overall, this section presented a new method, devised specifically for the purposes of this thesis, showing how the new test framework can be compared to a rival. The results of the six simulations that it presented showed that the new framework is more "efficient" than the V-Model for the sets of requirements of the five interfaces of FireControl when the application of each of the two (new framework vs. V-Model) is simulated with the requirements of each interface as well as when all communications requirements are combined into one set.

Lastly, a question that might arise in light of the simulation work presented in this section and in Section 8.7 could be the following: "can the simulation method be used instead of the new test framework to decide on the most optimal sequencing of the requirements?"

A pre-requisite to carrying out the simulation work is a detailed expert analysis of the requirements' interdependencies one-by-one. This is a type of complex test analysis effort the new test framework is intended to simplify, and needed to be carried out before the simulation work was possible. In the context of this thesis, the simulation work was done to evaluate the new framework and was not intended nor is suitable to replace its conceptual representation and simplification of a communications interface for test analysis and design purposes. However, subject to further work, it may be viable for the simulation method described in this section and in Section 8.7 to be adapted for other uses, but that's not likely to be in a context similar to that of the case study included in this thesis.

8.8 EV.6: Can the framework fulfil the initial intended criteria?

This section presents the EV.6 related evidence which was obtained from external participants as part of the user-based evaluation[109] presented earlier in Chapter 7.
The evaluation used a survey style format, with a number of questions asking the participants to provide their feedback and opinions about the viability and benefits of the framework based on their own past experience working on different CCLSS. Question 5 of the feedback form was tailored specifically around the two lists of "assumptions", "structural specifications"[110]/[111] as originally outlined in Chapter 4 (Sections 4.2 and 4.3). Question 5 was intended specifically to obtain the external participants' evaluation of the framework in a form that corresponds to evidence item EV.6. For clarity, the term "criteria" will be used for the rest of this subsection to also encompass "attributes" and "features".

The external participants were provided with a list of criteria and were asked to provide their opinions on whether the new test framework is likely to fulfil them. The form had four tick-boxes next to each of the criteria. The participants were asked to select one of the tick-boxes (Likely, Neutral, Not Likely, Don't Know) and also had space to provide further optional comments next to each of the criteria.

The list of criteria[112] included in Questions 5 of the feedback form is below:

- Suitability for testing communications-critical large scale systems
- Enabling testing to be linked to the system's requirements, and being the basis for evidence on whether the requirements have been fulfilled
- Enabling early detection of faults in a new system
- Applicable to the full lifecycle of a new system

[109] The user-based evaluation was not limited to the FireControl case study. The external participants were asked for their opinions and feedback about the new test framework with reference to their experiences working on other CCLSS

[110] These are terms used by (Verschuren & Hartog, 2005) equivalent to the criteria, attributes and features of the new test framework as outlined in chapter 4, particularly sections 4.2 and 4.3, explaining some of the early thinking and ideas that shaped the form and potential uses of the new test framework

[111] Relating to the form and uses of the new test framework

[112] For an explanation of the motivation for each of the criteria see Section 7.4 – Question 5

- Supporting close cooperation between the test team and the rest of an IT project team
- Useful as basis for defining a test strategy for a new system
- Provides a simplified conceptual view of a communications-critical large scale system's structure which can be used as an aid for the test analysis and design efforts
- Can be used as basis for review and verification activities of the system requirements, technical design and technical specification
- If used as basis for testing a communications-critical large scale system, allows the testers to start their verification and validation work from an early stage of the project
- Can facilitate test traceability and coverage analysis
- Have potential uses in an IT project outside purely testing, e.g. providing a shared view of a system for business analysts, developers, testers and suppliers/vendors

The responses were confirmatory that, on the whole, the new test framework does fulfil the initial intended criteria. The majority of the ticked boxes were "Likely", with six "Neutral" and three "Don't Know". To avoid duplication between this and the previous chapter, see the responses by the external participants to Question 5 in Section 7.4.

8.9 EV.7: Can the framework be applied by other potential users/participants?

This section presents the EV.7 related evidence in three subsections. The first two subsections (Subsections 8.9.1and 8.9.2) will be brief and will only outline the evidence obtained from external participants, which was already presented in detail in Chapter 7. This is to avoid duplication between the chapters.

The third subsection (Subsection 8.9.3) will also present evidence obtained from an external participant in the form of mapping of the Firelink requirements to the 19 test subcategories.

The subsection will present this evidence in more detail because it was not part of the user-based evaluation form presented in Chapter 7.

8.9.1 Evaluation by external participants

The evaluation by external participants was mainly[113] intended to answer the EV.7 question: *Can the framework be applied by other potential users/participants?* The evaluation form was designed to obtain feedback from expert potential users of the new test framework about its likely applicability to CCLSS projects and systems other than FireControl that they have direct experience of. To prepare them to carry out the evaluation, the participants were provided with a draft paper[114] explaining the ideas of the new test framework as a way to introduce them to it, followed by meetings[115] with the author to discuss the ideas further and answer any questions. The replies were either provided by a completed form emailed electronically by the participant to the author or provided during face-to-face meetings between the participant and the author.

The questions asked to the external participants in the evaluation form which are relevant to this subsection[116]/[117] were as follows:

- ***Question 1****: Please provide a brief description of the "communications-critical large scale" IT project/system that you were involved in which you will use as basis for evaluating the ideas presented in the attached paper titled "A new test framework for communications-critical large scale systems". This project/system will be referred to as "CS2" in the next five questions.*

[113] It also included Question 5 which related to EV.6

[114] Included in CD supplement No.2

[115] 30 – 60 minutes

[116] Questions 5, 6 and 7 of the evaluation form are not included in this subsection. Question 5 was relevant to the earlier subsection relating to EV.6 and has already been mentioned there (Subsection 0). Questions 6 and 7 are relevant to the next subsection (Subsection 8.9.2) and will be included in it.

[117] To reduce duplication between chapters, please refer to Chapter 7 for the actual form used which includes an explanation for the purpose of each of questions 2, 3 and 4. Question 1 is an introductory question intended to determine the relevance of the participant's experience to the evaluation, i.e. whether the participant worked on what can be described as CCLSS.

- ***Question 2****: Please provide an outline of your involvement in the CS2 project/system and your familiarity with how the testing was conducted on CS2 before it was delivered into live service*

- ***Question 3****: Based on the "new test framework" paper you read, and according to your professional opinion, could the ideas about the new test framework have been applicable and feasible for use with the CS2 project/system as basis for designing and conducting the testing?*

- ***Question 4****: What advantages and disadvantages do you think could have resulted if the new test framework was adopted as the conceptual basis for test analysis and test design for CS2?*

Please see Chapter 7 for the replies provided by the external participants. Overall, the replies were positive about the potential applicability and usefulness of the new test framework for CCLSS projects that the participants had direct experience of. They pointed towards an acceptance of the new test framework's ideas by the selected group of expert external participants acting as potential users of such a framework.

8.9.2 Mapping of CS2 requirements by an external participant

The evaluation form mentioned earlier in the previous Subsection 8.9.1 also included the following two questions[118]:

- ***Question 6****: Please map the set of communications interface requirements provided in Appendix B to the nineteen communications subcategories in the list below.*

- ***Question 7****: Please comment on how the categorisation of the list of communications interface requirements sample according to the nineteen test subcategories (from question 5) might affect the test analysis and test design for these requirements relative to other generic test methodologies you are familiar with, e.g. the v-model?*

[118] As with the previous questions, the evaluation form also includes an explanation for the purpose of questions 6 and 7

Section 13.8 in Appendix 4 includes a sample list of communications requirements taken from the requirements of a CCLSS comparable in scale and complexity to FireControl. It is included in Appendix 4 amongst other material related to the evaluation by external participants presented in Chapter 7. The sample list of CCLSS requirements was intended for use for further evaluation purposes to check the applicability of new test framework's ideas to another CCLSS by an external participant. The mapping of the sample list of requirements to the nineteen test subcategories was carried out by one of the external participants who also provided feedback for Question 7. The sample list of requirements is included in Appendix 4 (Section 13.8). The mapping of its requirements by the external participant is presented in Chapter 7. Please refer to both Chapter 7 and Appendix 4 for the details and the data used to conduct the mapping.

For the purpose of evidence EV.7, this mapping exercise represents the application of the 19 test subcategories to a set of requirements taken from a CCLSS other than FireControl[119]. It shows that it was possible for an external participant to take a sample list of communications requirements from a second case study CCLSS and independently map them to the nineteen communications test subcategories of the new test framework. The external participant was presented with the following as preparation to carry out the mapping exercise:

- A paper explaining the ideas of the new test framework.
- A table listing the 19 test subcategories of the communications layer.
- A list of FireControl's Firelink requirements.

This exercise provided further evidence relating to EV.7, that the new test framework, and specifically the nineteen communications test subcategories, can be applied by other potential users/participants.

8.9.3 Mapping of Firelink requirements by an external participant

[119] Hence the use of the acronym "CS2" for "Case Study 2"

This additional user-based evaluation step was carried out for two purposes. Firstly, to further evaluate whether an external participant, in this case the same participant "IA" referred to in the previous section, is able to map the set of FireControl's Firelink interface requirements to the nineteen test subcategories. Secondly, as the Firelink interface requirements were already mapped to the nineteen test subcategories during the case study, this would allow the two mappings to be compared.

The original hand-written sheet of the mapping by IA is shown in Appendix 4. The data of the new (IA) mapping were then used to carry out an additional simulation similar to those presented earlier in Section 8.7[120].

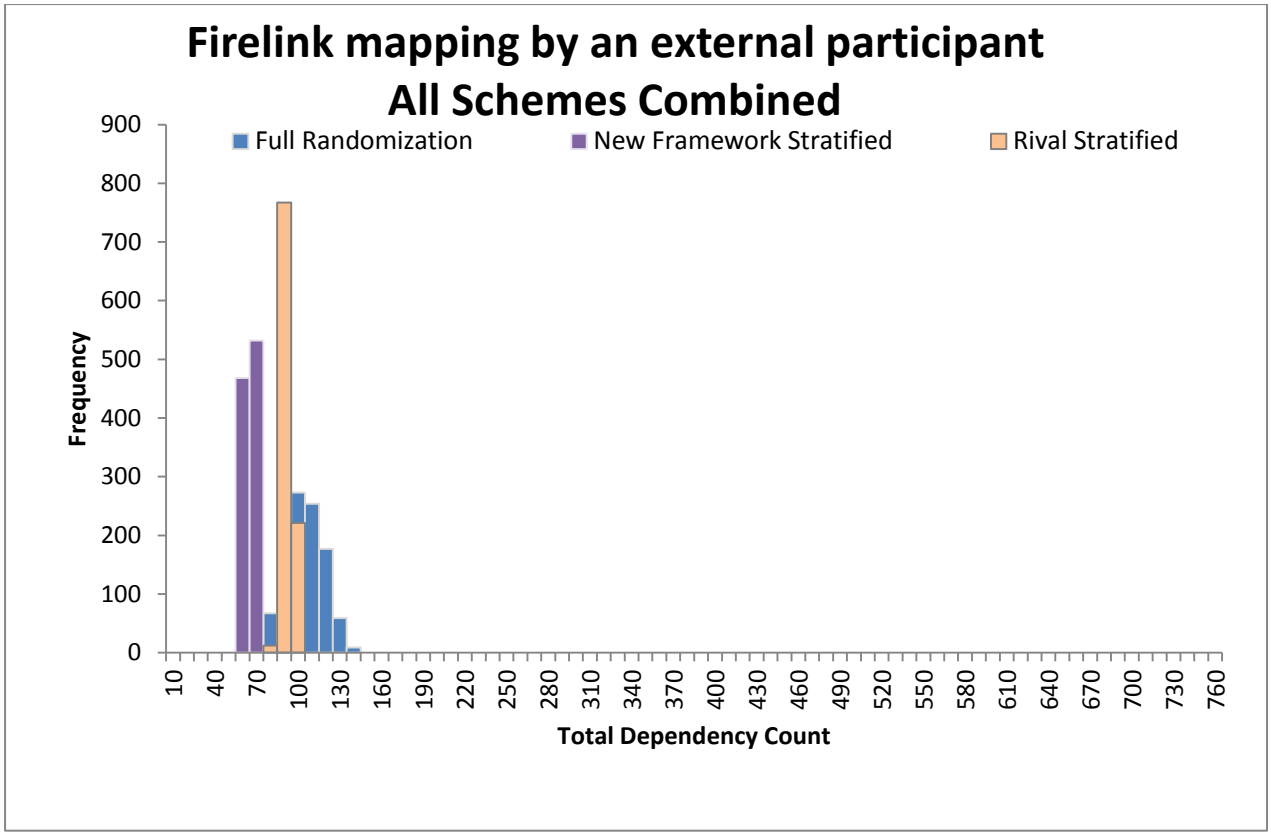
The table below shows IA's mapping and the original case study mapping.

Subcategory	IA mapping	Original case study mapping
Subcategory1: The structure/architecture of the network interface with CCLSS, its components and layout, hardware, and wiring	5.5.7.13.3.B	5.5.7.13.3.B
Subcategory2: The communications protocols used	5.5.14.3.23.B	5.5.5.1.8.A 5.5.14.3.23.B
Subcategory3: CCLSS user terminals	5.5.14.4.4	
Subcategory4: The data and messages that are transmitted by CCLSS over that network	5.3.4.3.D 5.5.5.1.2 5.5.5.1.2.A 5.5.5.1.3 5.5.5.1.5 5.5.5.1.6 5.5.5.1.7 5.5.6.15.5 5.5.6.57.G	5.5.14.4.4 5.5.5.1.6 5.5.6.57.G 5.5.5.1.2.A
Subcategory5: All possible types of CCLSS senders and receivers		
Subcategory6: The different possible modes of transmission used	5.5.5.1.8.A	5.5.5.1.1 5.5.5.1.4
Subcategory7: How the transmissions are acknowledged by the receivers		5.5.6.15.5
Subcategory8: How the performance characteristics of the		

[120] The V-Model mapping remained the same as was carried out by the author

network can affect CCLSS subsystems and processes		
Subcategory9: The services provided by the network to CCLSS	5.3.11.4.B 5.5.5.1.8	5.3.11.4.B 5.5.7.30.1
Subcategory10: Performance and volume limits of the services provided by the network	5.5.7.30.1	
Subcategory11: How CCLSS QoS requirements and SLAs are guaranteed, maintained and reported		5.5.5.1.8 5.3.4.3.D
Subcategory12: The ongoing operation, maintenance and administration of the network/its CCLSS interface		5.5.5.1.2
Subcategory13: Fault handling processes of the CCLSS interface, from detection to resolution		
Subcategory14: CCLSS certification/compliance requirements	5.3.11.3 5.3.11.4 5.3.19.5 5.5.3.1.7 5.5.7.13.3	5.5.5.1.3 5.5.5.1.5 5.5.7.13.3 5.3.11.3 5.3.11.4 5.5.3.1.7 5.3.19.5
Subcategory15: Resilience features (of the interface)	5.5.5.1.1 5.5.5.1.4 5.5.5.1.4.A 5.5.7.36.1	5.5.5.1.4.A 5.5.5.1.7 5.5.7.36.1
Subcategory16: Business continuity features (of the interface)		
Subcategory17: Documentation provided for CCLSS, user and technical documentation	5.3.7.2 5.3.18.1 5.3.18.6	5.3.7.2 5.3.18.1 5.3.18.6
Subcategory18: Risks		
Subcategory19: Operational readiness of the interface for CCLSS go-live		

The results of simulation of IA's mapping are shown below in distribution graph form and well as associated ANOVA data.



	New Framework	Rival Model	Fully Random
Is the data normal?	Yes	Yes	Yes
Are the variations equal?	Yes	Yes	Yes
N	1000	1000	1000
Mean	60.67	87.91	100.57
St Dev	2.33	3.49	13.42
Var	5.42	12.18	180.21
3-way comparison	There is significant difference		
	S_pooled	Effect Size	Comparison
New Framework vs Rival Model	2.97	-9.19	Different
New Framework vs Fully Random	9.63	-4.14	Different
Rival Model vs Fully Random	9.80	-1.29	Different

According to the simulation results above, the new framework remained more efficient than the V-Model ordering and the randomly generated sets even when the mapping of the new

test framework was carried out by an external participant. The ANOVA data also shows the following results for the new framework: Effect size of -9.19, Mean value of 60.67, Standard Deviation of 2.33 and Variance of 5.42. As shown in Subsection 8.7.2, the equivalent ANOVA data for the same simulation using new framework mapping data as per the case study are: Effect size of -11, Mean value of 52.99, Standard Deviation of 2.83 and Variance of 7.98.

Considering the range of ANOVA results for the simulations for the five interfaces as shown in Section 8.7 and summarised in Subsection 8.7.8 (e.g. effect size range from -11 to -4.74) provides a positive indication about the relative proximity of the results[121] of the mapping by the IA to those of the original mapping of the case study.

The fact that an external participant was able to map the Firelink requirements independently, and the results of the simulation of the mapping data were relatively close to the results of the author's own mapping, by itself represents supportive evidence relating to EV.7. However, to provide further insight into the differences between the mappings, a closer examination of the differences one-by-one is presented below.

The list of the requirements where there is a variation in the mapping is as follows: 5.3.4.3.D, 5.5.14.4.4, 5.5.5.1.1, 5.5.5.1.2, 5.5.5.1.3, 5.5.5.1.4, 5.5.5.1.5, 5.5.5.1.7, 5.5.5.1.8, 5.5.5.1.8.A, 5.5.6.15.5, 5.5.7.30.1.

Looking closely at the text of each of these requirements (Appendix 3, Section 12.1), there seem to be two instances where the difference between IA's mapping and the original case study mapping is quite noticeable and seem to be an "incorrect" application of the framework. These are 5.5.6.15.5 and 5.5.5.1.8. Requirement 5.5.6.15.5 is classed under Subcategory 4 by IA presumably because it is talking about data messages. However, this requirement is about how the acknowledgement of MRMS messages should function, therefore Subcategory 7 is more appropriate. Requirement 5.5.5.1.8 is classed as Subcategory 9 by IA, whereas

[121] As was stated earlier in this chapter, the simulation results are appropriate for analytical purposes rather than for use as data for statistical generalisation (Yin, 2009, p. 38).

Subcategory 11 is more appropriate because the requirement is related to the production of a routing transmission report for the data messages transmitted over Firelink.

Other differences seem to be less pronounced and less suitable for classification as correct vs. incorrect use of the framework, but are more due to the interpretation and style of the individual analyst. For example, five more of the differences were classified by IA under Subcategory 4 seemingly because they were data messages related. These are[122]: 5.3.4.3.D, 5.5.5.1.2, 5.5.5.1.3, 5.5.5.1.5 and 5.5.5.1.7. However, each requirement was related to how messages are managed by the network, e.g. their priority in 5.3.4.3.D, routing rules in 5.5.5.1.2, transmission standard in 5.5.5.1.3/5, transmission interruptions in 5.5.5.1.7. Therefore, during the case study analysis, it was deemed that they were closer to other subcategories. However, can their mapping by IA to Subcategory 4 be classed as "incorrect"? Probably not with certainty because their allocations seem to be valid interpretations of the meaning of Subcategory 4 in the absence of any further documented rules or training. One possible pattern that can be observed by the six differences in IA's mapping specifically for Subcategory 4 is that they seemed to be decided based more on the literal text of the requirements, whereas the original case study mapping involved more analysis of the meaning of each requirement and how it related to the other requirements.

The above observation regarding a possible pattern of differences between IA's mapping and the mapping of the case study remains valid for other differences. For example, 5.5.5.1.8.A was originally mapped during the case study to Subcategory 2, and was mapped by IA to Subcategory 6. This requirement relates to the Communications Gateway being able to support full-duplex data transmission. During the case study data analysis, this requirement was considered a feature of the communications protocol used. IA seemed to view it as a part of the functionality of the Communications Gateway. IA's interpretation is more direct and literal, whereas the case study mapping is based on the analysis of the meaning of the requirements.

[122] as well as 5.5.6.15.5 mentioned earlier

Furthermore, 5.5.14.4.4 was mapped during the case study analysis to Subcategory 4 and placed by IA in Subcategory 3. This requirement relates to the ability of the MDT to display GIS data. IA treated it as a feature of the MDT (a user terminal) whereas the requirement was mapped during the case study analysis to Subcategory 4 for data and messages transmitted by CCLSS over the network/interface. Once again, IA seemed to have used a more literal approach here whereas the case study categorisation involved more analysis beyond the literal wording of the requirement.

As for the remaining three mapping differences for 5.5.7.30.1, 5.5.5.1.1 and 5.5.5.1.4, no such pattern for the variations can be observed. The differences seem to simply be due to valid variations between IA's interpretations for these requirements and the case study analysis. For example, 5.5.7.30.1 is related to the conferencing features of the ICCS, but it also refers to a minimum and a maximum number of conferencing parties. This was classed as Subcategory 9 whereas IA classed it as Subcategory 10 as a performance requirement. 5.5.5.1.1 is a difference only due to simulation assumptions where duplicates were removed from the latter subcategory for simulation purposes, otherwise it was deemed during case study analysis to equally belong to Subcategory 6 as well as Subcategory 9. IA mapped it to Subcategory 9. Lastly, 5.5.5.1.4 relates to the ability of the MDT to route messages via a number of bearers. This was mapped during the case study analysis to Subcategory 6 regarding different modes of transmission, whereas IA mapped it to Subcategory 15 presumably because the routing functionality of the MDT is related to the resilience of its communications.

The discussion so far in this subsection focused on the differences between IA's mapping and the mapping of the original case study, as well as the possible reasons behind these differences. However, their consequences on the simulation results are relatively limited as was discussed earlier. A further look is needed to consider why the mapping differences lead to relatively similar simulation outcomes.

The differences between IA's mapping and the case study mapping consisted of the following two types of differences:

- Four requirements mapped by IA to "forward" (i.e. later) subcategories in the subcategories list. One of them, 5.5.5.1.1, was a difference only because duplication had to be avoided for simulation purposes; otherwise the case study analysis also placed this requirement in Subcategory 15 which agrees with IA's mapping of this requirement.

- Eight requirements mapped by IA to "earlier" subcategories in the subcategories list, six of them were mapped to Subcategory 4.

A review of the dependency sets in Appendix 3 Section 12.2 for the requirements in the two groups above was carried out to better understand the impact of each difference. This resulted in a number of observations. Of the four requirements mapped "forward" by IA, only 5.5.5.1.8.A was likely to have had a tangible impact on the simulation results. This is because it had a relatively large dependency set of twenty one requirements and it would have been placed by IA after some requirements that were in its dependency set. Requirements 5.5.5.1.1 and 5.5.5.1.4 each had a dependency set of two items. Their dependency sets consisted of each other and 5.5.6.15.5, which in turn was placed by IA to Subcategory 4. Therefore, the impact of these two differences on the simulation results would have been limited only to placing 5.5.6.15.5 in a subcategory ahead of them. The fourth requirement placed in a later subcategory was 5.5.7.30.1 which has no requirements in its dependency set.

The remaining differences mapped to "earlier" subcategories by IA were primarily for requirements that had no or few dependencies (up to three). Only two differences, 5.5.5.1.3 and 5.5.5.1.5, had a relatively large number of dependencies of nine. However, the new mapping retained three of their dependencies (5.3.19.5, 5.3.7.2 and 5.3.18.6) after them and retained both of them within the same subcategory. Furthermore, they were placed by IA ahead of three of their dependencies: 5.5.5.3.1.7, 5.3.11.3, 5.3.11.4, rather than within the same subcategory as was done during the case study analysis. This represents an advantage in terms of the simulation results. This advantage is offset by moving two of their dependencies, 5.5.5.1.1 and 5.5.1.4, after them which represents a disadvantage for

simulation purposes.

Therefore, it seems that the most tangible impact on the difference between the simulation results for IA's mapping and the original case study mapping was probably due to 5.5.5.1.8.A being placed by IA in Subcategory 6. The other differences seem to have had no or limited effect or counter-balanced each other. Additionally, there were agreements between IA's mapping and the case study mapping regarding six "empty" subcategories. This must have been a limiting factor to the simulation results being more tangibly different because it restricted the consequences of the differences on the dependency calculations by limiting the differences to a smaller subset of the nineteen subcategories. It also means that both mappings have a similar potential for identifying gaps in the requirements.

The factors discussed above, when combined, provide an explanation for why the difference between the simulation results of IA's mapping and the mapping of the case study were relatively close. This was an unpredictable outcome of the mapping exercise carried out by IA, which leads or confirms a number of observations and ideas presented below:

- Like any methodology or framework in IT, for the new test framework to be successfully adopted in industry, its needs to be supported by training and tools
- Familiarity and length of involvement with the requirements and their underlying purpose may have been a factor behind some of the differences
- Precision and clarity in the way the requirements are expressed is an important factor for their test effectiveness

Chapter 7 Section 7.5 presented another mapping exercise to the test subcategories carried out by the same participant, IA. It involves a new sample set of communications requirements from another CCLSS project; the project is different from but comparable to FireControl. This was carried out as part of the evaluation by external participants presented in Chapter 7. It allows comparison of the two mappings carried out by the same external participant.

Referring back to the original purpose of this Subsection (8.9.3) and the purpose of Section 8.9 in general, which is to present material relevant to evidence EV.7, the material presented in this section demonstrated that an external participant was able to map the test subcategories of the communications layer to the Firelink interface's requirements set[123].

8.10 Chapter Summary

There seems to be no established precedence in the IT profession for new conceptual frameworks to be adopted only following the availability of conclusive evidence of their benefits (Bertolino, 2007, p. 8) (Juristo, Moreno, & Strigel, 2006). Due to this and to the awareness that no single conclusive evidence item may be feasible due to the nature of research presented in this thesis, a variety of types of evidence from a variety of sources was sought. This approach was intended to provide a varied set of evidence items which involve external participants as much as feasible that, when combined, can be considered to be reliable and appropriate evidence.

The table below lists the evidence items presented and discussed within this chapter and the section that discussed each of them.

Evidence ID	The case study question	Section
EV.1	Can the new test framework be applied?	8.3
EV.2	Are there benefits from the framework?	8.4
EV.3	Can the benefits be generalised?	8.5
EV.4	Can the benefits be estimated numerically?	8.6
EV.5	Can the framework be compared to a rival?	8.7
EV.6	Can the framework fulfil the initial intended criteria?	8.8
EV.7	Can the framework be applied by other potential users/participants?	8.9

[123] The material presented already in Section 7.5 also demonstrated that the same participant was also able to apply the test subcategories to another sample of requirements for a second CCLSS project.

9. Chapter 9: Summary, conclusions, further work

This thesis started by explaining, in Chapters 1 and 2, that there is a need for domain-specific test framework for communications-critical large scale systems. It explained that, in practice, a general methodology such as the V-Model or other test standards derived from it leave key decisions about what should be tested to the judgement and experience of the individual tester. This leads to unpredictable and imprecise testing outcomes, an effect that is magnified when the system under test is a communications-critical large scale system. To achieve more effective testing of such a class of system, the testing activities such as design, specification and prioritisation need to be made more objective, more precise and be carried out earlier in an IT project's lifecycle. The ideas that the V-Model is out-dated or that domain-specific test frameworks are needed may not necessarily be new or unique to this thesis (Bertolino, 2007) (Scully, 1998) (Smith & Thompson, 2008). What this thesis does is that it provides further insight into why and how a generic test approach is out-dated and it also explains why a domain-specific approach is likely to be more appropriate.

The thesis then moves into explaining how this identified "gap" in the theory and practice of testing can be bridged for a specific type of systems, namely the type/domain termed in this thesis as the "communications-critical large scale systems" (CCLSS). The intention from this thesis was to create more than a high-level and theoretical framework but also to come up with ideas that are clearly defined on one hand but also adaptable on the other hand and ready to start being applied with real-life systems. Therefore, the ideas of the layered test framework that were presented at a high level in Chapter 4, and then detailed further for the communications layer in Chapter 5, were then applied to a real-life CCLSS project (FireControl). This real-life project case study was then used to discuss and evaluate the benefits that could be observed following the adoption of the new test framework.

In summary, the thesis started by explaining the need for a CCLSS domain-specific test approach, it then explained how such a domain-specific approach can be evolved and provided a detailed example, it then applied this example to real-life CCLSS project then discussed and evaluated the benefits.

The advantages that the proposed domain-specific approach offer emanate from it being a conceptual representation of a domain-expert's knowledge of the system's uses and structure, rather than being a representation of the (expected) phases of development cycle of the system being tested. This is what makes the framework more appropriate for adoption from the early stages of an IT project before the technical details of a system have been defined, as was demonstrated by the real-life CCLSS project case study; it also reduces the reliance on the individual tester's knowledge and experience and allows for more objective, predictable and early definition and prioritisation of the assurance and testing activities. Managing complexity through simplicity seemed in this case to make the complexity more manageable.

9.1 Contributions of this thesis

The contributions of this thesis to the collective knowledge of software testing are as follows:

Main contributions:

1- The new domain-specific test framework for Communications-Critical Large Scale Systems (CCLSS) as discussed in Chapter 4 and diagrammatically represented by the layered model in Section 4.4. This provides a new way of approaching the testing of CCLSS that was detailed, applied[124] and evaluated within this thesis and shown to be capable of delivering tangible efficiency benefits to the testing process of a real-life CCLSS.

2- The Communications layer specific test approach and its nineteen subcategories as discussed in Chapter 5. This provided a detailed real-life example of how the new test framework can be instantiated and applied to a real-life CCLSS.

3- Devising a new method used in the simulation based analysis in Chapter 8 as a simple but meaningful way to estimate "effectiveness" of the new test framework which can

[124] Partially, with the communications layer's nineteen test subcategories

also be used to identify whether a set of requirements is more or less complex from a test point of view. This method was extended in Section 8.7 to compare the efficiency of two test approaches. The author is not aware of a similar use of random simulations to evaluate a conceptual framework or to compare it to a rival framework, such as was done in this thesis in Sections 8.7 and 8.6.

4- Adapting the idea of "protocol layers" from telecommunications (Bochmann, Rayner, & West, 2010) and the idea of "value chain" (Value Chain Group, 2007) from business management and applying them to software testing as presented in Chapter 4.

Additional contributions:

5- As outlined in section 6.4.3 in notes 18.3 and 19.1, the use of the nineteen test subcategories provides objective basis for identifying gaps, inconsistencies and "test risks" in the requirements as well as provide a useful way of assessing test coverage for the communications interfaces of a CCLSS.

6- Presenting a relatively simple new approach to linking requirements to testing generally and to test cases (Barmi, Ebrahimi, & Feldt, 2011) using the nineteen communications test subcategories.

7- Combining engineering evaluation ideas from (Verschuren & Hartog, 2005) with software engineering research and combining some of the paper's ideas with case study methodology as a way to improve reliability of the case study work.

8- The evaluation work in Chapters 7 and 8, and this thesis in general, can be thought of as an example of how software engineering can *"utilize the strengths of case study research"* (Runeson, Höst, Rainer, & Regnell, 2012, p. 30) to devise new efficient approaches to testing CCLSS.

9.2 Thoughts arising from the use of simulation-based analysis for evaluation of test efficiency

The use of simulations of randomly generated orderings of the requirements sets, as presented in section 8.6 and extended further in section 8.7 to include comparison between the new test framework and the V-Model, was devised by the author for the purpose of evaluating the new test framework. With the thesis being about designing a conceptual framework for test analysis and design, it initially seemed difficult to come up with an evaluation method that can produce objective and tangible evidence. It was possible to offer considered discussions, analysis and explanations of the qualitative input and output data of the case study (i.e. requirements and outline test cases), but could that be sufficient as rigorous and objective academic evidence? The answer depended on whether the research methodology should be oriented towards experimental sciences or other research fields such as social sciences or business management. The subsequent decision was to include more numeric and statistical data in the evaluation of the benefits of the new test framework and to make them (the benefits) measurable. This is why the simulation-based analysis work was carried out.

Furthermore, the idea of analysing the interdependencies between requirements and turning the analysis into numeric "test efficiency" data could be viewed as a link between research in the testing of large scale systems and the research into deterministic systems testing using mathematical and statistical methods. Whereas the literature about testing small deterministic systems often and typically features precise numeric and statistical methods, the author is not aware of a similar use of random simulations or numeric data being used to evaluate a conceptual framework and compare it to a rival framework such as was done in this thesis in Sections 8.6 and 8.7.

9.3 Findings and implications of the user-based evaluation

The user-evaluation participants had varied backgrounds in IT: a technical architect, a project manager, a business analyst and a communications engineer. The CCLSS real-life examples

they cited were also varied by purpose, organisation and timeline. Despite the variety, the four participants were all able to relate the ideas of the new test framework to their individual experiences working with CCLSS, and to largely agree that the ideas were viable and useful.

The evaluation comments the participants provided reflected the variety of their experiences and backgrounds, IA's comments indicated he expected the framework to be feasible for the CCLSS he cited, and was more interested in whether it delivers tangible benefits in terms of time, cost and quality. IM also thought that the framework was probably feasible for his CCLSS example and identified the subcategories that he could link to his experience of working on a CCLSS. RJ thought the new framework could have helped provide a unified test framework that was otherwise missing and could benefit from using existing requirements management guidelines and standards. KK's response identified a need for a fully developed framework that covers all non-functional aspects of an IT project such as security and service management.

The comments from the four participants were supportive of the framework and reflected confidence in its viability. Despite their variety, the participants' comments seem to point to the need to develop the new framework further, which was already known before the user-based evaluation because the only the communications layer was detailed as part of the case study.

However, a new idea that emerges from the user-based evaluation is that there seems to be a desire amongst the participants that the new test framework needs to be cross-disciple and to help with closer inter-working on CCLSS projects between the testing function and the business analysts, technical architects and specialist communications engineers. Therefore, the participants' feedback can probably be used to formulate new ideas about *how* the new test framework can be developed further and in what direction, e.g. by evolving it to be linked to IT project management methodologies, business analysis guidelines and associating it with a defined set of engineering-like processes and efficiency metrics. In other words, rather than it being a standalone new test framework for CCLSS, it might be more beneficial to make it a part of overall set of software engineering guidelines and practices aimed at

CCLSS that define an overarching software engineering framework specifically intended for CCLSS.

For instance, further work can consider whether and how closer integration between the testing function and other software engineering functions could potentially lead to benefits and efficiencies for CCLSS projects. There is already a precedent to this idea in software engineering: closer integration of the various development functions and experts is already done within the scrums of the Agile development methods for smaller web-based systems. Therefore, an extension of this research could consider adapting Agile methodology ideas for use on CCLSS projects where already-developed systems (COTS), subsystems and technology layers (infrastructure, communications and data) are integrated to form a CCLSS where the scale may make the original Agile methodology less appropriate.

Furthermore, the mapping exercise by one of the external participants for both the Firelink requirements and the CS2 requirements presented an opportunity to compare the mappings done by an external participant and by the author, and to consider the level of their agreement when independently mapping requirements to the nineteen subcategories.

One statistical measure appropriate for formally expressing such level of "inter-rater" agreement is called the "Kappa statistic" (Carletta, 1996) which is used as a measure of agreement between two or more classifiers of the same set of data or items. There are different types of Kappa measures aimed at different purposes, the two identified as most appropriate for the purposes of the mapping of the Firelink and CS2 requirements to the nineteen test subcategories are: one is the "Cohen's Kappa" (Cohen, 1968) coefficient which is specific to measuring the agreement level between two raters (i.e. classifiers) only, and "Fleiss' Kappa" (Fleiss, 1981) coefficient which can measure the agreement between two or more raters as well as taking into account the probability of un-intentional or chance agreement. For the purposes of this thesis, there are only two raters, the external participant and the author, therefore both Cohen's Kappa and Fleiss' Kappa can be used. According to both measures, the level of agreement between the external participant and the author is represented using a coefficient from -1 to 1 for each of the two sets of mappings: Firelink and CS2.

Below are the results of the Kappa calculations presented in Appendix 4 (Section 13.9).

For Firelink requirements mapping, the agreement level between the external participant and the author are as follows (rounded):

- Cohen's kappa: 0.50, Fleiss' kappa: 0.57 when agreements regarding empty subcategories are not included
- Cohen's kappa: 0.60, Fleiss' kappa: 0.65 when agreements regarding empty subcategories are included

For CS2 requirements mapping, the agreement level between the participant and the author are as follows (rounded):

- Cohen's kappa: 0.91, Fleiss' kappa: 0.92 when agreements regarding empty subcategories are not included
- Cohen's kappa: 0.93, Fleiss' kappa: 0.94 when agreements regarding empty subcategories are included

Kappa figures are not meant to have an absolute meaning although there is one paper (Landis & Koch, 1977) which proposes the following ranges[125]:

Kappa Statistic	Strength of Agreement
< 0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate

[125] It is worth noting that these ranges are referred to in the paper (p.165) as “arbitrary” and as “benchmarks”, therefore they need to be adapted to the context and the source of the data. However, they appear to be appropriate for the purpose of interpreting the levels of agreement between the external participant and the author regarding the mapping of Firelink and CS2 requirements to the nineteen test subcategories.

0.61-0.80	Substantial
0.81-1.00	Almost Perfect

It is therefore clear that the level of agreement between the external participant and the author in the mapping of CS2 requirements was far higher than the agreement over the more complex and more fragmented Firelink requirements. Further to the analysis presented in Sections 7.5 and 8.9, the above kappa figures confirm formally the effect of better structuring and simplification of requirements, as was the case with CS2 requirements, on the precision of the testing.

Using Kappa measures to assess levels of agreement between users of the framework could be used as a formal approach to assess the precision benefit the test framework provides to the test analysis and design work for a CCLSS. This idea can be extended and explored with further work researching how the precision of the new test framework can be improved, e.g. through better description of the subcategories, amended subcategories, closer integration with business analysis and requirements capture standards, or simply better training and test tools support for the framework.

9.4 Reflections on the use of case study methodology for this thesis

The research methodology for this thesis is explained in the Research Methodology chapter (Chapter 3), further discussions related to research methodology are also included in the early sections of Chapter 8, particularly Section 8.1. This section is intended to evaluate and include further reflections related to the use of a case study as a vehicle for the research presented in this thesis.

A useful term sometimes used in engineering disciplines[126] is the term “trade-off”, used to describe when competing engineering or desirable design objectives need to be traded-off to

[126] e.g. software engineering (SEI, 2013) and communications systems design (Zheng & Tse, 2003)

arrive at an optimal solution or result. This term seems quite relevant when reflecting on the use of case study methodology for this thesis. This is because there were competing needs and objectives that had to be balanced and "traded-off".

For example, there was a need for academic rigour in the methodology adopted by this thesis, but on the other hand this thesis needed to be relevant to real-life industrial practices in testing. The ideas in thesis are intended to be applicable and relevant for generalised use in the specified domain (CCLSS) and it needed to be applied to real-life communications-critical large scale systems, however, how many such systems can be available for inclusion in one thesis which is time as well as effort constrained. Will it produce academically more reliable outcome to involve less data and smaller studies from multiple sources, or more data and more in-depth study from one source? What is more appropriate, a qualitative or a quantitative evaluation approach?

The above were considerations that were encountered during the work on this thesis. Such considerations need decisions and may not all be foreseen at the start of the research, meaning it may not always be possible to plan for them in advance. Planning in advance indicates that the course the research will take is determined from the outset, but exploratory and flexible research that does not fit in this category needs to be catered for.

Overall and based on the experience of this research effort, case study research seemed appropriate when the above mentioned trade-offs are needed. Firstly, it allows the researcher flexibility. It also is well-suited for researching real-life situations, especially for understanding and evaluating a "big picture" rather than, for example, statistically analysing data. A case study research could be helpful to the researcher when there is a need to combine qualitative and quantitative methods yet still adapt them to real-life use.

However, flexibility can also bring a risk of drifting away from a planned or intended route for the research. Real-life situations can also bring with a reduced quality of data and subsequent analysis, as well as the researcher potentially losing control of the data collection effort, it can also lead to blurring of the interpretations, analysis and the facts.

Furthermore, specifically for the use of case study methodology for software engineering research, most case study methodology literature seems to be related to and written for social sciences research. Therefore, the software engineering researcher who wishes to specialise in case study research may have to adapt case study methodology ideas from other fields before they can be applied to software engineering[127].

Overall and based on the experience gained while preparing this thesis, case study methodology brings significant potential benefits to software engineering research because of the way it can help link it and make it more relevant to real-life practices in software engineering. It offers a variety of research "tools" that can be adapted to fit specific research situations and objectives. However, the flexibility it offers the researcher come with risks that need to be mitigated through good practice, such as maintaining a defined case study design (even if flexible), constantly maintaining and reviewing the chain of evidence to ensure that the data collected, the analysis and conclusions are traceable back to the research questions. Case study research may at times seem an "easy" route to conducting software engineering research, but that must be confused with it being "easy to do well". Conducting good case research places more onus on the researcher to be competent in planning, analysis, reporting, and not least, able to identify then resolve potentially difficult trade-offs.

9.5 General guidelines for applying the new test framework to other IT domains

To ensure that the required effort and timescales are proportionate to a PhD thesis, the scope of this work was focused on defining a new test framework for a specific IT domain (CCLSS) then applying it through a real-life case study. However, the ideas presented in this thesis are expected to have a potential of being adapted for other IT domains and system types. Although this is further work outside the scope of this thesis, below is a list of ideas and guidelines on how it might be progressed.

[127] A recent book on case study methodology (Runeson, Höst, Rainer, & Regnell, 2012) in software engineering is an exception and seems to have collected case study experiences of a few software engineering researchers and presented them in an easily usable style for software engineering researchers

1. Create a conceptual domain-expert's view of the intended type of system, from a testing viewpoint, starting from the hardware physical infrastructure then ending by the business processes, as per the model described in Chapter 4. The high-level framework proposed by this thesis may well remain applicable to other types of systems whereas the details of the individual layers may vary according to the domain. However, there is likely to be a need to start over from the beginning for domains that use distinctive technologies such as virtualisation or interconnected distributed database systems.

2. For each of the layers, determine the testable features of that layer, independently from what the requirements or the design of a specific system of that domain might state, then organise these features into test subcategories starting from the basic and simple to the complex and advanced.

3. For applying the framework to a specific system: categorise that system's requirements according to the layers in the derived framework. During the requirements capture phase for that system, ensure that inter-layer and inter-subcategory dependencies are minimised when the requirements are defined.

4. Consider including further subcategories reflecting the non-functional attributes to be tested, as was discussed towards the end of Chapter 6 from section 6.9 onwards.

5. When defining the details of how the system will be verified, decide whether each requirement should be verified by review, demonstration, inspection and/or test and at which relative stage of the project. The aim of effective testing is to assure and test the more complex features early enough to allow for relatively easy remedial action is needed. This entails having ready from an early stage of an IT project, a representative test environment, the necessary test tools and test data needed to carry out complex tests.

6. Treat the prioritisation derived via the framework as a starting point, review the resulting schedule of tests to evaluate if it can be optimised further based on knowledge of the specific system's architecture and its development process.

7. Raise test risks where a layer's subcategories are not sufficiently defined by the requirements, to ensure that they are sufficiently addressed by the detailed design and would not be overlooked.

9.6 What is new in this framework?

The new test framework and the example of the communications layer detailed further in this thesis are both centred on the idea that the test analysis and design work should be organised according to a conceptual domain-specific testing view of the system under test. As explained in section 5.2.1, this is conceptually comparable to abstract models and frameworks long used in telecommunications such as the OSI layers (Zimmermann, 1980), TMN model (ITU-T, 2000), and more recently eTOM/FAB (Kelly, 2003). Each of these models or frameworks provides a unified view of aspects of telecommunications systems or services that is applicable to different types of networks and technologies and they all help (or helped) making telecommunications systems more precisely standardised than if they did not exist. The literature survey carried out for this thesis (Chapter 2) did not encounter such a framework for testing. Probably the closest comparable example found was for the Zachman Enterprise Architecture framework (Zachmann, 1987) (Zachman International, 2012) which does not include a testing view amongst the views of Enterprise Architecture that it defines.

According to the framework, and as was demonstrated in Chapter 6, the *expected or anticipated* contents of the requirements are represented by a set of test categories and subcategories that could be applicable to any CCLSS. The subcategories of the CCLSS under test would then be organised according to their complexity and priority, which is intended to help in the scheduling of the test activity as well as achieving more precise and objective definition of the test cases. The use of framework during the test analysis can also have the benefits of highlighting more objectively where the requirements have inter-dependencies, or where there are potential gaps in the requirements.

Adopting this approach instead of a generic approach such as the V-Model allows the test analysis activity to evolve from being somewhat subjective, general and a recipient of requirements to a more objective, domain-specific pro-active way of verifying the correct delivery of the requirements, and defining an affective set of tests that is more easily synchronised with other project activities.

Finally, whilst the proposed framework is meant to be clearly defined to aid objective analysis, it also allows for flexibility and can be adapted and expanded as was discussed in Section 6.9. In fact, none of the statements or the discussions made in this thesis is meant to rule out the use of any of the ideas of the V-Model or other generic methods/standards in conjunction with this framework where and if appropriate, e.g. for developing the tests for one of the framework's layers or subcategories.

9.7 How the test framework approach can be developed for other layers

Although outside the scope of this thesis, below is a list of (non-exhaustive) guidelines for how test approaches for the other five layers of the framework[128] may be detailed as part of potential further work:

1. Domain-specific expertise will be needed. The subcategories should capture a subject matter expert's view of what the testable features are for that particular layer, e.g. infrastructure, data, high-level or detailed functionality and business process.

2. The test subcategories should not be derived specifically to map to a particular system's pre-existing requirements or specifications but they should form an expert's view of what the requirements and specifications can be expected to contain.

[128] infrastructure, data, high-level functionality, detailed functionality and business processes

3. The initial few subcategories should represent the most fundamental features of the intended layer, i.e. what it is and how it is structured, how its valid and invalid use will be decided, what its inputs are and what its outputs are, who the users are and what modes of use should it support.

4. The later subcategories should represent the most complex aspect of the layer that need to be tested last, such as resilience, business continuity, risks and operational readiness. These are the aspects which, when proven by testing, can provide confidence that the other preceding subcategories have also been fulfilled. "Test risks" may be better organised as one subcategory for clarity, but in practice they are likely to span more than one subcategory.

5. Less critical but important aspects such as documentation, certification and standards compliance need to be included in the final or last few subcategories.

9.8 Additional thoughts and ideas arising during the preparation of the thesis

This section outlines additional thoughts and ideas that developed whilst working on this thesis but whose detail is outside the scope of this thesis. They could represent potential further work.

9.8.1 CCLSS projects early agreement being mapped onto the test framework

Incorporating a domain-specific test framework such as the one proposed by this thesis into a CCLSS project's requirements and commercial agreement, e.g. with vendors and sub-contractors, could reduce the ambiguity later on during the project for how the systems should be tested. The adoption of such a framework can be expected to offer significant

benefits because it will be domain-specific to the project's systems and technologies and less open to subjective interpretation.

9.8.2 Organisation of the requirements according to the test framework

Maximum benefits of the new test framework generally and the communications test approach specifically can be realised when the requirements of a CCLSS are initially developed with this test approach in mind and according to the proposed test layers described in Chapter 4, and where interlayer dependencies are kept to a minimum.

When the ideas of this thesis were initially applied to FireControl's requirements as discussed in Chapter 6, there was no distinct group of requirements representing a distinct communications layer. Instead, searches were used to identify, group then categorise all communications related requirements from amongst 2000+ requirements. This work required considerable analysis effort and time. If another comparable CCLSS project adopted the ideas of this test framework from the outset and organised the requirements according to the proposed test layers, then the test design effort should be more efficient, faster and more precise that was observed in the case study with FireControl.

9.8.3 Interdependencies between the test framework's layers

Each layer of a system organised according to the test framework needs to be defined as standalone as can be feasible, and its interdependencies with other layers need to be made easily identifiable and grouped together. Each layer should ideally be interdependent with adjoining layers only. Such ideas are common and well established in communications protocols. If such ideas are adapted to requirements management and systems engineering practices for CCLSS they can be expected to offer benefits of better project planning, better release management, better test scheduling and prioritisation.

9.8.4 The format of a requirement to facilitate more effective testing

Attaching detailed test ideas to individual requirements, or subcategories of requirements, during the requirements capture phase of a project should reduce ambiguity and subjectivity later on during an IT project, and minimise the time and effort needed for test design work.

Examples of what is meant by detailed test ideas:

- How the requirement should be tested
- The requirement's purpose
- Whether the requirement describes whole system activity/feature or is only a part of an overall feature.
- What the system should be able to do or not do according to this requirement.
- What non-functional attributes an individual requirement relates to, even if implicitly.
- Interdependencies between this requirement and other requirements within the same system layer, other layers or subcategories.

The above ideas can be viewed as akin to aspects of Test Driven Development (TDD) (Maximilien & Williams, 2003) where the test information and preparations start being produced from the outset of a development cycle (in TDD/Agile terms, this is called a "sprint"). In this case, the concept of early test preparations is applied at the requirements level instead of at code level of a system.

9.8.5 A way of bringing forward the assurance and test effort for a CCLSS project

The FireControl case study discussed in Chapter 6 demonstrated an example of how meaningful preparations for test activities can be started early during a project before technical design and specifications are available. In other word, the testers do not have to wait until the technical design and specification is made available before they start doing productive work identifying and resolving issues and risks for the system being developed.

Because the overall test framework (Chapter 4) and the subsequent communications test approach (Chapter 5) are based on domain expertise, it was possible to produce useable and tangible "test ware" (i.e. test design, high level prioritised test cases) that was requirements-based and independent of the vendor's technical design.

Providing a foundation to start conducting test activities early in the project's lifecycle is an additional benefit of the framework if the way it is applied can be optimised to ensure that the complex more fundamental features of the CCLSS are verified and tested early. The early testing of the more complex features means reduced risk to the IT project of encountering complex problems at a later stage such as final acceptance testing.

9.8.6 Potential benefit to a CCLSS release management

Organising CCLSS requirements into distinct layers and grouping their interdependent features into distinct layers and subcategories, as proposed by the new test framework, could make organising system releases easier if each release corresponded, for example, to a standalone layer or a number of adjoining subcategories. Being able to achieve such organisation of a system's delivery could mean the features of a release are less likely to be negatively affected or disabled due to dependencies on other features not becoming available until a later release. This in turn could help streamline the development, testing and rollout efforts of a CCLSS project.

9.9 How can the new test framework achieve industry wide adoption?

For the new test framework to achieve industry adoption, it needs to be integrated with, or be made an extension of, a widely accepted industry testing standard or project management methodology. Furthermore, for this framework to offer optimal benefits to the testing of a CCLSS, the requirements of that CCLSS need to be formulated from inception to map easily onto the framework's layers. This may be achieved by incorporating the ideas of the

framework with system engineering standards used in the specific domain that the CCLSS system belongs to, e.g. telecommunications, banking, ecommerce, health or others. For example, for telecommunications Operational Support Systems (OSS), the most relevant standards are likely to be the ITU/TMF design standards such as NGOSS and eTOM standards (Kelly, 2003) (TMF, 2013).

Additionally, the application of the framework needs to be supported by requirement management and test management tools, e.g. DOORS (IBM Rational, 2013) or Quality Center (HP, 2013) via templates, notations and functionality that support the concepts of the layered test framework, its test categories, subcategories, the assurance methods, and by allowing the creation of links between test cases and subcategories representing their interdependencies.

Following on from what was mentioned in Section 6.9, the integration within industry standards and tools of the test framework is outside the scope of this thesis, and can be the subject of potential further investigation.

For instance, further work can also be done to detail the most appropriate notations and test artefacts to be used for each of the framework's six layers and their subcategories, and the form with which test cases for each test subcategory can be specified, e.g. FSMs, structure diagrams, activity diagrams, MSCs, or others. For this work, UML 2.0 could be an appropriate notation to use but other graphical notations, such as TTCN, could also be considered. Any such work should have the objective of shaping the domain-specific approach into a more precise set of templates for specifying the resulting test cases and other related "test ware".

Finally, the IT profession is currently too familiar with projects for large scale IT systems being delivered late, over the original budget and below the originally intended specification. To achieve a paradigm shift takes more than the publication of this one thesis, but a collaboration effort across the profession to further crystallise, apply and develop its ideas by trialling it with more real-life communications-critical large scale systems.

Bibliography

Riungu, L. M., Taipale, O., & Smolander, K. (2010). Research Issues for Software Testing in the Cloud. *2nd IEEE International Conference on Cloud Computing Technology and Science.* IEEE.

Adams, S. (2009). *ITIL V3 foundation handbook* . Norwich: The Stationery Office (TSO).

Aichernig, B. K., Griesmayer, A., Johnsen, E. B., Schlatte, R., & Stam, A. (2009). Conformance Testing of Distributed Concurrent Systems with Executable Designs. *FMCO 2008. LNCS 5751*, pp. 61-81. Springer.

Alicherry, M., Bhatia, R. S., Nagesh, H., Phadke, C., & Poosala, V. (2003). Testing and Verification of Network Management and Design Tools. *8*(3).

Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing.* Cambridge University Press.

APM. (2012). *PRINCE2*. Retrieved March 2012, from Official PRINCE2 website: http://www.prince-officialsite.com/

Arcuri, A., Iqbal, M. Z., & Briand, L. (2012). Random Testing: Theoretical Results and Practical Implications. *IEEE Transactions on Software Engineering*, 258-277.

Australian Bureau of Statistics. (2013). *Sample Size Calculator*. Retrieved June 13, 2013, from NSS: http://www.nss.gov.au/nss/home.nsf/pages/Sample+size+calculator

Baker, P., Dai, Z. R., Grabowski, J., Haugen, O., Schieferdecker, I., & Clay, W. (2008). *Model-driven testing: using the UML testing profile.* Berlin: Spinger-Berlag.

Barmi, Z. A., Ebrahimi, A. H., & Feldt, R. (2011). Alignment of requirements specification and testing: A systematic mapping study. *2011 Fourth International Conference on Software Testing, Verification and Validation Workshops* (pp. 476-485). IEEE Computer Society.

Batteram, H. J., & Romijn, W. A. (2003). A Test Framework for CORBA Component Model-Based Software Systems. *Bell Labs Technical Journal, 8(3)*, 15-29.

BC Associates. (2012). *Continuity Management*. Retrieved October 15, 2013, from The ITIL and ITSM Directory : http://www.itil-itsm-world.com/

Becchina, W., Ciccarelli, L., & Kenny, J. (2007). Re-inventing the enterprise with communications-enabled applications. *Nortel Technical Journal*(5), 33-41.

Belgamo, A., Fabbri, S., & Maldonado, J. C. (2005). TUCCA: improving the effectiveness of use case construction and requirement analysis. *International Symposium on Empirical Software Engineering.* Noosa Heads, Australia: IEEE.

Bellware, S. (2008, May/June). *Code Magazine*. Retrieved February 06, 2014, from http://www.codemag.com/article/0805061

Bertolino, A. (2007). Software Testing Research: Achievements, Challenges, Dreams. *Future of Software Engineering (FOSE'07)* (p. 1). IEEE Computer Society.

Bertolino, A., De Angelis, G., Di Sandro, A., & Sabetta, A. (2011). Is my model right? Let me ask the expert. *The Journal of Systems and Software*, 1089–1099.

Binder, R. V. (2012). *2011 Model-based Testing User Survey: Results and Analysis.* System Verification Associates. System Verification Associates.

Bjorklund, T. A. (2013, March). Initial mental representations of design problems: Differences between experts. *Design Studies, 34*(2), 135-160.

Bochmann, G. v., Rayner, D., & West, C. H. (2010). Some notes on the history of protocol engineering. *Computer Networks*, 3197–3209.

Bochmann, G. v., Rayner, D., & West, C. H. (2010, May). Some notes on the history of protocol engineering. *Computer Networks*.

Boehm, B. (2006). Some future trends and implications for systems and software engineering processes. *Systems Engineering*, 1-19.

Boehm, B. W. (1984, January). verifying and Validating Software Requirememts and Design Specification. *IEEE Software*, 75-88.

Boehm, B., & Basili, V. R. (2001). Software defect reduction top 10. *Computer*, 135-137.

Boehm, B., Basili, V. R., & Rombach, H. D. (2005). *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili.* Berlin, Hiedleberg, Garmany: Springer-Verlag.

Boroday, S., Petrenko, A., & Groz, R. (2007). Can a Model Checker Generate Tests for. *190*, 3-19.

Boucharas, V., van Steenbergen, M., Jansen, S., & Brinkkemper, S. (2010). *The Contribution of Enterprise Architecture to the Achievement of Organizational Goals: Establishing the Enterprise Architecture Benefits Framework.* Utrecht University, Department of Information and Computing Sciences, Utrecht, The Netherlands.

Briand, L. (2011). Useful Software Engineering Research: Leading a Double-Agent Life. *IEEE International Conference on Software Maintenance.* Simula Research Laboratory - University of Oslo - Norway.

Briand, L. (2012). Embracing the Engineering Side of Software Engineering. *IEEE Software,* 92-95.

Briand, L., & Labiche, Y. (2002). A UML-Based Approach to SystemTesting. *Softw Systems Modeling, 1*, 10–42.

Budnik, C. J., Chan, W. K., Kapfhammer, G. M., & Zhu, H. (2011). Guest editors' introduction to the special section on exploring the boundaries of software test automation. *DOI 10.1007/s11219-011-9159-2, 19*, 689-690.

Campbell, R. V. (1952). Evolution of automatic computation. *ACM '52 Proceedings of the 1952 ACM national meeting.* Pittsburg: ACM.

Carletta, J. (1996). Assessing Agreement on Classification Tasks:The Kappa Statistic. *Computational Linguistics, 22*(2), 249–254.

Castro, L. M. (2011). Testing Data Consistency of Data-Intensive Applications Using QuickCheck. *Electronic Notes in Theoretical Computer Science, 271*, 41-62.

Chen, Z., & Pooley, R. (2009). Redescovering Zachman Framework using Ontology from a Requirement Engineering Perspective. *Annual IEEE Computer Software and Applications Conference.* IEEE.

Chen, Z., & Pooley, R. (2009). Rediscovering Zachman Framework using Ontology from a Requirement Engineering Perspective. *33rd IEEE International Computer Software and Applications Conference.* 2009.

Chernak, Y. (2001). Validating and Improving Test Case effectiveness.

Choi, J., Nazareth, D. L., & Jain, H. K. (2010). Implementing Service-Oriented Architecture in Organizations. *Journal of Management Information Systems*, 253–286.

Clark, J. O. (2009). System of Systems Engineering and Family of Systems Engineering From a Standards, V-Model, and Dual-V Model Perspective. *IEEE SysCon 2009 — 3rd Annual IEEE International Systems.* Vancouver, Canada.

Cohen, J. (1968). Weighed kappa: Nominal scale agreement with provision for scaled disagreement or partial credit". *Psychological Bulletin, 70*(4), 213–220.

Collins, E., & Lucena, V. (2010). Iterative Software Testing Process for Scrum and Waterfall Projects with Open Source Testing Tools Experience. In A. Petrenko, A. Simao, & J.

C. Maldonado (Ed.), *Proceedings of the 22nd IFIP International Conference on Testing Software and Systems: Short Papers* (pp. 115-120). Natal, Brasil: Centre de recherche informatique de Montréal (CRIM).

Conformiq Software Ltd. (2009, March 17). *Conformiq Qtronic*. Retrieved June 17, 2012, from Conformiq Qtronic: http://www.verifysoft.com/en_qtronic.html

Davis, F. D., & Venkatesh, V. (2004). Towards Prototype User Acceptance Testing of New Information Systems: Implications for Software Project management. *51*(1).

Demuth, H. B., Jackson, J. B., Klein, E., Metropolis, N., Orvedahl, W., & Richardson, J. H. (1952). MANIAC. *ACM '52 Proceedings of the 1952 ACM national meeting.* Toronto: ACM.

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

dos Santos, A. M., Karlsson, B. F., Cavalcante, A. M., Correia, I. B., & Silva, E. (2011). Testing in an Agile Product Development Environment: An Industry Experience Report. *IEEE*.

Easton, E. (2010). *One case study is enough.* Lancaster University Management School.

Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2001). Prioritizing Test Cases for Regression Testing. *ISSTA '00* (p. 1). Portland: ACM.

Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2002). Test Case Prioritization: A Family of Empirical Studies. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 28*(2).

Ericson, T., Subotic, A., & Ursing, S. (1997). TIM—A test improvement model. *Software Testing, Verification and Reliability, 7*(4), 229–246.

eTOM. (2011). *eTOM Process Customer QoS/SLA Management*. Retrieved October 15, 2013, from http://cw-innovations.de/beispiele/cp-etom9-framework/Publikation/etomprocess38.htm

ETSI. (2005). *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language.* ETSI.

ETSI. (2011, August). *Maintaining Better Standards*. Retrieved August 25, 2011, from ETSI: http://portal.etsi.org/mbs/

ETSI. (2012). *ETSI Conformance Testing*. Retrieved March 2012, from ETSI: http://www.etsi.org/WebSite/technologies/ConformanceTesting.aspx

ETSI/Fraunhofer-Institute. (2011, November 18-20). *MBT User Conference*. Retrieved May 13, 2012, from MBT User Conference: http://www.model-based-testing.de/mbtuc11/index.html

Everett, G. D., & McLeod Jn., R. (2007). *Software Testing: Testing Across the Entire Development Life Cycle.* Hoboken, New Jersey: John Wiley & Sons.

FiReControl. (2013, July). Retrieved July 14, 2013, from Wikipedia: http://en.wikipedia.org/wiki/FiReControl

FireControl project. (29 March 2007). *FIRECONTROL PROJECT Infrastructure Services Procurement Exhibit C - Requirements.* Post Evaluation Pre Contract version, Department for Communities and Local Government, Fire and Resilience Programme.

Fleiss, J. L. (1981). *Statistical methods for rates and proportions* (2nd ed.). New York: John Wile.

Flores, S., Lucas, S., & Villanueva, A. (2008). Formal Verification of Websites. *Electronic Notes in Theoretical Computer Science, 200*, 103-118.

Formal Methods and Tools (FMT) research group, University of Twente, The Netherlands. (2006, December 22). *TorX*. Retrieved June 17, 2012, from TorX: http://fmt.cs.utwente.nl/tools/torx/introduction.html

Gartner. (2012, June). *Gartner IT Glossary*. Retrieved June 10, 2012, from Gartner IT Glossary: http://www.gartner.com/it-glossary

Gelperin, D. (1996). A testability maturity model. *Proceedings of the Fifth International Conference on Software Testing, Analysis and Review.* Orlando, FL.

Gobet, F., Lane, P. C., Croker, S., Cheng, P. C.-H., Jones, G., Oliver, I., et al. (2001). Chunking mechanisms in human learning. *TRENDS in Cognitive Sciences*, 236-243.

Graham, D., & Fewster, M. (2012). *Experiences of Test Automation.* New Jersey, USA: Pearson Education Inc.

Graham, D., Veenendaal, E. V., Evans, I., & Black, R. (2008). *Foundations of Software Testing: ISTQB Certification.* London: Cengage Learning EMEA.

Grieskamp, W., Kicillof, N., Stobie, K., & Braberman, V. (2011). Model-based quality assurance of protocol documentation: tools and methodology. *Software Testing, Verification and Reliability, 21*, 55-71.

Grozand, R., & Hierons, R. M. (2004). Testing of Communicating Systems. *TestCom 2004, 16th IFIP International Conference. Lecture Notes in Computer Science 2978*, p. xii+223. Oxford, UK: Spinger-Verlag.

Guldali, B., Funke, H., Sauer, S., & Engels, G. (2011). TORC: test plan optimization by requirements clustering. *Software Quality Journal, 19*, 771-799.

Hartman, A., Katara, M., & Paradkar, A. (2007). Domain Specific Approaches to Software Test Automation. *ESEC-FSE'07.* Cavtat, Croatia: ACM.

Hemmati, H., Acruri, A., & Briand, L. (2010). Reducing the Cost of Model-Based Testing through Test Case Diversity. *ICTSS 2010, LNCS 6435*, 63–78.

Hemmati, H., Arcuri, A., & Briand, L. (2011). Empirical Investigation of the Effects of Test Suite Properties on Similarity-Based Test Case Selection. *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation* (pp. 327-336). IEEE Computer Society.

Henderson-Sellers, B. (2011). Bridging metamodels and ontologies in software engineering. *The Journal of Systems and Software*, 301-313.

Hierons, R. M. (2003). Editorial: Testing in the large through the small. *Software Testing Verification and Reliability, 13*, 139-140.

Holmes, A., & Kellogg, M. (2006). Automating Functional Tests Using Selenium. *Proceedings of AGILE 2006 Conference.* IEEE.

HP. (2013). *HP Quality Center Software*. Retrieved August 14, 2013, from HP Quality Center Software: http://www8.hp.com/us/en/software-solutions/software.html?compURI=1172141

IABG. (1993). *V-Model Lifecycle Process Model Brief Description.* IABG Information Technology Industrieanlagen Betriebsgesellschaft mbH.

IAP. (2013). Retrieved August 9, 2013, from http://www.iap.org.uk/main/

IBM. (2014). *Resilience plans: validation and testing*. Retrieved March 07, 2014, from http://www-935.ibm.com/services/us/en/it-services/resilience-plans-validation-and-testing.html

IBM Rational. (2013). *IBM DOORS*. Retrieved August 14, 2013, from IBM DOORS: http://www-03.ibm.com/software/products/us/en/ratidoor

IBM Rational. (2013). *Rational Quality Manager*. Retrieved August 14, 2013, from IBM Rational: http://www-03.ibm.com/software/products/us/en/ratiqualmana

IBM RUP. (2012). *IBM Rational Unified Process (RUP)*. Retrieved March 2012, from IBM RUP: http://www-01.ibm.com/software/awdtools/rup/

IBM UML 2. (2006). UML 2: A Model-driven development tool. *IBM Systems Journal, 3*, 607-620.

IEEE 1012. (2012). *1012-2012 - IEEE Draft Standard for System and Software Verification and Validation.* IEEE. IEEE Standards Association.

IEEE 829. (2008). *IEEE Standard for Software and System Test Documentation 829-2008.* IEEE Computer Society, Software & Systems Engineering Standards Committee. New York: IEEE Computer Society.

IETF. (2010). *An Overview of Operations, Administration, and Maintenance (OAM) Mechanisms.*

IETF. (2011). *Guidelines for the Use of the "OAM" Acronym in the IETF*.

Institute for System Programming of the Russian Academy of Sciences. (2006). *UniTESK*. Retrieved June 17, 2012, from UniTESK: http://www.unitesk.com/

ISO. (2005). *ISO 9000:2005 - Quality management systems - Fundamentals and vocabulary.* International Standarization Organisation.

ISO/IEC 12207. (2008). *IS0/IEC 12207:2008 Software life cycle processes.* ISO.

ISO/IEC 15504. (n.d.). ISO/IEC15504-5 Information Technology — Process Assessment — Part 5.

ISO/IEC/IEEE 29119. (2014, March). *ISO/IEC/IEEE 29119 Software Testing*. Retrieved March 7, 2014, from Software Testing Standards: http://softwaretestingstandard.org/

ISO/IEC/IEEE 42010. (2012, April 19). *Survey of Architecture Frameworks*. Retrieved June 10, 2012, from ISO/IEC/IEEE 42010: http://www.iso-architecture.org/ieee-1471/afs/frameworks-table.html

ISO/IEC/IEEE 42010. (2012, June). *Systems and software engineering — Architecture description*. Retrieved June 10, 2012, from ISO/IEC/IEEE 42010: http://www.iso-architecture.org/ieee-1471/index.html

ISTQB. (2012). *Certified Tester Advanced Level Syllabus Test Manager.* International Software Testing Qualifications Board.

ISTQB-WEB. (2012). *ISTQB*. Retrieved May 10, 2012, from ISTQB: http://www.istqb.org/

ITU-T. (1994). *X.200 Open Systems Interconnection - Model and Notation.* International Communications Union.

ITU-T. (1996). *Recommendation M.3010: Principles for a telcommunications management network.* ITU-T.

ITU-T. (1997). *Recommendation M.3400: TMN management functions.* ITU-T.

ITU-T. (2000). *Overview of TMN Recommendations: Recommendation M.3000.* ITU.

ITU-T. (2004). *Recommendation M.3050 eTOM Suppliment 1: ITIL Application Note.* ITU-T.

ITU-T. (2012, March). *ITU-T Recommendation with Test Specifications*. Retrieved from ITU-T Recommendation with Test Specifications: http://www.itu.int/net/ITU-T/cdb/Test-Specifications.aspx

Java Pathfinder Wiki. (2012). *JPF*. Retrieved June 17, 2012, from JPF: http://babelfish.arc.nasa.gov/trac/jpf

Jia, Y., & Harman, M. (September/October 2011). An Analysis and Survey of the Development. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 37*, 649-678.

Jovic, M., Adamoli, A., Zaparanuks, D., & Hauswirth, M. (2010). Automating Performance Testing of Interactive Java Applications. *AST '10.* Cape Town, South Africa: ACM.

Juristo, N., Moreno, A. M., & Strigel, W. (2006, July/Augist). Software Testing Practices in Industry. *IEEE Software*, 19-21.

Just, R., & Schweiggert, F. (2011). Automating unit and integration testing with partial oracles. *Software Qual Journal, 19*, 753-769.

Karlstrom, D., Runeson, P., & Sara Norden, S. (2005, January). A minimal test practice framework for emerging software organizations. *Software Testing, Verification and Reliability, 15*, 145-166.

Kasurinen, J., Runeson, P., Riungu, L., & Smolander, K. (2011). A Self-assessment Framework for Finding Improvement Objectives with ISO/IEC/IEEE 29119 Test Standard. *EuroSPI 2011. CCIS 172*, pp. 25-36. Berlin Heidelberg: Springer-Verlag.

Kasurinen, J., Taipale, O., & Smolander, K. (2010). Software Test Automation in Practice: Empirical Observations. (P. Laplante, Ed.) *Advances in Software Engineering, 2010*.

Kasurinen, J., Taipale, O., & Smolander, K. (2011). How Test Organizations Adopt New Testing Practices and Methods? *2011 Fourth International Conference on Software Testing, Verification and Validation Workshops* (pp. 553-558). IEEE.

Kelly, M. B. (2003). The TeleManagement Forum's Enhanced Telecom. *Journal of Network and Systems Management*, 109-119.

Khan, S. u., Rehman, I. u., & Malik, S. U. (2009). The Impact of Test Case Reduction and Prioritization on Software Testing Effectiveness. *International Conference on Emerging Technologies*, (pp. 416-421).

King, J. (2014). *Resource page: Generalized Kappa & Other Indices of Interrater Reliability*. Retrieved January 26, 2014, from http://www.ccitonline.org/jking/homepage/interrater.html

Kollanus, S. (2011). Critical Issues on Test-Driven Development. *PROFES 2011, LNCS 6759* (pp. 322–336). Berlin: Springer-Verlag.

Koomen, T., & Pol, M. (1999). *Test Process Improvement—A Practical Step-by-step Guide to Structured Testing.* Boston, MA: Addison-Wesley.

Landis, J.R.; & Koch, G.G. (1977). "The measurement of observer agreement for categorical data". Biometrics 33 (1): 159–174.. JSTOR 2529310. PMID 843571. (n.d.).

Lei, B., Liu, Z., Morisset, C., & Li, X. (2010). State Based Robustness Testing for Components. *Electronic Notes in Theoretical Computer Science, 260*, 173-188.

LSCITS. (2013). *Large Scale Complex IT Systems*. Retrieved August 2013, from http://lscits.cs.bris.ac.uk/

LSCITS. (2013). *LSCITS Initiative Overview*. Retrieved October 26, 2013, from LSCITS Overview: http://lscits.cs.bris.ac.uk/overview.html

LSCITS. (2013). *LSCITS Research Programme*. Retrieved October 26, 2013, from LSCITS Research Programme: http://lscits.cs.bris.ac.uk/research.html

Lyu, L. R. (2007). Software Reliability Engineering: A Roadmap. *Future of Software Engineering (FOSE2007).* IEEE.

Machado, P., Vincenzi, A., & Maldonado, J. C. (2010). Software Testing: An Overview. *Lecture Notes in Computer Science* , 1-17.

Mantis. (2012). Retrieved 2012, from Mantis Testing Tool Manual: http://www.mantisbt.org/

Marathon. (2012). Retrieved from Marathon Integrated Testing Environment: http://www.marathontesting.com/blog

Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2007). 'Good' Organisational Reasons for 'Bad' Software Testing: An Ethnographic Study of Testing in a Small Software Company. *29th International Conference on Software Engineering (ICSE'07).* IEEE.

Mathur, S., & Malik, S. (2010). Advancements in the V-Model. *International Journal of Computer Applications, Volume 1*(No. 12).

Mattiello-Francisco, F., Martins, E., Cavalli, A. R., & Yano, E. E. (2012). InRob: An approach for testing interoperability and robustness of real-time. *The Journal of Systems and Software*, 3-15.

Maximilien, M. E., & Williams, L. (2003). Assessing Test-Driven Development at IBM. *Proceedings of the 25th International Conference on Software Engineering.* IEEE Computer Society.

Microsoft. (2014). *Operational Readiness Checklists*. Retrieved from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/cc296893(v=bts.10).aspx

Microsoft Research. (2006). *XRT*. Retrieved June 17, 2012, from XRT: http://research.microsoft.com/apps/pubs/default.aspx?id=77413

Microsoft Research. (2012, June). *Microsoft Research*. Retrieved June 17, 2012, from Model-based Testing with SpecExplorer: http://research.microsoft.com/en-us/projects/specexplorer/

Myers, J. G. (1978, September). A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections. *Communications of ACM, 21*(9), 760-768.

Myers, J. G., Sandler, C., & Badgett, T. (2012). *The Art of Software Testing* (3rd ed.). Hoboken, New Jersey: John Wiley & Sons.

Naslavsky, L., Ziv, H., & Richardson, D. J. (2007). Towards Traceability of Model-based Testing Artifacts. *AMOST'07* (pp. 105-114). London: ACM.

National Audit Office. (2011). *The failure of the FiReControl project.* Norwich: The Stationery Office.

Neto, A. C., Subramanyan, R., Vieira, M., & Travassos, G. H. (2007). A Survey on Model-based Testing Approaches: A Systematic Review. *WEASELTech'07.* Atlanta Georgia, USA: ACM.

Network World. (2008). *Chapter 1: Building a Simple Network*. Retrieved August 13, 2013, from Network World: http://www.networkworld.com/redesign08/subnets/cisco/053008-ch1-ccna-prep-library.html?page=4

NIST. (2011). *The NIST Definition of Cloud Computing.* U.S.A Department of Commerce.

Offutt, J. (2008). Editorial: Software testing is. *SOFTWARE TESTING, VERIFICATION AND RELIABILITY*, 191-192.

Orden, A. (1952). Solution of systems of linear inequalities on a digital computer. *ACM '52 Proceedings of the 1952 ACM national meeting.* Pittsburgh: ACM.

Pacione, M. J., Roper, M., & Wood, M. (2004). A Novel Software Visualisation Model to Support Software Comprehension. *11th Working Conference on Reverse Engineering (WCRE'04)* (pp. 1095-1350). IEEE.

Page, A., Johnston, K., & Rollinson, B. J. (2009). *How we test software at microsoft.* Microsoft Corporation.

Papadakis, M., & Malevris, N. (2011). Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing. *Software Quality Journal, 19*, 691-723.

Parker, J. (2005). Three key ingredients to effective IT management. OpenWater Solutions, LLC.

Paugh, K. (2011). *Lean-Agile Acceptance Test-Driven Development.* Addison-Wesley.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). *Capability maturity model for software, version 1.1.* Software Engineering Institute.

Ponte, D., Rossi, A., & Zamarian, M. (2009). Cooperative design efforts for the development of complex IT-artefacts. *Information Technology & People*, 317-334.

Pretschner, A., Prenninger, W., Wagner, S., Kunel, C., Baumgartner, M., Sostawa, B., et al. (2005). One evaluation of Model-Based Testing and its automation. *ICSE'05.* St. Louis, Missouri, USA: ACM.

Quadri, S. M., & Farooq, U. S. (2010, September). Software Testing – Goals, Principles, and Limitations. *International Journal of Computer Applications, 6*(9), 7-10.

Rayport, J. F., & Sviokla, J. J. (1995). Exploiting the virtual value chain. *Harvard Business Review*.

Reid, S. (2012). The New Software Testing Standard. In S. Reid, C. Dale, & T. Anderson (Eds.), *Achieving Systems Safety* (pp. 237-255). London: Springer.

Reid, S. C. (2000). BS 7925-2: The Software Component Testing Standard.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empir Software Engineering, 14*, 131-164.

Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). *Case Study Research in Software Engineering.* Hoboken, New Jersey: John Wiley and Sons.

Schatz, B., & Pfaller, C. (2010). Integrating Component Tests to System Tests. *Electronic Notes in Theoretical Computer Science, 260*, 225-241.

Schmidt, D. C. (2006). Model-Driven Engineering. *IEEE Computer*, 25-31.

Scully, J. K. (1998). The hidden crisis in test effectiveness. *AUTOTESTCON '98. IEEE Systems Readiness Technology Conference* (pp. 59-66). Salt Lake City, UT , USA: IEEE.

SEI. (2012). *Capability Maturity Model Integration (CMMI)*. Retrieved March 2012, from The Carnegie Mellon Software Engineering Institute (SEI): http://www.sei.cmu.edu/cmmi/

SEI. (2013). *Architecture Tradeoff Analysis Method*. Retrieved August 14, 2013, from Software Architecture : http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm

SEI ULS Study Team. (2006). *Ultra-Large-Scale Systems: The Software Challenge of the Future.* Software Engineering Institute Carnegie Mellon.

SEI ULS System Research . (2012, March). *SEI ULS System Research* . Retrieved March 2012, from Ultra-Large-Scale Systems : http://www.sei.cmu.edu/uls/

Selic, B. (2006). UML 2: A model-driven development tool. *IBM Systems Journal, 45(3)*, 607-620.

Sheffield, J., & Lemétayer, J. (2013). Factors associated with the software development agility of successful projects. *International Journal of Project Management, 31*(3), 459-472.

Siegl, S., Hielscher, K.-S., & German, R. (2010). Introduction of Time Dependencies in Usage Model Based Testing of Complex Systems. *2010 4th Annual IEEE Systems Conference* (pp. 622 - 627). IEEE.

Sinha, A., & Smidts, C. (2006, July). HOTTest: A Model-Based Test Design Technique for Enhanced Testing of Domain-Specific Applications. *ACM Transactions on Software Engineering and Methodology, 15*(5), 242-278.

Smith, M., & Thompson, N. (2008). The Keystone to Support a Generic Test Process: Seperating the "What" from the "How". *IEEE Conference on Software Testing Verification and Validation Workshop (ICSTW'08).* IEEE Computer Society.

Sogeti. (2012). *TPI NEXT*. Retrieved May 10, 2012, from SOGETI: http://www.sogeti.com/Curious-about-Sogeti/publications/TPI-NEXT/

Somekh, B. (2006). *Action research: A methodology for change and development.* McGraw-Hill International.

SQS. (2012). *SQS Group*. Retrieved May 10, 2012, from SQS Group: http://www.sqs.com/en/group/index.php

Stamatis, D. H. (2003). *Failure mode and effect analysis: FMEA from theory to execution* . Milwaukee: American Society for Quality.

Steiner, M., Blaschke, M., Philipp, M., & Schweigert, T. (2010). Make test process assessment similar to software process assessment—the Test SPICE approach. *Journal of Software Maintenance and Evolution: Research and Practice*.

Strasser, A., Mayr, H., & Naderhirn, T. (2010). Harmonizing the Test Support for Object-oriented Legacy Systems Using State-of-the-Art Test Tools. *ECOOP '2010.* Maribor, Slovenia, EU: ACM.

Svensson, R. B., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R., et al. (2011). Prioritization of Quality Requirements: State of Practice in Eleven Companies. *2011 IEEE 19th International Requirements Engineering Conference*, (pp. 69-78).

SWSOL. (2014). *http://www.swsol.org/articles//Dates_expert_panels*. Retrieved January 26, 2014, from http://www.swsol.org/articles//Dates_expert_panels

TestLink. (2012). Retrieved 2012, from TestLink Testing Tool Documentation: http://www.teamst.org/

TGV. (2012). Retrieved June 17, 2012, from The test sequence generator TGV: http://www.irisa.fr/vertecs/Logiciels/TGV.html

The Royal Academy of Engineering. (2004). *The Challenges of Complex IT projects.* London: The Royal Academy of Engineering.

TMF. (2011). *tmforum Frameworx*. Retrieved August 13, 2013, from TMF eTOM: http://www.tmforum.org/Models/eTOM/etom_9.0_publication/Framework/etomprocess165.htm

TMF. (2013). *Business Process Framework (eTOM)*. Retrieved August 12, 2013, from eTOM: http://www.tmforum.org/BusinessProcessFramework/1647/home.html

TMMi Foundation. (2012). *TMMi Foundation*. Retrieved May 10, 2012, from TMMi Foundation: http://www.tmmifoundation.org/html/tmmiorg.html

Utting, M., & Legeard, B. (2007). *Practical Model-Based Testing: A Tools Approach.* San Francisco, USA: Elsevier.

v Steenbergen, M., Bos, R., & Brinkkemper, S. (2008). The application of a prescriptive and a constructive perspective to design-science research. *Proceedings of the 3rd International Conference on Design Science Research in Information Systems & Technology.* Georgia State University, Atlanta, Georgia, US.

Value Chain Group. (2007). Retrieved August 12, 2013, from Value Chain Group: http://www.value-chain.org/

Verschuren, P., & Hartog, R. (2005). Evaluation in Design-Oriented Research. *Quality & Quantity,* 39:733–762.

Yanga, L.-R., Huang, C.-F., & Wua, K.-S. (2011). The association among project manager's leadership style, teamwork and project success. *International Journal of Project Management, 29,* 258–267.

Yin, R. K. (2009). *Case Study Research: Design and Methods* (Fourth Edition ed.). Southern Oaks: SAGE publications.

Yoo, S., & Harman, M. (2007). Pareto Efficient Multi-Objective Test Case Selection. *ISSTA'07* (pp. 140-150). London: ACM.

Yoo, S., Harman, M., Tonella, P., & Susi, A. (2009). Clustering Test Cases to Achieve Effective & Scalable Prioritisation Incorporating Expert Knowledge. *ISSTA'09.* ACM.

Zachman International. (2012). Retrieved June 10, 2012, from Zachman International: http://www.zachman.com/about-the-zachman-framework

Zachmann, J. (1987). A Framework for Information Systems Architecture. *IBM Systems Journal, 12*(3).

Zheng, L., & Tse, D. N. (2003, May). Diversity and Multiplexing: A Fundamental Tradeoff in Multiple-Antenna Channels. *IEEE Transactions on Information Theory, 49*(5), 1073-1096.

Zimmermann, H. (1980). OSI Reference Model - The IS0 Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications, 28*(4), 425-432.