SOFTWARE COMPONENT TESTING - A STANDARD AND THE EFFECTIVENESS OF TECHNIQUES

STUART CHRISTOPHER REID

A submission presented in partial fulfilment of the requirements of the University of Glamorgan/Prifysgol Morgannwg for the degree of Doctor of Philosophy

December 1997

SOFTWARE COMPONENT TESTING -

A STANDARD AND THE EFFECTIVENESS OF TECHNIQUES

Stuart C. Reid

ABSTRACT

This portfolio comprises two projects linked by the theme of software component testing, which is also often referred to as module or unit testing. One project covers its standardisation, while the other considers the analysis and evaluation of the application of selected testing techniques to an existing avionics system. The evaluation is based on empirical data obtained from fault reports relating to the avionics system.

The standardisation project is based on the development of the BCS/BSI Software Component Testing Standard and the BCS/BSI Glossary of terms used in software testing, which are both included in the portfolio. The papers included for this project consider both those issues concerned with the adopted development process and the resolution of technical matters concerning the definition of the testing techniques and their associated measures.

The test effectiveness project documents a retrospective analysis of an operational avionics system to determine the relative effectiveness of several software component testing techniques. The methodology differs from that used in other test effectiveness experiments in that it considers every possible set of inputs that are required to satisfy a testing technique rather than arbitrarily chosen values from within this set. The three papers present the experimental methodology used, intermediate results from a failure analysis of the studied system, and the test effectiveness results for ten testing techniques, definitions for which were taken from the BCS/BSI Software Component Testing Standard.

The creation of the two standards has filled a gap in both the national and international software testing standards arenas. Their production required an in-depth knowledge of software component testing techniques, the identification and use of a development process, and the negotiation of the standardisation process at a national level. The knowledge gained during this process has been disseminated by the author in the papers included as part of this portfolio. The investigation of test effectiveness has introduced a new methodology for determining the test effectiveness of software component testing techniques by means of a retrospective analysis and so provided a new set of data that can be added to the body of empirical data on software component testing effectiveness.

Contents

INTRODUCTION

This is the overview document for a Portfolio PhD in 'Software Component Testing'. The portfolio covers two projects: one on the standardisation of software component testing and the other on the effectiveness of the techniques used in software component testing.

Within this document 'software component testing', or more simply, 'component testing', is considered to be the dynamic testing of software components, where a component is defined as a minimal program for which a separate specification is available. Component testing is also known as unit or module testing, and although these terms are often used to refer to the testing of larger programs, made up of a number of components, this is not the case here, where component testing is solely considered to be that dynamic testing at the bottom of the 'V' life cycle model. The terms 'component' and 'module' are used interchangeably in this document.

The first project considered is the standardisation of software component testing ('the Standard'). This project was based on the production of the BCS/BSI Software Component Testing Standard [4]. This was conceived in early 1989 with the aim of it being the first of a number of software testing standards developed by members of the BCS Specialist Interest Group in Software Testing (SIGIST). After over eight years of development, copyright for this document, and an associated Glossary of terms used in software testing [5], was assigned to the British Standards Institution (BSI) in April 1997, allowing them to edit and publish it, and hopefully subsequently present it to the International Organization for Standardization (ISO) for 'fast-tracking' to international status.

The second project documents what has been described as a 'retrospective analysis' [17] to determine the relative effectiveness of a number of testing techniques when applied to components from an operational avionics system ('the Investigation'). This study has generated a number of interesting results, which, although obviously specific to the analysed system, raise questions about the rationale on which test techniques (and their corresponding test completion criteria) are currently chosen.

This overview document first presents a list of the documents that make up the portfolio, followed by a brief rationale for performing component testing, and then considers how the two projects are related. Next, the two projects are considered in more detail, by reference to the documents in the portfolio, commentary on the projects, recommendations for further work, and conclusions. In the case of the standardisation project a detailed description of the author's role in the project is provided. Finally, the contribution to knowledge of this portfolio is presented.

1. ACRONYMS

ANSI	American National Standards Institute
BCS	British Computer Society
BSI	British Standards Institution
DoD	(US) Department of Defense
DPC	Draft for Public Comment

ESA	European Space Agency
ESSI	European System and Software Initiative
GQM	Goal - Questions - Metrics
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
IEEE	Institute of Electrical and Electronic Engineers
IPR	Intellectual Property Rights
ISO	International Organization for Standardization
LCSAJ	Linear Code Sequence And Jump
MOD	Ministry of Defence
MPSE	Master Plan for Software Engineering
MQG	Metrics - Questions - Goal
NWI	New Work Item
PET	The Prevention of Errors through Experience-driven Test Efforts
SEI	Software Engineering Institute
SIGIST	Specialist Interest Group in Software Testing
SMARTIE	Standards and Methods Assessment using Rigorous Techniques in Industrial Environments
URL	Uniform Resource Locator
WP	Working Party

2. COMPOSITION OF THE PORTFOLIO

The portfolio comprises the following eight documents:

- The BCS Software Component Testing proto-Standard [1]
- The Software Testing Standard - how you can use it [2]
- Popular Misconceptions in Module Testing [3]
- BCS/BSI Glossary of terms used in software testing [4]
- BCS/BSI Standard for Software Component Testing [5]
- Test Effectiveness in Software Module Testing [6]
- An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing [7]
- Module Testing Techniques - which are the most effective? Results of a Retrospective Analysis [8]

3. THE RATIONALE FOR COMPONENT TESTING

It is recognised best practice in software development to detect and remove faults at the earliest opportunity. This is due to the increasing costs of correction later in the life cycle, and supports the widespread quality goal of minimising rework. Component testing is *cost effective* as long as the faults it

detects (and which are then removed) cannot be detected and corrected more cheaply at a later phase of development.

It is often argued that the faults found by component testing could, just as easily, be found during integration testing. However, the combination of components into an integration 'build' makes for a complex piece of software that is difficult to comprehend, and thus test; one of the reasons that designers partition software into components is to ease understanding - not just for the coder, but also for the tester. The complexity of builds also makes the achievement of test coverage criteria more difficult - it is often difficult enough achieving the required coverage levels when testing components in isolation. Myers [46] presents two more reasons for component testing. He argues that component testing eases the task of debugging since, when an error is found, it is known to exist in a particular component. He also considers the project management perspective, pointing out that component testing introduces parallelism into the testing process by presenting the opportunity to test multiple components simultaneously.

Component testing has further potential benefits when maintenance and reuse are considered. Components are more likely to be reused if they have been tested in isolation, rather than as part of an overall system, as they are then much more likely to be correct in terms of meeting their individual specification (rather than performing as part of a larger system) and so reusable in a different context based on their individual specification. Maintenance is made easier when it is possible to retest modified components in isolation and this is obviously helped when the original test case suites are available.

Apart from the aforementioned reasons there is also a growing body of empirical data supporting the inclusion of component testing in the software life cycle. For instance, the ESSI PET Project Report [59] concluded that the lack of component testing in the projects studied was the second largest cause of bugs and that a quantifiable improvement resulted from its introduction with suitable tool support. Similarly, figures from Jones [44], show that it cost nearly twice as much to detect and fix faults during integration testing than during component testing, and more still in later phases, for the systems he studied.

Study in this area needs to continue, and the increasing popularity of metrics initiatives will hopefully provide the necessary data at little cost, but at present the arguments *for* component testing appear overwhelming.

4. THE RELATIONSHIP BETWEEN THE TWO PROJECTS

The projects are primarily connected by the examination in the second project (the Investigation) of the component testing techniques defined in the first project (the Standard). This has provided the following benefits:

- The application of the test case design techniques (as defined in the Standard) to real components in the Investigation provided feedback on their accuracy and ease of use. This led to both the definitions in the normative part of the Standard and the guidelines on their use being improved.

- Results from the Investigation on the test effectiveness of different test case design techniques provided evidence for the retention of techniques in the Standard (none of the techniques fared badly enough to warrant removal). More significantly, in the case of random testing, the results from the Investigation were used to justify its inclusion in the Standard when otherwise it would have been left out.
- The provision of definitions of component testing techniques by the Standard allows their consistent and repeatable use. Thus, by using these defined techniques in the Investigation, it was possible to be consistent in their application across components in the Investigation and, in the future, it will also allow the results to be safely compared with experiments using the same definitions. Similarly, practitioners basing their choice of component testing techniques on the results of the Investigation (or other experiments using the definitions in the Standard) can be sure of using the same techniques by using the definitions in the Standard. Until the Standard provided these definitions, the comparison of techniques and the ability to consistently repeat them were difficult as there was no guarantee that different testers meant the same thing by the same-, or similarly-named techniques.
- The Standard currently provides guidelines on how to choose test case design techniques and test coverage measures as an Annex. But, due to the lack of consensus in this area (consensus is a pre-requisite for standardisation), it provides no more than the subsumes ordering for a subset of the white box techniques defined in the Standard. Hopefully, the Investigation will provide a small amount of the data required to make up a cohesive body of knowledge on the effectiveness of test techniques that will allow future versions of the Standard to provide more comprehensive guidelines in this area.

5. THE STANDARDISATION OF SOFTWARE COMPONENT TESTING

5.1 Introduction

The documents related to the Standard are considered in chronological order. Document details and a synopsis are provided for each document and the original abstract and conclusions are included for the papers. Next, a commentary on the project is presented; this is not restricted to the documents alone, but also includes remarks on the overall progress with the Standard and Glossary. The Standard and Glossary were produced by a Working Party (WP) formed by volunteers from the BCS Specialist Interest Group in Software Testing (SIGIST), and a section detailing the input of the author is provided. Finally, recommendations for further work in this area and conclusions are presented.

5.2 Aims and objectives

The aim of this project was to produce a standard on software component testing to fill a perceived gap in both the national and international standards arenas.

The objective was a standard that enabled the measurement and comparison of testing performed on software components to enable users of the Standard to directly improve the quality of their software testing, and improve the quality of their software products.

5.3 Documents related to the Standard

5.3.1 The BCS Software Component Testing proto-Standard [1]

5.3.1.1 Document details

This paper was presented at, the 2nd International Conference on Software Quality Management, Edinburgh, in July 1994, and appears in the proceedings of the conference.

5.3.1.2 Abstract of paper

Software testing should provide both a means of assessing the quality, and a degree of confidence, in a software product. However, the actual quality of testing carried out by different organisations, or expected between customers and suppliers, varies alarmingly. Standards currently available fail to define any test design techniques and provide little, if any, means of measuring and comparing the quality of testing performed on software. Without these attributes the user cannot improve the quality of testing undertaken and, hence, the quality of the software product. This paper describes the Software Component Testing proto-Standard, which has been developed to fill this gap by defining a generic test process, test case design techniques, and a set of objective measures for carrying out dynamic testing of software components, as well as providing guidelines on their use.

5.3.1.3 Conclusions of paper

The experiences involved with development of the software component testing proto-standard have been described and the similarities between the development methodologies for the proto-standard and for software have been discussed. It is suggested that this analogy could be extended from project management to process improvement and a process maturity model.

The difficulty of choosing which test case design techniques and measures to use from the proto-standard has been addressed and it has been concluded that given current technology it is not yet possible, except in the broadest terms, to define best practice in this area. Therefore it has been left to the user of the proto-standard to choose the techniques and measures most suitable for their situation. The level of confidence afforded to users of the proto-standard is thus determined not only by conformance to the proto-standard, but also by this choice.

However, a number of other problems remain that also need to be investigated further. It can be argued that the test coverage measures are only objective for structural coverage and that functional coverage is subjective, making conformance checking difficult. The proto-standard also mandates that some independence is achieved in the testing, but the choice of level is left to the user. Finally, the quality of

specifications is outside the scope of the proto-standard, but its affect on the product of component testing needs to be quantified.

5.3.1.4 Synopsis

This paper defined the attributes of a 'good' software engineering standard and attempted to demonstrate that the Standard had these attributes. It related the history and development of the standard from its conception in 1989 up to early 1994, and optimistically predicted a delivery to BSI in mid-1994. This date was especially hopeful given the requirement, identified in the paper, to perform both informal and formal reviews prior to the delivery.

Analogies between the software development process and the development of software standards were drawn. This was primarily to explain the move to a more disciplined development regime in early 1993, but also to support the proposition that ideas from software process improvement and the SEI process maturity model could be applied to the development of standards.

The paper described how, at that time, it was believed the Standard could form part of the new framework of software testing standards instigated by the IEEE in their Master Plan for Software Engineering Standards (MPSE). This belief, it was explained, led to the creation of the Glossary of software testing terms as a separate document (by extracting and expanding the definitions section from the main body of the Standard), which would be able to support all the proposed testing standards.

Technical issues were then addressed. Having considered the rationale for component testing, the scope of the Standard, as defined in the Standard itself, was considered item by item. The difficulty of choosing which of the test case design techniques and measures defined in the proto-standard was addressed and it was concluded that given the current state of the field it was not yet possible, except in the broadest terms, to define best practice in this area. The paper presented the subsumes hierarchies for both black and white box testing techniques, which were provided in the Standard at that time, as the best agreed basis for choosing techniques. The effect of the quality and form of component specifications on test quality was also considered and used to reinforce the point that the Standard would not be able to solve the problems of software test quality alone. Finally the appendix considered the poor coverage of component testing provided by a number of other standards.

5.3.2 The Software Testing Standard - how you can use it [2]

5.3.2.1 Document details

This paper was presented at the 3rd European International Conference on Software Testing, Analysis and Review, in London, November 1995, and appears in the proceedings of the conference.

5.3.2.2 Abstract of paper

The software testing community has, until now, lacked both standard definitions of test case design techniques and standard definitions of test coverage measures. This has led to the actual quality of

testing carried out by different organisations, or expected between customers and developers, to vary alarmingly. Standards currently available fail to define any test design techniques and provide few, if any, means of measuring and comparing the quality of testing performed on software. Without these attributes the user cannot improve the quality of testing undertaken and, hence, the quality of the software product. The BCS Software Component Testing Standard was produced to satisfy these shortcomings; it defines a generic test process, test case design techniques, and test coverage measures for use in component testing (otherwise known as unit testing).

This paper initially provides a brief introduction to this Standard, its history and current status, before concentrating on how the Standard is expected to be used. Two viewpoints are considered; that of the developer and that of the customer. Finally its relationship to other standards and the question of choosing test techniques and coverage measures are addressed.

5.3.2.3 Conclusions of paper

This paper initially described the development of the Software Component Testing Standard. It then went on to describe its structure and content, and suggested how it may be used, and considered it relationship with other standards. Finally the necessary, but sometimes difficult, choice of test case design techniques and measures was briefly covered.

The Standard is planned for release at Issue 3 at the EuroSTAR '95 conference and is freely available to potential reviewers from the author of this paper. The working party commend the Standard to your attention and look forward to receiving comments on its use.

5.3.2.4 Synopsis

This paper again initially provided the history and development of the Standard, but considered progress up to mid-1995, approximately 18 months further on from paper [1]. By this time the high overheads of reviewing each clause of the Standard had been discovered, with the paper citing four iterations of review and rework for a typical sub-clause. The paper also explained that the WP decided to use formal software inspection techniques on the Standard before Issue 2 was released at the EuroSTAR conference in November 1994. The consideration of comments from external reviewers and another formal inspection were described as final preparation for the planned delivery of the Standard to BSI in November 1995 at the EuroSTAR conference. Other planned work on the Standard was presented, such as the aim of increasing its audience and therefore its reviewers; this was to be achieved both by converting it to HTML for the Web and increasing publicity. The addition of further test case design techniques and measures was also described as planned.

The paper then explained how standards can be considered as 'agents for change' to assist organisations in improving their development process and how the Standard would perform this role for component testing in particular. It then presented the structure of the Standard and explained how the different clauses defined conformance to the Standard from both the perspective of the software component tester and the

perspective of the buyer of software component testing. Finally it considered, as in paper [1], the problem of choosing techniques and measures, using the subsumes ordering.

5.3.3 Popular Misconceptions in Module Testing [3]

5.3.3.1 Document details

This paper was presented at the 13th International Conference on Testing Computer Software, in Washington DC, June 1996, and appears in the proceedings of the conference.

5.3.3.2 Abstract of paper

Software module testing should provide a measure of confidence in the quality of the tested module. However, testing can only provide confidence if it is both well-defined and well-understood. In 1989 the definitions of test case design techniques and coverage measures were considered in need of standardisation, although the subject area was considered to be well-understood. It was thus believed to be eminently suitable for standardisation. This paper describes a number of the technical issues encountered during this process.

The British Computer Society Software Component Testing Standard is an attempt to define both the software module testing process and the techniques used within it to provide a basis for the definition, measurement, and comparison of software module testing. This Standard was presented to the British Standards Institution in November 1995 (after nearly seven years work) with the expectation that it will subsequently be accepted as an international standard.

One reason for the lengthy development period was due to the discovery that the standardisation of module testing raised a number of technical questions, the answers to which were not as "well-understood", as first thought. This paper describes a number of the technical problems investigated during the development by posing the following questions:

- *Why can't we measure Syntax Testing coverage?*
- *Is there a simple State Transition Testing measure other than Chow's?*
- *Why do testing tools provide different measures for branch and decision coverage?*
- *Why doesn't Branch Condition Testing subsume Branch testing?*
- *Is Modified Condition Decision Testing useful if it only applies at 100%?*
- *Does someone know which is the 'best' set of test techniques and measures?*
- *What code features can render the Subsumes Hierarchy useless?*
- *Is Equivalence Partitioning subsumed by Boundary Value Analysis?*

5.3.3.3 Conclusions of paper

This paper has described a number of the technical (rather than managerial) issues that arose during the development of the BCS Software Component Testing Standard. These highlight how the supposedly well-understood area of module testing is still open to various different interpretations.

The BCS Standard was intended to be the first of a framework of software testing standards, module testing being considered an ideal starting point due to its maturity. Efforts at standardisation, however, were constantly hampered by disagreements on what were considered to be the some of the most fundamental aspects. These disagreements highlight a problem with the standardisation of any mature, well-understood area - everyone has an opinion, and opinions held for long periods are difficult change. However, the disagreements also confirmed the working party's view that the area required standardisation.

The filling of this gap, by providing a basis for the definition, measurement, and comparison of software module testing, should have several benefits. Standards have a technology transfer role and the BCS Standard can improve knowledge of 'good' module testing practice. The definitions of the tcdts and tcms can provide both a contractual basis between suppliers and procurers, and repeatability that will makes the comparison of testing easier.

5.3.3.4 Synopsis

This paper, after introducing the Standard, described a number of the technical discussions on component testing that took place during its development. It explained how discussion of these issues and similar questions caused unexpected delays in the production of the Standard. The paper emphasised the fact that many of these topics were initially considered to be mature and well-understood, but, under the close scrutiny required during standardisation, were found to require closer analysis.

5.3.4 BCS/BSI Standard for Software Component Testing [4] and BCS/BSI Glossary of terms used in software testing [5]

5.3.4.1 Document details

The versions of these standards included in the portfolio are version 3.3 of the BCS Standard for Software Component Testing and version 6.2 of the BCS Glossary of terms used in software testing, both dated April 1997. These standards were produced by the BCS SIGIST Standards working party and are available from the author of this overview document.

5.3.4.2 Synopsis

These standards, which are 63 pages and 17 pages long respectively, were started in 1989 and finally delivered to BSI in April 1997. The Standard is intended for use by both those performing and those specifying software component testing. It covers the dynamic testing of software components, where a component is the smallest program for which a specification is available. The Glossary provides

definitions of software testing terms in general, although it is biased towards those terms used in software component testing, due to the feedback of definitions from the development of the Standard.

5.4 Commentary

One of the first decisions of the newly-formed WP in January 1993 was to follow a software-like development process for the Standard. Before any development work was begun, it was decided that the WP should determine whether there was still a requirement for the Standard. This involved an investigation into the currently-available software testing standards and the overall result was that there *was* still a requirement for a software component testing standard (some of the specific results were included in the appendix to paper [1]). No work in progress on such a standard was discovered either, although when the author attended the Software Engineering Standards Symposium in August/September 1993 he was informed about a new framework of software testing standards instigated by the IEEE. The Standard was shown to representatives from the IEEE at the Symposium and they agreed that it should, when complete, form a part of this new framework. Although the IEEE have been kept appraised of progress on the Standard, their initiative appears to have disappeared.

In following an engineering approach to the Standard's development, a requirements document [9] and PERT chart [10] were produced. The requirements were found to be very useful in deciding later arguments on the inclusion of new clauses, while the PERT chart was only really useful in determining dependencies between activities as the enforcement of deadlines was not possible. Overall the application of traditional management techniques to the voluntary group work involved in producing the Standard was found to be impractical, and the prime means of motivating members of the WP to meet deadlines was the (somewhat empty) threat of passing the work on to someone else.

Prior to the formation of the WP, which introduced defined roles for its members and a constitution, the development of the previous versions of the Standard had been relatively uncontrolled. Small groups or individuals would work alone on areas they were interested in, and create sections (properly known as clauses in a Standard) that were neither independently checked, nor planned as part of an integrated document. The formation of the WP and the new development approach were mainly as a result of the recognition of these problems - that an integrated document was required and that this document must reflect the consensus of all those working on it.

A problem arose in the use of the previously-produced definitions of techniques because they had not been widely reviewed among the WP and so no consensus had been reached. Papers [1] and [2] described the work pre-1993 as prototyping and, as is known in software development, the delivery of prototypes is fraught with difficulties. This was the case here, and the discussion of material from pre-1993 used up much time and resulted in many rewrites, much of the time necessitating starting again from scratch. In general, it was found that those techniques that were considered the most well-known and well-understood generated the most debate (and therefore took the longest to define), probably because in these cases all

the members of the WP had an opinion. In fact, a considerable proportion of *this* discussion was wasted due to a foreseeable problem that arose during the production of the guidelines clauses of the Standard.

It had been decided that all the guidelines should include examples. For each test case design technique its application to an example component was documented, and for the test process it took the form of a set of example documentation describing the test process for an example project. Although the definitions had been reviewed and considered at length before the writing of the corresponding guidelines clauses began, in *every* case the definitions had to be changed due to feedback from this task, including the original definition of the component test process. In several cases discussion of the guidelines highlighted ambiguities in the corresponding definitions. Often, where a definition had eventually been agreed after much discussion, the example, demonstrating a particular interpretation of this definition, re-started debate over the original definition. In hindsight, the definitions and guidelines clauses should have been produced together as this would have avoided lengthy debates on the finer points of definitions that were later drastically changed due to the feedback from producing the corresponding guidelines clauses.

Paper [1] reflected the over-confidence and over-ambition of the Working Party (WP) before the aforementioned problems had been recognised. Having moved from what was perceived to be a chaotic development process into a defined development process, and with a large number of the test techniques and test coverage measures already defined, the WP believed that delivery to BSI was imminent. In fact paper [1] had already identified the lack of suitability of material produced prior to 1993 and, secondly, the requirement to evaluate, or test, the Standard before its delivery - but it had not been realised how long this would take. Given that the WP was comprised mainly of testing specialists, and that the analogy with the software life cycle had already been accepted, this was a major oversight. By the time paper [2] was written the problems of performing reviews by a committee that met, at most, once a month had been recognised. Even though a core group within the WP attempted to speed up the process by reviewing, and rewriting drafts between meetings, their efforts were often in vain. This was because there were inevitably some members of the WP who insisted on being allowed to review all changes that had been made, but who would only review material at actual meetings (and often only attended every other meeting). Reviewing and re-drafting out of session were normally carried out via email (if participants had this facility) and orchestrated by the author.

Prior to publication of a new version of the Standard it was formally inspected by the WP. The use of these formal inspections was a painful experience for those members of the WP who had not used them before. The number of issues arising and the subsequently calculated number of errors remaining in the Standard seemed high, but we were assured they were quite normal by Dorothy Graham, who led the inspections and is an author on the subject. The main problem that arose from these inspections was the short timescale available to address the issues before that version of the Standard was published.

Paper [2] was written as a result of changes made to the Standard from comments received from *external* reviewers. It was pointed out that the current version of the Standard did not tell them how to determine whether they (or any other user) had conformed to the Standard. The WP had been so involved in

defining the test case design techniques and measures (and their corresponding guidelines) that no-one had considered this fundamental point, most assuming that conformance had already been built-in to the structure of the Standard. It was a fairly easy task to add the text to define conformance, but this did require the inclusion of clauses to allow users to define their own techniques, as long as they were defined in a similar manner to those already in the Standard. Paper [2] attempted to show how conformance to the Standard would encourage users to move to better component testing practices.

Changes were also made to the Standard due to feedback from attempting to use the definitions for equivalence partitioning (EP) and boundary value analysis (BVA) in the Investigation. To allow the comparison of results of the application of these techniques to various components (and to be able to expect future use of them to be similar) required that the interpretation of the definitions be consistent. It was found during the Investigation that there were a number of possible interpretations of the definitions and that these different interpretations yielded values of test effectiveness that varied substantially (this area is considered in some detail in paper [7]). Feedback of these results by the author to the WP led to the guidelines clauses for EP being modified to explicitly describe the possible range of interpretations. For some unknown reason the corresponding guidelines clause for BVA was not similarly changed.

To further widen the audience of external reviewers, the author began a project in the Summer of 1995 to put both the Standard and Glossary on the Web. This was accomplished in Spring 1996 with the assistance of the Computing Service at RMCS, who employed a student over the Summer of 1995 to help translate the Word document to HTML (the main difficulty was the representation of the many tables in the HTML document, which were included as images). The Web page also allowed visitors to download the latest versions of the documents in Word format. The author is still responsible for this Web page, although the Standard itself has had to be removed at the insistence of BSI, who felt they would not be able to sell a Standard that was also freely available on the internet.

Changes continued to be made to the Standard until April 1997 when it was finally handed-over to BSI; these were mainly due to internal reviews and external comments. The one major exception to this was the inclusion of a new test case design technique in the Standard - random testing. Initially there had been a reluctance to include random testing in the Standard as no-one on the WP had much experience of using it and there was little published evidence of it being used successfully in a commercial environment. The author, however, presented results from the Investigation to the WP demonstrating its test effectiveness, and, based on this, random testing was included.

The idea for paper [3] grew from a presentation made at the Software Engineering Standards Symposium in Brighton in 1993 [34]. This presentation pointed out that much of the underlying technical discussion by working parties developing standards was never properly documented, and so the results were prone to misunderstanding and the discussion prone to repetition by future working parties. The experience of the WP supported this argument as a number of the topics covered in this paper had to be addressed a number

of times due to members missing discussions the first time, new members joining the WP, and members forgetting the first discussion (due, perhaps, to the long intervals involved).

Two of the problems described in paper [3], of the exclusion of syntax test coverage from the Standard and the use of Chow's switch coverage for state transition test measurement, highlight a problem the WP encountered throughout. This was due to a self-imposed rule that only techniques that had been published elsewhere could appear in the Standard. This rule was intended to stop the WP 'inventing' techniques and thereby only include in the Standard publicly-available techniques, so enforcing a measure of consensus in the Standard. This was a difficult rule to apply, especially when members of the WP had experience of applying 'better' versions of the techniques than those that had been published. This was frustrating as the available published material was often quite old (for instance, Chow's switch coverage dates from a 1978 paper [23]) and published material on several techniques appeared to have stagnated since Myers' definitive work [46] of 1979. The solution was to add notes to the relevant guidelines clauses, as was done for the negative testing of state transition diagrams.

Probably the longest-running and unexpected discussion from the Standard's development is presented in paper [3] and arose from what was considered by the WP as the best-understood test coverage measure: branch coverage. The problem first came to light when the author noticed that example branch coverage values in the guidelines clause appeared to be wrong. The clause had been produced by the representative on the WP from IPL, a testing tools vendor, who had checked the figures using one of their tools. The discrepancy was due to one value being a measure of decision outcomes exercised (which is how most testing tools measures decision coverage), while the author's measure was based on the proportion of arcs on a directed graph of the component that were exercised. As described in the paper, these can differ at values of less than 100% coverage. Further investigation by the author uncovered more complications using the directed graph model, as different ways of drawing the directed graph resulted in different graphs that could lead, in turn, to different branch coverage values. It was eventually decided, due to the wide use of the different measures, to include both branch and decision coverage in the Standard.

When papers [1] and [2] were written both functional and structural techniques were included in the subsumes hierarchies (both in the paper and in the Standard). It was subsequently discovered that that the definitions agreed for equivalence partitioning and boundary value analysis did not support the subsumes relationship; this is described in detail in paper [3]. In addition to the subsumes ordering, paper [3] also presents a number of other factors to consider when making the choice of techniques and measures. This topic is obviously an integral part of the Investigation and is covered in more depth in section 7 of this overview document.

From its very inception, it was intended that the Standard should become an international standard. The WP understood that this must be achieved by the Standard first becoming a British Standard, via BSI. The first step to becoming a British Standard is to get the standard accepted by the relevant BSI working group (in this case IST/15) as a New Work Item (NWI). Each NWI needs a sponsor on the BSI working group, but for the Standard this presented no problem as Martyn Ould, who had first suggested the Standard in

1989, was a member of IST/15 and gladly took on this role. The Standard was accepted as a NWI in spring 1996, less than six months after it was presented to BSI, but it then took approximately a year before BSI were ready to take delivery of the WP's final version of the Standard in early 1997. This version was then edited to the format required by BS0 [18], and will be distributed as a Draft for Public Comment (DPC), to elicit comments to be submitted by the end of December 1997. These comments will be considered by a technical panel, which, it was suggested by BSI, would be made up mainly of volunteer members from the WP, before any final revisions are made and the Standard published as a full British Standard.

Once BSI decided to publish the Standard themselves they approached the WP and requested that they remove it from the Web and assign them copyright of both documents. This would then allow BSI to publish and sell the finished Standard. The assignment of copyright caused some problems for the WP as several members used it in their work and wished to continue to be able to distribute copies of it both within their own organisations and to their clients. They felt that as they had produced it, they should continue to be allowed to copy and distribute it. There was also reluctance to remove the Standard from the Web as the WP felt that it should be as widely available as possible. BSI's position was that as they were now an agency, and had to make a profit, they must sell standards and they could only do this if they owned the copyright (this would also allow them to protect it from illegal copying, if necessary). The WP discussed this problem and eventually decided that the overriding goal should be to advance the Standard to the status of an international standard and therefore the demands of BSI should be accepted. At one point it was suggested that the Standard could still achieve this aim if it was taken up by ANSI, via the IEEE, but as contact with the IEEE had been one-way for the previous couple of years (the WP sent them new versions, but heard little in return), it was agreed to remain with the BSI route. BSI have stated that, after the Standard is an official British Standard, then they will propose it for fast-tracking by ISO to make it an international standard.

Having decided what to do in principle, the actual assignment of the copyright to BSI proved somewhat complicated, as BSI gave the appearance of having never done this before. Assignment agreements drawn up by BSI lawyers contained so many errors and inconsistencies that the author finally rewrote them, had them checked by an IPR barrister, and these were eventually used. This has meant that although BSI now own copyright of the final version delivered to them on April 29th, 1997, the WP still retains the copyright for all previous versions (including a version produced on April 28th 1997, that differs from the final version only in that the copyright notices have been changed). Although the Standard was removed from the Web, the Web page directs enquiries to the author, who emails copies of the version dated April 28th 1997 to anyone who wants it. According to the agreement BSI must also publish the Standard within a year of the assignment, otherwise copyright reverts to the WP - this was to ensure that BSI did not 'sit' on the Standard so that it was lost from public view for too long.

Added complications arose during this period of negotiation with BSI. Firstly the MOD approached the WP, asking if they could make the Standard a Defence Standard until it became a British Standard (apparently MOD can only specify standards in contracts that are either official national, international, or defence standards). The WP had no problem with this, and, having checked that BSI felt the same, passed the contractual negotiations over to BSI. Secondly, both the BCS and the BCS SIGIST, decided individually, at this time, that they had the right to publish and distribute the Standard. The WP felt that both these claims were spurious and invited both groups to a meeting to discuss copyright (along with the MOD and BSI), which they failed to attend.

Paper [1] suggested that applying process improvement techniques from software development to the development of standards would require positive action by the standards bodies, such as BSI. Unhappily this has not been forthcoming, and the change of BSI from a public body to an agency required to support itself seems to make any initiatives in the near future unlikely. This makes it all the more important that the experience of the WP is widely disseminated.

5.5 Relationship to other work

A number of existing standards covered software component testing to different degrees when work on the Standard first began, but it was felt that there was no useful software component testing standard. As stated earlier, one of the first tasks of the WP was to research the coverage of component testing in other standards in order to determine that there was still an actual requirement for the Standard to ensure that it did not duplicate material already available. Among others, the following standards were considered: [13], [14], [15], [20], [24], [25], [37], [38], [39], [52] and [53]. The result of this research confirmed the original belief that there was still a requirement for the Standard. This position was supported by Wichmann, who stated that the use of ISO-9001 for software development would encourage a move towards software test measurement and its standardisation [66] and specifically suggested that the Standard [4] could fulfil the role identified [65].

Estimates of the number of software-related standards vary from "more than 300" [58] to "over 1100" [67], but there is no disagreement that within software standards there is duplication, inconsistencies, overlap and omissions. In 1987 IEC and ISO formed a joint technical committee to attempt to impose some structure in this area. The result is the proposed JTC1/SC7 Architecture [58]. One aim of this architecture is to position software engineering standards in relation to quality systems, concentrating on the ISO 9000 series [68]; the Standard [4] was produced with the expectation that it would be used to specify the software testing requirements of an ISO 9000 quality system. The JTC1/SC7 architecture is based on a process foundation of the software life cycle processes standard [40]. At the next level, users will require standards to support particular phases of the life cycle, such as testing. There are a number of standards that concentrate solely on testing including [13], [14], [15], [19], [36] and [42]. As stated earlier, however, none of these covers component testing satisfactorily, leading to the requirement for the

Standard [4]. It is too soon to know how successful this particular architecture will be, but it demonstrates where the Standard fits into such a framework of standards.

An orthogonal set of software standards that include software testing requirements are the application-specific standards, such as [25], [26], [35], [37], [45], [49] and [55]. These standards define software requirements for particular areas, such as avionics or railway signalling systems, and specify component testing requirements, but without defining the specified techniques. Any updates to such standards or new application-specific standards should specify and component testing requirements by referencing the Standard [4].

As part of the four year SMARTIE project, Fenton [28] reported on a measurement-based approach to assessing software engineering standards. He stated that it was impossible to make an objective assessment of conformance to the majority of existing standards and went on to make recommendations on the future writing of standards. When reporting on software testing standards [29] he stated that the Standard [4] was "as close to a true engineering standard as any seen by SMARTIE".

The ISO/IEC Directives [51] state that "*The same term shall be used throughout each standard or series of standards to designate a given concept. The use of an alternative term (synonym) for a concept already defined shall be avoided.*". An IT terminology standard [41] is available, which is made up of 22 parts, for a wide range of areas within the IT field. Unhappily it does not include software testing. No other standard has been found that supplied a suitable range of definitions (that were considered to be correct) and so the Glossary of terms used in software testing [5] was produced.

5.6 The author's role in the development of the Standard and Glossary

Papers [1], [2], and [3] were produced solely by the author, however the Standard [4] and Glossary [5] were produced by the WP as a whole. The activities of the author as part of the WP are listed below. The roles of Secretary and Chairman of the WP were only assigned from January 1993 onwards.

- Member (1990 to present).
- Secretary (7/1/93 to 16/2/95).
- Chairman (16/2/95 to present).
- Editor (Glossary 7/1/93 to 15/4/94 and 17/10/96 to present).
- Editor (Standard, in part from 16/11/92, and completely from 17/3/95 to present).

Since the WP started meeting regularly with a defined constitution and roles in January 1993, the author has attended all 35 meetings. For these, although he was Secretary for only about half this time, he produced and distributed the minutes for 31 meetings. Of the actions detailed in these minutes, the author was responsible for 32%.

Since January 1993 the author has been a contact point for members of the public to request copies of the Standard. Cranfield University has funded the copying and distribution of individual requests in this

period, which the author handled, although bulk runs for conferences were funded by the BCS SIGIST. More recently the majority of copies have been sent using the internet (for instance, in the last year the author has handled over 500 email messages on the subject of the Standard).

5.7 Recommendations for further work

Although mention has been made of a new framework of software testing standards instigated by the IEEE as part of their Master Plan for Software Engineering Standards this initiative appears to have disappeared. When work on the Standard first started the intention was that it would be part of such a framework (although the time taken to produce a standard was then thought to be much shorter!). If the IEEE initiative has failed then a new one needs to be set up. The WP, when it has finished the Standard, should not be dissolved, but move on to work on another testing standard within this framework (although the membership of the WP will, of course, continue to change). This will hopefully allow the expertise built up in the WP, most especially in the areas of procedure, to be maintained. The new framework would be expected to accommodate standards on integration, system and acceptance testing, as well as application-specific testing standards, such as GUI testing, client-server testing, internet/intranet testing, etc.

Members of the WP have already suggested that once the British Standard is published there will be a requirement for a book providing extra guidance on the Standard including examples of its use for various application areas (and this would presumably be amenable to translation into a hypertext version). Similarly there should also be a requirement for guidelines on auditing against the standard.

The WP wasted much time and effort in moving from what appeared to be a chaotic development process to a defined process. Time was also wasted by both BSI and the WP, in the hand-over of the Standard to BSI. In the author's experience, writers of standards appear to lack experience in each of these areas of the standardisation process and material on them is not readily-available. Even the structure and processes of BSI are poorly-understood by those outside the organisation. There is a requirement for information on writing standards, and specifically British standards, to be made widely-available for future standards work. This material *should* be provided by BSI.

As soon as the Standard becomes available as an official British Standard, then it should start to be used as the definitive source of software component testing definitions. At present many application-specific standards mandate that component testing is performed but rarely define the requirement clearly. A study should be performed to identify where component testing is specified in other standards. Where this occurs, in both new and existing standards, the authors of these standards could then be made aware of the Standard so that their documents may refer to it, rather than attempt to define the required component testing themselves.

5.8 Conclusions - Standard

The Standard, once released by BSI, should fill a gap in the existing software engineering standards framework, and will hopefully go on to become an international standard. It has been reviewed and tested

both internally, by the WP, and by external reviewers. As with all standards it is not perfect - any product of the standardisation process, which is driven by the need for consensus, cannot be perfect - but in comparison to the majority of draft standards it is of particularly high quality. It is already being used commercially by several major organisations, including government agencies, the banking sector, software houses, etc. The MOD also wish to use it to specify their component testing requirements.

Whether the objective of producing a standard that improves test quality by enabling the measurement and comparison of testing has been achieved remains to be seen. More feedback is required from users of the Standard to confirm this. Although the response of early users is wholly positive, it is also presumably subjective, as no empirical data on its use has been published. That the Standard has the potential to meet this objective seems to be confirmed by the results of the SMARTIE project [29].

The Standard has taken a long time to produce and during the first four years of development, before a systematic approach was taken, much effort was wasted. Only after the adoption of a structured development process, similar to that used in software development, was real progress made. The move from chaotic development to a more defined process was difficult, but hopefully the experience of the WP will not be lost and the lessons learnt applied to any future projects undertaken. It is assumed that those working under the auspices of BSI are given more guidance in this respect, but there certainly remains a requirement for more information in this area, and, as the national standards body, this should be BSI's responsibility. BSI would also help the developers of standards (who are not working as part of BSI) if they provided more information on the standardisation process itself - that is, the procedures by which a new standard eventually becomes a national and international standard. The WP were given the impression that they were the first group to propose a new standard from outside BSI, which was not the case. The difficulties that arose over the copyright of the Standard were typical of the relationship between the WP and BSI.

The WP was made up of voluntary members, who received no recompense nor expenses for the time spent working on the Standard. Meetings were generally held monthly in London and for most of those attending a fair proportion of the time they gave up was spent travelling. The author believes that the increasing availability of communication across the Web opens up new opportunities in the development of standards. At present it is possible to communicate quickly and cheaply using email, allowing the easy dissemination of minutes and changes for review. Although actual meetings are currently still necessary, when videoconferencing becomes more widely available then virtual meetings may be held using this facility. This will allow the WP to include a wider range of members, as those who are unable to afford the time to travel to meetings will now be able to take part, and participants may be located anywhere in the World, not just within travelling distance of London. The Web will also allow Standards to be reviewed by a potentially vast audience, thereby making them true consensus standards.

The Web can affect a more fundamental change to the standards arena. Standards should serve the common good, and so the more widely-available they are, the better. The Web provides an opportunity to

make standards much more widely accessible, but, presently, standards organisations have prevented its use for this means as they will thereby lose a source of income. Although these bodies have an argument for selling standards whose production they have funded, it seems unreasonable that, as a monopoly (the national body alone can put forward standards to ISO), they should be allowed to charge for standards produced by others at no cost to themselves. The author hopes that the Standard will some day return to being freely available on the Web, along with other standards.

6. INVESTIGATION INTO THE EFFECTIVENESS OF SOFTWARE TESTING TECHNIQUES

6.1 Introduction

The documents related to the Investigation are considered in chronological order. Document details and a synopsis are provided for each document, as well as the original abstract and conclusions. Next, a commentary on the project is presented; this is not restricted to the documents alone, but also includes remarks on the overall progress of the Investigation.

The basic aim of the Investigation was to investigate the effectiveness of different software component testing techniques and measures in order to be able to provide firmer guidelines on which techniques should be used. Many experiments and case studies have been performed with similar aims, but still there is no consensus on the result, perhaps due to the problems inherent in such activities. One of the main problems is basing such experiments on real data, which the author believed was imperative and so a company was approached to provide access to data from a commercial project. The author was given unlimited access to all the software and data related to the project and was also allowed to talk to the developers, while they were still working on the project.

The first of the three papers presents a high-level view of the Investigation, which, at that point, aimed to not only identify the relative effectiveness of different testing techniques, but also to identify those attributes of a module that would allow the most appropriate set of techniques to be chosen for each individual module. This paper also presents the results of the failure analysis of the studied system. The other two papers present the experimental methodology in depth and present relative test effectiveness results for a wide range of testing techniques. By this time, it was recognised that the original aim of matching a module's attributes with a set of techniques was not achievable as too little data was available, but that the Investigation would 'simply' provide relative test effectiveness values for the testing techniques studied.

6.2 Aims and objectives

The aim of this project was to identify those attributes of a module that would allow the most appropriate set of software module testing techniques to be chosen for each individual module.

The objective was first to perform a failure analysis of an operational system to identify those modules with faults that could have been exposed by module testing. The ability of selected testing techniques to expose these faults was then to be determined using a new methodology to determine the test effectiveness of the studied techniques. Finally, a correlation analysis was to be performed to identify those module attributes that could be used to determine the most effective testing technique for the module.

6.3 Documents related to the Investigation

6.3.1 Test Effectiveness in Software Module Testing [6]

6.3.1.1 Document details

This paper was presented at the 2nd European International Conference on Software Testing, Analysis and Review, in Brussels, October 1994, and appears in the proceedings of the conference.

6.3.1.2 Abstract of paper

Software developers would like to know which testing technique(s) to apply to a module, or unit, to provide the satisfactory level of 'testedness'. There have been several studies carried out into the relative effectiveness of various software testing techniques, which have generated contradictory results. Different standards require different test techniques to be used and test coverage levels to be achieved with no apparent basis for agreement.

There are several possible reasons for the conflicting results of such research. A number of experiments have used bug-seeding, where the criteria for error insertion will bias results and cannot produce a representative set of errors. The use of random test data to satisfy test coverage criteria or testers with varying levels of ability also present difficulties in comparing results both within and between experiments and in transferring results to the real world. It is the author's belief that a major reason for the non-agreement is that test effectiveness results are not necessarily universally applicable and will vary dependent on factors such as application domain and language. Even within such subsets the possible range of module functions could mean that test effectiveness results are applicable only to 'similar' modules. In fact, an aim of the research has been to identify a module's 'signature' (that could be based on design and/or code metrics) that can be used to identify the most effective test technique(s) for that module.

This paper presents interim results of research which attempts to address some of the above problems. First the research approach is described, which comprises three stages: Initially an error analysis is carried out to identify the system faults, followed by the generation of test cases from a range of test techniques and an analysis of their effectiveness in detecting the previously identified faults. Finally those module attributes that can be used to choose the 'best' testing technique are identified. The second part of the paper presents results derived from the first stage of this research, which is an error analysis of an avionics system implemented in Ada.

6.3.1.3 Conclusions of paper

This paper has performed two roles: Firstly it has presented an approach to determining a means of selecting the most effective TCDTs for a given module. Secondly it has presented interim results from the study so far, an error analysis of the case study project, an avionics system implemented in Ada.

The research approach can be considered as three stages. In the first an error analysis is carried out on an operational system to determine the faults in the system when module testing was first carried out. In the second TCDTs are applied to the original module specifications and code to generate test cases, which are then used, along with the fault data, to determine the relative effectiveness TCDTs for the system's modules. The final stage involves identifying attributes of modules that can be used to select the 'best' TCDT for a given module.

Error analysis is a proven, if time-consuming, means of identifying the faults in a system, and is preferable to seeding with artificial faults, which would lower the validity of the final results. Several papers have investigated the relative effectiveness of different TCDTs, but with little consensus. This paper has attempted to identify those factors that characterise software module testing and emphasises the difficulty of comparing study results from different environments. The use of standard definitions for the TCDTs should provide repeatability that will make comparisons within the study more valid and their similar use in industry easier. The varying capabilities of different TCDTs to detect certain types of errors is well-recognised and the ability to predict errors from design and code attributes is also feasible, so achieving the overall goal should be possible given enough time and data.

The error analysis of the case study project provided a number of interesting results, among which were:

- *The optimum size of modules varied depending on your viewpoint. Smaller modules exhibited higher error density, but when the number of modifications required to modules was considered (due to both errors and other changes), then small modules fared better than their larger counterparts.*
- *It was not possible to predict errors in modules, except in large specifications and bodies (that did not include subprogram code).*
- *The developers accounted for less than half the changes made to modules, the rest were due to requirements changes, standards changes, etc.*
- *15% of changes were explicitly made due to project standards changing in the course of development and being implemented retrospectively.*

The results of the case study are not representative of all software, and cannot be extrapolated to other environments, however the characteristics of this project are described in detail earlier.

Error analysis is very time-intensive, it would be far better to collect the required metrics at source (and automatically), but the requirement for a system with known real faults in it (ie an operational system) was considered more important. The lack of cost data for the original development has restricted

possible analysis in this area, although this is not considered of primary importance to this research. The two main lessons learned from the error analysis were that the limited amount of validation carried out was insufficient and that increased statistical rigour must be employed in future.

Finally it is hoped that despite the long-term process improvement role of this research the interim results can have a shorter term effect by exhibiting benefits of metrics collection and prompting development organisations to introduce metrics programs as soon as possible.

6.3.1.4 Synopsis

Paper [6] comprised two parts. In the first, the Investigation was introduced, while in the second the results of the first stage of the Investigation, a failure analysis, were presented.

The paper initially presented a high-level view of the proposed research to be performed. It started with the hypothesis that it was possible to improve the quality of module testing by identifying a way of choosing the test case design techniques and test completion criteria for a given module by an analysis of that module's specification and/or source code. It then went on to explain how influence diagrams from System Dynamics and the GQM approach were used to identify those measures required for the research. The difficulties of determining the effectiveness of testing techniques were then considered, before the overall experimental approach, described as a hybrid of experiment and case study, was presented. This first part of the paper concluded by explaining how the research could support a process improvement strategy that used the research results to improve the quality of module testing.

The second part of the paper reported the results of the failure analysis, and then considered what other conclusions could be drawn from the collected data. To do this the paper described a variation of GQM that starts with the metrics from which questions and then goals (in this case, representing any interesting results) are derived, referred to as MQG. The results from the failure analysis included values for the size and complexity distribution of modules, the fault distribution across module versions, the classification of faults and their distribution by type, and test coverage levels achieved. Analysis of the data provided a number of interesting insights that were included in the paper.

The paper concluded by stating that, following the failure analysis, two more stages of the research were required. The first stage would be to determine the relative test effectiveness of different test case design techniques when applied to the studied system's faulty modules. The second would be to identify the attributes of these modules that could be used to determine which test techniques would be the most effective, and it also stated that this goal should be achievable *given enough time and data.*

6.3.2 An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing [7]

6.3.2.1 Document details

This paper was presented at the 4th International Software Metrics Symposium, in Albuquerque, November 1997, and appears in the proceedings of the conference.

6.3.2.2 Abstract of paper

An experiment comparing the effectiveness of equivalence partitioning (EP), boundary value analysis (BVA), and random testing was performed, based on an operational avionics system of approximately 20,000 lines of Ada code

The paper introduces an experimental methodology that considers all possible input values that satisfy a test technique and all possible input values that would cause a module to fail, (rather than arbitrarily chosen values from these sets) to determine absolute values for the effectiveness for each test technique.

As expected, an implementation of BVA was found to be most effective, with neither EP nor random testing half as effective. The random testing results were surprising, requiring just 8 test cases per module to equal the effectiveness of EP, although somewhere in the region of 50,000 random test cases were required to equal the effectiveness of BVA.

6.3.2.3 Conclusions of paper

This paper has presented the results of a study into the test effectiveness of a number of black box testing techniques. The experimental methodology used here has differed from previous experiments as the derived test effectiveness is based on an analysis of all inputs that satisfy the test technique rather than arbitrarily chosen inputs from this set.

The results, in summary, are:

- *BVA was the most effective technique studied, achieving a highest mean probability of detection of 0.79, compared with 0.33 for EP. However, to achieve this, nearly twice as many test cases were required (13.6 for BVA , 7.6 for EP).*
- *For lower levels of effectiveness, random testing appears to have the advantage over EP. Just eight random test cases per module were required to achieve the same level of effectiveness as EP, and random testing will also be less expensive to perform as no test case design is required.*
- *Although random testing appears effective for achieving lower levels of test effectiveness, it requires a prohibitive number of test cases (50,000) to reach the levels achievable by BVA.*
- *Minimised BVA focuses testing more on the extremes of the input domain than one-to-one BVA; the study found little to choose between the two approaches.*
- *Minimised EP was consistently slightly less effective than one-to-one EP, although it required marginally fewer test cases. Again, there was little to choose between the two approaches.*

- *Using complementary techniques (rather than a single technique) appears sensible, and, if choosing a black box technique to complement branch testing, then BVA appears to be more effective than either EP, or random testing.*

Obviously the results of this study have taken little account of cost, except in general terms; measures could not be taken as the techniques were not used to detect faults, but rather to create test case sets. More study of the costs of applying testing techniques needs to be performed, although the relatively large differences in test effectiveness results presented here may well overshadow relatively small differences in the cost of their application.

A number of the faults were difficult to detect using any of the techniques studied (see Table I). A number of white box testing techniques, supposedly more effective than branch testing, warrant investigation and it is the intention to reuse the experimental methodology with these techniques to determine their relative effectiveness. The next step will then be to re-apply the methodology to other systems to determine if the results are system-specific.

As with all such experiments the results are characterised by the conditions under which it took place (described in Appendix A) and their extrapolation to other environments is not implied.

6.3.2.4 Synopsis

This paper introduced an experimental methodology for determining the effectiveness of module testing techniques and presented the results of its application to a number of black box techniques on an operational avionics system.

Initially the studied techniques were described, which included random testing and two variants of equivalence partitioning (EP) and boundary value analysis (BVA) - these were 'minimised' and 'one to one'. It explained that these variants were derived from the definitions in the BSI/BCS Software Component Testing Standard [4] and went on to provide an example to illustrate the differences between them. Next, the difficulties with experiments on test effectiveness and related work in the area were considered briefly before the experimental methodology was described.

The four stages of the methodology were described in some detail, having first presented the hypotheses, which consisted of five comparisons between the studied techniques. The equations used to calculate the test effectiveness were given along with an example calculation.

The probabilities of detection for each of the faults identified from the failure analysis were then presented for each of the studied techniques, along with 'means' and 'sums' for the techniques across all faults. The results of the statistical analysis were presented that showed whether there had been significant differences in the test effectiveness of the compared techniques. Each of the comparisons was then discussed in turn, followed by branch testing, the technique originally used by the developers.

Finally, some limitations of the experiment were presented, followed by conclusions and some recommendations on further work.

In summary, the paper comprised two major themes. The first was the presentation of the experimental methodology, which, it was explained, differed from other experiments in that *all* possible inputs that satisfied a testing technique were considered rather than arbitrarily chosen representative values from this set. The second was the presentation of the test effectiveness results. The most notable results were that BVA was by far the most effective technique studied and that the effectiveness of random testing was comparable to that of EP.

6.3.3 Module Testing Techniques - which are the most effective? Results of a Retrospective Analysis [8]

6.3.3.1 Document details

This paper was presented at 5th European International Conference on Software Testing, Analysis and Review, in Edinburgh, November 1997, and appears in the proceedings of the conference.

6.3.3.2 Abstract of paper

If you are module testing as part of software development then you face a predicament: Which techniques, do you use to design the test cases, and which measures of test coverage do you require to be achieved? Industry guidelines and standards often mandate the required testing and many developers believe they know the appropriate level of testing for their system. However, what is the basis of this knowledge? This paper argues that we are currently basing our testing decisions on unsafe foundations and presents results from an empirical study, which appear to refute common testing wisdom.

A retrospective analysis of an operational system was performed to determine what would have been the effectiveness of the most commonly-used testing techniques if applied to an operational avionics system of approximately 20,000 lines of Ada code, produced to satisfy an 'essential' level of criticality.

The test case design techniques covered were equivalence partitioning, boundary value analysis, branch, branch condition, branch condition combination, modified condition decision coverage and random testing; all the techniques were applied as defined in the BCS Software Component Testing Standard.

The results of the analysis show, for this system, that several widely-held assumptions, some apparently supported by industry standards, do not hold. For instance, testing to achieve 100% branch coverage would have detected fewer faults than using equivalence partitioning. Also, random testing would have been surprisingly effective. Random testing would have achieved similar levels of effectiveness for the same number of test cases as any of the other techniques studied except for boundary value analysis, and with only six random test cases per module would have outperformed branch, branch condition, branch condition combination or modified condition decision coverage (the level required for safety-critical avionics software). In contrast, boundary value analysis fared particularly well in this study, and over 50,000 random test cases per module would have been required to equal its test effectiveness.

6.3.3.3 Conclusions of paper

This paper has presented the results of a study into the test effectiveness of a number of testing techniques. The experimental methodology used here has differed from previous experiments as the derived test effectiveness is based on an analysis of all inputs that satisfy the test technique rather than arbitrarily chosen inputs from this set. The results, in summary, are:

- *BVA was the most effective technique studied, achieving a highest mean probability of detection of 0.75, compared with 0.16 for EP and 0.17 for the most effective white box technique studied (BCC). However, to achieve this, nearly twice as many test cases were required (on average, 13.7 for BVA, 5.1 for EP and 4.7 for BCC).*
- *For lower levels of effectiveness, random testing appears to have the advantage over both EP and all the white box techniques considered. Just six random test cases per module are required to achieve a better level of effectiveness than any of these techniques, and random testing will also be less expensive to perform as no test case design is required.*
- *Where multiple conditions are present and condition testing is worthwhile, the use of MCDC rather than BCC appears pointless due to the extra effort required to define the test case sets, and the (albeit marginally) reduced levels of test effectiveness.*
- *Although random testing appears efficient for achieving lower levels of test effectiveness, it requires a prohibitive number of test cases (over 50,000) to reach the levels achievable by BVA.*
- *If choosing a black box technique to complement branch testing then BVA appears to be more effective than either EP, or random testing.*
- *It would appear that reliance on a single test technique is not the best approach as even BVA would always have missed faults, and so the identification of which techniques complement each other most effectively appears to warrant further study.*

Obviously the results of this study have taken little account of cost, except in general terms; measures could not be taken as the techniques were not used to detect faults, but rather to create test case sets. More study of the costs of applying testing techniques is required, although the relatively large differences in test effectiveness results presented here may well overshadow relatively small differences in the cost of their application.

Using the same example avionics system, the methodology has still be applied to other testing techniques, such as those based on data flow. The next step will then be to re-apply the methodology to other systems to determine if the results are system-specific.

As with all such experiments the results are characterised by the conditions under which it took place (described in Appendix A) and their extrapolation to other environments is not implied.

6.3.3.4 Synopsis

This paper followed the format of paper [7] very closely. It reported on the same experiment, but presented test effectiveness results for a different, and larger, set of techniques. EP, BVA and random testing were covered, as before, but five white box techniques, based on control flow and predicate conditions, were also included. A correspondingly larger number of hypotheses were also considered. All testing techniques in this paper were considered in their minimised form, where the smallest possible test case suite that satisfies the corresponding test coverage measure to 100% is used.

The results presented in this paper also corresponded closely to those in the previous paper [7]. Despite the inclusion of white box techniques, BVA was still the most effective technique, by far. Random testing also fared well, again. It compared favourably against both EP and the white box techniques, whose probabilities of detection were surprisingly low, being similar to those for EP.

6.4 Commentary

When paper [6] was written, the aim of the research, as described by the hypothesis, was to identify the most effective techniques for a given module by analysis of the module's specification and/or source code. As stated in paper [6], this could *only* be achieved if test effectiveness data for enough modules and test techniques were collected. The scale of the problem was obviously not recognised and, to date, only the generation of test effectiveness values for the one system have been produced. This has not provided enough data to achieve the original aim.

An unsuccessful attempt was, however, made to identify attributes that could be used to predict fault-prone modules. Paper [6] did note that it was possible to identify modules that would require revision by their size (using lines of code, although the close correlation between lines of code and complexity measures would also have allowed their use). This was not considered, however, to be particularly useful, as an analysis of the reasons for change found that over half of the revisions were not due to the modules themselves, but instigated by either requirements changes from the customer or changes to development standards. This highlights a potential problem when researchers use a simple measure of module revisions as a surrogate measure for module quality [56].

A problem can also arise if the information contained on problem reports is blindly accepted as being correct. Paper [6] presented a classification of faults based on the information provided on the problem reports. Only when this information was closely scrutinised at a later stage of the Investigation was it found to be flawed. For unknown reasons, the cause of faults was often found to have been attributed to coding/detailed design when this was incorrect. In the paper a value of 28% was attributed to problem reports with these faults, when the true figure should have been less than 10%.

More than half of paper [6] is dedicated to what it describes as an error analysis, but more correctly would be known as a failure analysis. A number of interesting results were presented in this section. A contribution was made to the seemingly continuous debate on the optimum size of a module. Smaller

modules were found to have higher error densities, and, surprisingly, this could not be explained by the influence of interface errors.

The structure of papers [7] and [8] were very similar - presenting the same experimental methodology, which was applied to the same system, but comparing different testing techniques. Three black box testing techniques (and five hypotheses) were considered in paper [7] while paper [8] considered a further five white box techniques (and thirteen hypotheses).

Two alternative implementations of EP and BVA were studied in paper [7]. The guidelines clause in the Standard describes "two distinct approaches" for EP - these are the 'minimised' and 'one-to-one' as described in the paper. Both were included in the Standard as there was some disagreement over which was the optimum way to implement EP. One aim of investigating the two EP approaches was to gain some insight into which was the better implementation. An apparent oversight in paper [7] is that it explains the difference between partition and subdomain testing techniques, and then describes the 'minimised' and 'one-to-one' approaches, but then fails to explain that the minimised approach corresponds to true partition testing.

The calculation of the probabilities of detection for branch coverage highlighted problems with three of the modules/faults studied. Values of one were achieved for the probability of detection for branch testing for each of these modules/faults, implying that any test cases that satisfied branch coverage would have detected the faults. As the modules in the study had already been tested to achieve 100% branch coverage, and these faults not been found, then something was obviously wrong. After investigation it was found that the reason the faults had not been found by the original branch testing was that they had been masked by the setting of incorrect variable values in the test harness. As these faults would have been detected by every technique (except random testing), and should have been found by the original branch testing, these modules/faults were discounted from the analysis of results in paper [8]. Unhappily, these results were not available when paper [7] was written and so the analysis of results in this paper should be viewed with this in mind.

The generation of probabilities of detection for all subsequent techniques after the first one was relatively fast as the fault-finding sets for the modules had already been created to determine the results presented in paper [7]. Thus, just the production of test case sets and the calculation of probabilities of detection were necessary to produce the extra results. The automation of much of the calculation of the statistical measures also made the production of those results far easier. This calculation was performed within the Access database that was used to store all the experimental data. The author's inexperience in using databases led to some difficulties in checking the correctness of some of the necessarily complex queries. In hindsight, the use of a dummy database with the same structure, but populated with only a small amount of data to make the verification of queries relatively easy, would have increased confidence in its use and made the checking of results far easier.

When taking into consideration the number of test cases (and so indirectly, cost) along with the values for probabilities of detection, just two of the five hypotheses were considered to be proven in each of papers [7] and [8]. The statistical results presented in paper [8] confirmed the author's interpretation of the data and both statistical techniques also agreed throughout. For paper [7] there was just one disagreement between the statistical techniques, and this was probably due to the inclusion of the three modules/faults that were discounted in paper [8].

The small differences between the 'one-to-one' and 'minimised' approaches for EP and BVA in paper [7] were not surprising, although the inability to absolutely distinguish between the effectiveness of EP and random testing was certainly surprising, despite the seminal papers by Duran and Ntafos [27] and Hamlet and Taylor [32]. This counter-intuitive success of random testing caused the author to spend considerable time re-checking the calculations, and, from conversations with other researchers and practitioners, he was not alone in finding it difficult to believe the random testing results.

The probabilities of detection for the white box testing techniques presented in paper [8] were lower than had been expected. These results were expected to be relatively low, as the modules had all been tested to 100% branch coverage prior to the study, however, they appeared low to the author despite this. The lack of any faults that were directly related to multiple conditions in predicates also contributed to the apparent low level of effectiveness, and this made comparisons between these techniques difficult. Secondly, relative to the white box techniques, random testing continued to perform better than expected, and only BVA was statistically significantly more effective.

The author started the study inexperienced in the application of statistical techniques. The techniques used in the study were chosen from text books and the choices validated by more expert colleagues. The usefulness of the statistical measures was initially unclear as the author felt that a straightforward interpretation of the experimental results was sufficient. The use of the statistical techniques did, however, highlight any unusual results requiring additional investigation, and eventually confirmed the author's initial conclusions. If anything, the statistical techniques identified significant differences between techniques that the author would not have considered significant. In these cases, such as the comparison between 'one-to-one' EP and 'minimised' EP, the author believed the differences to be so small, that, when other considerations, such as the cost of applying the techniques, were considered, no particular technique could be said to have an advantage over the other.

For paper [7] the full text (rather than an abstract) was required for submission to the conference, and was one of 19 papers accepted out of more than 60 submissions. The original call for papers required a maximum of twelve pages, but, on acceptance, this was reduced to ten pages, with a penalty of $100/page for longer papers. As the originally submitted paper was exactly 12 pages this meant that two pages of material had to be removed for the final paper. The definition of random testing was completely removed, the sections on test effectiveness experimental difficulties and related work were both drastically cut, and other sections reduced, to comply with the ten page limit.

6.5 Relationship to other work

At a high level the Investigation attempted to follow one of the proposed research directions from Osterweil and Clarke's paper [48], that suggests the identification of characteristics to allow the exploitation of particularly effective tools and techniques. There appears to be no published material, however, on such studies that consider the same characteristics and techniques used in the Investigation. The reason for this could be its over-ambitious nature, as explained in the commentary in section 7.4. The Investigation started out with the aim of performing a study of test effectiveness and then extending the study by using the generated test effectiveness results to match module characteristics with the most effective testing technique. As was found, insufficient test effectiveness results were generated to allow the correlation analysis with module characteristics and so the Investigation, as performed, is closely related to studies into test technique effectiveness.

A number of papers have been published in the area of test technique effectiveness, covering both theoretical ([21], [63], [43], [32], [22], [33], [61], [62], [50]), and empirical studies ([27], [16], [47], [64], [57], [60], [31]). Perhaps one of the surprising features of these studies is that the empirical studies tend to pre-date the theoretical work. There is still not enough data available to validate much of the theoretical work and a software tester reading the empirical studies would find that there is little consensus on which testing techniques are the most effective. The reason for this can be explained, in part, from an examination of the factors influencing module testing quality. When carrying out experiments module testing quality cannot be measured directly so this is achieved using surrogate measures, known as response variables, such as the number of defects missed, or the probability of detecting a fault. The remaining factors that are not measured, but affect the results, are known as state variables, and are set to representative values. These state variables then characterise the experiment or case study - that is to say the results of the comparison should be applicable elsewhere *given the state variables are similar*. There are several state variables and the inability of researchers to agree on, or set, *similar* values could account, on its own, for the lack of consensus.

It is difficult for researchers to find a representative set of software testers with the time, and motivation, to act as 'guinea pigs' and frequently students are used. Basili and Selby [16] in comparing code reading, functional testing (using EP and BVA) and structural testing (using 100% statement coverage) attempted to overcome this problem by using groups of programmers with different levels of expertise ranging from students to professionals with over 20 years experience, although this is uncommon. Studying four modules they found that code reading was most effective, and among other results, that the test effectiveness was dependent on the type of software tested. Myers [47] performed an experiment on three testing techniques, including reviews, black box techniques and white box techniques, and found equal levels of test effectiveness. His experiment employed 59 subjects, who he described as, for the most part, "professional programmers" although he described them as above-average, so they were explicitly non-representative. He later states that his subjects were 'judged to be highly motivated during the experiment.'

This highlights the danger of the Hawthorne Effect that can arise when the testers know their efforts are being assessed as part of an experiment. Frankl and Weiss [30] overcame the difficulties of using testers by generating test case sets randomly, determining their coverage of data flow and branch coverage criteria, and then measuring their effectiveness. They concluded that the test sets that satisfied the data flow criteria were significantly more effective in five of the nine subject programs and guaranteed to detect the error in four of them. In a similar manner, the Investigation requires no testers as no actual test cases are used. Instead test effectiveness is calculated analytically by considering all possible inputs that satisfy a test criterion. This use of test case sets, which were generated by the author using definitions from the Standard, ensures both no bias and the ability to re-apply the techniques by reference to the Standard.

The problem of choosing representative test cases does not apply to random testing. Duran and Ntafos [27] compared random testing to partition testing using both simulations and experiments. They concluded that random testing can be cost effective for many programs, and their counter-intuitive results were subsequently validated by Hamlet and Taylor [32], who performed simulations using the failure-rate model used in the earlier paper as well as a number of other models. These results appear to directly refute the statement of Myers [46] that "Probably the poorest methodology of all is random-input testing...", an opinion that is still widely-held, despite the results from [27] and [32].

The obvious way to ensure that the faults used in research are truly representative is to use real code containing real faults. The difficulty then is to know what faults are contained in it. This is only practicable if the code has already been thoroughly tested, or better still, used operationally as well, so that *most* of the faults in it are known, which was the situation with the Investigation. The alternative is for the researchers to put the faults in themselves. This approach is easy to criticise and Abbott [12] states that "seeding suffers from too many problems to be considered a reliable technique for evaluating test thoroughness and completeness". In the study by Girgis and Woodward [31] in which mutation testing, data flow testing and control flow testing are compared, error seeding is used as the only source of errors. This study concluded that the control flow strategy (based on LCSAJ coverage) was the most effective, although it also argued for the use of complementary test techniques.

As can be seen, there is no consensus on an experimental methodology nor on the results of these empirical investigations. The wide variation in the state variables, not least the difficulty in knowing if the same (named) techniques are being consistently applied, makes sensible comparison and practical use of the results impossible. Future empirical studies must provide sufficient details to enable their results to be compared and used to build up a cohesive body of knowledge in the area. A parametric framework has been suggested [54], which would allow experiments to be classified and so both compared and repeated. The papers describing the Investigation ([6], [7] and [8]) have all attempted to provide as complete a description of the state variables as possible.

The theoretical work performed on test effectiveness appears to be more cohesive than the experimental work. A number of researchers have performed theoretical comparisons of techniques, such as Weyuker

and Frankl [61], although many of these studies have compared generic partition testing techniques with random testing, rather than considering actual systematic techniques such as EP, BVA and branch testing. Where actual techniques have been studied there has been a tendency to consider 'new' techniques (e.g. [60] and [64]), rather than compare those already in use commercially. Most papers have agreed on using the same underlying model as described in [63], previously referred to as the failure rate model, although the earlier papers assumed true partitioning (i.e. no overlapping), whereas the practical application of most techniques tends to generate overlapping subdomains - a fact allowed for in later papers (e.g. [22] and [62]). The model used to calculate the probabilities of detection in the Investigation is based on the same underlying model as these papers and allows for overlapping subdomains. Several of the theoretical papers (e.g. [32], [61] and [62]) suggest that further empirical work should be carried out by comparing specific partition testing techniques, as was performed in the Investigation.

6.6 Recommendations for further work

These recommendations are aimed at both expanding the scope of the Investigation and mitigating its limitations.

As the fault finding sets have been created already, then application of the experimental methodology to other techniques is relatively easy as only the creation of test case sets and the calculation of probabilities is necessary. The most obvious techniques to consider next are the data flow testing techniques, as proposed and investigated by Elaine Weyuker and others [64]. Similarly, the experimental methodology could be applied to more than one technique at a time, to try to determine which is the most effective mix of techniques to use. Another relatively low cost re-application of the experimental methodology would be to determine empirically probabilities of detection for the test techniques already considered but with several test cases per subdomain. Weyuker and Jeng [63] have considered this area theoretically, as have Chan et al [21].

A limitation of the Investigation so far is that it has only been applied to one system. It is not known how typical the studied system is of other systems and so how applicable the results are to other system developers. The obvious answer is to repeat the Investigation on other systems. Two main difficulties arise from this. The first is that it is difficult to gain access to the amount of data necessary to perform the analysis, which must come from an operational system. The second is the amount of time required to perform the analysis. The process is very time-consuming, and there is little opportunity for automation. If more systems are studied and enough data becomes available, then the original aim of the Investigation - to determine rules for identifying the most effective test techniques for a given module - could be achieved. This would require relationships to be determined between the attributes of a faulty module, such as code, structure and specification-based metrics, and the test effectiveness for the test techniques applied to the module.

A modification to the experimental methodology could remove the necessity for analytically calculating the probability of detection from the overlap of the fault-finding and test case sets, and for even creating the fault-finding sets. This would involve using a tool to randomly generate test cases based on the test case sets. In order that these test cases are representative of the complete test case set then a very large number of test cases would have to be generated. This number of tests would necessitate the automation of the process of determining if a fault would have been detected. This could be achieved by running each test case through both the original, faulty module and the corrected module and then comparing the two outputs. If the outputs differ then the test case would, presumably, have detected the fault. The proportion of test cases that detect the fault would then give the probability of detection for the test technique. This modification would not be too difficult to implement as the author already has access to a syntax testing tool that generates random inputs based on a defined syntax and the test case sets could easily be defined using this syntax. The use of commercially-available Ada testing tools would make the execution and comparison of results for the subsequent calculation of probability of detection relatively simple. A first use of this system would presumably be to apply it to the system already studied in the Investigation. This could then validate the results already calculated, in order to determine the accuracy of the manually-performed analysis and also to determine the effectiveness of the new approach. It would, for instance, be useful to know how many test cases per test case set were required to achieve similar results to those absolute values already calculated analytically.

Ideally the experimental methodology should be re-applied to a system where the state of the modules prior to module testing was available. This would allow probabilities of detection to be calculated for each technique alone. The modules studied in the Investigation had already been branch tested to achieve 100% coverage and so all results had to be considered in this light. If the modules had not been module tested at all then the test effectiveness of techniques in isolation, and in conjunction with other techniques apart from branch testing, could be calculated. Unless an organisation could be found that stored modules prior to module testing, then it would be necessary to deliberately store modules at this point in their development purely for the purposes of the experiment. It would then be necessary to wait until the completed system had been operational for some time so that nearly all of the faults that could have been determined by module testing had manifested themselves as failures. The obvious problem with this approach is its long-term nature.

Two aspects that affect test effectiveness were not considered in the Investigation. Firstly, no account of the criticality or scope of the faults was considered. This information could be extracted from the problem reports for the faults and weightings applied to determine weighted values of test effectiveness. Secondly, the cost of applying the testing techniques was not considered except in the broadest terms. If available, the cost of applying each technique could be included in the calculation of test effectiveness, however, this would first require such cost data to be available, and no source of such data is known to the author.

6.7 Conclusions - Investigation

The Investigation did not satisfy its initial aim of identifying those attributes of a module that would allow the most appropriate set of software module testing techniques to be chosen for each individual module. To identify the correlation between the most appropriate testing technique and module characteristics would have required a much larger number of modules to have been analysed in the Investigation. There was not time to do this. The Investigation was terminated after the test effectiveness results for the complete avionics system had been compiled for ten testing techniques. More systems need to be analysed to provide sufficient data to satisfy the initial aim.

Despite this, the author considers the Investigation to be a success in two major areas - it validated a new methodology for determining test effectiveness empirically and also generated test effectiveness results for a number of techniques. Due to the methodology used the results suffer from fewer limitations than those from many other experiments in this area. Probably the main limitation of most other test effectiveness experiments has been their difficulty in determining absolute values of probability of detection based on all the input values that satisfy a test technique - most experiments are based on arbitrarily chosen input values from this set. Conversely, the Investigation considered all values in this set, which is particularly important when considering hard-to-find faults. The time-consuming nature of the Investigation has meant that it has only been applied to a single system and the results are therefore characterised by this system, although a wide range of techniques has been considered. Many other experiments, however, have been based on much smaller, less realistic data sets, and only considered two or three techniques.

The author has found no reference to any other use of the experimental methodology used in the Investigation, which, in hindsight, is closely related to the theoretical work on partition testing. Although the methodology is limited to the analysis of code with linear predicates in a similar manner to symbolic execution, this was not a problem as none of the studied modules failed to satisfy this criterion.

As branch testing had already been used on all of the studied modules, the results indicate which is the best complementary technique to 100% branch coverage. This is BVA. The results also show that a complementary technique to branch testing is necessary - as there were several faults left after 100% branch coverage had been achieved that were relatively easy to detect using many of the studied techniques. The relatively high test effectiveness results achieved for random testing were surprising despite the knowledge that others had already published similar results, albeit mostly theoretical. There is a lack of empirical data in the area of test effectiveness (as there is for most areas of software engineering) and, when possible, it should be made available to the community of researchers and practitioners. The author intends to publish the results of the Investigation in a journal in the near future.

The results of the Investigation have caused changes to be made in several areas. The organisation which provided the studied system has now introduced a metrics program and collects and analyses fault data automatically, using the fault classification scheme produced by the author for the Investigation. They have only recently been presented with the test effectiveness results, and their reaction is awaited. The

author has presented feedback both on the application of the testing techniques and the test effectiveness results to the Working Party producing the Standard, and this has resulted both in improvements to several parts of the Standard and the inclusion of random testing, which otherwise would have been left out.

7. CONTRIBUTION TO KNOWLEDGE

Of the two projects the development of the Standard has necessarily broken less new ground than the Investigation. Standards are not intended as a forum for new ideas, but as a medium defining best practice, which must be agreed by consensus. The creation of the BCS/BSI Software Component Testing Standard and the BCS/BSI Glossary of terms used in software testing has filled a gap in both the national and international software testing standards arenas. Their production has required its authors to gain an in-depth knowledge of software component testing techniques, identify and use a development process, and learn how to negotiate the standardisation process at a national level. The technical insights into component testing will prove invaluable in any future standardisation projects in this area, as well as to researchers and practitioners in component testing, such as those implementing test coverage tools. The experience of the standardisation process will be of use to anyone producing a standard, but especially to those who intend taking a standard to national status through BSI.

The knowledge gained during this process has been disseminated by the author in a number of ways. The papers, [1], [2] and [3], included as part of this portfolio, and this overview itself, provide a means of promulgating this knowledge. The author has presented papers at conferences, BCS SIGIST meetings and seminars, and talking to delegates at such events also disseminates this knowledge. The imminent publication of the Standard by BSI will provide the author with a further opportunity to talk about the experiences of the WP and especially the relationship with BSI. The presence of a well-publicised Web page for the Standard, the URL of which is listed on several search engines, has meant that the author is regularly contacted by those interested in the Standard.

The Investigation has introduced a new methodology for determining the test effectiveness of software component testing techniques by means of a retrospective analysis. This methodology has differed from previous experiments as the derived test effectiveness is based on an analysis of all inputs that satisfy the testing technique rather than arbitrarily chosen inputs from this set. The methodology has been validated by its successful application to a complete operational avionics system and has thereby provided a new set of test effectiveness data that can be added to the body of empirical data on software module testing effectiveness. This data is a valuable first step in the building of a cohesive body of research where the experimental variables have been both controlled and documented. It is only with the results of such research that the "holy grail" [54] of placing software testing techniques on an ordinal scale can eventually be achieved. The author intends to make this data available to a wider audience by presenting the results in a journal publication. An example of how the results of the Investigation can contribute to the community was demonstrated when the favourable test effectiveness results for random testing were used to convince the WP to include it as a technique in the Standard.

8. REFERENCES

[1] -----, The BCS Software Component Testing proto-Standard, Proceedings of the Second International Conference on Software Quality Management, Edinburgh, July 1994.

[2] -----, The Software Testing Standard - how you can use it, *Proceedings of the 3rd European International Conference on Software Testing, Analysis and Review*, London, November 1995.

[3] -----, Popular Misconceptions in Module Testing, *Proceeding of the 13th International Conference on Testing Computer Software*, Washington DC, June 1996.

[4] BCS/BSI Standard for Software Component Testing, Version 3.3, April 1997 (available from the author).

[5] BCS/BSI Glossary of terms used in software testing, Version 6.2, April 1997 (available from the author).

[6] -----, Test Effectiveness in Software Module Testing, *Proceedings of the 2nd European International Conference on Software Testing, Analysis and Review*, Brussels, October 1994.

[7] -----, An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing, *4th International Software Metrics Symposium*, Albuquerque, November 1997.

[8] -----, Module Testing Techniques - which are the most effective? Results of a Retrospective Analysis, *5th European International Conference on Software Testing, Analysis and Review*, Edinburgh, November 1997.

[9] BCS SIGIST Standards Working Party, Reqts Document for the Software Component Testing Standard, Draft 5 (available from the author).

[10] BCS SIGIST Standards Working Party, Software Component Testing Standard Project Plan (available from the author).

[11] BCS SIGIST Standards Working Party Minutes (available from the author).

[12] Abbott, J., 'Software Testing Techniques', NCC Publications, 1986.

[13] ANSI/IEEE Std 1008, Standard for Software Unit Testing, 1987.

[14] ANSI/IEEE Std 1012, Standard for Software Verification and Validation Plans, 1986.

[15] ANSI/IEEE Std 829, Standard for Software Test Documentation, 1983.

[16] Basili, V.R. and Selby, R.W., 'Comparing the Effectiveness of Software Testing Strategies' *IEEE Trans. on Software Eng.*, Vol.SE-13, No.12, pp. 1278-1296, Dec. 1987.

[17] Beizer, B., Cleanroom Process Model: A Critical Examination, *IEEE Software*, pp. 14-16, March/April 1997.

[18] BS 0, A standard for standards, BSI, 1991.

[19] BS 5887, Codes of practice for Testing of Computer-Based Systems, 1980.

[20] BS 7165, Recommendations for the Achievement of Quality in Software.

[21] Chan, F. T., Chen, T.Y., Mak, I.K. and Yu, Y.T., 'Practical considerations in using partition testing', *Proc. SQM '94*, Edinburgh, July 1994.

[22] Chen, T.Y. and Yu, Y.T., 'On the Expected Number of Failures Detected by Subdomain Testing and Random Testing', *IEEE Trans. on Software Eng.*, Vol. 22, No. 2, pp. 109-119, Feb. 1996.

[23] Chow, T.S. 'Testing Software Design Modelled by Finite State Machines' *IEEE Trans. on S/W Eng.*, Vol.SE-4, No.3, May 1978.

[24] Def Stan 00-17, Modular Approach to Software Construction, Operation and Test (MASCOT), Issue 1.

[25] Def Stan 00-55, The Procurement of Safety-Critical Software in Defence Equipment, Interim, 1991.

[26] Development Guidelines for Vehicle-Based Software, MISRA, 1994.

[27] Duran, J.W. and Ntafos, S.C., 'An Evaluation of Random Testing', *IEEE Trans. on Software Eng.*, Vol. SE-10, No. 4, pp. 438-444, July 1984.

[28] Fenton, N. and Page, S., 'Towards the Evaluation of Software Engineering Standards', *Proceedings of the Software Engineering Standards Symposium*, Brighton, United Kingdom, 1993. IEEE Computer Society Press, 1993.

[29] Fenton, N., 'Measurement and Testing Standards', *Presentation to BCS SIGIST*, London, May 1993.

[30] Frankl, P.G. and Weiss, S.N., 'An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing' *IEEE Trans. on Software Eng.*, Vol. 19, No. 8, pp. 774-787, Aug. 1993.

[31] Girgis, M.R. and Woodward, M.R., 'An Experimental Comparison of the Error Exposing Ability of Program Testing Criteria', *Proc. of the Workshop on Software Testing*, Canada, 15-17 July 1986.

[32] Hamlet, R. and Taylor, R., 'Partition Testing Does Not Inspire Confidence', *IEEE Trans. on Software Eng.*, Vol. 16, No. 12, pp. 1402-1411, Dec. 1990.

[33] Hamlet, R., 'Theoretical Comparison of Testing Methods', *SIGSOFT Software Eng. Notes*, Vol. 14, Iss.8, pp. 28-37, 1989.

[34] Hawkes, D.J., Struck, W.F. and Tripp, L.L. 'An International Safety-Critical Software Standard for the 1990s', *Proceedings of the Software Engineering Standards Symposium*, Brighton, United Kingdom, 1993. IEEE Computer Society Press, 1993.

[35] IEC 1508, Functional Safety: Safety-Related Systems, Draft, 1995.

[36] IEC 56(Sec)372, Guide to test methods for dependability assessment for software, 1994.

[37] IEC 880, Software for Computers in the safety systems of nuclear power stations, 1986.

[38] IEC/ISO 9126, Information Technology - Software Product Evaluation - Quality Characteristics and guidelines for their use.

[39] IEEE 1003.3, Standard for Information Technology - Test methods for measuring conformance to POSIX.

[40] ISO 12207, Information Technology - Software life cycle processes, 1995.

[41] ISO 2382, An Information Technology Terminology Standard.

[42] ISO/IEC 12119, Information Technology - Software Packages - Quality Requirements and testing, 1994.

[43] Jeng, B., and Weyuker, E.J., 'Some Observations on Partition Testing', *Proc. ACM SIGSOFT 3rd Symposium on Software Testing, Analysis, and Verification,* ACM Press, pp. 38-47, Dec 1989.

[44] Jones, *Applied Software Management*, McGraw-Hill, 1991.

[45] MIL-STD-498, Software Development and Documentation, 1992.

[46] Myers, G., *The Art of Software Testing*, Wiley Interscience, 1979.

[47] Myers, G.J., 'A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections' *Comms. of the ACM*, Vol. 21, No. 9, pp. 760-768, 1978.

[48] Osterweil, L. and Clarke, L., 'A Proposed Testing and Analysis Research Initiative', *IEEE Software*, Sept. 1992.

[49] prEN 50128, Railway Applications - Software for Railway Control and Protection Systems - Draft European Standard, 1995.

[50] Ntafos, S.C., 'A Comparison of Some Structural Testing Strategies', *IEEE Trans. on S/W Eng.*, Vol.13, No.6, pp. 868-874, 1988.

[51] *ISO/IEC Directives - Pt 3. Drafting and presentation of International Standards,* ISO, Geneva, 1989.

[52] RAC CRTA-TEST, Testability Design and Assessment Tools.

[53] RAC/CSA Q396, Software Quality Assurance Program Standards.

[54] Roper, M. et al, 'Towards the Experimental Evaluation of Software Testing Techniques, *Proc. EuroSTAR '94*, Brussels, Oct. 1994.

[55] RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification, RTCA, Dec 1, 1992.

[56] Schneidewind, N.F., 'Software Metrics For Integrating Quality Control And Prediction', *Proc. 9th International Symposium on Software Reliability Engineering*, pp. 402-411, Albuquerque, Nov 1997.

[57] Selby, R.W., 'Combining Software Testing Strategies: An Empirical Evaluation', *Proc. of the Workshop on Software Testing*, Canada, 15-17 July 1986.

[58] Tripp, L. and Voldner, P., 'A Market-Driven Architecture for Software Engineering Standards', *Proceedings of the Software Engineering Standards Symposium,* Montreal, Quebec, August 1995. IEEE Computer Society Press, 1995.

[59] Vinter, et al, The Prevention of Errors through Experience-driven Test Efforts, *Final Report*, ESSI Project No. 10438, March 1996.

[60] Vouk, M.A., Tai, K.C. and Paradkar, A., 'Empirical Studies of Predicate-Based Software Testing', *Proc. Fifth International Symposium on Software Reliability Eng.*, Monterey, Nov. 1994.

[61] Weyuker, E.J. and Frankl, P.G., 'A Formal Analysis of the Fault-Detecting Ability of Testing Methods', *IEEE Trans. on Software Eng.*, Vol. 19, No. 3, pp. 202-213, Mar. 1993.

[62] Weyuker, E.J. and Frankl, P.G., 'Provable Improvements on Branch Testing', *IEEE Trans. on Software Eng.*, Vol. 19, No. 10, pp. 962-975, Oct. 1993.

[63] Weyuker, E.J. and Jeng, B., 'Analyzing Partition Testing Strategies', *IEEE Trans. on Software Eng.*, Vol. 17, No. 7, July 1991, pp 703-711.

[64] Weyuker, E.J., 'More Experience with Data Flow Testing', IEEE Trans. on Software Eng., Vol. 19, No. 9, pp. 912-919, Sep. 1993.

[65] Wichmann, B., 'Problems and Strategies for Software Component Testing Standards', *Journal of Software Testing, Verification and Reliability*, Vol. 2, pp. 167-185. 1992.

[66] Wichmann, B., 'Why are there no measurement standards for software testing?', *Computer Standards and Interfaces*, Vol. 15, pp 361-364, 1993.

[67] Nash, S.H. and Redwine, S.T., 'Map of the World of Software-Related Standards, Guidelines, and Recommended Practices', *Computer Standards and Interfaces*, Vol. 6 No. 2, pp. 245-265, 1987.

[68] ISO 9000-3, Quality Systems Guide to the Application of ISO 9001 to the Development, Supply and Maintenance of Software, 1991.

9. BIBLIOGRAPHY

Agresti, W.W. and Evanco, W.M., 'Projecting Software Defects From Analyzing Ada Designs', *IEEE Trans. on S/W Eng.*, Vol. 18, No. 11, pp. 988-997, 1992.

Bache, R., 'Graph Models of Software', Ph.D. Thesis, South Bank Polytechnic, London, 1990.

Basili, V.R., 'Applying the Goal/Question/Metric Paradigm in the Experience Factory', *Proceedings of the 10th Annual Conference on Application of Software Metrics and Quality Assurance in Industry*, Amsterdam, 1993.

Basili, V.R. and Perricone, B.T., 'Software Errors and Complexity: An Empirical Investigation', *Comms of the ACM*, Vol. 27, No. 27, pp. 42-52, 1984.

Beizer, B., 'Software Testing Techniques', Van Nostrand Reinhold, 1990.

BS0, A standard for standards, British Standards Institution, 1991.

Cargill, C.F., Information Technology Standardization, Digital Press, 1989.

Chilenski, J.J. and Miller, S.P., 'Applicability of modified condition/decision coverage to software testing', *Software Engineering Journal.*, pp. 193-200, Sept. 1994.

Cooper, B.E., 'Statistics for Experimentalists', *Pergamon Press*, 1969.

Coward, P. D., 'Symbolic Execution Systems - A Review', *Software Eng. Journal*, Vol. 3, Part 6, pp. 229-239, 1988.

Davey, S., Huxford, D., Liddiard, J., Powley, M. and Smith, A., 'Metrics Collection in Code and Unit Test as Part of Continuous Quality Improvement', *Proceedings of 1st European Conference on Software Testing, Analysis and Review,* 52-28 October 1993, London, UK.

Elliott, B. 'A Standard for Software Component Testing', *Proceedings of the 2nd European Conference on Software Testing, Analysis and Review*, Brussels, October 1994.

Fairley, R.E., 'Software Engineering Concepts', McGraw-Hill, 1985.

Gilb, T. and Graham, D., Software Inspection, Addison Wesley, 1993.

Hetzel, B, and Gelperin, D., 'Software Testing: Some Troubling Issues', *American Programmer*, pp. 22-27, April 1991.

Hetzel, W.C., 'The Complete Guide to Software Testing', Collins, 1985.

Hetzel, W.C., 'Making Software Measurement Work', QED Publishing Group, 1993.

Humphrey, W., 'Characterizing the Software Process: A Maturity Framework', *IEEE Software*, Vol.5 No. 2, pp. 73-79, 1988.

Jones, C., 'Applied Software Measurement: Assuring Productivity and Quality', McGraw-Hill, N.Y., 1991.

Jones, G.W., 'Software Engineering', John Wiley and Sons, 1990.

MacNamara, G. and Murphy, C.M., 'People and the Pursuit of Excellence', *Proceedings of the 1st European International Conference on Software Testing, Analysis & Review*, London, United Kingdom, 1993.

PSS-05-0, ESA Software Engineering Standards, 1992.

Roper, M., 'Software Testing', McGraw-Hill, 1994.

Stevens, R. and Scheffer, A., 'Developing a Structured Set of Standards', *Proceedings of the Software Engineering Standards Symposium*, Brighton, United Kingdom, 1993. IEEE Computer Society Press, 1993.

Weiss, D.M. and Basili, V.R., 'Evaluating Software Development by Analysis of Changes: Some Data from the Software Engineering Laboratory', *IEEE Trans. on S/W Eng.*, Vol.SE-11, No. 2, pp. 157-168, 1985.