A new test framework for communications-critical large scale systems

M. A. Nabulsi, R. M. Hierons

***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 paper argues for the need for a specifically tailored framework to achieve effective and precise testing of communications-critical large scale systems (CCLSSs). The paper briefly discusses how generic test approaches are leading to inefficient and costly test activities in industry. The paper then outlines the features of an alternative CCLSS domain-specific test framework, and then provides an example based on a real case study. The paper concludes with an evaluation of the benefits observed during the case study and an outline of the available evidence that such benefits can be realized with other comparable systems.**

***Index Terms*— testing, test framework, communications-critical, large scale IT systems, test case prioritization, requirements prioritization**

I. Introduction

During the 1980s and early 1990s, formal software test methodologies, terminologies and practices were not clearly established in the IT 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. The way testing is done and how efficient it will be still relies 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 organized 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 [1, p.1]. This creative aspect is a feature inherent in testing and need not be viewed negatively, but reliance on the subjective judgment 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. 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 results in inadequately tested systems with lower reliability and availability levels than is needed, as well as project delays and higher project costs [2]. The impact of the effectiveness and precision of testing is further magnified for large scale IT systems that are prevalent today. These are systems that combine multiple technologies, multiple hardware platforms, multiple software components, multiple internal and external communications interfaces, and can be 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 paper, these systems will be called "Communications-Critical Large Scale Systems" or "CCLSSs".

CCLSSs are increasingly prevalent, more complex and critical. In fact, developed societies can no longer continue to function normally without this class of systems. Examples of CCLSSs are: emergency mobilization applications, distributed banking applications, trading systems, web based portals and ecommerce sites, supply chain applications, fleet management, automatic vehicle location systems (AVLS), e-health systems, telecommunications network management and operations support systems (NMS/OSS).

Can the IT industry's vendors, clients and users adopt test approaches that would ultimately lead to more efficient testing of communications-critical large scale systems? The discussion within this paper will start by explaining factors that lead to inefficiency in testing such systems.

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE

II. Test Methodologies and Standards

When testing a communications-critical large scale IT system, there are numerous commercial test methodologies and standards that can be adopted. Examples of these are the V-Model [3], Agile testing [4], IEEE 829 Standard for Software and System Test Documentation [5], BS 7925-2 standard for component testing [6], software life cycle standard ISO/IEC 12207 [7], verification and validation standard IEEE 1012 [8], software assessment and improvement standard ISO 15504 [9], the more recent emerging ISO/IEC/IEEE 29119 [10], as well as many other company specific methodologies.

Such methodologies and standards are familiar in the IT industry, although not necessarily adopted universally or strictly. They define how testing activities for a system should be formalized and structured. Despite their variety, they share a common premise that effective testing can be achieved by adhering to or tailoring a pre-defined process. With significant investment committed towards the adoption of such standards and methods by industry, why does the IT industry suffer regularly from problematic deliveries of new systems [11]?

For a new test framework to be capable of delivering more effective testing, it needs to improve the precision and reduce the subjectivity of the test design effort. It also needs to "front-load" the testing of the complex features of the system, i.e. test them as early as possible. These are the main underlying premises behind the ideas outlined in the remainder of this paper.

III. The Proposed Domain Specific Test Framework

Most methodologies and standards listed in Section II 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. Development methodologies moved from being procedural in the 1980s, to object oriented in the 1990s. Nowadays, the trend is for model driven or service oriented architectures (MDA, SOA) and Agile development approaches. Commercial test practices for large scale communications-critical systems have not kept up with these changes. For example, when defining tests for communications technologies and services, the traditional distinction between functional and non-functional features is often less appropriate when compared to transactions processing systems, and test cases cannot always be expressed in the traditional test case style of initial condition/input/procedure/output. This is because communications networks and interfaces are for setting up connections between senders and receivers and transporting data rather than accepting inputs and calculating or producing outputs.

Furthermore, large scale IT systems have been increasingly dependent on their communications interfaces, yet the widely accepted approaches to their testing have not kept up with this convergence between IT and communications. The adoption of widely used standards and methods based on the V-Model, such as IEEE 829, BS 7925-2, ISO/IEC 12207, or IEEE 1012, would only define the structure of the test process rather than the precision and completeness of coverage of the test cases or their suitability for testing CCLSSs. Therefore, we believe that the IT industry needs a new test framework that allows for CCLSSs to be tested more precisely and predictably.

What would the proposed test framework be like?

An effective test framework intended for a specific type of system could benefit from being derived from a domain expert's view of the structure of such systems [1, p.12], as well as the order with which a system's components, or groups of components, should be tested. Having such a starting point for the test design could help the test analyst in identifying gaps, inconsistencies or ambiguity in the system requirements or the system design, and reduces the subjectivity of the test design effort. In other words, such a framework would be a conceptual representation of a domain-expert's knowledge from a testing viewpoint. This contrasts, say, with the V-Model which is a conceptual representation of phases of the development cycle. Such a domain-specific test framework can be thought of as the testing equivalent to Zachman's Enterprise Architecture Framework [12] in the sense that it provides a simplified "model" upon which the test analysis and test design can be based. This is comparable to how an Enterprise Architecture Framework such as Zachman's can be used as the basis for deriving a system's architecture.

How can such a test framework be formulated?

The idea of layers is a fundamental and well established feature of communications protocol design and testing in the world of telecommunications. Protocol testing [13] 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 large scale systems with significant communications layers.

Motivated by the ideas discussed in this section, a domain-specific framework was defined for use as a template for high-level test design for CCLSSs **(Figure 1)**.

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE

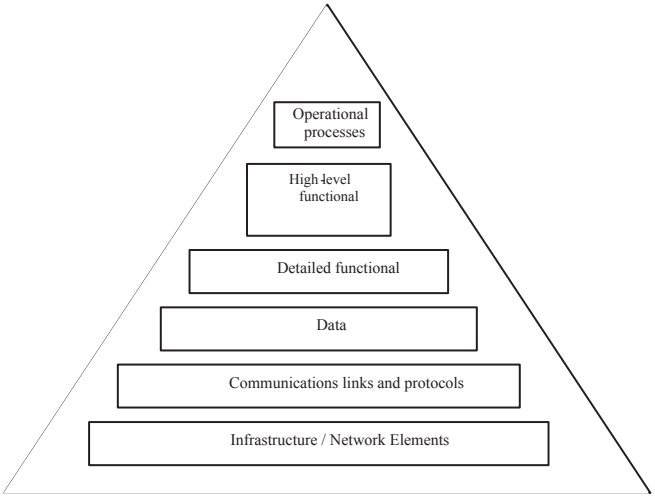


Figure 1: Tester's view of a communications-critical large scale system

Under the proposed approach, test subgroups and phases are categorized according to one of the following categories:

- Non-IT commercial: these are outside the scope of testing.
- Infrastructure: communications hardware, IT hardware and software packages, configuration and setup needed for the infrastructure.
- Communications links and communications features.
- Data.
- Detailed functional: 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: the functional features that describe how the system is meant to achieve the intended business and operational processes.
- Business and operational processes: these represent what the system is meant to achieve.

Each of the five layers should in turn have its layer-specific test approach.

The industrial case study described in the next section evaluates the feasibility and the benefits of adopting such a framework as the starting point to organizing the test design and test activities for a real-life CCLSS project.

IV. THE COMMUNICATIONS LAYER

This section describes an industrial case study in which the ideas of the layered test framework discussed in the previous section were detailed further for the communications layer and then applied to the communications layer of a significant CCLSS. For confidentiality, the actual name of the system is not disclosed; it will be referred to as Sys.

The case study applied the layered test approach to the requirements based test design for Sys, and focused on the communications interfaces requirements. The requirements were not intended to fit with the layered model, and contained a large number of technical and non-technical requirements with varying degrees of granularity and specificity. Therefore, significant analysis effort was needed to identify and group together the core technical communications requirements of Sys before the ideas presented in this paper could be applied and evaluated.

It was possible to group the requirements describing the "communications interfaces layer" according to the five individual communications interfaces: the primary radio network interface, secondary/fallback radio network interface, telephony interface, wide area network interface and local area network interface.

For each interface, the requirements were further organized according to the nineteen subcategories shown in Figure 2 which summarize potential generic testable features of a communications interface for any CCLSS.

The nineteen subcategories were defined by the authors as further detail to the overall domain-specific test framework specifically for the communications layer. They were derived using ideas from the telecommunications field (e.g. OSI layers [14], TMN layers [15], eTOM/FAB process model [16]) where abstract layers and common logical processes are used as the conceptual basis for unifying and standardizing the approaches to managing complex and technically varied telecommunications networks. The authors also intended the nineteen subcategories to represent a "value chain" of the elements that deliver the services of a network interface. One other intention behind this categorization was 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 was to allow for simpler prioritization of testing and to minimize the interdependency between the subcategories.

The groupings of requirements of Sys were used as templates for defining high level requirements-based communications test cases for the communications interfaces of the system. These were the primary output of the case study: a precise, more objective and prioritized set of outline test cases together with other related specific QA actions, i.e. inspections, demonstrations, or reviews.

Figure 2 shows the 19 subcategories, with the intention being that each communications requirement is placed in one of these. The test cases for a particular requirement would then be placed in the same subcategory as the requirement, with this inducing a partial order on the test case. Specifically, if test

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE

case t appears in an earlier subcategory than test case t' then t would be used before t'. Figure 2 also describes the dependencies between the subcategories. Typically, subcategory C depends on subcategory C' if either features from C' are required in order to implement/test features from C or changes in C' are likely to lead to retesting of features in C. There are also cases where dependence arises for non-technical reasons such as project management or commercial factors.

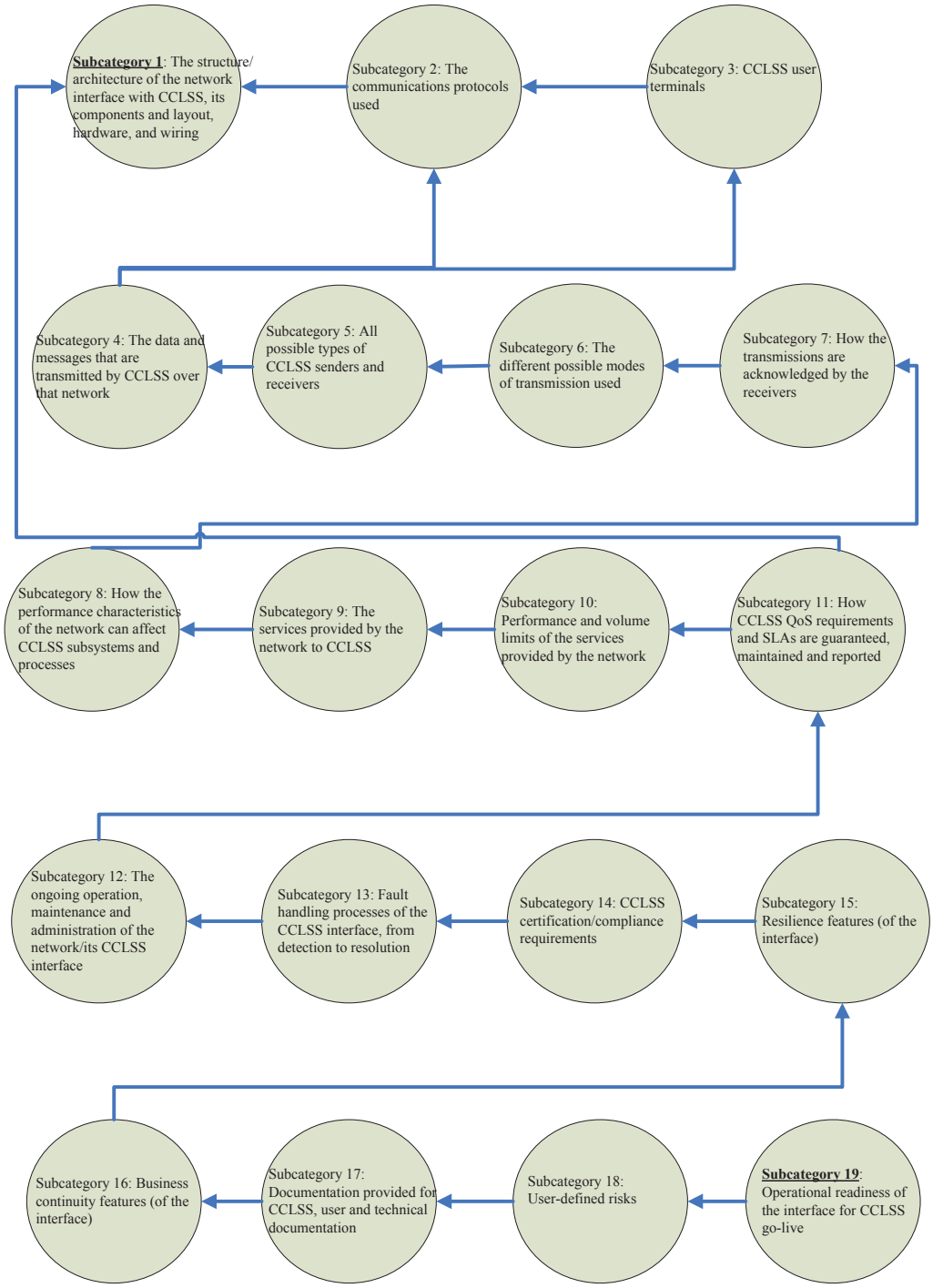


Figure 2: The subcategories

We produced a diagram for each communications interface of Sys, with each diagram showing the nineteen test subcategories with requirements allocated to each subcategory. Where no requirements were identified then this was also indicated. The diagram for each interface was then cross-checked for consistency against the outline test cases for that interface.

Our findings from the case study are summarized in the next section.

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE

V. EVALUATION

This section outlines evidence and observations that emerged through the case study, firstly to indicate that the proposed framework is applicable to other comparable systems, and secondly to demonstrate its benefits both qualitatively and, where possible, quantitatively.

The evidence

Although the requirements were not written to fit with the proposed domain-specific test framework, it was possible to categorize them according to the layered test framework and to isolate a distinct communications layer. This provides confidence that the framework can be applied to other comparable systems.

The nineteen test subcategories mapped well to the requirements for each of the communications interfaces of Sys, although the requirements were not originally written to adhere to such a structure. The same diagram template was usable for organizing the requirements for each of the five interfaces.

The dependencies and sequences, represented by arrows in Figure 2, remained valid for each of the communications interfaces of Sys. This provides further confidence that the framework can help organize and prioritize testing for the communications layers of other comparable systems. Like in other engineering-type activities, prioritization is an indicator for good quality testing [17]. Specifically, better prioritization and scheduling of tests can lead to more faults being found early in the development cycle, hence costing significantly less to rectify and requiring less re-testing. This aspect of test effectiveness will be referred to further in the randomized simulation subsection.

The benefits

Based on the observations and evidence that emerged from the case study, we expect the following four main advantages to be realistically realizable if the framework is adopted as the basis for testing comparable CCLSS:

1- Effective prioritization of the testing, as the simulations described below show.

2- Effective identification of gaps and inconsistencies in the requirements and the technical design through the use of the test subcategories. This can help identify areas of potential contradiction or ambiguity in the requirements, e.g. by using the diagram in Figure 2 as a model for analysis and review of the requirements.

3- Improved synergy between testing and the overall project activities and phases because the framework helps maintain a continuous link between test activities and requirements and the layers of the framework can be used to define the phases of an IT project.

4- Improved confidence in the results of tests during each phase of the project. This is due to the efficient prioritization of the tests, meaning fewer tests will be run too early or too late.

Test effectiveness randomized simulations

One key estimate of "test effectiveness" is the re-testing effort needed once a fault is identified during testing. An effective test framework should lead to well prioritized test cases *("prioritized" in this context is used to mean "optimally ordered")*, which in turn should lead to early detection of faults as well as reduced re-testing effort.

We devised a way to estimate the efficiency of an ordering of a set of requirements-based test cases, by deriving a numeric indication of the impact of the interdependencies as explained in the next paragraph. Given requirements X and Y, a "dependency" refers to the situation in which X cannot be fulfilled correctly until Y is fulfilled correctly, i.e. X is "dependent" on Y. Here, if a fault is found in Y and fixed then this is likely to necessitate the re-testing of the functionality of X.

Let us suppose that we have a sequence X_1, ..., X_n of requirements. For a requirement X_m we count the number of requirements that are before X_m in the sequence and that also depend on X_m and let this count be C_m. The overall dependence count is the sum of the C_m over m = 1, ..., n. The purpose of such a simplified "test efficiency" calculation method is to produce a simple quantitative estimate of how "optimal" a particular ordering of a set of requirements is from a re-test effort viewpoint.

We adopted this method in a number of randomized simulations for the communications requirements of Sys. A single run of the simulation would randomly order the requirements and then determine the overall dependency count. For each of the five communications interfaces of Sys we carried out the following:

- A randomized simulation of 1000 orderings of the requirements, stratified according to the nineteen subcategories of the new framework.
- A randomized simulation of 1000 orderings of the requirements, stratified according to the V-Model test phases of: review, unit testing, integration testing, system testing and user acceptance testing.
- A simulation of 1000 randomly generated orderings of the requirements.

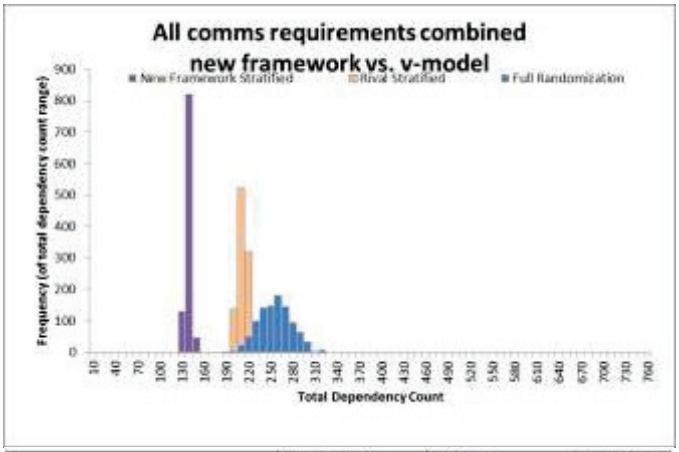
The term "stratified" in this context means that randomization was carried out within each of the nineteen test subcategories or five test phases, but the sequence of the subcategories and phases was maintained. The results of the simulations were

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE

represented using graphs similar to the one shown in Figure 3. The X axis, "Total Dependency Count", gives the overall dependency count as explained earlier. The Y axis, "Frequency", is the number of simulations whose total dependency count falls within a particular range.

The sets of results for an interface were compared visually as well as using ANOVA analysis. For each of the five interfaces, the new test framework was more efficient than both the V-Model (Shown as "Rival" in the graph) and the fully random simulation. There were variations between the five interfaces but all differences were significant at 95% confidence according to the ANOVA analysis.

Finally, we combined all communications requirements in one set and repeated the simulations for this combined set; the results are presented in the graph in Figure 3. The new framework clearly has lower dependency counts than both the V-Model and fully random, i.e. is more "test efficient" than both. The resulting ANOVA analysis data is presented in the table in Figure 3.



	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	16.55	-2.69	Different

Figure 3: Results of the Simulation

VI. CONCLUSION

The test framework made what otherwise could have been technically complex and fragmented test analysis and design work relatively objective and simple. It made it easy to derive a well prioritized set of test cases that would have otherwise required specialist technical knowledge of communications technologies generally and of the requirements of the specific CCLSS project as well as its detailed technical design.

The adoption of the framework can also lead to indirect benefits for an overall IT project developing a system comparable to CCLSS, i.e. not just for the testing activities. For example, by reducing the ambiguity of the requirements and improving the objectivity of testing and QA, the adoption of the framework early on during an IT project can reduce the potential for disagreements between IT users and vendors.

Through this work, we demonstrated how a new test framework can be devised and then applied to produce an objective requirements-based test design for a communications-critical large scale system, we then evaluated the outcome. We also presented a new method for how "test efficiency" can be estimated and how two test frameworks can be compared. We gave an example of how requirements and test design can be mapped and how both activities can be aligned [18] to lead to more effective testing, and ultimately to better quality CCLSSs.

Future work may include one or more of the following: defining the layer-specific test approaches (both functional and non-functional) for the remaining five layers, developing further the use of simulation to evaluate the prioritization benefit of the new test framework (Section V), devising notations and formats to support more precise definition of the test cases and possibly the automatic generation of the test cases, and incorporating the ideas of this research into, or as an extension of, one of the existing established test or software engineering standards. Last but not least, the ideas of the new framework need to be trialed further via an industrial collaboration program covering more real-life CCLSS case studies and involving real-life defects and re-testing metrics.

REFERENCES

[1] A. Bertolino, Software Testing Research: Achievements, Challenges, Dreams, *Future of Software Engineering (FOSE'07)*, IEEE Computer Society, 2007

[2] S. Bullock and D. Cliff, Complexity and Emergent Behaviour in ICT Systems, *Hewlett-Packard Company*. October 2004

[3] Ammann, P., & Offutt, J. (2008). Introduction to Software Testing. Cambridge University Press

[4] Susan D. Shaye, "Transitioning a Team to Agile Test Methods," *AGILE Conference, pp. 470-477*, Agile 2008, 2008

[5] IEEE Std 829-2008, Standard for Software Test Documentation, 2008

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE

[6] S. C. Reid. 2000. BS 7925-2: The Software Component Testing Standard. IEEE January 2000

[7] IS0/IEC 12207, Information Technology - Software life cycle processes, 1995

[8] IEEE Std 1012-2012, Standard for Software Verification and Validation Plans, 2012

[9] Thamm, J. and Kollmar, C. 2004. Assessments according to the ISO 15504. *Information Management & Consulting* 19, 62-9

[10] ISO/IEC/IEEE 29119 software testing standard http://softwaretestingstandard.org/ (March 2014)

[11] The Challenges of Complex IT Projects, British Computer Society, *Royal Academy of Engineering*, April 2004

[12] J. A. Zachman, A Framework for Information Systems Architecture, *IBM Systems Journal*, Vol. 26, No 3, 1987, pp. 276-292

[13] ETSI Conformance Testing http://www.etsi.org/WebSite/technologies/ConformanceTesting.aspx (March 2014)

[14] Zimmermann, H. (1980). *OSI Reference Model - The IS0 Model of Architecture for Open Systems Interconnection.* IEEE Transactions on Communications, 28(4), 425-432.

[15] ITU-T. (2000*). Overview of TMN Recommendations: Recommendation M.3000.* ITU

[16] Kelly, M. B. (2003). The TeleManagement Forum's Enhanced Telecom. *Journal of Network and Systems Management*, 109-119.

[17] Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2002). Test Case Prioritization: A Family of Empirical Studies. *IEEE Transactions on Software Engineering*, 28(2)

[18] 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.

Authors

Rob Hierons received a BA in Mathematics (Trinity College, Cambridge), and a Ph.D. in Computer Science (Brunel University). He then joined the Department of Mathematical and Computing Sciences at Goldsmiths College, University of London, before returning to Brunel University in 2000. He was promoted to full Professor in 2003.

Contact:
Email: rob.hierons@brunel.ac.uk
Tel: +44 (0)1895 266002

Mohammad Nabulsi is a test practitioner with over 20 years' experience in the field. He was recently awarded his PhD in Computer Science Research by Brunel University, the thesis for which is reflected in this paper. He holds an MSc in Telecommunications from University College London and a BSc in Computer Science and Business Studies from the University of Buckingham. He is also a member of the British Computer Society.

Contact:
Email: mnabulsi@transit.co.uk
Tel: +44 (0)7973 418905

Digital Object Indentifier 10.1109/MS.2014.53 0740-7459/$26.00 2014 IEEE