Number 11

THE TESTER

March 2005

NEXT CONFERENCE

Have I Got Tests for You

Tuesday 8 March 2005

- **Forensic Software Engineering: Patterns of Failure**
- **Communication and Assertiveness**
- **Future Challenges for Testing**
- **IT Service Management: ITIL & BS15000, Drivers for Testing**
- **A Practical Model for Program Test Management**
- **Tips on Performing a Test Process Assessment**
- **Recent Case Studies in Testing: Parallel Inspections and Testing After Delivery**

IN THIS ISSUE:

BCS

THE BRITISH COMPUTER SOCIETY

FROM THE EDITOR

We are delighted to see that the attendance continues to increase following our reduction in cost of attendance to £100. We are paring down costs in all areas to facilitate this and we hope that you are not only spreading the word of the new, lower cost to your testing colleagues but you are also booking yourself on the conferences!

I am sure you will find that we have still managed to keep the quality of our conferences very high both in terms of the new location and the speakers. The December conference was a very good example. We had Martin Pol, always a very entertaining and informative speaker, providing two papers as well as a Special Session. I was particularly interested in his experiences of outsourcing.

We try to keep you up to date too. Andy Redwood related his experiences of implementing the Sarbanes-Oxley (SOX) regulations that have been enforced for US related companies. He also informed us of the probable Euro-SOX regulations that are likely to be introduced in Europe.

In keeping you up to date, please see in this issue the latest in the plans for the ISEB qualifications to be internationally recognised.

We hope you all had a wonderful and relaxing Festive Season and we look forward to seeing your refreshed selves at the SIGIST conferences in 2005!!

Pam Frederiksen
Communications Secretary
Tel: 01483 881188 (Leysen Associates)
Fax: 01483 881189
email: pam@leysen.com

BCS SIGIST website: www.sigist.org.uk

SIGIST Standards Working Party: www.testingstandards.co.uk

FUTURE SIGIST CONFERENCE DATES

21 June 2005

20 September 2005

9 December 2005

NEXT MEETING - PROGRAMME

BCS SIGIST - Have I Got Tests For You

Tuesday 8 March 2005
Royal College of Obstetricians and Gynaecologists, 27 Sussex Place, Regent's Park, London NW1

08:30	Coffee & Registration, Exhibition opens	
09:25	Introduction and Welcome – Stuart Reid, SIGIST Chairman	
09:30	**Featured Speaker** **Forensic Software Engineering: Patterns of Failure** *Les Hatton, Oakwood Computing*	
10:30	Networking session and commercial break	
10:45	SIGIST Best Presentation Award for 2004	
10:50	Coffee & opportunity to visit the exhibition	
11:20	**Communication and Assertiveness** *Paul Lister,* *CheckFree HelioGraph*	**Workshop** **IT Service Management: ITIL & BS15000, Drivers for Testing** *Gary Holmes* *Techpractice Ltd.*
12:05	**Future Challenges for Testing** *Susan Windsor,* *IBM*	
12:50	Lunch & opportunity to visit the exhibition	
13:50	**A Practical Model for Program Test Management** *Graham Thomas,* *Independent Consultant.*	**Special Session** **Any Questions?** *Les Hatton,* *Oakwood Computing*
14:35	**Tips for Testers** *Julie Gardiner, QST Consultants Ltd.*	
14:50	Tea & opportunity to visit the exhibition	
15:20	**Tips on Performing a Test Process Assessment** *Lee Copeland, SQE*	
16:05	**Featured Speaker** **Recent Case Studies in Testing: Parallel Inspections and Testing After Delivery** *Les Hatton, Oakwood Computing*	
16:50	Closing Remarks	

WORKSHOP

This Special Session at 11:20 is an 90 minute workshop with Gary Holmes of Techpractice Ltd. Places may be limited and will be available on a first-come, first-served basis on the day, there is no advanced booking and no additional fee.

SPECIAL SESSION

The Special Session at 13:50 is a 60 minute workshop with Les Hatton our featured speaker. Places may be limited. They will be available on a first-come, first-served basis on the day. There is no advanced booking and no additional fee.

Les will leading a discussion of current topics in software testing. If you would like to attend this session please submit at least one question in advance of the day by emailing it to admin@sigist.org.uk.

The SIGIST committee reserves the right to amend the programme if circumstances deem it necessary.

PRESS RELEASE: ISTQB MOVES FORWARD

Geoff Thompson (the UK rep to the International Testing Qualification Board - ISTQB) provides the following two press releases. The first, dated the 1 December 2004, from ISTQB:

- The ISTQB Foundation syllabus (1st level) is planned to be released Q1 2005
- At the 2nd level, there are currently two schemes and two syllabi:
    - ASQF (also known as ISQI) Advanced level[1]
    - ISEB Practitioner level[2]
- Both are recognised by ISTQB as professional qualifications for testers as they have gained a respect over many years in the testing community.
- There is currently no ISTQB agreed 2nd level or advanced level international syllabus.
- ISTQB intends that these two 2nd level schemes will be integrated into a future single unified 2nd level ISTQB Advanced level qualification which should supersede both existing schemes.
- All existing certificates from either of these schemes will remain valid and recognized.
- It is also planned to have international qualifications and syllabi at a 3rd level, the ISTQB Expert level.

Geoff explained that ISTQB has issued this press release to announce the new Foundation level and to clarify the position regarding both the British Computer Society's ISEB and the ASQF 2nd level syllabi and qualifications. He also confirmed that this means any qualification above Foundation level, currently being advertised as an 'International ISTQB qualification' is not truly international. The first truly international qualification will be the ISTQB Foundation Syllabus in Q1 2005.

Once released, it will be a minimum of 6 months, whilst the training providers update their material, before the new Foundation courses and exams will be available to the public. The BCS have confirmed that until the first ISTQB Foundation exam is available they will continue to offer the ISEB Foundation, which will be dual branded as ISTQB (see below). This will ensure that anyone interested in taking an International qualification can take the current version and still be qualified at the International level.

The release of the Foundation Syllabus is a great step forward in standardising testing across the globe. The syllabus has been produced by the ISTQB Working Party -Foundation Level, who are Thomas Müller (chair), Dorothy Graham, Klaus Olsen, Erik van Veenendaal, Rex Black, Sigrid Eldh, and Maaret Pyhäjärvi.

Geoff also confirmed that the BCS is now dual branding the ISEB Software Testing Foundation Certificate to be both ISEB and ISTQB compliant. The official statement from ISEB reads:

ISEB/ISTQB dual-branded Software Testing Foundation Certificates

ISEB are now dual branding their certificate with 'This certificate denotes the holder to have gained approved ISTQB certification', in light of furthering the ISTQB mission and promoting the new international standard. Candidates taking ISEB Foundation exams will gain ISTQB certification at a foundation level.

ISEB recognition of the ASQF Software Testing Foundation Exam

ISEB formally accepts the ASQF foundation exam as equivalent to the ISEB Foundation exam. By doing so, candidates who hold the ASQF Foundation exam are eligible to sit for the ISEB practitioner exam.

For those who don't know, ISTQB is a voluntary organisation set up to establish truly International Software Testing qualifications. As well as the UK there are many other member countries, such as Germany, Austria, USA, Israel, Denmark, Sweden, The Netherlands, Switzerland and Portugal. Further information on ISTQB can be found at www.istqb.org.

If anyone would like to contact Geoff regarding ISTQB his email address is Thompson@istqb.org.

[1] developed by German Testing Board and exams run by ASQF / iSQI and SAQ
[2] developed by UK Board and exams run by ISEB

ARTICLE: A COMMERCIAL TESTING FUNCTION - ASPIRATION OR REALITY?

***Chris Shaw, Director at Mission Testing**, discusses some significant findings from Mission Testing's recent survey "Software Testing – The IT Executive's View" and explores key success factors in the move towards Managed Testing Services.*

Testing has never been more important to organisations in both the public and private sectors as they invest in new or upgraded systems. There are many examples of high profile cases when things have gone wrong, which as a result have had a significant impact on service levels. So, bearing in mind the type of risks involved, are organisations taking a long hard look at how their testing is undertaken?

Testing is a function about to change

Mission Testing recently undertook a survey of organisations in the private and public sectors to find out how IT Executives view software testing. The responses show that testing is now recognised as a clearly defined business function and that the respondents acknowledge the real benefits of investment in testing. The results demonstrate that the primary reasons behind testing include an improvement in software quality which can lead to an increase in customer confidence. The business value is clear - should customer confidence be lost, this can quickly impact on reputations, the brand and ultimately share price.

Over 75% of the Senior IT Executives questioned said they felt that the way in which testing is carried out is about to change. There appears to be a shift away from the traditional testing approach which has often been to engage users or IT staff to undertake testing at the end of the IT development cycle. Instead, organisations are moving towards a more comprehensive and risk-based approach using dedicated professional testing resources. This is also reflected in plans to increase investment in training and recruitment of dedicated testers in the next 12 months.

There is also a clear move towards the use of managed testing services. These can range from flexible resourcing solutions, to handle the peaks and troughs in testing requirements, to lower cost offshore resources or outsourcing, particularly when large, high risk projects are planned. In our view at Mission Testing, these observations represent a step-change in the way testing is perceived by Senior IT Executives compared to customer feedback which we were receiving a year ago.

The path towards successful managed testing services

Many organisations have been drawn to outsourcing and using offshore resources because they believe it will bring them cost savings. As an objective, financial savings can be extremely difficult to quantify in the short term and to have this as a sole goal will invariably result in disappointment.

In order to achieve the results organisations are generally looking for from managed services, such as a reduction in risk and improvements in efficiency, there are some crucial points that they need to consider - particularly at the full outsourcing end of the spectrum. The following is by no means an exhaustive list but serves as an illustration.

First, they need to be already fully in control of their testing having developed robust test strategies and plans. They also need to make sure they have strong processes and procedures, the right levels of expertise in-house and appropriate control measures.

Second, the relationship between an organisation and its outsourcing partner should be one of mutual respect. It should, after all, be a long term association. Having a single and accountable point of contact in both client and supplier will help to manage the relationship smoothly. A clear understanding and communication of roles and responsibilities are also vital to ensure improved software quality is achieved.

Third, whilst agreed service levels are fundamental, flexibility needs to be built in to the contract at the outset. Anecdotally, 80% of contracts are modified during their life time. The right level of flexibility allows the contract to be more responsive to the client's changing needs.

The survey results show organisations aspire to invest more in software testing as well as change the way they manage such a process. The reality is that while many organisations are moving steadily along the path to maturity in their testing, those seeking to use managed test services may also need to make a further step change to ensure their success***.***

ARTICLE: AGILE TESTING AND TEST-FIRST DEVELOPMENT

Don Mills, Macroscope Services

I coined the term, "Test-First Development", in connection with the test training course I devised fifteen years ago in New Zealand, so I read David Putman's and Charlie Poole's "Agile Testing" (*The Tester*, 10: p. 11) with great interest.

Test-Driven Development (TDD), as they presented it, isn't the same as "Test-First Development", but they share the basic idea expressed in the old engineering principle, "Don't start to build it till you know how to test it" (or, as Dave Gelperin and Bill Hetzel put it twenty years ago this year, "Test -- *then* code!"). TDD also has the advantage that it's got programmers interested in doing testing, if only because, in the agile environment, it equates to writing more program code ... .

However, I have reservations about TDD, in which "requirements are specified as a set of tests". Here are a couple.

Complexity, clarity, completeness

If testers develop the test cases and "let the developers know what the tests are before they even start coding," as Putman's and Poole's imaginary "head of testing" objected, "the developers would then only write code to pass the tests!"

So what's to object to? Wouldn't "passing the tests" be good -- ideal, even? Well, yes, to a point. Trouble is, there are often many aspects of real-world requirements that cannot easily be represented in the fragmented view afforded by test cases.

For example, look at this set of test cases, and try to figure out the actual requirement:

1. Animals = small white mouse: FAIL
2. Animals = large blue tropical fish: FAIL
3. Animals = big black cat: FAIL
4. Animals = large white mouse, big black cat: FAIL
5. Animals = small grey mouse, big black cat: FAIL
6. Animals = small white rat, big black cat: FAIL
7. Animals = small blue tropical fish, big black cat: FAIL
8. Animals = large red tropical fish, big black cat: FAIL
9. Animals = large blue temperate fish, big black cat: FAIL
10. Animals = large blue tropical bird, big black cat: FAIL
11. Animals = small white mouse, small black cat: FAIL
12. Animals = small white mouse, big white cat: FAIL
13. Animals = small white mouse, big black dog: FAIL
14. Animals = large blue tropical fish, small black cat: FAIL
15. Animals = large blue tropical fish, big white cat: FAIL
16. Animals = large blue tropical fish, big white cat: FAIL
17. Animals = small white mouse, big black cat: PASS
18. Animals = large blue tropical fish, big black cat: PASS
19. Animals = small white mouse, large blue tropical fish, big black cat: PASS

Did you work out what the pet-shop customer wants, or did it become lost in the mass of testing detail? And have the test cases captured the requirement *completely*?

Here it is, written out as a compound requirement statement:

"I must have a big black cat. At the same time, I must also have either a small white mouse, or a large blue tropical fish, or both. Nothing else will do."

I contend that this is simpler, clearer, and less redundant than specifying multiple repetitive test cases, even if we restrict the test cases to the last three. Those are the only ones you need in order to identify the requirement, provided you also know that there are no other PASS cases. I also think it would be far easier to develop code from the requirement statement, than from the test cases, since the program code must incorporate the underlying generalisation ("business rule"), not the specific examples of applying it. But I admit that I've never tried developing software from particularised test cases rather than from generalised specifications.

Clarifying complex choices by generalising the rules that underlie them, then, is one area where writing conventional requirements may be superior to conveying the requirements via test cases. A second is ensuring coverage (completeness): it's easier to check that the conditions of a rule are complete, than to check that a complex test set is complete when its rules have not been made explicit.

Conversely, a good test set may reverse the situation and clarify the *meaning* of complex requirements by providing concrete examples ("scenarios"). However, that merely makes the point, that a test case is not a requirement, but an illustration of one way to satisfy -- or disappoint -- a requirement.

Requirements, features

Another potential problem with "Test-Driven Development" (test cases as requirements specifications) is the danger of confusing customer requirements with product features. Some comments written by Bret Pettichord illustrate the problem (http://www.testing.com/cgi-bin/clipper.pl?ProductTestsFirst):

"I am having an office built right now and am working with an architect and a builder. We're still in the early stages. My architect is eager to have me avoid describing the size and shape of the office. Instead, he wants to know what kinds of activities will happen there and what my requirements are for desk space, meeting space, book shelving and what not. I actually wrote up some requirements, we discussed them at length. But the goal even here was understanding, not to draft the requirements in a form that everyone could agree to. Rather the formal document will be the blueprints. ... I'm not in love with analogies that compare software to buildings, but even here, notice that the building experts don't worry too much over requirement documents."

It seems to me that Brett was confusing requirements ("kinds of activities ... desk space, meeting space," etc.) with features ("the size and shape of the office"); I suspect that the "requirements" he wrote were feature-rich and requirement-light -- a situation his architect (evidently a skilled Requirements Engineer) sought to avoid. This very common problem (almost universal where use cases are employed) is largely responsible for the widespread perception that "the requirements keep changing" during a project. Putman and Poole refer to this: "At the very beginning of any project longer than say a month, it is extremely unlikely, if not impossible, for the customer to know what will be required at the end of the project."

Amongst many other authors, Capers Jones has tackled this problem. His paper, "CONFLICT AND LITIGATION BETWEEN SOFTWARE CLIENTS AND DEVELOPERS", which is updated roughly annually, based on ongoing research, claims that "State of the art requirements gathering and analysis techniques can reduce the volume" of perceived requirements change by (extrapolating from his statistics) up to 90%. The secret is to separate real requirements (what the business needs to achieve) from product features intended to satisfy the requirements ("blueprints"). Requirements tend to be quite stable in most business environments, whereas solution products are often very *un*stable. It's perfectly possible for testing to make this distinction between Requirements and Features (that's the purpose of separating Test Case Specifications and Test Procedure Specifications in IEEE 829); but TDD, especially when done by programmers, runs the very severe risk of confusing the two. The dangers of this are far too well documented for me to need to repeat them here.

Test-First Development

I introduced the term, "test-first development", at the start of this essay, and wrote that it's different from "test-driven development". In TFD, test cases are designed in conjunction with the writing of a "test basis" document. At the beginning of a project, this may be a Business Requirements Specification.

As sections of the BRS are written, they are supplied to the test team for the design of test cases. A Test Case Specification documents the satisfaction of a business goal, without specifying *how* it will be satisfied. For example, we can specify that our bookshop customer will buy (say) three titles, qualify for a 5% discount, and have them delivered to a particular address. We can do this without knowing anything at all about whether the bookselling system is a software system or a "peopleware" system, or what interfaces the customers and staff will use to operate it. The requirements underlying the test case will remain substantially stable, even though the solution product and its interfaces may undergo radical change. The underlying requirements, and the implementation-independent test cases, may even be carried forward unchanged into future, different, solution products.

By developing test cases first from the Business Requirements Specification, before even thinking about the solution product and its interfaces, we achieve at least four things:

1. The clarity of the "as-written" requirements can be checked. If ambiguities or evident omissions prevent us from even specifying test cases, the product developers are going to be in trouble when they come to design a conforming product. It's far easier to check the clarity of a written statement than that of an oral statement.

2. The accuracy and completeness of the requirements can be checked. Each test case serves as an illustration of what the written requirements really mean, and its specified outcome can be checked by the customer. When the test case shows that the book sale results in a 5% discount instead of the intended 7% discount, the requirement error can be tracked down and fixed.

3. As problems are found with the ways in which requirements are specified, the specifiers (usually Business Analysts) learn to overcome their own weaknesses and to write clear, complete, unambiguous requirements statements. These may even directly incorporate basic test cases (see Tom Gilb, *Competitive Engineering*, to be published).

4. The completed, verified test cases serve as illustrations to help the development team understand exactly what any solution product must do with the requirements. Downstream, it may be possible to take a statistical approach to quality control by executing only a subset of the full test set (statisticians have suggested about one test case in thirty) to gain an accurate insight into how well the requirements have been met. (See Beizer on "The Threat of Testing" in *Software System Testing and Quality Assurance.*)

   Note that the test cases should cover both the required functions, and the required levels of performance.

5. In Test-First Development, we distinguish carefully between an agreed statement of what is to be achieved ("requirements") and the features that will achieve it ("design"). Given requirements, we first develop Test Cases based on them, *before* the developers design a conforming product that the Test Cases will validate. Downstream, the Product Design Specification is created on the basis of the Business Requirements, using the "Business Test Case Specifications" to validate its operation. But before building the Product, we use the PDS to specify operational Test Procedures which will interact with the designed features to *implement* the Test Cases; and this Test Procedure design process, again, serves to test and clarify the product design.

6. At first sight, this looks like a very linear, "Waterfall" approach to software development, but it can be applied equally well in iterative and evolutionary projects. "Despite what many would like to believe, any lifecycle that is worth its salt is going to be based on the waterfall lifecycle. Just as water flows downhill, it is best to go with the flow and understand what you are going to do; before you work out how you'll do it; before you'll actually do it; before you'll confirm that you did it. To not follow this approach is to invite disaster" (Jim Brosseau, "Editorial", *E-ssentials*, February 2002). But we should remember that "It's not a question of whether you are adopting a waterfall lifecycle, it's a question of how many drops and what size. Heading over Niagara Falls in a barrel can seem foolhardy, and smaller drops can readily be navigated with less risk."

7. Of course, Test-Driven Development can conform very well to Jim's prescription. Whether the resultant lack of a generalised specification is a problem or not depends on many factors, including the relative downstream stabilities of the requirements and the product features. Test-First Development is known to offer dramatic reductions in the time and cost of software development, and in the defect density of the delivered product, and can be implemented equally well in "waterfall" and "agile" environments. Whether Test-Driven Development (test cases as requirements) or Test-First Development (test-case design as requirements validation) is best for you will depend on the nature of your product -- and your customer's real requirements.

ARTICLE: A RECIPE FOR SUCCESS – NOT LIKE THIS

May I recommend to fellow computer cooks the following recipe for that old favourite, Savoury System Crumble. Far from being a delicacy, this dish is so gruesome that its taste will sometimes (perhaps always) linger for years.

Unfortunately, many people never master this rudimentary recipe, yet they refuse to leave the kitchen and constantly get under the feet of the more advanced practitioners.

The ingredients for the recipe are quire flexible and can be varied according to what is available or according to the way the mixture is turning out. The more skilled among you (the black belt-and-braces brigade) may notice the absence of any "coherent strategy", which would normally act to bind the other ingredients together. This is not an oversight, but is in keeping with the preferences shown by most people. In fact, this is the secret ingredient whose inclusion would be guaranteed to spoil the fun.

Take the following ingredients:

10 leaves of user requirements (well dried)
1 gallon of stock answers
3 analysts (or 1 ripe bunch of herbs)
8 medium programmers (or 4 rare ones)
cream of excellence
a pinch of enterprise (pinched from somebody else)
1 pint of experience extract (from a previous recipe)
1 or 2 fresh ideas (crushed)
8 bags of stuffing
1 flash of rare brilliance
essence of panic

Take the user requirements, and shred them finely into a bowl. Add the stock answers and stir well until all traces of the user requirements have dissolved.

Add the three analysts and continue stirring. At this point, the mixture may well curdle. If it does, discard it and begin again as there is no hope of a successful outcome. Provided that this stage is safely passed through, the colour should soon change to a bright green

Now the programmers should be added, slowly, and the mixture will begin to thicken. Actually, there is a strong likelihood that the whole lot will seize up solid, in which case you will be spared the remaining unpleasant steps.

Stir in the cream of excellence and simmer slowly for at least six months, allowing an occasional agitation from the chief programmer. His interventions have no effect on the mixture, which infuses itself (not to be mistaken for enthuses) into many different forms, quite oblivious from the outside world. During this period, there is a great risk of disintegration, as the different elements tend to separate out into quite distinct entities.

In the course of simmering, the characteristics of the base ingredients will have been entirely lost and replaced by a sticky mess that fouls up everything with which it comes into contact. Now the users (remember the users?) must be given a test sample (or specimen if it is particularly obnoxious). They will almost certainly find it totally unpalatable, with a quite unrecognisable flavour.

You should now add the pinch of enterprise and experience extract and turn the heat up high. Boil hard for two weeks, ignoring any spillages and vile odours. On no account should the fresh ideas be added, since this would make the mixture become quite clear and call into question the effectiveness of some of the other ingredients.

At the next sampling session, several of the users are likely to show signs of serious nervous disorder. Quickly pile in the stuffing, the flash of brilliance and the essence of panic. Inevitably the horrible mess will begin to fall apart, producing the well-known crumble mixture.

All that can now be done is to spoon the execrable remains onto a bed of soured relations, garnish with ladles of sweetener and grill until browned off. This last stage unfortunately causes the cream of excellence to evaporate.

This story was obtained from 'Datalink' in the mid 1980's by Peter Morgan, whose copy of the original article is lost. If it come to hand, he will give credit to the author on his web site, where the article also appears - www.nicemove.biz.

ARTICLE: COMMENTS ON THE ISEB PRACTITIONER EXAM

By… Anon

It is sometime since any feedback was given on the Practitioner exam, so here are some comments to candidates. These remarks are intended to assist students in both their preparations and in the exam sitting itself. Some items refer to general examination techniques, whilst others are more related to this particular exam, and as is always the case in such reports, parts will be more relevant to individual students, depending upon their background and experience.

Know how many questions you have to answer, and answer them in full.

It is surprising that some exam scripts are received with 5 or 6 questions attempted; it is only a matter of time before a paper is marked that answers all questions EXCEPT the first compulsory question. Burn into your mind: Question 1 is compulsory (worth 40%) and then choose any 3 from 5. Again, each portion of the chosen questions should be attempted. Candidates frequently throw marks away by not attempting all parts of questions.

Examination techniques alone will not get you a pass

The examiners are looking for both theoretical knowledge, and evidence that this can be applied. Therefore, candidates should ensure that they know the subject matter. Most candidates will know parts of the syllabus matter better than others, but the compulsory question can encompass ANY part of the syllabus. A recent 'compulsory question' had one section on Belbin team roles, and any well justified Belbin roles for the individuals detailed would score highly. Some candidates did not name any Belbin roles at all. They displayed no real knowledge of Belbin, and so were rewarded accordingly (zero marks for that section).

Plan your time

The following is a suggestion that will work for some individuals (if it does not work for you, find something that does and use it)

- 10 mins – read thoroughly through the whole paper
- 5 mins per question – PLAN your answer
- 30 mins (60 for compulsory question 1) – write your answer
- Any remaining time – read through the exam script and amend detail as necessary

Use the appropriate amount of time for each part question

From the outline scheme above, 2 marks will be awarded for every 3 minutes effort. If a diagram is requested in one of the earlier parts of a question, a good, well thought through drawing will probably make marks easier to accumulate in some later parts of the same question. In a diagram that merits (up to) 4 marks, it appeared that some candidates spent less than 1 minute (judging by the quality of the work), rather than the 'guideline' 6 minutes.

If under time pressure, write short notes

One of the best exam answers ever achieved by an exam marker was in a (computing-related) mock exam, where there were 6 minutes available to attempt the final question. Bullet-point short notes achieved 18 / 20, and an answer that was cited as a model answer before the rest of the class.

Three hours is a long time for an exam

So prepare yourself physically (eat before you write, and visit the bathroom) and mentally (ideally, attempt a full 3-hour practice examination). Many candidates have not taken a full 3-hour essay-style exam for 10 or 15 years. It is hard work. Your writing hand may be aching towards the end of the exam.

Answer the question set, rather than attempting to adapt it

There is an overall marking scheme set, and candidates that answer a 'different question' will score ZERO marks. The broad marking scheme allows discretion, so that a candidate who gives a valid and plausible reason that is 'an answer' but not on the marking scheme will be credited However it has to answer the question asked. One recent candidate used the acronym RIAD (Recognise, Investigate,

Action and Disposition – for the stages in the defect life cycle) three times in different questions. In all three instances, 'RIAD' was not appropriate or relevant.

Do not necessarily choose which questions you will answer right at the beginning, or rather be prepared to alter your choice as time passes

Frequently, an 'impossible' question becomes answerable as the subconscious gets to work. Reading through the whole exam paper at the beginning of the exam is essential and gives the subconscious something to work upon.

The first 50% of marks in any part of a question are generally the easiest to obtain

Write down what you know, and you will probably get some marks for it. A score of 1 out of 4 in a part question is not brilliant, but several such marks can be the difference between a pass and a failure, provided that there are some good solid answers elsewhere on the paper.

It is suggested that you do not leave the compulsory question until last, partly because time pressures can be significant on the last question

It is rather unusual for a candidate to pass the exam when scoring < 17 on the compulsory answer, and nearly impossible if the question 1 total is in single figures.

Many questions are scenario based

In these instances, answers should relate to the situation provided, and apply knowledge to the business problem, or sample of code. If the question asks what errors might static analysis have highlighted in the example, candidates should include real errors, rather that the kind of problems that static analysis could reveal. The question wanted an answer referring to the scenario. Failure to do this will probably be 'rewarded' by zero marks.

This is the PERSONAL view of one examination marker, who has been a regular marker over the last year. It is submitted with no name, as the individual does not want anyone taking issue with THEIR examination results. It has the broad agreement of several other members of the examination panel, without being officially sanctioned by that body.

Issue 12

THE TESTER

June 2005 Issue

NEXT CONFERENCE

Tuesday 21 June 2005

They Think Test's All Over

- **The View from Outside: Stories from an Outsourcing Lab**
- **We have decided to consolidate and update our IT Infrastructure ... How will we know it will work?**
- **UML: How do we test from Use Cases?**
- **When You're Tested: Strategies for the Test Audition**
- **Web sites that work: RNIB's Campaign for Good Web Design**
- **Failure Analysis and the Expert Tester**
- **Good Enough Software: Can We Ship Yet?**

IN THIS ISSUE:

THE BRITISH COMPUTER SOCIETY

FROM THE EDITOR

We had extremely good feedback from our Conference in March The papers proved to be not only informative but there was also some good humour injected throughout!

The winner of the Best Presentation Award for 2004 was Lloyd Roden, so many congratulations to Lloyd!

You will be interested to know that we are hoping to be able to fund special project(s) relating to testing again at some point in the near future. This depends on your attendance at our conferences to increase our income over expenditure, and we would welcome any suggestions you may have that can be brought to the attention of the Committee for consideration. So a double whammy to your attendance – you keep up to date on testing issues and also further testing related projects!

We are starting in this issue a regular feature about the SIGIST library. The library is available for you to borrow testing books free of charge and you will find in this edition, and on our web-site, how this process works. We would also welcome book reviews that you have written so please let us know if you would like to volunteer for inclusion in The Tester or for our conference sessions.

Well it's that AGM time of year (see Agenda) so we look forward to hearing your comments/questions at the June Conference. We do want to hear your views!

Pam Frederiksen
Communications Secretary
Tel: 01483 881188 (Leysen Associates)
Fax: 01483 881189
Email: pam@leysen.com

BCS SIGIST website: www.sigist.org.uk

SIGIST Standards Working Party: www.testingstandards.co.uk

FUTURE SIGIST CONFERENCE DATES

21 June 2005

20 September 2005

9 December 2005

BOOKING INSTRUCTIONS

1. Download a booking form from:
   http://www.sigist.org.uk/bookingForm.pdf

2. Complete and fax to:
   Colin Chivers
   01793 480270

   OR Post to:
   Colin Chivers
   Specialist Groups & Branches Accounts
   1 Sanford Street
   Swindon
   SN1 1HJ

NEXT MEETING - PROGRAMME

BCS SIGIST - They Think Test's All Over

Tuesday 21 June 2005
Royal College of Obstetricians and Gynaecologists, 27 Sussex Place, Regent's Park, London NW1

08:30	Coffee & Registration, Exhibition opens	
09:25	Introduction and Welcome – Stuart Reid, SIGIST Chairman	
09:30	**Featured Speaker** **The View from Outside: Stories from an Outsourcing Lab** *Jonathan Bach, Quardev Laboratories*	
10:30	**AGM**	
10:40	Networking session and commercial break	
10:50	Coffee & opportunity to visit the exhibition	
11:20	**Book Review**	**Workshop** **UML: How do we test from Use Cases?** *Richard Warden* *Software Futures Ltd.*
11:35	**TBA**	
	We have decided to consolidate and update our IT Infrastructure ... How will we know it will work? Danny Williams, IBM Global Services	
12:50	Lunch & opportunity to visit the exhibition	
13:50	**Web sites that work: RNIB's Campaign for Good Web Design** *Julie Howell, RNIB*	**Special Session** **When You're Tested: Strategies for the Test Audition** *Jonathan Bach, Quardev Laboratories*
14:50	Tea & opportunity to visit the exhibition	
15:20	**Failure Analysis and the Expert Tester** *Geoff Thompson, Experimentus Ltd.* *Stuart Reid, Cranfield University*	
16:05	**Featured Speaker** **Good Enough Software: Can We Ship Yet?** *Jonathan Bach, Quardev Laboratories*	
16:50	Closing Remarks	

WORKSHOP / SPECIAL SESSION

This Special Session at 11:20 is a 90 minute workshop with **Richard Warden** of Software Futures. Places may be limited and will be available on a first-come, first-served basis on the day. There is no advanced booking and no additional fee. If you are planning to attend this workshop please send example UML models or problems from your own projects to Richard at warden@globalnet.co.uk. This will be an interactive workshop with 2-way interaction. Richard will be looking to help those attending by discussing their concerns with UML.

The Special Session at 13:50 is a 60 minute workshop with **Jonathan Bach** our featured speaker. Places may be limited. They will be available on a first-come, first-served basis on the day, there is no advanced booking and no additional fee.

The SIGIST committee reserves the right to amend the programme if circumstances deem it necessary.

BCS SIGIST ANNUAL GENERAL MEETING 2005 AGENDA

To take place at 10.30am 21 June 2005.

1. Minutes of Previous AGM and Matters Arising

2. Reports
   - Chair
   - Treasurer
   - Standards committee

3. Constitutional amendments

   Change clause 4 (c) from:

   "The officers shall be elected by the Annual General Meeting (AGM) and shall serve from the end of the meeting at which they are elected until the end of the AGM following"

   To:

   "The standard term of office for all officers is three years from election. The officers shall be elected by the Annual General Meeting (AGM). Re-election is possible."

   Change clause 6 (a) from:

   "Each year the SIGiST shall hold an AGM in May."

   To :

   "Each year the SIGiST shall hold an AGM with in 14 months of the last. "

4. Elections

   - Committee
     - Chair
     - Vice-chair
     - Secretary
     - Programme Secretary
     - Communications Secretary
     - Marketing
     - Web site Secretary

5. Any Other Business

SIGIST Library

Looking for a testing book but not sure which topics are covered? Or are you trying to decide which testing book to buy? Or do you simply want to increase your testing knowledge? If the answer to any of these questions is 'yes' then the SIGIST Library could help!

The SIGIST Library has lots of testing books covering a variety of topics and they are available to borrow for a period of 4 weeks - free of charge. Extended loans are allowed as long as the book has not been requested by another SIGIST member.

Topics include (amongst others) Requirements testing, Reviews/Inspections, Test Management, Techniques, Test Process Improvement

If you would like to know more about the library and books available, or for any queries, please contact Julie Gardiner on 07974 141436 or email her at gardinerjulie@yahoo.co.uk. Alternatively, download the book loan form on the SIGIST website www.sigist.org.uk. Happy Reading!

SPEAKER ABSTRACTS AND BIOGRAPHIES

Featured Speaker:

Jonathan Bach

Quardev Laboratories

The View from Outside: Stories from an Outsourcing Lab

Abstract:

Outsourcing is a hot topic these days. Much has been said in conferences and forums about the failures when companies hire other companies to do testing for them and what others must do to avoid perils such as communication breakdowns, time zone differences, and shoddy workmanship.

But what's the view from the vendor providing the services? Imagine a group of trained testers in a lab, having a nice game of darts. The hotline rings, and it's someone in need of help. But what next? Is it a scramble for bunker coats and axes, or is it an organized slide down a fire pole to the waiting firetruck below? What can you expect from these trained strangers who show up at your door?

This talk is about the life of a tester in an outsource lab. It's about the questions to ask of us and the questions you may be asked when you tell us there is more testing to be done than your existing staff can handle. It's about the many clients who call us, the discussions we have, and the training we undergo in waiting for that hotline to ring so that when we show up at your door, we are armed with the tools to help your project succeed.

Biography:

Jon Bach is a senior test consultant and manager for Quardev Laboratories (www.quardev.com) - a Seattle, Washington outsourcing lab specializing in rapid, exploratory testing. In his ten-year testing career, Jon has led projects for small companies and large corporations, including 3 years at Microsoft where he was a test manager.

Jon is both a testing philosopher and a practitioner. He has presented testing techniques at testing conferences and workshops and has published articles in testing magazines. He has developed a new orientation method for testers called "Open-Book Testing" and is a co-inventor (with his renowned test expert brother James) of Session-Based Test Management. In spring 2006, he will serve as Program Co-Chair for the first conference sponsored by the Association for Software Testing.

Danny Williams

IBM Global Services

We have decided to consolidate and update our IT Infrastructure...How will we know if it works?

Abstract:

A Human Resources consultancy decided that they should consolidate their IT infrastructure by centralising their core services in a new Data Centre. Such a bold move was the first of this type of project for them globally and they wanted to make sure they did it correctly. A large bank decided that it was time to refresh their UNIX platform in preparation for a server consolidation. They wanted to ensure that the new platform was able to deliver what was required of it. Danny had the honourable title of Testing Manager for both customers. He will lead you through the trials and tribulations of trying to get a bunch of techies to design, build and test a new IT infrastructure without using fag packets and bits of wet string.

Biography:

Danny Williams is both a Systems Management Consultant and Testing Consultant working for IBM Global Services in the e-business Infrastructure & Management Practice. His knowledge of testing and systems management processes and methodologies is built upon a strong foundation of technical experience across a wide range of industries including human resources, banking, retail and public-sector.

Featured Speaker:

Jonathan Bach

Quardev Laboratories

When You're Tested: Strategies for the Test Audition

Abstract:

Software testing is a process of information discovery. But not all problems are easily found in the user interface, so it takes a broad combination of skills to unearth and report them -- creativity, technical ability, scientific thinking, diplomacy, and communication, to name a few.

The same is true with job interviews. Finding a person with all of these skills takes the *same* testing skills -- creativity, technical ability, scientific thinking, diplomacy, and communication.

Just like software, candidates come to an interview with their problems hidden. The interviewer has to quickly obtain enough information about the candidate's strengths and weaknesses to be confident that the candidate can be useful on the job. The talk will focus on experience interviewing hundreds of candidates over the past 10 years and a demonstration of a variety of proven interactive techniques (and a few unproven ones) to help you make the most of the testing interview, no matter which side of the clipboard you're on.

Biography – see page 5

Richard Warden

Software Futures Ltd.

UML: How do we test from Use Cases?

Abstract:

In UML the Use Case model describes the functional requirements of an application, and may be the basis for systems and acceptance testing. However there seem to be as many ways of writing Use Cases as there are organisations writing them. Some contain enough information to start test design but others are too abstract, incomplete or obscure to be of much use.

The aim of the workshop is to help delegates understand what testers need from a Use Case model for effective test design. We know that testers cannot live by Use Cases alone. We can examine the UML model set and identify which additional models we would like. For example an Activity diagram for each Use Case or, if there is significant state behaviour, we may need a Statechart diagram probably at the business object or systems level. Understanding the data is a vital element of test design but how does UML help? Use Cases do not model data, which is primarily the role of static models, e.g. Class and Type diagrams, and interaction models, which can show how data is passed between objects. Furthermore Use Cases do not describe non-functional requirements, so how are they integrated with the UML models?

We would like delegates to bring along models or experiences from their UML projects so we can explore these issues using meaningful examples. We wish to encourage an active debate so the format is relatively free and open for people to contribute. This workshop cannot solve all the problems. However it is an opportunity for you to raise your concerns, share experiences with others and leave with some new ideas and insights.

Since UML 1.1 was published in 1997 testers have been running to catch up with its significant implications for the way we work. In February 2003 the workshop leader, Richard Warden, gave a presentation to the SIGIST on the challenges that UML presents to testers (available as a download on the Papers page at www.softwarefutures.co.uk). Then, of 120 at the meeting, only six showed an interest in UML. We know that in the last two years involvement with UML has grown very considerably, with major companies adopting it as part of their IT strategy. What we have not achieved within the testing community is a critical mass of interest so we can develop our best practices. So please join us for this workshop.

Biography:

Richard Warden has been an independent IT consultant for the last 14 years trading as Software Futures Ltd (www.softwarefutures.co.uk). For the last seven years he has worked extensively with UML; testing systems, developing techniques and courses and providing training and consultancy. For nearly two years he was a test consultant to the Swiss Exchange, working on UML-based trading systems. Following that he was the principal test consultant to Semaphore Europe, an object-technology company. In partnership with Testing Solutions Group they recently formed the UML Testers' Forum, which holds its inaugural meeting on 28th February 2005.

In terms of personal history Richard wrote his first computer program in 1970, and it is still Millennium compliant! His 29 years experience in the DP/IT/ITC industries encompass work as an analyst, designer, programmer and tester on RAF defence support systems, followed by work on business and CAD systems for RACAL Electronics in management roles of testing, development and quality assurance. He then worked for K3 Group as product and research manager, with an emphasis on client-server systems in financial applications, before gaining his independence.

Julie Howell

RNIB

Web sites that work: RNIB's Campaign for Good Web Design

Abstract:

There are more than 9 million disabled people in the UK. Developments in assistive technology mean that blind and partially sighted people, deaf and hard of hearing people and people with physical and learning disabilities can enjoy the web. However, the quality of a disabled web user's online experience is dependent on the way a web site is designed. RNIB's Campaign for Good Web Design reminds web designers and commissioners of the importance of universal design principals, dispels myths about disability and technology and provides guidance on the requirements set out in the Disability Discrimination Act. Julie Howell leads the Campaign and will describe the campaign aims and achievements to date. She will include information about the Disablity Rights Commission's 2004 research into web accessibility in the UK and will provide an update on the development of the British Standards Institute Publically Available Specification (PAS) on web accessibility.

Biography:

Julie Howell joined RNIB, the UK's largest charity for people with disabilities, in 1994. Initially a member of the Research Library team, she was appointed to the position of Web Site Editor in 1997. In 1999, she was promoted to the Public Policy Department where she established RNIB's Campaign for Good Web Design, a national initiative to raise public awareness of the benefits of making web sites accessible to disabled people.

In 2003, Julie was promoted to the new position of Digital Policy Development Manager. She works with policy makers, information architects, manufacturers and software designers, businesses across all sectors and the Government to ensure that digital information, products and services are accessible to disabled people. In 2003, Julie was named among the top 50 most influential people in the new media industry by New Media Age magazine.

Julie has been an active member of the W3C Web Accessibility Initiative (WAI) since 1999. She was on the project team that produced the 1999 WAI film 'Web Sites That Work'. She is author of the 2000 RNIB campaign report 'Get the message online: making Internet shopping accessible to blind and partially sighted people', and edits 'Get the net', a column about Internet technology in RNIB's monthly 'New Beacon' magazine.

Julie advises the UK Government on web design policy. She has provided consultancy on drafts of the Cabinet Office guidelines for Government web sites. She represented RNIB on the Cabinet Office committee that published 'The Quality Framework for UK Government Website Design'.

She has advised numerous UK and international companies and organisations - notably Tesco.com, BBC, the British Bankers' Association, Adobe Systems and Macromedia - on strategies for making consumer-facing e-services and solutions accessible to disabled consumers. She represents RNIB on numerous new media industry awards panels, including the Government Forum Internet Awards, Local Government Internet Awards, Charity Times Awards, BAFTA

Interactive Entertainment Awards, British Interactive Media Association Awards, Revolution Awards and New Statesman New Media Awards. During 2003, Julie assisted the Disability Rights Commission in its Formal Investigation into Web Accessibility.

In 2004, Julie was invited to represent the needs of disabled people on the NHS IT Task Force. Following this she was invited to join the NHS Care Record Development Board as a patient representative, working to ensure that the forthcoming NHS National Care Records Service is accessible to disabled people. She is also a member of the NHS Direct New Media Committee.

In 2005, Julie assumes the role of Technical Author on a new BSI process standard for accessible web design.

Julie undertakes a rigorous programme of outreach and education to raise awareness of the value and importance of inclusive design to the digital design community and those who influence and regulate it. She has presented papers at more than 50 national conferences, and regularly provides commentary on digital access issues on international TV, radio and in the press.

Julie is a chartered member of the Chartered Institute of Library and Information Professionals (CILIP) and a member of CILIP's UK e-Information Group (UkeIG). She is a Fellow of the Royal Society for the Encouragement of Arts, Manufactures and Commerce (RSA). She is also a member of the British Web Design And Marketing Association (BWDMA) and the Usability Professionals Association (UPA).

Geoff Thompson & Stuart Reid

Experimentus Ltd. Cranfield University

Failure Analysis and the Expert Tester

Abstract:

Otto von Bismarck stated that “Fools say they learn by experience, I prefer to profit from other people’s experience”. In the software development world, where fully successful projects are so rare, we have the perfect opportunity to learn from the failures of others. Software disasters range in scale enormously - they can impact our everyday lives in many ways from the inconvenience of a lost telephone call (perhaps along with millions of other callers) to the tragic failure of a life support system. How should we learn from these disasters to make ourselves better testers? How should we profit from other projects’ painful experiences? This presentation will introduce the philosophy that expert software testers are those that know the most about how systems fail, as they are in the best position to recognise the symptoms early in their own projects.

Today the demand for new technology is continuously driving down the time to market for new applications and so the time available to understand and manage the risks of this new technology is getting shorter and shorter. Also, the potential scale of disaster that can be caused by software is now practically immeasurable. So far the software industry has been “lucky”; major software disasters have lost millions of euros, but have not caused major loss of life. But is our growing dependence on new technology just a disaster waiting to happen? It will be argued that there is an alternative to trusting to luck - engineering. A mature engineering process relies on learning from experience; and we can apply this philosophy to software testing. By using historical data it is argued that the software development and software testing processes can together be less prone to failure, and therefore be more risk averse.

The subject of software disasters has been the basis of many presentations at testing conferences worldwide over the years. Many of these presentations have tried to show how specific failures could have been prevented if a particular technique had been applied. However, in this presentation we consider failures to be more useful when they have been made generic – into what we have called patterns of failure. In order to identify such patterns causal analysis of specific failures is applied, from which generic patterns of failure are then derived. It is then argued that it is these patterns that provide the knowledge in a form that experts can use both to prevent and detect future failures.

A number of case histories of real-world software disasters are used to provide an insight into how and why software systems fail. Software failure analysis is reviewed to show how the root cause of failure may be identified, or in some instances why it is never identified!

To illustrate the learning process, throughout the presentation disaster examples are drawn from the complete range of computer applications, such as transport, medical, telecoms, financial, space, emergency services, military, even to the demise of high-tech toilets. Software system failures are analysed from a number of contrasting viewpoints, providing management and technical as well as theoretical and practical perspectives.

The presentation will evidence that failure analysis is an integral part of a risk based approach to development, and therefore of a mature development process.

The conclusion being:

- That by following the basic principles of failure analysis testers can move from being craftsmen to engineers.
- That the communication of the results of these failure analyses is an integral part of being a professional member of the software testing community.
- That by knowing about potential failures makes the tester better able to prevent and detect future faults.

Biographies:

Geoff Thompson has been involved in testing for over 16 years. He has managed testing on many £M projects. Geoff is recognised by his peers as an expert in test process and Test Strategy, Test Management and process improvement. He has recently established his own successful Testing Consultancy in the UK, Experimentus.

He was the UK nomination for the EuroSTAR Testing Excellence award in 2003, and won the Test Manager of the Year award in 2004.

Geoff has many testing interests outside of those that earn him a living. He is the Vice-Chairman of the BCS SIGiST committee. He is also a founder member of the Information Systems Examination Board (ISEB), having specific responsibility for the delivery of the Practitioner level syllabus that is in use today. He was directly involved in the creation of, and is currently UK Representative on, the International Software Testing Qualification Board (ISTQB) where he is also one of the authors working on the new ISTQB Software Testing Foundation certificate. Lastly he is a founder member of the TMMi Foundation which is looking to standardise the use of the Test Maturity Model.

Stuart Reid has a PhD in Software Testing and is a Senior Lecturer in Software Engineering for Cranfield University at the Royal Military College of Science. He has previously worked on the development of high-integrity systems, such as command and control, radar and avionics. He is currently involved with research on the effectiveness of software testing techniques, high volume automated testing and standardisation.

He is Chair of the BCS SIGIST Committee and the associated Standards Working Party, which was responsible for the development of BS 7925-1 and BS 7925-2. This working party is now developing a standard on non-functional testing techniques.

Until September 2003, he was Chair of the ISEB Software Testing Certificate Board, which is responsible for the Foundation and Practitioner Software Testing qualifications.

He has presented papers and tutorials at conferences worldwide.

Stuart was awarded the EuroSTAR Testing Excellence Award at Stockholm in 2001. This annual award is a peer-group and industry award given to people who have had significant influence in improving the industry.

Featured Speaker:

Jonathan Bach

Quardev Laboratories

Good Enough Software: Can We Ship Yet?

Abstract:

Some QA professionals mistakenly think that "Good Enough" software development principles are irresponsible. They say it's a method that lowers quality because it's an excuse to ship mediocre products. But this perception is due to a simple misunderstanding of what "Good Enough" means.

"Good Enough" does not mean "substandard" or "mediocre", but is actually an optimal and responsible economic principle we all use. Since quality is expensive, the "Good Enough" framework provides criteria to help stakeholders make proper decisions about whether or not it's time to ship, because it is always a discussion about economy. The four elements of the "Good Enough" paradigm will be explained as well as how testers and test managers can help project stakeholders know if they are shipping with too little quality or are unnecessarily striving for too much quality.

Biography see page 5

BCS CMSG CONFERENCE & EXHIBITION - 21 & 22 JUNE 2005

The World of Change, Configuration and Release Management

Following on from the success of our 1st Conference, the 2nd British Computer Society Configuration Management Specialist Group (BCS CMSG) Conference and Exhibition will be on 21 & 22 June 2005, at Homerton College in Cambridge, UK.

Configuration Management goes hand in hand with testing to support quality development practices. How do we ensure that we are testing the correct configurations? Can we reproduce problems from older releases?

We welcome both newcomers and old hands to the BCS CMSG sharing experiences and lessons learnt. The event has been planned to encourage a high degree of peer-to-peer interaction and discussions in both formal and informal situations. This will be a chance to renew old acquaintances and interact with those who have a lot of real experience in change, configuration and release management. To this end we look forward to welcoming delegates on Monday night for drinks and dinner before the first conference session on Tuesday morning.

Who Should Attend

For managers and professionals involved in change, configuration and release management.

Whether you are managing or doing, this conference will provide pragmatic tools and information for controlling and delivering your projects and programmes and the subsequent services more effectively.

What You Will Learn

The conference themes include large-scale systems programmes and projects, CM in software development in commercial environments and CM across the lifecycle to including service management. Each track features both 'how to' case studies as well as the latest theories. You will find experienced change, configuration and release practitioners on the podium and in the audience. Ample time has been built-in for networking with other delegates who are undertaking the same journey as you. You will also be able to take advantage of leaders in the industry who will be available to you to answer your questions and to impart their hard won wisdom.

Delegate comments from the previous conference included:

"The event has surpassed my expectations. I found the presentations both relevant and interesting. It's wonderful to meet people who appreciate the everyday problems and resistance encountered to CM and to find answers to many questions."

For details on the full programme and online registration, please see:

http://www.bcs-cmsg.org.uk/conference/2005/index.shtml

ARTICLE: THREE CHEERS FOR THE BLACK, WHITE, AND GREY!

Don Mills, Macroscope Services Ltd., London

What everybody knows

Most testers know about Black Box Testing and White Box Testing.

At least, most know *something* about them, though there seems to be some confusion about exactly what they refer to and exactly when they're relevant to the testing experience.

I have to acknowledge that there is no World Testing Authority that has mandated the uses and meanings of these terms from on high (though there have been attempts, of course), and that my interpretations may not be the same as yours. Mine are, of course, the *correct* interpretations, but then, yours are too.

Let me start with what motivated me to write this article. Many of you who read it will have sat and passed the ISEB "Foundation" exam. Some of you will have delivered, or even created, ISEB training courses. Anyone who's been involved with Foundation-level training will realise that "black boxes" and "white boxes" (but not "grey boxes") cast long shadows in the *ISEB Software Testing Foundation Syllabus* ("the *Syllabus*", see www.bcs.org/BCS/Products/Qualifications/ISEB/Areas/SoftTest/syllabus.htm). They appear in the third major Section, "Dynamic Testing Techniques", where (under the first Topic, "Black and White box testing") we read as follows:

> Black box is relevant throughout the life cycle whereas, in general, additional white box is appropriate for sub-system testing (unit, link) but becomes progressively less useful towards system and acceptance testing. System and acceptance testers will tend to focus more on specifications and requirements than on code.

There's nothing objectionable in this, of course; it's the next two Topics of the *Syllabus* that cause me problems, as a testing trainer and practitioner myself, because they tell us that there are "Black box techniques as defined in the BCS standard", and "White box techniques as defined in the BCS standard", and there (in fear and trembling) I embolden myself to disagree. Not absolutely—there may be some truth in this division—but, for the most part, I respectfully disagree.

It must be true—it's there in black and white!

The relevant standard is the BCS *Standard for Software Component Testing* ("the *Standard*"), which you can view or download at http://www.testingstandards.co.uk/bs_7925-2.htm. There's also the related BCS *Glossary of Testing Terms* ("the *Glossary*"), which you can find at http://www.testingstandards.co.uk/living_glossary.htm

Here are some relevant definitions from the *Glossary*:

> **black box testing:** See functional test case design.
>
> **functional test case design:** Test case selection that is based on an analysis of the specification of the component without reference to its internal workings.
>
> **white box testing**: See structural test case design.
>
> **structural test case design:** Test case selection that is based on an analysis of the internal structure of the component.

"Black box testing" is so called (obviously) because the box you're testing is painted black so you can't see what's inside. "White box testing," on the other hand … hmmm, okay it's also (less commonly) called, "glass box testing" (as the *Glossary* indeed notes).

I have a quibble about the "white box" definition (which we'll come to at the end), but these definitions are reasonable. The mischief (as I see it) comes from the way the *Standard* classifies its "test design techniques" as either "black box techniques" or "white box techniques", and the requirement in the *Syllabus* to teach the same division.

In fact, the *Standard* doesn't classify all of the thirteen techniques it discusses; the term, "white box", doesn't appear at all (except in passing, in an illustrative Test Plan), but four techniques are explicitly

described as “black box” and three others explicitly as “structural”. Reading between the lines for the remainder, we derived this table of “black box techniques” and “white box techniques”:

Black Box	***White Box***
Equivalence Partitioning	Statement Testing
Boundary Value Analysis	Statement Testing
State Transition Testing	Data Flow Testing
Cause-Effect Testing	Branch Conditions
Syntax Testing	Branch Condition Combinations
Random Testing	Modified Condition Decisions
	LCSAJ Testing

Table 1: "Black box techniques" and "white box techniques"

In terms of the definitions already given, then, the techniques on the left could be described as “requirements-based”, and those on the right as “code-based”. This, in effect, is what students on ISEB-compliant courses learn; and never the twain shall meet.

But life isn’t nearly so black-and-white(!). Between requirements and code, there lies the system architecture, the realm of integration testing. The system architecture identifies what components are required, and their interrelationships; “integration” fits them together; and “integration testing” measures the goodness of the fit. Where does this lie in terms of “black box” and “white box” testing?

Enter “grey box testing”.

The product architect’s view lies between the “black box” of requirements and the “white box” of code; she or he sees both the inside (HOW—the “solution space”) and the outside (WHAT—the “problem space”), and provides the mapping from one to the other. This leads to *design-based* or *architecture-based* testing, a.k.a. “grey box testing”.

In a waterfall-style process, the three types of testing stack upon one another as shown in *Figure 1: Stacked Boxes*. (If you don’t practice a waterfall method, the principles may still remain true.) The top two boxes, if we’re purist about it, represent respective “external” views of what the product is *required* to do, and what it *will* do. The three bottom boxes show how component testing has opportunities to use black box, white box, *and* grey box techniques. And in the middle, architecture-based testing sees the components that are inside the system, but not what’s inside the components themselves.

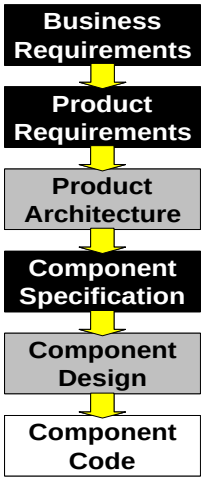

Figure 1: Stacked boxes

There’s not a lot about grey (or “gray”) box testing in software testing books, and I’m not going to go into any detail here (that’s not my purpose), but the following quotation from Kaner, Bach, and Pettichord (*Lessons Learned in Software Testing*, Lesson #289) provides some insight:

> Even though you probably don't have full knowledge of the internals of the product you test, a test strategy based partly on internals is a powerful idea. We call this gray box testing. The concept is simple: If you know something about how the product works on the inside, you can test it better from the outside. This is not to be confused with white box testing, which attempts to cover the internals of the product in detail. In gray box mode, you are testing from the outside of the product, just as you do with black box, but your testing choices are informed by your knowledge of how the underlying components operate and interact.
>
> Gray box testing is especially important with Web and Internet applications, because the Internet is built around loosely integrated components that connect via relatively well-defined interfaces. Unless you understand the architecture of the Net, your testing will be skin deep. Hung Nguyen's Testing Applications on the Web (2000) is a good example of gray box test strategy applied to the Web.

In the *Standard* and in the Syllabus, “black box” and “white box” are applied to specific techniques of test modelling (a better term than “test design”, in this context); I’m not prepared to go so far, for

reasons we'll see below. Those two *styles* of testing, along with "grey box", don't refer to the types of test model we build (or they shouldn't, and I'll come to that below), but the *test bases* (sources of testing information) we use in constructing the models. My own definitions (for what they're worth) are:

> **black box testing.** Testing which uses requirement specifications as the test basis for testing externally visible properties of a test item, while ignoring internal structural details.
>
> **grey box testing.** Testing which uses internal software design specifications (such as architectural use cases) as the source of test requirement information.
>
> **white box testing.** Testing which uses program source code as the test basis.

Test modelling techniques

The BCS *Standard for Software Component Testing* is a very useful reference in many ways, and its being a freebie is a big plus. All its parts are useful, even if there are quibbles about their content, and here comes a quibble.

"Annex B" contains reasonably detailed descriptions, with worked examples, of the thirteen "test design techniques" outlined in Clause 3. My quibble is that the names given to the techniques are the names either of *test analysis techniques* or *test modelling techniques*. Each technique operates around a more-or-less formal model of what the test item does, or of how it does it. For this reason, I prefer to think of "cause-effect graphing" or "control flowgraphing" as "modelling techniques": they result in "test models" (of types called "fault models"), out of which we abstract test paths or test points, from which, in turn, we design test cases and test procedures.

Okay, quibble expressed, now for the non-quibble part. It's unfortunate, to my mind, that ISEB students are taught that "*these* techniques are used with requirements", and "*these* techniques are used with code", and (as said before) never the twain shall meet. Unfortunate, because I don't believe it's true.

Example: "White box techniques"

This is most obvious in the case of the "white box techniques", all of which use some form of flowgraph (or flow graph) as an underlying model. The BCS *Glossary* doesn't define the term, *flowgraph*, though it appears seven times in the BCS *Standard* (as "flow graph"), so here's my own definition:

> **flowgraph.** (also **flow graph**) A form of directed graph representing the flow of control through some process (such as a program or a business system).

A **directed graph** is one in which all links—the lines between the activity boxes or "nodes"—are "directed links", which have "a mandatory direction from a source node to a target node". The arrowheads on the lines between boxes in flowcharts illustrate the concept of "directed links".

Flowcharts are a form of flowgraph familiar to many people—more so outside of the IT industry than within it nowadays, I suspect. And that brings me to the nub. Flowcharts, flowgraphs, UML Activity Diagrams, are all ways of modelling a *process*, and the process *doesn't have to be inside a software component*.

Scott Ambler's *The Object Primer* is described as "an important book for agile modelers, describing how to develop 35 types of agile models including all 13 UML 2 diagrams." In chapter 9, he introduces flowcharting as "a modeling technique introduced in the 1940/50s and popularized for structured development in the 1970s (Gane and Sarson 1979) as well as business modeling".

Ambler comments, most pertinently, that "because object methods are much smaller [than COBOL routines] flowcharts have dropped out of favor with programmers in recent years. That's okay, they're still useful for process modeling."

He shows how modelling an existing use case specification as a flowchart "picked up some logic errors" in the use case, and remarks that going through the flowchart model with the project stakeholders would help resolve the issues "right away".

His example illustrates the use of flowcharting as a "static testing" technique. "But," he says, "once we've fixed that problem the flowchart isn't going to be of much value any more."

Well, wrong. Any of the techniques listed in *Table 1* above as "white box techniques" could be applied to develop covering test sets—or, better, a basis test set—of the use case. Flowgraphs (or the more detailed flowcharts) are very useful models of the *processes* an actor will follow in interacting with a system or component. Flowgraphs are also a good solution if you are faced with UML Sequence Diagrams, since a Sequence Diagram is required to show only a single thread through a use case. Bob Binder remarks (*Testing Object-Oriented Systems*), "This fragmentation creates opportunities for errors and makes it difficult to decide when a [covering] test suite has been developed." His solution?—"derive a composite control flow graph from a [set of Sequence Diagrams, and use] established control flow test models". These are the ones listed in the "White box" column of *Table 1*, and described and illustrated in Annex B of the BCS *Standard*.

Example: "Black box techniques"

In the ISEB tradition, "Equivalence Partition Testing" and "Boundary Value Analysis" are among the "black box techniques", as also is cause-effect testing ("cause-effect graphing"). The implication is that these techniques are for testers working from specifications, and have no relevance to testers working from code. Nothing could be further from the truth.

This is not the place to go into details, but cause-effect testing (for instance) derives a model in which the *external* behaviour of the test item is represented by a decision table.

At least, some aspects of the item's behaviour are, specifically those to do with possible combinations of input values, and the combinations of output values (or other types of externally visible behaviour) that are supposed to result. What the model *doesn't* include is any of the internal mechanisms (internal components and internal processes) that the test item uses to *generate* its outputs.

From that description, the application to "requirements-based testing" should be obvious. However, it's perfectly possible, and sometimes very helpful, to generate a cause-effect table from program source code. You do this by identifying and abstracting, from the code, "input conditions" that are supposed to trigger behaviours, and "output conditions" that are supposed to result, and suppressing the process between them. As those of you who were programmers in the 1970s may remember, decision tables of this sort, with their powerful but simple verification rules, were potent ways of uncovering bugs in program logic—especially combinatorial bugs, and bugs of omission. Doing the analysis and creating and navigating the model can perform very useful static testing of software components, just as it can with requirements specifications and functional specifications. And when you've debugged the model, and the test basis document you derived it from, you can then reuse the model to derive a covering set of test cases.

Before I leave this topic, let me present one more illustration of where "black box" techniques play a role in "white box" testing. Suppose you've drawn up a flowgraph of a process specification (it may be at the requirements level, or you may be working from code, it doesn't matter in the least). The process computes the price for sending packages by courier. You've used your flowgraph model to derive a basis set of paths, from which you are going to design test cases. The second decision point in your first path specification reads thus:

> **if** PackageWeight >= 2 kg **then** …

From your knowledge of Boundary Value Analysis, what sort of values come to mind that might be useful for a "true" case and for a "false" case out of this condition?

Wrapping it all up

What I have tried to do in this article is to demonstrate how the techniques called "white box techniques" in the ISEB materials, and which most Foundation students will therefore think are "not for them", may be exactly what they need for certain kinds of testing situation: those where the test basis includes a specification of processual behaviour, whether by a user of the product, by the product itself, or by a component of the product. Their use should not be limited to testing from program code: Boris Beixer's *Black Box Techniques* applies almost all of them (not LCSAJ or random testing) to deriving test cases from US tax forms (and finds the odd bug as a result).

Conversely, the so-called “black box techniques” are just as useful when your test basis is program source code. The trick is to abstract *externally visible behaviour*, while suppressing *internally defined procedures*.

Black, white, and grey: what do they refer to? Not the techniques you use to analyse the test basis for test conditions, or the models you build out of those test conditions, or the analysis you apply to the models to identify test paths and test points, nor yet to the way you design test cases and procedures to exercise those test paths and test points. What they refer to is your *source of information* for building the models:

But there is a clear difference between the types of technique in the “Black box” column of *Table 1*, and those in the “White box” column. The former model *behaviour*, regardless of how that behaviour is brought about; the latter model *process*, regardless of the level at which that process is defined.

And what about “structure”, the term used in the BCS *Glossary* definition of “white box testing”? (This brings me back to the quibble I mentioned at the outset.) “Structure” is modelled by techniques such as class diagrams or other, more concrete, software architecture modelling tools; it relates to the *interactions* of components (or, indeed, of systems).

The following definitions may be useful:

> **behavioural testing:** Derivation of test requirements and test cases from behavioural models of a system or component, regardless of internal process.
>
> **procedural testing:** Derivation of test requirements and test cases from process models at any level of description (requirements specification through to program source code).
>
> **structural testing:** Derivation of test requirements and test cases from structural models of a system or component, regardless of internal process.

In the software world, as we all know, the opportunities for error are innumerable yet ever increasing. Whether we are testers, analysts, designers, or coders, we need all the help we can get, and to present test modelling techniques in a way which seems to say, “the majority [by a small margin!] are only of use and interest if you can read the code”, seems to me to be an unfortunate lapse in the ISEB perspective.

Testers need capacious toolbags, and lots of tools to put in them.

Issue 13

THE TESTER

September 2005

NEXT CONFERENCE

Tuesday 20 September 2005

Whose Test is it Anyway?

- **How To Break Software**
- **What Testers can learn from Other Professions**
- **Supercharging Keyword-Driven Testing Using Model-Based Techniques**
- **Patterns for Testing**
- **Listening to What Your Bugs are Telling You**
- **Really Agile: Testing for ASP**
- **How To Break Software Security**

IN THIS ISSUE:

BCS

THE BRITISH COMPUTER SOCIETY

FROM THE EDITOR

It turned out to be a very lively AGM in June! It had been decided to have the AGM in the middle of the day when all attendees would be present, rather than its usual beginning of the day slot when there has been limited attendance in the past. This resulted in a very animated discussion on changes to the constitution, and although this ran well over time, it was good to have more input from members of the SIGiST!

However, we did get feedback that a number of attendees would not have attended the AGM out of choice, and because it was held between presentations were unable to opt out. It would therefore help us if you would email me to let me know your views on this so that we can discuss the timing at the next committee meeting - and plan for the AGM accordingly next year. If you can place 'AGM' in the subject line this will help.

Also for your information, following a dictum from the BCS, the word 'interest' has been removed from our title. The SIGiST now stands for Specialist Group in Software Testing.

With regards to the June conference, we had very good feedback on the varied programme of speakers. Jonathan Bach, brother of James, scored well for his Special Session in particular. Please note that attendance at Special Sessions is limited in number, and places are allocated on a first come, first served basis.

To ensure your attendance at the next conference please don't forget to send in your booking form in good time!

Pam Frederiksen
Communications Secretary
Tel: 01483 881188 (Leysen Associates)
Fax: 01483 881189
Email: pam@leysen.com

BCS SIGIST website: www.sigist.org.uk

SIGIST Standards Working Party: www.testingstandards.co.uk

FUTURE SIGIST CONFERENCE DATES

9 December 2005

15 March 2006

15 June 2006

19 September 2006

14 December 2006

BOOKING INSTRUCTIONS

1. Download a booking form from:
   http://www.sigist.org.uk/bookingForm.pdf

2. Complete and fax to:
   Colin Chivers
   01793 480270

   OR Post to:
   Colin Chivers
   Specialist Groups & Branches Accounts
   1 Sanford Street
   Swindon
   SN1 1HJ

NEXT MEETING – PROGRAMME

BCS SIGIST - Whose Test is it Anyway?

Tuesday 20 September 2005
Royal College of Obstetricians and Gynaecologists, 27 Sussex Place, Regent's Park, London NW1

08:30	*Coffee & Registration, Exhibition opens*	
09:25	*Introduction and Welcome – Stuart Reid, SIGIST Chairman*	
09:30	**Featured Speaker** **How To Break Software** *James A. Whittaker, Florida Institute of Technology*	
10:30	**Networking session and commercial break**	
10:50	*Coffee & opportunity to visit the exhibition*	
11:20	**Book Review**	
11:30	**What Testers can learn from Other Professions** *Peter Morgan, Nicemove Ltd.*	
12:10	**Supercharging Keyword-Driven Testing Using Model-Based Techniques** *Dr. Kelvin Ross,* *K. J. Ross & Associates Pty Ltd.*	
12:50	*Lunch & opportunity to visit the exhibition*	
13:50	**Patterns for Testing** *Don Mills* *Macroscope Services Ltd.*	**Special Session** **Listening to What Your Bugs are Telling You** *James A. Whittaker,* *Florida Institute of Technology*
14:50	*Tea & opportunity to visit the exhibition*	
15:20	**Really Agile: Testing for ASP** *Bogdan Bereza-Jarocinski, bbjTest*	
16:05	**Featured Speaker** **How To Break Software Security** *James A. Whittaker, Florida Institute of Technology*	
16:50	Closing Remarks	

WORKSHOP / SPECIAL SESSION

The Special Session at 13:50 is a 60 minute workshop with James Whittaker our featured speaker. Places may be limited. They will be available on a first-come, first-served basis on the day, there is no advanced booking and no additional fee.

The SIGIST committee reserves the right to amend the programme if circumstances deem it necessary.

SOFTWARE QUALITY CONFERENCES INCORPORATING ICSTEST® 2005

Queen Elizabeth II Conference Centre, London. 27th & 28th September 2005.

Tutorials 26th and 29th September 2005.

Join us at the UK's leading conference addressing the most important topics in quality and testing today.

An independent committee has selected real-life, case study based presentations that will share how they achieved success. Hear how Ericsson, EA games, HBOS. Marks & Spencer, npower, T-Systems, BCS, KLM, Philips, Dixons and Xansa improved quality.

Keynotes from British Airways, Ovum, Microsoft and T-Systems will be complemented by a presentation from Joe Simpson of "Touching the Void" fame addressing the issues of survival, self motivation and personal achievement.

Free interactive workshops run alongside the conference and full day tutorials the day before and after make SQC the premiere learning experience for quality and testing professionals.

BCS members benefit from a 10% discount to all conference events. To see more about the exceptional programme and the opportunities to listen to the leading exponents, visit www.sqs-conferences.com. To take advantage of the BCS discount book online and use the code BCS in your booking.

SIGIST Library

Looking for a testing book but not sure which topics are covered? Or are you trying to decide which testing book to buy? Or do you simply want to increase your testing knowledge? If the answer to any of these questions is 'yes' then the SIGIST Library could help!

The SIGIST Library has lots of testing books covering a variety of topics and they are available to borrow for a period of 4 weeks - free of charge. Extended loans are allowed as long as the book has not been requested by another SIGIST member.

Topics include (amongst others) Requirements testing, Reviews/Inspections, Test Management, Techniques, Test Process Improvement

If you would like to know more about the library and books available, or for any queries, please contact Julie Gardiner on 07974 141436 or email her at gardinerjulie@yahoo.co.uk. Alternatively, download the book loan form on the SIGIST website www.sigist.org.uk. Happy Reading!

SPEAKER BIOS AND ABSTRACTS

Featured Speaker:

James A. Whittaker

Florida Institute of Technology

Author: *"How to Break Software"*
Co-author (with Hugh Thompson): *"How to Break Software Security"*

How to Break Software

Abstract:

What is it that makes a really good tester? I mean a really good tester! What intuition do they have that leads them so surely to the best bugs?

We sought to answer this question by analyzing thousands of bug reports. We attempted to understand each bug in the context of its underlying causal faults, its failure symptoms and the best method to discover its existence. Thus, we would end up with understanding how bugs come into being, how to look for a bug's telltale signature and how we go about finding bugs in the first place. To our great surprise, we invented testing techniques that were not published anywhere. Thus, How to Break Software was created.

This talk is a fun, entertaining and educational journey through the testing techniques that make up How to Break Software. You will learn many new and novel ways to force software applications (of whatever variety) to fail. You will also see old, tried-and-true techniques explained in fresh ways. Prepare to learn why software fails and to force it to do just that.

Learn about:

- The life-preserver model of software behavior. What does software really do and how can it fail doing it?
- 17 ways to break software through its user interface or API.
- Ways to tell of your software is going to fail—before it actually does.
- How to simulate a faulty environment in order to test error paths and exception handlers.

And much, much more. How to Break Software is more than a discussion about software testing. It's about what it takes to increase your skills to the Jedi level—may the force be with you.

Biography:

James A. Whittaker is a professor of computer science at the Florida Institute of Technology. He earned his Ph.D. in computer science from the University of Tennessee in 1992. His research interests are software testing, software security, software vulnerability testing and anti cyber warfare technology.

He is the author of How to Break Software, How to Break Software Security (with Hugh Thompson) and over 50 peer-reviewed papers on software development and computer security. He holds patents on various inventions in software testing and defensive security applications and has attracted millions in funding, sponsorship and license agreements while a professor at Florida Tech. He also has served as a testing and security consultant for Microsoft, IBM, Rational and many more US companies.

In 2001 he was appointed to Microsoft's Trustworthy Computing Academic Advisory Board and was named a "Top Scholar" by the editors of the Journal of Systems and Software based on his research publications in software engineering. His research team at Florida Tech is known for its testing technologies and tools, which include the highly acclaimed runtime fault injection tool Holodeck. His research group is also well known for their development of exploits against software security, including cracking encryption, passwords and infiltrating protected networks via novel attacks against software defenses.

James A. Whittaker

Workshop: Listening to What Your Bugs are Telling You

Abstract:

Bugs are corporate assets. There is nothing that tells us more about what we're doing wrong and how to do things right than the bugs that we ship in our products. But how does one go about learning from bugs? This workshop shows a methodology for "listening" to your bugs. Join us to find out a simple process for listening to the lessons that bugs teach. There are lessons for developers, testers and manager in these bugs stories, but first you have to learn how to listen.

Biography:

James A. Whittaker is a professor of computer science at the Florida Institute of Technology. He earned his Ph.D. in computer science from the University of Tennessee in 1992. His research interests are software testing, software security, software vulnerability testing and anti cyber warfare technology.

He is the author of How to Break Software, How to Break Software Security (with Hugh Thompson) and over 50 peer-reviewed papers on software development and computer security. He holds patents on various inventions in software testing and defensive security applications and has attracted millions in funding, sponsorship and license agreements while a professor at Florida Tech. He also has served as a testing and security consultant for Microsoft, IBM, Rational and many more US companies.

In 2001 he was appointed to Microsoft's Trustworthy Computing Academic Advisory Board and was named a "Top Scholar" by the editors of the Journal of Systems and Software based on his research publications in software engineering. His research team at Florida Tech is known for its testing technologies and tools, which include the highly acclaimed runtime fault injection tool Holodeck. His research group is also well known for their development of exploits against software security, including cracking encryption, passwords and infiltrating protected networks via novel attacks against software defenses.

James A. Whittaker

How to Break Software Security

Abstract:

What is it that differentiates security bugs from run-of-the-mill functional bugs? What knowledge, insight and intuition must testers develop in order to train themselves to find and recognize security vulnerabilities?

We sought to answer this question by analyzing thousands of security vulnerabilities that shipped in major products, from operating systems to browser plug-ins, routers to firewalls. We attempted to understand each bug in the context of its underlying causal faults, its failure symptoms and the best method to discover its existence. Thus, we would end up with understanding how security bugs come into being, how to look for vulnerabilities' telltale signature and how to go about finding vulnerabilities in the first place. The end result was a body of knowledge called How to Break Software Security.

This talk is a fun, entertaining and educational journey through the security testing techniques that make up How to Break Software Security. You will learn many new and novel ways to force software applications (of whatever variety) to fail in ways that are exploitable by hackers.

Learn about:

- The life-preserver model of software behavior. What does software really do and how can it fail doing it?
- Why security bugs are often invisible to the human eye.
- 19 ways to break software in ways that are remotely exploitable.
- Ways to tell of your software is going to fail—before it actually does.
- How to simulate a faulty environment in order to test error paths and exception handlers.

And much, much more. How to Break Software Security is more than a discussion about software vulnerabilities. It's about what it takes to increase your skills to the Jedi level—may the force be with you.

Biography:

James A. Whittaker is a professor of computer science at the Florida Institute of Technology. He earned his Ph.D. in computer science from the University of Tennessee in 1992. His research interests are software testing, software security, software vulnerability testing and anti cyber warfare technology.

He is the author of How to Break Software, How to Break Software Security (with Hugh Thompson) and over 50 peer-reviewed papers on software development and computer security. He holds patents on various inventions in software testing and defensive security applications and has attracted millions in funding, sponsorship and license agreements while a professor at Florida Tech. He also has served as a testing and security consultant for Microsoft, IBM, Rational and many more US companies.

In 2001 he was appointed to Microsoft's Trustworthy Computing Academic Advisory Board and was named a "Top Scholar" by the editors of the Journal of Systems and Software based on his research publications in software engineering. His research team at Florida Tech is known for its testing technologies and tools, which include the highly acclaimed runtime fault injection tool Holodeck. His research group is also well known for their development of exploits against software security, including cracking encryption, passwords and infiltrating protected networks via novel attacks against software defenses.

Peter Morgan

Nicemove Ltd

What Testers can Learn from Other Professions

Abstract:

Ideas are peculiar, in that you cannot smell them, weigh them, or follow their progress. Thus the origin of this presentation is lost in time. What is clear is that several snippets of conversation, observations or newspaper articles seemed to cry out loud to me, indicating that there was a carry over from the activity observed or described that could be of us to those engaged in the broad area of 'testing'.

The aim is to take seven professions, and draw from each of these a simple idea that can easily be applied for testing professionals.. Of course some of this will be simplistic, for not all engaged in, say 'carpet fitting' are as professional as the example would wish them to be. However, neither are all 'testers' perfectly socially adjusted, helpful, friendly and individuals who welcome constructive criticism at all times of day or night!

The seven activities from which simple lessons are drawn were the first ones that excited me in this respect. Since then, there have been several pretenders who have not made the final cut. These may be included in a future article or presentation. Until that time, you will have to make do with amongst o5thers, a carpet fitter and a police cadet.

Biography:

Peter Morgan is a testing professional who has been involved in the ICT industry for more than 20 years, and worked in the freelance marketplace for much of that time. His time has sometimes moved from testing to 'development', but he would add "always using the mindset of a tester". He is passionate about testing and a firm advocate of ISEB qualifications. Peter was one of the first tranche of testers to sit and pass the ISEB Practitioner's Certificate in Software Testing, and he vociferously encourages other testers to obtain this qualification. An occasional speaker and author, Peter tries to base his output on hand-on experience, attempting to relate fine sounding ideas back to how it will affect Joe or Jane Tester in their everyday working lives in the war of attrition that we call software testing.

A strong believer in "testers for testers roles", Peter makes a virtue out of a varied business background; MOD, Health, Gas Supply Industry, Post Office automation, interface checking on 3rd party credit agencies, etc. As with others in the industry, he has occasionally been involved in a project that sunk without trace, and one such was cited before the Public Accounts Committee as a fine example of how NOT to undertake a software development project. That is a story waiting to be told at some future date.

Now that he has passed the other side of 50, Peter still has large areas of unfulfilled professional ambition. He feels that he has benefited tremendously from the input of fellows in the worldwide family of software testing, and is keen to actively contribute to raise both the profile of software testing, and improve the standing of professionals who engage in this thoroughly rewarding activity. If his activities assist others to progress, this is a part payment of the enormous debt of gratitude that he still owes to the 'fathers of the faith'. Some individuals in this category are simply superb hands on testers, former colleagues who few others would have heard of.

Dr Kelvin Ross

K. J. Ross & Associates Pty Ltd

Supercharging Keyword-Driven Testing Using Model-Based Techniques

Abstract:

Over the past few years we have found keyword-driven techniques to be an extremely effective way of building test automation frameworks for large and complex software applications. Through keywords we have found that we build an interface through which non-technical testers can contribute to the automation project without having to be automation gurus or scripting geeks. Furthermore we have found the frameworks, when compared to other automation approaches, scale well and do not suffer as heavily from the impact of interface changes to the application under test.

However, we still find that Keyword-driven testing remains significantly labour-intensive. Firstly, defining test scenarios and input data requires chaining together hundreds and sometimes thousands of individual keywords into meaningful sequences. Furthermore, maintenance of these sequences becomes significant whenever there are changes to the preconditions and/or input data requirements for each keyword. While we have achieved a reduction in maintenance from more traditional approaches, such as capture/playback, the maintenance effort has shifted to coordination of the sequences themselves.

Always conscious of the overhead of automation, we have been following Model-based techniques over the past few years as a mechanism to make automation more effective, and importantly reduce the effort required to setup and maintain current automation frameworks. Our research has shown modelling to be a natural extension to Keyword-driven automation, which assists with generation of test sequences, and importantly simplifies maintenance should data and sequencing behaviour be altered.

Our selection of modelling approaches has been largely influenced by our focus on data-driven applications rather than control systems. We found that the generation and correctness of input data used in data-driven applications is as critical as the sequencing of test scenarios. This led us to investigate models that use Abstract States, rather than pure state-based models, such as Statechart and Markov models. The examples we will present use Spec#, a high level modelling extension to C# provided by Microsoft Research.

We have found the up-front cost of modelling remains significant as producing a model has a steep learning curve. However, we have found a number of later benefits:

1. The maintenance of test scenarios and input data is minimised as we simply fine-tune the model for changes and discrepancies and regenerate.
2. We can achieve very high volumes of test sequence generation, useful for testing scenarios with high combinatorial effect.
3. We have significant potential for reuse, as models may be adapted to suit different variations in behaviour. This last point we have been investigating further as we have been identifying patterns as a mechanism for rapidly building models of new behaviour and perhaps reducing the up-front cost of automation setup.

This presentation will review our approaches to extending Keyword-driven testing with Model-based techniques. We will present a number of real case studies which show both successes and failures of our attempts and present lessons learned.

Biography:

Dr Kelvin Ross is Director of K. J. Ross & Associates (KJRA), a consulting firm currently employing of 16 specialist software testing consultants. KJRA also incorporates SMART Labs, their ISO17025 accredited software testing laboratory.

Kelvin started his IT career with Hornet Radar Systems Group, DSTO, developing software support tools for F/A-18 operational flight programs. He then undertook and "finally" completed his PhD at the University of Queensland, where he researched change and configuration management within high-integrity software development. While studying towards his PhD, Kelvin was also employed by the Software Verification Research Centre as a consultant, where he was involved in safety-critical development projects, professional training, and applied research.

In 1997, Kelvin established K. J. Ross & Associates on the Gold Coast, Australia, which has since developed a national reputation as the "Australia's software testing experts" working on many commercial testing projects throughout Australia and overseas. Test consulting and outsourcing projects in which they have been involved have included shrink-wrapped software products, e-commerce and web applications, gaming applications, distributed and fault-tolerant systems, and safety critical medical, defence and transportation systems.

Don Mills

Independent

Patterns for Testing

Abstract:

Craftspersons in all walks of life use patterns to avoid their having constantly to reinvent the wheel. A pattern captures a problem solution that someone (you, someone else ...) has found useful in one set of circumstances, and may find useful again in the same or a similar situation.

The architect Christopher Alexander famously formulated the pattern language concept for expressing (architectural) design patterns, and was as surprised as anyone when it was picked up with enthusiasm in the IT community via the publication of the "Gang of Four" book, Design Patterns: Elements of Reusable Object-Oriented Software (1994).

It didn't take too long for some testers to realise that testing, too, involves design activities; and while test patterns have yet to take the testing world by storm, there's a growing body of knowledge about them, and of experience with them.

Don Mills' presentation, Patterns for Testing, is a beginner's class in the topic. Informative and entertaining, it examines the general nature of patterns, describes a basic patterner (pattern for patterns), examines a template for test patterns, and explores the history and nature of pattern languages. If you are not enlightened, you may be titillated; and if not titillated, you may be enlightened.

With luck, you'll be both!

Biography:

Don started working as a software developer when the bank that employed him computerised in the late 1960s. In 1972, he emigrated to New Zealand to work for Burroughs (later Unisys) in software development and customer and engineering support. He quickly gained an international reputation, providing customer and engineer training in Australia, Canada, South-East Asia, and even the USA, as well as in NZ.

In 1991, now freelance, he was participating in redevelopment of the premiere NZ Government software system. Coincidentally, his NZ-born wife Margaret Fordyce was appointed Analyst in Charge of Testing on the same project, and soon found that none of the 500 developers really knew much about it. Don researched the subject, and they put together a method applying cause-effect testing to requirements specifications before software design began--an approach Don christened, "test-first development". When the method proved very successful in finding requirements and specification bugs, and preventing code bugs, they proposed a training course.

The customer accepted, and Don developed the course, but retained ownership and delivery rights.

Both Don and Margaret then committed themselves to careers in testing and test training, and were involved in around 20 different testing projects over the next fourteen years.

In 1994, the government customer recommended the course to New Zealand's leading IT training provider, to replace the course they were offering in NZ and Australia. It proved to be their most popular course, notching up its 1000th public student in mid-2004, the week before Don and Margaret left NZ to further their careers in the UK.

SoftEd still deliver the course, which was reworked to ISEB requirements in 2003, and Don is about to rejig it again to fit ISTQB requirements, without losing its special character.

Bogdan Bereza-Jarocinski

Independent

Really Agile: Testing for ASP

Abstract:

A growing phenomenon in the software industry are the so called Application Service Providers (ASP). ASP vendors provide licences for products which are available on their own servers, thus relieving customers from "being their own system administrators". There is no need for cumbersome installations, patches and incompatible versions. The risk of data loss due to laptop failure disappears. ASP application is available from any machine from which you work. This approach has significant influence on QA and testing as well:

- All users have access to the same version of the application
- Single faults can be immediately removed – no need for costly release process
- You have a number of small programs instead of one large application
- Easy to gather performance metrics
- Re-test responsibility on programmers rather than QA department
- Better (and known) quality of released SW
- Faults discovered immediately instead of long time after release is completed
- Much simpler procedures for error tracking
- Faults removed as soon as discovered by one user
- Faults removed while still "fresh"
- Less interconnected faults
- Continuous usability testing performed by end-users
- Performance testing using real-life load profiles
- Practical experience: not so easy as it seems?

Key points:

- General advantages of ASP delivery of SW products
- Advantages of CM for ASP products
- Better testing and quality for ASP products

Biographies:

Bogdan has a degree in computer science and post-graduate studies in organisational psychology. He worked with requirements engineering, system design, construction and quality assurance – especially testing - for Swedish, German and Polish companies. Bogdan translated into Polish Ron Patton's book "Software Testing", published many articles and co-authored "Practice and Theory of SW Testing", to be published (in Polish) in 2005. Frequent speaker at international conferences, such as "EuroSTAR" and "ICSTEST". Bogdan has authored training material for a number of courses, teaches ISEB and IEEE CSDP training. He works now as independent consultant.

Contact details:

bbjTest, Bogdan Bereza-Jarocinski
www.bbjtest.com, info@bbjtest.com
Phone +48-22/ 498 34 48, mobile +48-509 123 362, , fax +48-22/ 629 79 47
Office: Nutki 2/8, PL 02-785 Warsaw, Poland

TEST MODELLING, BUG PREVENTION, AND BASIS PATHS

Don Mills, Macroscope Services Ltd., London

Why you should read this article (it's quite long and a bit technical)

Well, of course, perhaps you shouldn't—especially if you already know what it says.
What the article is about, though, is a powerful and (once you've unravelled the jargon) simple testing technique which is available to both "white box" testers and "black box" testers, but which for several years has been mistakenly presented as solely the concern of programmers and not testers.
This technique offers important testing (and debugging) advantages, including:

- A simple and systematic way to ensure complete process coverage of both internal and external processes.
- Simple prediction of the maximum size of the test set needed to achieve this.
- "High-yield" test sets that offer a good return of unique bugs for your investment of test effort.
- Rapid diagnosis of the location of a bug in even a complicated process.
- Simpler creation of powerful test data designed specifically to locate bugs.

If you're a tester, you should read this article if you are ever faced with testing *processes*. You may meet these in various ways—as business process maps, say, or (as the ISTQB Foundation Level Syllabus recognises) as use case specifications. If you're a programmer, there are probably software tools that can do all this for you automatically from program source code, but testers working from specifications aren't so fortunate—which doesn't reduce it's usefulness one jot.
Here's an illustration of what it's about.

Gimme a (Circuit) Break

A number of years ago, while my wife and I were still living in New Zealand, we had an electrical problem: one of the kitchen power sockets had stopped working. We arranged for the electrician on a day I'd be working at home, and he conscripted me into being his assistant. "Isolating the circuit break isn't as easy as you might think", he told me; "it may not be the wire that leads directly into the malfunctioning socket, and we don't want to have to rip your walls out, so we're going to have to do a bit of testing to find exactly where the break is. D'you mind if I call you when I need you?" I said "Okay," and went back to work.
When he called me, he said he'd taken a look at the house wiring diagram which had been conveniently left by a previous electrician, and reckoned we could find the break with a maximum of eight tests. He would go round the house, sticking his probes into various sockets to find which ones worked and which didn't, and my job was to toggle selected circuit breakers on his command. It took five goes to locate the break, and then I was able to return to my computer keyboard while he got on with the repair.
I was quite fascinated by the process, because it was a living practical application of *basis path testing* in the world of *cyclomatic graphs*—a powerful testing technique I taught (and teach) on my testing courses. It's old stuff, but powerful stuff, simple but not well understood by many testers.

Basis Path Testing

The term, "basis path testing", doesn't appear in the BCS *Glossary of Terms Used in Software Testing* (a.k.a BS7925-1), but a related term does: **basis test set**. The *Glossary* defines a basis test set this way:

> **basis test set**. A set of test cases derived from the code logic which ensure that 100% branch coverage is achieved.

(A very similar definition appears in the published version of the ISTQB "Glossary of terms used in software testing", Version 1, 8th December 2004).
The fact is, though, that test case specification techniques are not necessarily "code-based" or "requirements-based" (see my previous article on Black- and White-Box testing: "Three Cheers for the Black, White, and Grey", *The Tester*, June 2005). Testing using "basis test sets" should not be restricted to program code logic—indeed, our electrician's use of the technique showed this quite

clearly. It happens that a basis test set may be derived wherever there is a *flow of information* or *flow of control*—as in the examples mentioned above, business process maps and use case specifications. The BS7925-1 definition (and the ISTQB definition) is also deficient in another, more technical, way, which is really what this article is about. A basis test set provides you with much more than just "100% branch coverage" of program code.

Let's look at an alternative definition of "basis test set" (from my own *Software Testing Glossary*):

basis test set. A set of test cases derived from a directed graph model and constructed to include all linearly independent paths. *Synonym*: basis test set. *See*: basis path set, basis path testing.

> A covering set of linearly independent paths also achieves both node coverage and link coverage. All paths through a graph may be constructed by simple addition and subtraction of the links traversed by a basis test set. The cyclomatic complexity of a graph specifies the maximum size of a corresponding basis test set.

Okay, perhaps not the most *transparent* of definitions. What's a "directed graph model"? What's a "linearly independent path"? What are "node coverage" and "link coverage", and what does "all paths" mean? Have patience: all will be revealed!—but where to begin?

Directed Graphs

Let's start with the idea of a **graph**. Being a maths dummy, I used to think of a graph as, well, you know—a set of lines connecting dots, or perhaps a segmented pie, or a series of coloured pillars. Turns out I was only half-right: those are graphs, in one sense, but what they really are is *visual representations* of graphs, a graph being (to a mathematician) a set of abstract relationships between "entities" of some type or another—such as observations in a timed series, or steps in a process description.

A graph is, in fact, a **model** of such a set of relationships (what's known as a "computational model"). A model is a simplified representation of some portion of reality that we want to understand, interact with, or control. It has properties which are not identical to the real thing being modelled, but which have useful analogies to the real thing. Graphs provide a set of well-defined and widely-used techniques and components for modelling processes, systems, or objects, and are commonly used in software testing.

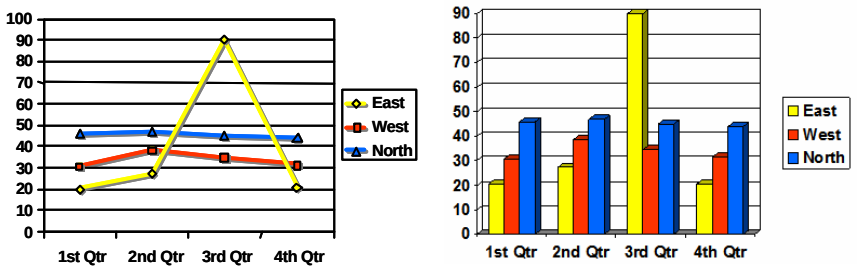

Figure 1: Two graph representations of the same set of relationships

The "entities" that are included in a graph are called **nodes** (or "vertices"). In an X-Y graph (like the leftmost example in Figure 1), the nodes are observations plotted on the graph (represented by triangles, squares, and circles). The relationships between them are called "arcs" or "edges" of the graph. In an X-Y graph, they are represented visually by lines joining the observations. In the software testing world, we often call them **links**.

In a **directed graph**, the links have a required direction, which is often represented diagrammatically by an arrowhead. A familiar example of a directed graph is a *flowchart* (though often enough, the arrowheads are left out because we *know* that the direction of flow is from top to bottom and from left to right—unless the direction is deliberately reversed, as in a loop).

Flowcharts Model Processes

A **flowchart** is a model of a process in which the individual steps, actions, or events of the process are represented as nodes, and the "flow of control" between them as links. As Figure 2 shows, the nodes in a flowchart are usually given different geometrical shapes, representing different types of process step. Conventions differ, but the start- and end-points of a process are often modelled with ellipses (nodes 1 and 4), decision points with diamonds (node 2), and "activities" of the process with rectangles (nodes "A" to "D"). Circles are often used as "connector points" (node 3), representing points at which links meet one another again after having separated at a decision point.

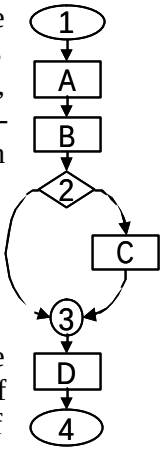

Figure 2: A Flowchart

Figure 2 also shows that the nodes of a flowchart may be *labelled* with unique identifiers. I've written "may be", because there is no international legislation that says you have to; but in fact, I'd strongly recommend it, for ease of identification and reference (as in the previous paragraph). You'll see that (in this case) I've provided two streams of node-id; capital letters for process actions, and numbers for points at which "control of flow" is exercised.

Control of flow, or **flow of control**, means deciding what to do next. Figure 2 illustrates the five basic ways in which this happens:

- **Initiation** of a process, as at node 1 (an **entry node**).
- **Fall-through** from one step to the immediately following step (for example, from step "A" to step "B", or from node "C" to node 3).
- **Selection** from amongst alternative next steps, as at node 2 (a **decision node**).
- **Conjunction** between two subsidiary flows, as at node 3 (a **junction node**).
- **Termination** of a process, as at node 4 (an **exit node**).

Control Flowgraphs

When our New Zealand electrician was investigating our power socket problem, he wasn't very much concerned with what *devices* might be plugged in to any of the sockets, or what they did, so much as with whether any electrical current was getting through to them, and if so by which sub-circuit. Similarly, when we set out to do basis path testing, we are not initially concerned with what the steps of a process do, so much as with the flow of control from one step to "the next". Consequently, it's common in testing to simplify flow charts to **control flowgraphs**, in which individual process steps are suppressed except where they explicitly control the flow through the process. Figure 3 shows a control flowgraph which is logically identical to the flow*chart* in Figure 2. "Fallthroughs" between the kind of process steps represented by rectangular boxes are *implicit* flows of control, so they've been left out, together with the rectangles themselves. But the remaining four control flow mechanisms are all *explicit* controls, and are all included.

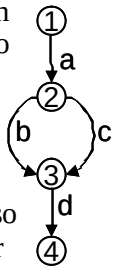

Figure 3: Control Flowgraph

In Figure 3:

- Node 1 is an **entry node** (no **inlinks**, or incoming links);
- Node 4 is an **exit node** (no **outlinks**);
- Node 2 is a **decision node** (multiple outlinks); and
- Node 3 is a **junction node** (multiple inlinks).

Notice that I've "weighted" (labelled) the links with lower-case letters as **link names**, for the same reason I earlier recommended labelling nodes. (And notice too that there are different conventions for drawing control flowgraphs. The examples in BS7925-2, the *Standard for Software Component Testing*, use a slightly different set of conventions.)

Paths

In a graph, a **path** is a sequence of nodes joined by links. Processes can't usually start or end somewhere in the middle, so process-based testing is normally concerned with **entry-exit paths**,

which each start at an entry node and end at an exit node. When I use the term "path" from here on, I will mean "entry-exit path"; anything else is a **subpath**.
There are two entry-exit paths through Figure 2, which we can individually identify by naming the links that lie along them (thus giving each path a "path name"):

Path 1: abd
Path 2: acd

Notice that both paths cover the same *nodes* (1, 2, 3, and 4, in that order), yet they're two distinct paths because one takes the "true" link from node 2 (a decision node) to node 3, while the other follows the "false" link. Adopting an on-the-spot convention that "true" links branch out to the right, the "true" case corresponds to Path 2 (the path that includes link "c"). Referring back to Figure 1, then, both paths will include steps A, B, and D, but only Path 2 will also include step C.

Node and Link Coverage

"What do you do when you see a graph," asked Boris Beizer (*Black Box Testing*)? His answer: "Cover it!"
Coverage is a measure of how much of the test item (whatever you're testing) gets included in the testing. Different types of **coverage item** can be measured, such as the *statements* of a component (**statement coverage**), or explicit *transfers of control* (**branch coverage**), or *paths* through a process (**path coverage**). The measure is usually expressed as a percentage of the total number of coverage items *available* for inclusion.
In graphs, coverage is provided by entry-exit *paths* through the graph. Different sets of paths will provide different kinds and different degree of coverage. To understand this, let's examine Figure 4:

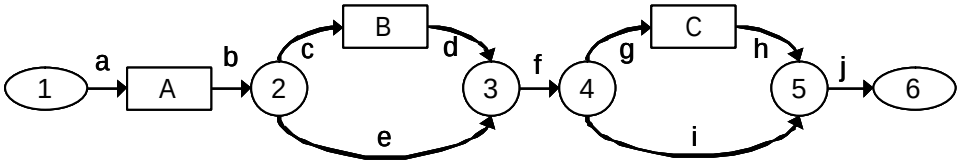

Figure 4: Flowchart weighted with link names

Covering Path Sets

Nodes in a graph are equivalent to "statements" (steps) in a process; outlinks from decision nodes (from nodes **2** and **4** in Figure 4) represent conditional branches. A set of paths that covers **all nodes** provides *100% statement coverage* ("**all statements**"). A set of paths that covers **all links** provides *100% branch coverage* ("**all branches**")—and as long as the graph is not "pathological", also provides 100% statement coverage.
A set of paths that provides a stated type and degree of coverage is called a **covering path set** (or "covering set" for short—but you may have "covering sets of other things, such as test cases). In Table 1 (below), the greyed cells identify the different types of coverage achieved by four different covering path sets. Each path is uniquely numbered, and also uniquely "named" by stringing together the names of the links that it traverses.

Path	Path name	Coverage provided		
Path 1	a b c d f g h j	all nodes = all statements	all links = all branches (and all statements)	all paths
Path 2	a b e f i j			
Path 3	a b e f g h j		all links = all branches (and all statements)	
Path 4	a b c d f i j			

Table 1: Covering paths for Figure 4

Path 1 covers "all nodes" by taking the "true" outlinks from nodes **2** and **4** of Figure 4. (With the flowchart lying on its side, the convention now is that "true" goes up, "false" goes down.) A test case based on this path will achieve *all statements* coverage, but would not detect bugs along links **e** and **i**. (What bugs could there be, along an empty link? Well, perhaps a test case would show that one or other of them shouldn't have been "empty" after all!)

Path 2 takes the "false" outlinks from nodes **2** and **4**. Path 1 and Path 2 between them cover all outlinks from all decision nodes in the graph, and since the graph includes no unconditional branches, Path 1 and Path 2 *together* achieve *all branches* coverage, as well as *all statements* coverage.

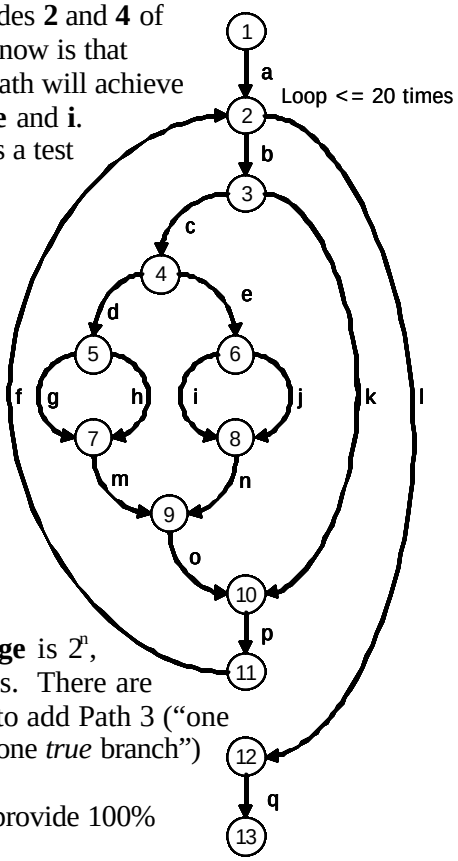

Figure 5: Flowgraph with loop

All Paths Coverage

But Path 1 ("all *true* branches") and Path 2 ("all *false* branches") don't cover all *paths* through the flowchart, from entry to exit. For this, we need to include all possible *combinations* of "true" and "false" outcomes for the decision nodes.

In principal, the number of paths needed for **all paths coverage** is 2^n, where *n* represents the number of binary (true-false) decisions. There are two binary decisions in Figure 4 (nodes **2** and **4**), so we need to add Path 3 ("one *true* branch, one *false* branch") and Path 4 ("one *false* branch, one *true* branch") to provide the four paths necessary for *100% path coverage*.

(Note that there's a choice of two *covering path sets* that each provide 100% branch coverage: Paths 1 and 2 together, or Paths 3 and 4 together.)

In Annex C to BS 7925-2 (*Standard for Software Component Testing*) there is a diagram ranking different types of coverage measurement by the degree of overall coverage they achieve. The accompanying discussion doesn't say so, but the rankings correspond quite well with effectiveness in finding bugs.

"All paths coverage" is right at the top of the hierarchy, which means that it's tops at finding bugs. But excellence comes at a cost, and the cost is the number of test cases you might have to run in order to achieve it. The "2^n" ratio mounts rapidly: two paths for a single binary decision, four paths for two decisions in a row, eight for three, 1024 for ten, 12.7^{29} for 100 …

It's true that nested decisions, and "infeasible" combinations of conditions, will both reduce the number of combinations actually **achievable**, but the presence of loops will increase them.

Consider Figure 5, in which there are five binary decision nodes (nodes **2**, **3**, **4**, **5**, and **6**). The "2^n" rule would yield 32 paths, but because all decisions are nested, except for node **2**, the total number of paths through the flowgraph would be reduced to only six.

Apart from the loop, that is: the return from node **11** to node **2**, which can execute any number of times from 0 to 20. Because of this loop, the set of *all paths* contains 10^{14} possibilities. Let's assume for the sake of argument that they all represent achievable combinations, and that we can execute (by hand) one test case a minute, uninterrupted by breakdowns, blocker bugs, or coffee breaks: Do we have 190 million years up our sleeves to complete the testing? Probably not …

"All paths" testing (but you've probably spotted this already) is pretty much the same as **exhaustive testing**, which means testing all possible combinations of test conditions—paths, inputs, outputs, environmental conditions …: it's physically impossible, except for trivial cases.

Like Figure 4? Well, that depends on the range of inputs, outputs, environmental conditions, etc., for the corresponding process.

(I'll make repeated use of the process model in Figure 5. Initially, it'll be an abstract process—I won't specify what questions are asked or what actions result—but I'll turn it into a concrete example at the end.)

Linearly Independent Paths

So far, then, we've explored the concepts of *entry-exit paths*, *covering path sets*, *node coverage* (statement coverage), *link coverage* (branch coverage), and *all paths* coverage.
But the definition I gave earlier used the term, **linearly independent path**. What's one of those, when it's at home?
Linearly independent paths were described by Thomas McCabe in his 1976 paper, "A Complexity Measure" (*IEEE Transactions On Software Engineering*, Vol. Se-2, No. 4). A linearly independent path is an entry-exit path with two important properties:

- After the first path has been selected, each subsequent path is constructed by selecting one decision node in the immediately preceding path, and selecting one outlink from that node that has not been traversed previously. This means that each new path introduces at least one "new" link, or (in programming terms) one new code strand not previously executed. Boris Beizer (*Software Testing Techniques*) calls this "the Scientific Testing method".
- If a path traverses a loop, it includes the minimum number of loop iterations permitted by the loop control mechanism. If the loop control requires a minimum of (say) five iterations, a path must respect that restriction; but no path through the loop control needs to execute six or more iterations. The exception to this rule is where no iterations of the loop at all are required; coverage rules still require us to traverse the loop body, so we will need at least one path that does so once. If the loop body includes nested decisions, we will have to construct enough "minimum-iteration" paths to ensure that all nodes and links within the loop are covered.

Building a covering set of linearly independent paths

Let's illustrate both parts of the above by constructing a covering set of linearly independent paths through Figure 5. It might help you to follow the argument if you "cover" the successive paths (sequences of links) with a highlighter as we go.
We'll start by selecting "the simplest, most obvious path from entry to exit" (Beizer). Because the loop control specifies a maximum of 20 iterations for the loop body, but no minimum number, the simplest path bypasses the "loop body" altogether, like so:

- Path 1: ***a l q***

There's only one decision node in this path (node **2**), so node **2** presents our only opportunity to apply the "Scientific Testing Method" at this stage. Path 1 bypasses the loop body by taking the "true" outlink from node **2**, so Path 2 (to be different) *must* take the "false" outlink, and enter the loop body. (The sequence of link names, "alq", is known as the **path name**.)
Taking the "simplest" path through the loop means avoiding the nested structures that lie along link **c**, so Path 2 consists of:

- Path 2: a ***b k p f*** l q

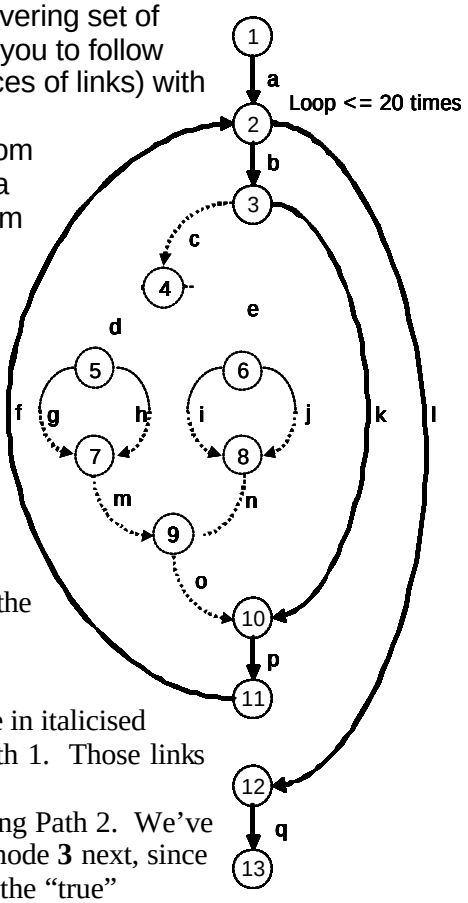

Figure 6: Path 2

This new path (Figure 6) traverses links **b**, **k**, **p**, and **f**, which are in italicised boldface in the path name because they weren't covered by Path 1. Those links that *were* covered by Path 1 are in normal type.
To construct Path 3, we examine the decision nodes that lie along Path 2. We've already taken both outlinks from node **2**, so we have to look at node **3** next, since that's the only other decision node along the path. Path 2 took the "true" outlink from node **3**, so Path 3 must take the "false" outlink and traverse link ***c*** to node **4**.

Which is the "simplest" outlink from node 4? To decide that, we might have to consult the process description (program source code, use case specification, etc) that the graph is modelling. I'll stipulate that four process steps lie along link **j**, three along link **i**, two along link **h**, and one along link **g**. A path containing link **g** (one process step) will be "simpler" than one taking any of the other three links, so we construct Path 3 as follows:

- Path 3: a b ***c d g m o*** p f l q

The path segment "c d g m o" consists of new links, not previously traversed, while the segment "p f l q" has already been traversed at least once before.
To construct Path 4, we introduce the "simplest" change over Path 3. Changing node **4** to take link **e** would introduce a whole new "strand" of logic with complex nested structures, so the simplest change is to switch the outlink from node **5** to take link **h** instead of link **g**:

- Path 4: a b c d ***h*** m o p f l q

Paths 3 and 4 have exhausted the possibilities led to by link **d** out of node **4**, so for Path 5 we take link **e** out of node **4** instead, followed by the "simpler" outlink from node **6** (link **i**). And for Path 6, we take the last remaining link not covered by any path, link **j**.
We now have a *covering set* of *linearly independent paths*, each (except for the first) exploring a small path segment not traversed by any other path, as follows:

- Path 1: ***a l q***
- Path 2: a ***b k p f*** l q
- Path 3: a b ***c d g m o*** p f l q
- Path 4: a b c d ***h*** m o p f l q
- Path 5: a b c ***e i n*** o p f l q
- Path 6: a b c e ***j*** n o p f l q

The "scientific testing principle" means that each successive path is narrowly focussed on one small process strand that hasn't previously been tested. If (for example) we run test cases corresponding to Paths 1, 2, 3, and 4, without finding any bugs, but the test case for Path 5 *does* find a bug, it's a safe bet that the *location* of the bug is directly linked to what happens along links **e**, **i**, and **n**.

Cyclomatic Complexity

According to the note attached to my definition of **basis test set** (above), "The cyclomatic complexity of a graph indicates the maximum size of a basis test set derived from it."
I want this article to be practical, so I'm not going to discuss why "cyclomatic complexity" is so called; nor will I describe why it's represented by the symbol, V_G. (I normally drop the subscript G, which merely indicates the cyclomatic complexity of "this" graph.) Its benefit is that it's an easily-computed number which has two practical uses:

- It's a good predictor of the relative bugginess of process descriptions, including source code for programs or components. Programmer productivity, and the stability of the code they write, are both likely to fall off rapidly when the number passes somewhere between 10 and 15. (The same may be true of other types of process description such as use cases, though I don't know of any studies that would show this.)
- It's an accurate predictor of the *maximum* number of paths you'll need for a basis test set. (And you can easily apply it to other types of testing technique, besides path-based testing, as I'll show you in my next article.) **There is no other simple method for reliably predicting the number of paths you will need for achieving process coverage.**

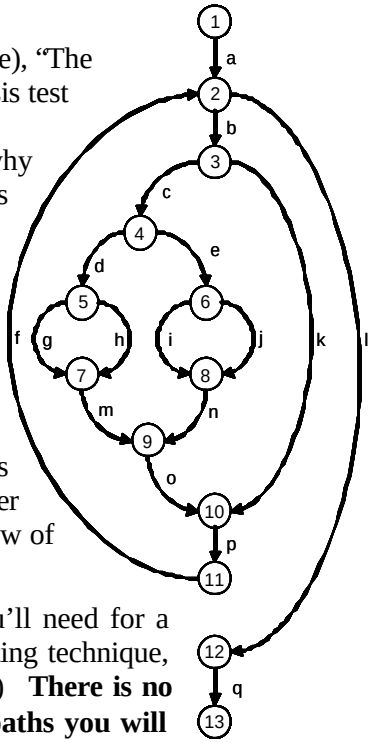

Figure 7: Flowgraph with loop

Why "maximum" number of paths? Some paths that look possible on paper may turn out to have combinations of conditions that make them **unachievable** (or "infeasible") in practice.
There are numerous ways of computing cyclomatic complexity, though most of the well-known ones are approximations which don't work for all cases. The simplest of these is to count the number of **decision nodes** and add 1:

- **V = D + 1**

In Figure 7, there are 5 decision nodes (**2**, **3**, **4**, **5**, and **6**), so the cyclomatic complexity of the graph is **6**—which agrees exactly with the number of linearly independent paths in our basis test set on the previous page.
However, that very simple method only works when all the decisions are **binary decisions**, i.e., they all ask "Yes/No" (or "True/False") -type questions, giving you only two choices of what to do next.
Our next formula is about as simple, but can cope with multi-way decisions. It requires you to count the number of **enclosed regions** in the graph. An example of an enclosed region is the space enclosed within nodes **6** and **8** and links **i** and **j** in Figure 7. A larger example is enclosed by nodes **4**, **5**, **6**, **7**, **8**, and **9**, and links **d**, **e**, **h**, **i**, **m**, and **n**.
Figure 7 has 5 enclosed regions. The formula for *V* that uses this number is:

- **V = R + 1**

—so the cyclomatic complexity is (again) **6**. This is because the number of enclosed regions is actually defined by the occurrence of decision nodes, whether for binary decision or multi-way decisions.
The most robust formula requires you to subtract the number of *nodes* from the number of *links*, and add on the number of *entry nodes* and the number of *exit nodes*. (Yes, that does mean you count each entry node and each exit node twice.) Here's the formula:

- **V = L – N + E + X**

In Figure 7, there are 17 links (**L**) and 13 nodes (**N**), of which 1 is an entry node (**E**), and 1 is an exit node (**X**). Do the arithmetic (17 – 13 + 1 +1) and, once again, you get **V = 6**. This is another nice confirmation that we have the right number of paths in our basis test set.
Programmers will tell you that no process should have more than one entry point or more than one exit point. They will also admit that there may be occasions when they violate that rule—for example, when something inappropriate happens during the execution of a process, and an "emergency exit" has to be taken. Real-world processes (e.g., business processes) may be the same, and may also (occasionally) have more than one entry point.
But when you are certain that your process has only one entry and only one exit (and that *should* be most of the time), you can use the formula most commonly quoted:

- **V = L – N + 2**

(Sometimes, this is modified to **V = L – N + 2P**, where "**P**" represents the number of "process parts" in a larger process, each "part" being modelled by a separate graph. Programmers can use this to compute the overall cyclomatic complexity of a program with multiple "subroutines". When you only have one process model, as in Figure 7, the number of process parts is "1", so you get the same result as from "L – N + 2". As long as there's only one entry and only one exit.)

Simple Loop Testing

"Annex B" to the British Standard on component testing (BS 7925-2) counts boundary value analysis among the "black box" techniques, with the implication that it has no place in "white box" (process-based) testing. This is a misleading impression, however. Boundary value analysis, and boundary cases, are often very important in process-based testing.
In Figure 7, for example, there are some obvious boundaries in terms of the control condition for the loop: the loop must execute at least zero times, and at most 20 times.
Terms such as "at least" or "at most", or "minimum" or "maximum", should evoke a knee-jerk response in testers: "BOUNDARY VALUE TESTING". In fact, whenever you see a numeric field being compared with another numeric field ("AMOUNT <= BALANCE"), or with a specific numeric value ("AMOUNT > £590"), you should think, "BOUNDARY VALUE TESTING". This applies just as much for the decisions within a process, as for its input fields.

Boundary testing for simple loops is much the same as boundary testing for simple numeric input fields. As far as you can, you test:

- The minimum value (minimum number of loop iterations), with the expectation that it should work.
- The maximum value (maximum permitted iterations), with the expectation that it should also work.
- The minimum value "-1" (one less than the minimum required number of loop iterations), with the prediction that this shouldn't be valid.
- The maximum value "+1" (one more than the maximum permitted number of loop iterations), again with the prediction that it won't be valid (shouldn't in fact be possible).

For the process modelled in Figure 5 (and in figures 6 and 7), we specified that there was no required minimum number of loop iterations, but that the maximum number was 20.
Our first linearly independent path ("Path 1") bypassed the loop body, and achieved the minimum number of iterations (zero). But we haven't tested the other possibilities.
For this example, "minimum – 1" is physically impossible (I can't imagine what it would mean, to execute a loop "-1" times!). "Maximum" would require 20 trips through the loop body; "maximum + 1" would require 21. Whether we could build a test case for "maximum + 1" iterations would depend on the nature of the loop control mechanism. If it's supposed to deal with up to 20 records for a given customer, for instance, what happens if we give it 21 records for the same customer?
Similar problems *shouldn't* affect the "maximum iterations" case (if they do, we've found a bug already). We could model a "maximum iterations" test case like this:

- Path 7: a ([b .. f] * 20) l q

Two other cases that are often interesting are "minimum iterations + 1" (which is achieved anyway by Paths 2—6), and "maximum iterations – 1":

- Path 8: a ([b .. f] * 19) l q

Notice that I haven't specified which subpaths are covered from link **b** to link **f** in Paths 7 and 8. There are only five subpaths through the loop body, and a test case that performs 20 (or 19) loop iterations gives us the chance to test each subpath multiple times. These additional tests probably won't find any bugs that once through each subpath wouldn't have found; but on the other hand, they might. Either way, the *purpose* of Path 7 is to see what happens when we hit the maximum intended number of iterations: can the system cope, or can it handle only 19 (or 18, or 10 … We don't necessarily need to know the actual maximum number, only that it's less than it's supposed to be).

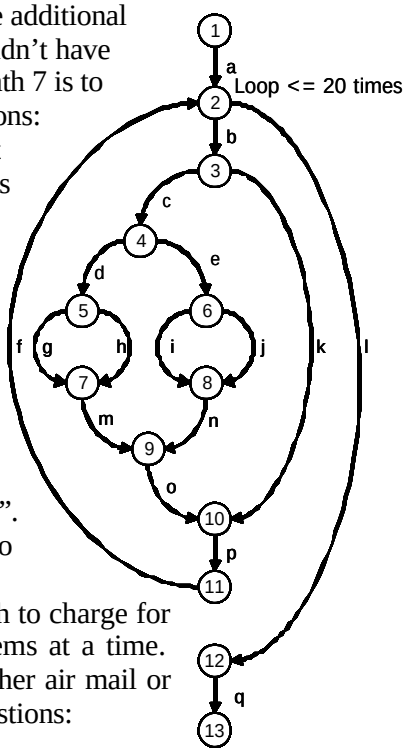

Figure 8: Flowgraph with loop

Input Data and Test Results (Test Requirements)

When you've identified your basis path set (with or without the loop coverage extensions), the next thing is to work out what input data values and/or preconditions are necessary to **force** (technical term!) execution of each path. Building a set of *path predicate expressions* will help you (one per path).
Quite simply, a **path predicate expression** lists each decision outcome along the path you're examining and strings them together with "ANDs". This is a little difficult to illustrate with the example we've been using so far, since I haven't described what the process does.
Let's suppose it's a process by which a mail clerk works out how much to charge for individual postal items (letters and parcels), to a maximum of 20 items at a time. Internal items don't require postage; external items do, and may be either air mail or surface mail. The first four decision nodes might ask the following questions:

- Node 2: Have I run out of items *or* done 20?
- Node 3: Is this an internal item?

- Node 4 (for external items): Is this a prepaid item?
- Node 5: Is prepaid postage adequate for weight?

Note that Node 2 represents a *compound condition*. These present minor coverage problems of their own—do you treat them as a single decision, or as two or more separate decisions? In this case, though, the compound condition is necessary to enable us to exit the loop after fewer than 20 iterations (like, none at all).

Here are the **path predicate expressions** for some of the paths we've drawn up:

- **Path 1 (a l q):**
    - I have no items to post.
- **Path 2 (a, b k p f, l q):**
    - I have items to post AND
      first item is internal AND
      I have no more items to post.
- **Path 3 (a, b c d g m o p f, l q):**
    - I have items to post AND
      first item is external AND
      first item is prepaid AND
      prepaid postage is adequate for item weight AND
      I have no more items to post.

Note how commas can be used to mark off significant subpaths—in the examples above, iterations through the loop body. Note also how the question at node 2 ("Have I run out of items?") has to be re-asked at the end of each pass through the loop body.

We can also use the paths to predict what "the process" should do in each case. Drawing on the (imaginary) full process description:

- **Path 1:** No items to post.
    - **Result:** I go away again (come back later …).
- **Path 2:** One internal item to post.
    - **Result:** I drop the item in the "Internal" bag. (This occurs along link **k**.)
- **Path 3:** One prepaid item with adequate postage.
    - **Result:** I drop the item in the "External" bag. (This occurs along link **j**.)

Input Data and Test Results (Test Case Specifications)

The combination of the path predicate expression for a path (or rather, the precondition and input values it represents) and the results, taken together, form a set of **test case requirements**. Plugging in actual input values (such as specific parcel weights for different general *ranges* of weight), and working out exact output values (such as the amount of postage for external items), would turn them into **test case specifications** (one per path).

A complication: some of the questions asked along a path may not be directly about input values, but about values derived from them. For example, there might be a question about the final amount of postage for a non-prepaid item—e.g., "Does computed postage exceed £10?". "Computed postage" isn't an input to the process; it's part of the mail clerk's job to work it out, based on parcel weight, delivery method, destination, and perhaps other matters specified in the process. When you trace your way through a path to identify its path predicate expression, you may have to perform **predicate interpretation** to work back to the original input conditions that downstream values are calculated from.

I made the claim earlier that the basis path technique can simplify the "creation of powerful test data designed specifically to locate bugs". How so? The answer lies in the "scientific testing technique", also known as "test one condition at a time". In principle, each successive path tests one decision not

previously tested. This provides the "specifically locating bugs" element. And depending on the nature of the process, and the questions asked on the way through it, it may be possible to build a test data set (a set of database records, say) in which each successive record differs from the preceding record by only one or two data values—the ones to which "this" path is specifically sensitive (a process known as **path sensitization** of test data). This provides the "simplification" in the design (and creation) of test data sets.

Debugging by Modelling

As promised, this article is "quite long and a bit technical", so I'll have to bring it to a close shortly. It's worth outlining, though, how model-based testing of the sort I've illustrated above (and basis path testing is only one form of model-based testing) can help find bugs even without your building actual test data and running actual test cases. You have several opportunities for this:

1. Before you can build your model (the control flowgraph, in this article), you have to analyse your **test basis document(s)** (a.k.a. your "test baseline") to define the elements that will go into your model and their logical relationships. First problem: The test baseline is unclear, inconsistent, or ambiguous, and you can't build your test model without some clarification from the baseline author (often quite a lot of clarification). Since the developers will face the same sort of problems as you, you can get the baseline debugged for them *before* they start designing and building the product.

2. Once you've got the baseline clarified, you can start building the model. Here again, you may find that the baseline still doesn't provide adequate clear information in some places, or that some parts of the model just don't fit together (often indicating inconsistencies in the baseline). Now you can get these second-level ambiguities resolved, again to the developers' benefit if they haven't yet started design and construction.

3. With the model built, you can start tracing test paths through it and identifying the test requirement sets. Now you may find that some parts of the process don't work: The questions don't make sense, asked in *that* order; or these two questions together mean that *this* subpath can never be entered (so what's it for?) … Once more, you have the opportunity to raise issues about the baseline document and get bugs fixed.

4. Eventually, you have a set of paths, and corresponding test case specifications, that seem to work. But are they working the way the customer *needs* them to work? This would be a good time for a **review** involving the baseline document author and a customer representative. Treat each test case specification as a **scenario**: trace through what happens, according to your model, and have the customer and analyst verify that what happens is what's *supposed* to happen. When your review partners protest that you've got it wrong, investigate whether the problem lies in your model (through your error) or in the baseline document (either because it's wrong, or because it could be misinterpreted), and get the problem fixed.

When your test basis (baseline) document is a business requirements specification, or a software specification or use case specification, *and the software hasn't yet been built*, test scenario reviews with the customer are an enormously powerful way of finding and removing specification bugs. And we all know the statistics showing that, on average, the largest and most damaging group of bugs is in the requirements specification (and the next largest in the software design spec).

Summary and Conclusions

On the surface, this article has been about basis path testing (with a look sideways at loop testing), because basis path testing is a test-technique that has been much misunderstood (and misrepresented) by the testing community. Although associated with "white box testing", it provides potent benefits when testing *any* kind of process, black box or white. The modern popularity of use case specifications has brought it back into prominence, since by definition (in the *UML Specification*) a use case specification defines the *process* by which an actor (user) interacts with a system or system component.

I listed benefits in two places. At the beginning, under **Why you should read this article**, I looked towards the test data construction, test execution, and product debugging activities. At the end, under

Debugging by Modelling, I showed how building and using test models can be a very effective method of "static testing", *preventing* product bugs by finding and removing specification bugs.
There are other reasons why basis path testing is a Good Thing, especially in the way that, uniquely, it's possible to build a set of All Paths from its elements by simple arithmetic. This "latent all paths coverage" adds robustness to its performance in isolating defects.
But in this article I've concentrated on its practical use. As you might expect, it isn't always as simple as in the elementary (and partial) example I've used, but I've provided most of the information you'd need to resolve any difficulties on your own. Rather than "good luck", I'll wish you "good skills" in using it!

Issue 14

THE TESTER

December 2005

NEXT CONFERENCE

Friday
9 December 2005

Little Test

- **Just-in-Time Testing Techniques and Tactics**
- **Risk is a Tester's Four Letter Word**
- **Measuring Return on Investment: the Agony & the Ecstasy!**
- **Condition Hierarchy Test Analysis**
- **Implementing a Performance Test "Centre of Excellence"**
- **Making Sense of Metrics!**
- **The Elevator Parable: Seven Steps to Building a Better Bug Workflow System**
- **Deciding What Not to Test**

IN THIS ISSUE:

BCS

FROM THE EDITOR

Everyone I have spoken to who attended the September conference said how much they had enjoyed the day, and this was borne out by the scores for each of the presentations.

We do take great interest in your feedback, whether it relates to the presentations, the organisation, lunch, location etc – so please make sure that you fill in the feedback form at the end of the conference. Wherever possible we try to improve future events on the basis of your comments.

We have an interesting programme for you in December, with Robert Sabourin as our Keynote Speaker. As usual there will be parallel sessions, in fact this time there are three. These are restricted in numbers with no advanced booking so please plan for your attendance on a first-come first-served basis.

I look forward to seeing you at the conference. If you have any announcements then please join me on stage for the Networking Session.

As this is the last newsletter for 2005, the committee wish you all a very Happy Christmas and New Year!

Pam Frederiksen
Communications Secretary
Tel: 01483 881188 (Leysen Associates)
Fax: 01483 881189
Email: pam@leysen.com

BCS SIGIST website: www.sigist.org.uk

SIGIST Standards Working Party: www.testingstandards.co.uk

FUTURE SIGIST CONFERENCE DATES

15 March 2006

15 June 2006

19 September 2006

14 December 2006

BOOKING INSTRUCTIONS

1. Download a booking form from:
   http://www.sigist.org.uk/bookingForm.pdf

2. Complete and fax to:
   Colin Chivers
   01793 480270

   OR Post to:
   Colin Chivers
   Specialist Groups & Branches Accounts
   1 Sanford Street
   Swindon
   SN1 1HJ

NEXT MEETING – PROGRAMME

BCS SIGIST – Little Test

Friday 9 December 2005
Royal College of Obstetricians and Gynaecologists, 27 Sussex Place, Regent's Park, London NW1

08:30	Coffee and Registration	
09:25	Introduction and Welcome	
09:30	**Featured Speaker** *Just-in-Time Testing Techniques and Tactics* *Robert Sabourin, AmiBug.Com Inc.*	
10:30	Networking session and commercial break	
10:50	Coffee & opportunity to visit the exhibition	
11:20	**Risk is a Tester's Four Letter Word** *Julie Gardiner,* *QST Consultants Ltd.*	**Condition Hierarchy Test Analysis** *Peter Mullins,* *Abbey National*
12:05	**Measuring Return on Investment: the Agony & the Ecstasy!** *Brian Wells,* *Experimentus Ltd.*	**Implementing a Performance Test "Centre of Excellence"** *Richard Bishop,* *HBOS*
12:50	Lunch & opportunity to visit the exhibition	
13:50	**Book Review**	**Featured Speaker**
14:00	**Making Sense of Metrics!** *Graham Freeburn,* *Sopra Newell & Budge*	**The Elevator Parable: Seven Steps to Building a Better Bug Workflow System** *Robert Sabourin, AmiBug.Com Inc.*
14:50	Tea & opportunity to visit the exhibition	
15:20	***Performance Testing for Managers*** *Scott Barber, PerfTestPlus Inc.*	
16:05	**Featured Speaker** **Deciding What Not to Test** *Robert Sabourin, AmiBug.Com Inc.*	
16:50	Closing Remarks	

PARALLEL SESSION

There are three parallel sessions today offering alternative presentations to those held in the main lecture theatre. These are held in a separate room that restricts the number of attendees. Places will be available on a first-come, first-served basis on the day, there is no advanced booking and no additional fee.

The SIGIST committee reserves the right to amend the programme if circumstances deem it necessary.

SIGIST Library

Looking for a testing book but not sure which topics are covered? Or are you trying to decide which testing book to buy? Or do you simply want to increase your testing knowledge? If the answer to any of these questions is 'yes' then the SIGIST Library could help!

The SIGIST Library has lots of testing books covering a variety of topics and they are available to borrow for a period of 4 weeks - free of charge. Extended loans are allowed as long as the book has not been requested by another SIGIST member.

Topics include (amongst others) Requirements testing, Reviews/Inspections, Test Management, Techniques, Test Process Improvement

If you would like to know more about the library and books available, or for any queries, please contact Julie Gardiner on 07974 141436 or email her at gardinerjulie@yahoo.co.uk. Alternatively, download the book loan form on the SIGIST website www.sigist.org.uk. Happy Reading!

SPEAKER BIOS AND ABSTRACTS

Featured Speaker:

Robert Sabourin

AmiBug.Com Inc.

Author: *"I am a Bug!"*

Just-in-Time Testing Techniques and Tactics

Abstract:

As the Boy Scout credo goes, "Be prepared." This presentation explores how to be ready for just about anything in a software testing project within the volatile environment of a Web or e-commerce software project. Which techniques can be used to manage and track software testing in chaotic environments with continuously changing requirements and shifting priorities. How can you work with minimal information, but still develop test and converge the product development effort.

Biography:

Robert Sabourin is the president and principal consultant of AmiBug.Com, Inc. He has more than 20 years management experience leading teams of software development professionals to consistently deliver projects on-time, on-quality and on-budget.

As a respected member of the software engineering community, Robert has trained and mentored literally hundreds of top professionals in the field.

Robert is an Adjunct Professor of Software Engineering at McGill University and often writes and speaks at conferences around the world on software engineering, Software Quality Assurance, testing and management issues.

Robert is also author of the popular children's book "I am a Bug!" that explains what testers really do at work!

Julie Gardiner

QST Consultants Ltd.

Risk Is a Tester's Favourite Four Letter Word

Abstract:

In their book Waltzing with Bears, Tom De Marco and Tim Lister wrote "Greater risk brings greater reward, especially in software development." Project managers speak the language of risk. Their understanding of risk guides their decisions. Testers can contribute to an organization's decision making process by speaking that same language. But applying a risk-based testing approach, knowing where to start and how to influence project managers, can be daunting.

During this session you will learn how to evaluate risk in both quantitative and qualitative ways. Identifying risk is one thing but controlling risk is often difficult and time-consuming to do well. This presentation will show managers how to prioritize, direct effort and report on the most important risks.

You will also learn how to identify and deal with various misunderstandings managers have about risk-based testing, such as:

- Testing is always "risk-based"
- Risk-based testing is nothing more than prioritizing tests
- Risk-based testing is a once only activity
- Risk-based testing is a waste of time
- Risk-based testing will delay the project

If risk-based testing is to be adopted and encouraged within your own company then tangible benefits must be seen by Senior Management. I shall be sharing the benefits that I have experienced when implementing a risk-based methodology and how this can be reported in an effective way to Senior Management in order to obtain their support for future projects.

This session will present:

- A variety of approaches to risk evaluation and where they can be used.
- Vital areas to consider when choosing your approach such as the development methodologies being used and the organizational culture and project goals
- Applying risk analysis to testing in a practical way to:
  - Plan & estimate testing
  - Choose testing techniques
  - Monitor, control and report testing
  - Misconceptions of management regarding risk-based testing
  - Benefits that can be achieved

Biography:

Julie is founder and Principal Consultant of QST Consultants Ltd and is a Grove Trainer working with Grove Consultants.

She has over 14 years experience in the IT industry including time spent as an analyst programmer, Oracle DBA and Project Manager. Julie has first hand experience in the roles of test analyst, test team leader, test consultant and test manager. At QST Consultants, she provides consultancy and training in all aspects of testing, specialising in risk-based testing, test management and people issues.

Julie has presented training courses on a range of testing topics and is an accredited trainer for the ISEB/ISTQB Foundation Certificate and ISEB Practitioner Certificate courses.

Her experience has been gained across a broad range of industries including financial, utilities, retail, web and the public sector using various software development approaches from traditional to RAD/DSDM/agile methodologies.

Julie is a regular contributor to Testing in the UK and is a committee member for the British Computer Society's Special Interest Group in Software Testing and has been England's Country Co-ordinator for EuroSTAR for the past 3 years.

An enthusiastic and motivated presenter Julie is a regular speaker at software testing conferences including EuroSTAR, STAREast, ICSTest and the BCS SIGIST. At STAREast 2005, she won Best Presentation and is a keynote speaker at STARWest this year too.

Peter Mullins

Abbey National

Condition Hierarchy Test Analysis

Abstract:

I've sometimes found it difficult to manage projects because the styles of test analysis employed by individual members of the team are often different. This is invariably the case when the test team is created from a mix of professional testers, developers and end users, but (more surprisingly) is also the case where the test team comprises entirely of professional testers. Whilst there are lots of methods that explain in detail how to plan and manage testing, there isn't a great deal on detailed methods of test analysis, and hence everybody seems to "do their own thing".

All the test staff will join the team with different skills and experiences, yet (as Test Manager) I don't have time or brain power to let them explain to me exactly what test analysis they have carried out or how far they have progressed. I don't have time to assess the quality of work they have completed. I've also run into problems when staff leave and the handover material is so bad, we have to start again.

Unless you have a rigid and comprehensive approach, test assets developed by different Test Analysts tend to have a different style, and be inconsistent. Variable quality test analysis results in limited future benefits when the next "generation" of Test Analysts have to carry out an impact assessment. It's also difficult to "mix" the different styles of test analysis - is one approach more or less thorough in a given circumstance? As a "permanent" Test Manager I need to give a lead in providing a consistent test analysis method.

Condition Hierarchy Test Analysis gives a consistent, fairly rigid method. Its based on Work Breakdown Structures (used for Project Planning), so is familiar for non-test staff to understand. Its flexible - it uses the terms used by the "client" (be that business or development), and is also flexible to changes in requirements or functions. It allows prioritisation of areas of testing so that the most important get tested as early as possible. Its applicable across all phases of testing (each using their own hierarchy), and also applicable for different methods of testing (e.g. scripted testing v unscripted but "directed" testing).

Many Test Analysts use their own "fixed" approaches, and they don't like change. Some contractors aren't particularly keen on a method that allows easy review of their work, or one which means they won't become "indispensable" through the knowledge they don't document. Other contractors are content to keep their heads down. They don't produce much, but if the team is large, and the project is long, they survive happily because they don't rock the boat and they don't need to prove anything. Whilst many testers are self motivated, they may be misguided - they will rely on the Test Manager for support and direction. Other testers need to be closely monitored. Condition Hierarchies aren't easy for everyone to pick up, but they are an excellent method to help Test Managers control their projects.

Not all Test Analysts find it easy to differentiate between a process and a condition (even experienced test analysts fall into this trap). Also "crisis" test analysts who rely on experience to "test what we can in the time available" find the concept of a structured plan difficult to countenance. With appropriate training material and examples, it is possible to overcome most problems - or at least to demonstrate that the problems an individual test analyst is experiencing have not been resolved by the training (so as Test Manager you can step in early).

Condition Hierarchy Test Analysis is very much a "straight bat" approach. Follow a set process and the result will be good, clear, well documented test analysis, regardless of input source material. The method gives improved planning, consistency of analysis, good quality documentation, and cross-project understanding of what the test team will (and won't) test.

Biography:

I've worked for Abbey for 18 years in a variety of IT and Business roles covering programming, analysis, project management, and test management. I came into test management full time around 5 years ago, after getting rather disillusioned with Project Management, which I found rather "generic". Testing was (and is) interesting to me because its full of the "inventiveness" that attracted me into IT at the start of my career. It gives me scope for invention, and the chance to experiment with psychology.

Brian Wells

Experimentus Ltd.

Measuring Return on Investment: the Agony & the Ecstasy!

Abstract:

Measurement is something that everyone seems to want; not least to demonstrate Return on Investment (ROI). But why is it that few organisations allocate sufficient priority, funds and resources to implement a basic, robust measurement programme?

Using a case study, we identify (changing) business requirements/needs, nature of a basic, start-up measurement programme, it's definition, implementation across project delivery structure (including off-shore out-sourcing, internal delivery streams and central testing). Also review results accrued in first 6-9 months.

This presentation is not only about initial, actual results that emanated from measurement programme but also how this influenced the view/perspective of the organisation to metrics.

We will cover:

- The actual business drivers/issues that eventually prevailed
- The base structure, goals/processes: initial meaningful objectives that were achievable
- The approach addressing the culture, awareness, training & mentoring (including easy to understand messages) issues/needs
- How we managed the quality and uptake of measurement process
- What happened to information at all levels of organisation (including ensuring positive interpretation)
- What the initial results of introduction of measurement programme influenced (business, IT, thinking, culture etc)
- As a consultant, outline frustrations, positives and impart to audience what would have tried to do differently!

3 Key messages are:

1. How to define/implement a (base 1) measurement programme that will produce quick results and acceptance at organisational level encompassing entire project life cycle (including off shore outsourcing elements)
2. How to approach education, awareness and culture change at all levels to overcome historical "blockers"
3. Indicators in assisting to persuade organisations to attach much greater importance and priority to the right kind of practical, realistic measurement programmes to deliver valid process improvements and demonstrate Return on Investment (ROI)

Biography:

Brian Wells is a professional IT consultant with over 16 years testing and quality related experience working for both large and small projects in a wide range of industry sectors and in many different countries. Before specialising in testing, he worked in project management, structured analysis & design, programming and other IT disciplines.

In recent years, he has gained an in depth understanding of the development, testing, quality and other issues and problems surrounding "geographically challenged" delivery life cycles utilising off-shore organisations in India, the Czech Republic and elsewhere.

Over the years, he is frequently engaged as a specialist public speaker for a number of different events throughout Europe and elsewhere (including EuroStar, ICSTest) covering a wide range of testing and quality-related subjects. He has also completed the ISEB/ISTQB Foundation & Practitioners Certificates in Software Testing and he is a certified ISEB/ISTQB trainer. He is also a qualified CMM Assessor. In addition, he is the current Chair of the TMMi Foundation; a non-profit making organisation formed to promote a standard TMM reference model to the industry at large.

Richard Bishop

Senior Performance Test Analyst, HBOS

Implementing a Performance Test "Centre of Excellence"

Abstract:

HBOS has invested a significant amount of time and effort into developing a performance testing team to work alongside their developers when preparing new software for their network systems. The performance test team works on both internal-facing software, for use in the branch and call centre network and external-facing software for customers' online-banking. The testing predominantly focuses on web applications, but the team also has experience of RTE, DCOM and Citrix scripting. The team is predominantly focussed on testing applications prior to deployment, but is increasingly becoming involved in performance tuning work and the use of diagnostics software to pinpoint application problems and bottlenecks.

In his presentation. Richard will give an overview of the best-practice approach to performance testing adopted by HBOS. He will discuss planning, preparation, test execution and results analysis. Richard will show examples of test results sites and will be able to answer questions based on his practical, real-world experience as a performance tester.

Biography:

Richard Bishop is a Senior Performance Test Analyst at HBOS plc. He and his team function as the centralized testing team for HBOS Group and are responsible for setting procedural standards as well as supporting the testing efforts of other HBOS organizations. In his current role, Richard is focused on various aspects of testing, from performance tuning and application optimisation to performance testing. He joined HBOS in 2002 and has extensive testing experience from his previous position as a performance-testing consultant.

Graham Freeburn

Sopra Newell & Budge

Making sense of metrics!

Abstract:

Metrics mean many things – the message 'received' is often very different from the message 'intended'. Test cases and defects are just noise to those outside testing. Can you present a view that shows status simply? How do you cope when over time what constitutes 'a serious problem' can change?

Mediated metrics can simplify presentation and deliver a clear message by bringing together a small number of metrics into a risk profile.

Based on an idea developed for managing the 'project end game' this presentation demonstrates examples of mediated metrics in action using a simple but effective tool – a 'risk spider'.

The following shows some of the key points.

1. What mediated metrics are and what they are not
2. How their use helps test managers achieve greater influence
3. How a specific example of their use, the risk-spider, can be transform the way you report the status of testing and its perceived value.

Biography:

Graham is a Principal Testing Consultant in the Testing Services Division of Sopra Newell & Budge whom he joined in 2002. He is responsible for all aspects of the testing training service including provision of ISEB Foundation and Practitioner training as well as bespoke training tailored to specific client requirements. He has an extensive background in testing consultancy and training and says he is "a fully paid-up testing enthusiast!" His areas of special interest are Rapid/Exploratory Testing, Risk Based testing and Test Process Improvement and he is a firm believer in the application of good test practices that fit the context of the project / mission.

For the five years prior to this Graham was the Principal Testing Consultant at Ajilon Consulting where he provided testing consultancy, technical support and training to a variety of clients. During this time he had a significant role in the development of leading edge approaches to test automation. As a result of this work he co-authored one of the advanced automation case studies in Software Test Automation by Mark Fewster and Dorothy Graham (ISBN 0-201-33140-3, Addison Wesley, 1999).

Before he joined the ranks of the consultants he was the Testing Process Owner at Scottish Widows where he was instrumental in the development and implementation of wide-ranging testing improvements in support of key business programmes, including Y2000. His involvement in testing began much earlier in British Telecom in 1983 as an Analyst Programmer, before moving to the flagship CSS project as Team Leader responsible for co-ordination of all user acceptance testing and live conversion.

Graham is a frequent speaker at international testing conferences including EuroSTAR, BCS SIGIST, SQAM and STG. He was a member of the Programme Committee for EuroSTAR2002 and is also the EuroSTAR country co-ordinator for Scotland. He has chaired tracks at several EuroSTAR conferences and is also the Chairman of the recently formed Scottish Testing Group (STG).

He holds a first class honours degree in Information Systems from Staffordshire University and the ISEB Practitioner Certificate in Software Testing.

Scott Barber

PerfTestPlus, Inc.

Performance Testing for Managers

Abstract:

One of the few common themes across all 50+ performance testing projects that I have been directly involved in, and at least most of the hundreds that I have either been indirectly involved in or heard tales of, is education. Performance testing as an activity is widely misunderstood, particularly by managers and executives. As you can imagine, this misunderstanding causes a wide variety of difficulties to include outright project failure. This presentation details the top topics that I have found myself teaching managers and executives time and time again over the last 6 years. Learning, understanding and applying these nuggets of knowledge on your current or future performance testing projects will dramatically increase your chances of success.

Overview:

For the Executive

1. Experienced performance testers will speak your language and guide you through the process to meeting your goals, even if you can't yet verbalize them.
2. Performance testing can and should begin long before the application is "fully functional"
3. "Delivery" is an informed decision based on risks and should not be confused with "Done".

For the Test Manager

4. Quality performance testers are senior members of the project team in terms of depth and breadth of skills and experience.
5. Quality performance testers need quality tools& and to know how to use them.
6. Extrapolating production loads from data collected on test systems is at best "Black Magic".

Supporting Information:

"High Performance Testing" Featured Article, Better Software Magazine, May/June 2005 by Scott Barber

http://www.stickyminds.com/BetterSoftware/magazine.asp?fn=cifea

Biography:

Scott Barber is the CTO of PerfTestPlus, Inc. (www.perftestplus.com) and Co-Founder of the Workshop on Performance and Reliability (WOPR www.performance-workshop.org).

PerfTestPlus is a software testing consultancy that focuses on technical and otherwise challenging software testing, mentoring, methodology development and effective test resource utilization. Scott's particular specialities are testing and analyzing performance for complex systems, developing customized testing methodologies for organizations, embedded systems testing, testing biometric identification and security systems, group facilitation and authoring instructional materials.

Scott is a columnist for Software Test and Performance Magazine, regularly contributes to Better Software Magazine and authored the widely recognized article series' "User Experience, not Metrics" and "Beyond Performance Testing." Scott has presented at STPCon, WSE, WOPR, AWTA, STMR, STiFS, PNSQC, STAR, 3WCSQ, the Rational User's Conference, SQE Testweek, various local users' groups and workshops, and has been a guest lecturer at both the Massachusetts Institute of Technology and the Florida Institute of Technology. He is a member of the Association for Software Testing, IEEE, American MENSA, the Context-Driven School of Software Testing and is a signatory to the Manifesto for Agile Software Development. You can view Scott's resume and a sampling of his work at www.perftestplus.com.

Featured Speaker:

Robert Sabourin

AmiBug.Com Inc.

Author: *"I am a Bug!"*

The Elevator Parable: Seven Steps to Building a Better Bug Workflow System

Abstract:

This workshop addresses one of the fundamental questions of software engineering: "How do we know we are finished?" Managing bugs is a critical part of any software development project.

In this highly interactive workshop, we'll explore the concepts of bug priority and severity, and you will learn how the priority and severity of bugs varies depending on a blend of the business and technical context. Development, project and SQA managers will learn a systematic approach to defining how defect data can be managed. Lead developers and testers will learn how they can contribute to the entire bug workflow life cycle. Which bugs should we fix? Which bugs should we keep? How can we decide consistently?

Featured Speaker:

Robert Sabourin

AmiBug.Com Inc.

Author: *"I am a Bug!"*

Deciding What Not to Test

Abstract:

Software project schedules are always tight. There is not enough time to complete planned testing. Don't just stop because the clock ran out.

This presentation explores some practical and systematic approaches to organizing and triaging testing ideas. Testing ideas are influenced by risk and importance to your business. Information is coming at your from all angles - how can it be used to prioritize testing and focus on the test with the most value?

Triage of testing ideas, assessing credibility and impact estimation can be used to help decide what to do when the going gets tough! Decide what not to test on purpose - not just because the clock ran out.

Biography:

Robert Sabourin is the president and principal consultant of AmiBug.Com, Inc. He has more than 20 years management experience leading teams of software development professionals to consistently deliver projects on-time, on-quality and on-budget.

As a respected member of the software engineering community, Robert has trained and mentored literally hundreds of top professionals in the field.

Robert is an Adjunct Professor of Software Engineering at McGill University and often writes and speaks at conferences around the world on software engineering, Software Quality Assurance, testing and management issues.

Robert is also author of the popular children's book "I am a Bug!" that explains what testers really do at work!

BOOK REVIEW

"Tasty Morsels from Rainsberger"

JUnit Recipes

J. B. Rainsberger, Manning 2005. ISBN 1-932394-23-0.
Paperback. Price £24.02

This thick volume (700 pages including good references and reading list) is aimed at three groups: Java developers in general, JUnit users, and lastly software testers. It is very useful to all three groups, and is marked as **** as a software tester (am I in a minority of one?). It may merit top marks for the other two readership groups.

JUnit is one of a series of language-dependent packages for those engaged in Test Driven Development. Important points to draw out are that this is describing both the building blocks for JUnit, and a process. The authors (important contributions from Scott Stirling and others, in addition to J.B. Rainsberger) make little in the way of assumptions. Not every reader will be an expert Java programmer, nor will everyone have used JUnit before.

Starting from the notion of building little tests as coding progresses, some tough questions are introduced early-on. How is production code to be separated from testing code? It is great having a fully rounded regression pack available, but is it possible to invoke only a subset of the tests available? Real questions from genuine situations that scratch where it itches! All this from the three principles of JUnit; create an object, invoke a method and test the result. Practical examples abound, and there are coded examples, for the most part very clear. Later on, some parts were beyond my level of Java technical understanding (particularly testing JavaBeans), but some testing points still emerge.

To build in the future-proofing of test packs, 're-factoring' is mentioned both early, and often. There are also some very common testing items. Here is one for a taster: if a module / class / object has no noticeable effect, why test it (and perhaps more pertinently, why code it!) There is also the idea that some items are not building blocks, and are too simple to test.

This is surely worth considering.

Development can learn from testing. This volume shows that all the traffic is not one way, and is a valuable addition to (some of) those engaged in software development."

Let me know

Peter Morgan
Software Tester
morganp@supanet.com

TEST MANAGEMENT CERTIFICATION SURVEY

Paul Gerrard, Technical Director, Systeme Evolutif Limited.

At the Test Management Forum meeting on Wednesday 26th October 2005, I introduced and facilitated a discussion of the "Skill Set of a Test Manager". Inevitably, the potential for a Test Management Certification scheme was also discussed. The discussion was lively and covered a broad range of issues relating to skill-sets, the ISEB/ISTQB scheme – other attributes of good (and bad) Test Managers, assessment and recruitment.

Although most of the participants had taken the ISEB Foundation certificate and some had taken the Practitioner certificate, none of the Test Managers in the room felt that the ISEB scheme was appropriate as a single certification for Test Managers. Some dissatisfaction with the scheme was expressed and there was definitely interest in the potential for a dedicated Test Management Qualification. In the session, there was quite a lot of interest in the room in getting involved in a group effort to define a Test Manager's skill-set.

One of the actions arising was that I said I would conduct a survey of attitudes to the ISEB/ISTQB scheme amongst the TM Forum membership and ask some questions about the potential for a Test Management Certification scheme. I created a survey accessible from the TM Forum home page and sent emails to the 150 or so email addresses of people who have attended the TM forum in the past. The survey was anonymous – we collected no personal information, but we are confident that the responses are from genuine Test Managers.

51 Test Managers responded. This is around 35% which, for an online survey, is a very healthy response. Normally, you would expect only 5% of the people on one's email list to respond. Clearly, there is a lot of interest in a Test Management Certification scheme.

A Personal Interpretation of the Survey

Statistics - Highlights

The vast majority of Test Managers in the UK have taken the ISEB Foundation or plan to take it. Clearly, the Foundation scheme is regarded as a useful base level qualification for the testing industry, often referenced in job specifications and recruitment advertisements. However, less than one in ten Test Managers believed the Foundation scheme significantly helps them to be better Test Managers. This is unsurprising as the Foundation scheme really covers basic testing concepts and only touches upon Test Management issues.

The Practitioner scheme, more recently developed, more expensive and time consuming, is less popular with Test Managers. One in three Test Managers said they weren't interested in the qualification at all. Only one in five Test Managers said the Practitioner Certificate would make them a better test manager 'a great deal'. The rest remain to be convinced. If the Practitioner were promoted as an 'essential qualification for Test Managers', it would fail. Few respondents thought the scheme fitted the needs of test managers 'a great deal'.

Only a quarter of test Managers felt that the ISEB/ISTQB scheme gave significant benefit to the IT Industry and Business. A minority thought it gave NO benefit at all. This seems to indicate that the perception of test Managers is that the Practitioner scheme is more focused on advanced testers, rather than Test Managers.

Only four out of ten Test Managers felt that the ISEB/ISTQB could create a useful Test Management certification scheme. The rest believed is possible, but were not positive in their beliefs.

A clear majority of Test Managers thought that a Test Management skills set definition or body of knowledge description would be very useful.

Patterns in the General Comments

The Test Managers who responded clearly had strong opinions on how the Test Management discipline should be supported by a certification scheme. In this section I have summarized and interpreted some of the, often strident, feelings expressed in these comments.

Few respondents believed that the ISEB certification scheme could be used as a differentiator between individuals who were Test Managers. There seemed to be little support for the ISEB Practitioner scheme as a useful benchmark qualification for Test Managers. Several respondents believed the Practitioner was too ideal-world, academic and focused on theory. Some expressed serious complaints that the scheme was too expensive, time consuming and not worthwhile.

A common complaint was that the Practitioner syllabus focused too much on developer testing, and technical testing skills not relevant to Test Managers. The Practitioner is not focused on practical, experience-based issues, but on peoples' ability to memorise course content, trot out theory and pass written exams.

Quite a lot of scepticism was expressed that current or future schemes will benefit Test Managers less than training providers or the certification administrators.

The consensus seems to be that a Test Manager's skill set would be predominantly project management and interpersonal skills. The technical testing skills promoted by the ISEB schemes are helpful, but are by no means the dominant skill set.

Many specific skills were mentioned in the comments received: the ability to build and lead a team; to negotiate test resources; to manage change; how to be an advisor; to identify ways to rescue a project, even. There seems to be a broad belief that Test Management skills set cannot be described in a 'one-size fits all' definition.

The most apparent variation in skill sets seems to be related to the Test Managers span of influence. For example, a programme level test manager might manage processes and senior test managers for a range of projects – the skill set is primarily interpersonal. A project test manager might have to deal with suppliers, developers, perform risk analyses, formulate test strategies, negotiate resources as well as have a good grasp of test methods. On a small project, the Test Manager's span of influence might extend only to his team – clearly, technical leadership, planning, scheduling monitoring and control are the key skills.

Perhaps a Test Management skill set could be structured along these three lines?

I would like to establish a group that would create a useful definition (or definitions) of a Test Management skill set. Would you like to help me?

Email paulg@evolutif.co.uk or keep an eye on the Test Management forum online web forum: www.evolutif.co.uk/tmforum

Paul Gerrard, 11 November 2005.

On the following pages, the response statistics and the detailed comments from respondents are reproduced.

By the way, I did not respond to the survey myself.

Questionnaire Results Summary

The analysis of answers provided to the nine questions are reproduced below.

Have you taken the Foundation Certificate?

Taken and passed	***38***
I'm in the process of taking it	***2***
I plan to take it sometime	***4***
I'm not interested	***7***

Would this make you a better Test Manager?

A great deal	5
Some	15
Not much	30
I have no opinion	1

Does the FOUNDATION Fit the needs of test managers?

A great deal	4
Some	15
Not much	30
I have no opinion	2

Have you taken the Practitioner Certificate?

Taken and passed	13
I'm in the process of taking it	9
I plan to take it sometime	12
I'm not interested	17

Would this make you a better test manager?

A great deal	10
Some	21
Not much	14
I have no opinion	6

Does the PRACTITIONER fit the needs of test managers?

A great deal	6
Some	19
Not much	17
I have no opinion	9

Does the ISEB/ISTQB scheme benefit the IT industry and business?

Significant benefit	13
Some benefit	23
Marginal benefit	8
No Benefit	3
I have no opinion	4

Could ISEB/ISTQB create a useful Test Management Certification scheme?

Yes	19
Possibly	26
No	4
I have no opinion	2

Do you think a Test Management skills set definition could be useful?

Very useful	37
Some use	13
No use	0
I have no opinion	1

Detailed Comments from Respondents

A free-format text area was provided for respondents to lodge any general comments on the ISEB/ISTQB scheme and a potential Test Management Certification. Of the 51 respondents, 34 provided a comment. With one exception (where an individual might recognize the situation referred to) the comments are reproduced as logged. Spelling and occasional grammatical mistakes have been corrected. (Interestingly, most people spelt Practioner incorrectly). Each bullet point represents one response. They are in the order received.

- It was my understanding that there was a further ISEB qualification being developed. Surely this would cater for any certification programme - although what is the objective here? Better test managers or more revenue for the training suppliers?
- The Practitioner course really covers two distinct skill sets - one is a test architect/senior test analyst. The other is a senior test manager/test strategist and for me the simple way forward would be to split the course in two - one for coneheads; one for managers and strategists.
- Practitioner syllabus useful. Exam a waste of time. The Test Management skills set definition would be similarly useful but an exam based on formula and memory would be similarly pointless.
- These certifications will not make a person a better test manager. These certifications will only teach the fundamentals.
- Although the Practitioner Course does cover some Test Management it is not the bulk of the course. The Practitioner course is considered very expensive. Maybe companies would be more willing to pay for a Test Management qualification rather than a System Testing qualification if it had the same recognition as Project Management and PRINCE2.
- The ISEB practitioner course does of course include test management but also includes a huge amount of material that test managers only need to know in outline (very detailed descriptions of test techniques for example). A Test Management certification could concentrate more on the management aspects to encourage more managers and potential managers to take the certificate.
- Be sure it is more relevant than the Foundation certificate. The ISEB material is so grounded in irrelevant non-real world material that you'd have to wonder whether it could ever arrive at a useful TM certification.
- Both the ISEB certifications are being used as a 'Qualification' not as a confirmation of knowledge. I have two testers one with ISEB Foundation one who has not. It is the tester who has no qualification who 'thinks' more.
- In my opinion the Foundation is just that. The Practitioner should be aligned to experience and work with people as they obtain knowledge of Acceptance, System, Performance testing etc. The Test Manager series is more where the practitioner is currently set.
- The ISEB Practitioner is a starting point - you are a better test manager with it than without it. It does not act as a certificate proving that somebody is a good Test Manager. It is hard to envisage how this could work when the important Test Management skills are soft skills and could only really be judged by an interview by Peers.
- I sent some of my people on the ISEB Foundation. They didn't learn much of use in their jobs – it was too broad and covered lots of areas that we simply had no need for. They learned nothing useful that our internal training program couldn't deliver anyway. Was great for their CVs though. I think the ISEB Foundation is of more use as an employment currency and less useful actually in the workplace. As for test management, the most powerful training I ever received was a 3 day SQE Test Process Mgt course followed up by some very well delivered management and leadership training. Plus a couple of good books: Martin Pol's TPI and Rex Black's Test Process Mgt. The ISEB Practitioner sounds good in principal but effort wise it's practically degree level. I just did a Software Engineering Masters instead which coupled with Test Mgt experience is much more useful in the job market as I'm finding out.

- I think that the ISEB/ISTQB has some value but limited for the Test Manager - I think a test manager scheme could be of value but only if it involved clear pre-requisites. e.g. so many years demonstrable experience and involved practical work as well as theoretical. I also think that the course needs to be aimed at the more commercial market and the development of the syllabus needs to be done by active practitioners - again those who wear the war wounds - "been there done it and seen it" rather that just written a book about it!
- I do not think you need a separate scheme. However, a hard and soft skills profile which identifies the wider skills required of a good Test Manager (ie Project Management skills etc) would be useful.
- The real problem is that it would possibly take years to establish and in its present form would by implemented by very few companies. All you need to do is look around at how many people in our trade actually hold ANY formal qualification of any sort and you get some idea of how many companies are likely to pay for their people to achieve ISEB certification. As an employer myself folks are only as good as their track record. I've met people who are fully 'qualified' who turned out to be incompetent and one of the best people I have ever worked with had a degree in Modern Chinese! The theory of qualification is great, but would it really make any practical difference? Thanks but I shall continue to read CVs, interview folks thoroughly and implement the 'probation period' option on the very few occasions where necessary.
- My main concern is that like a number of other certifications (e.g. Prince2 PM) accreditation becomes the panacea of competence within the job market and in my experience a high number of IT related practitioners are extremely good at studying and passing IT qualifications but have woeful ability in applying the knowledge thereafter.
- I don't think that ISEB Foundation or Practitioner in isolation is sufficient for certification. Perhaps some other evidence of the level of other skills should be required. People Management skills for example.
- When you look at SFIA and other skills initiatives the role of Testing Manager has been moved down the responsibility ladder to a technical role. This will be a strong counter force to this initiative.
- The scheme would need to include reference to the 'Soft' skills needed to perform the role e.g. leadership/management assertiveness ability to manage staff and products effectively ability to deliver under pressure (time usually) pro-active confrontation management high level understanding of the testing market place e.g. tools and their uses different software and their uses
- ISEB certification provides a useful indication of someone's ability to pass exams but for Test Managers, other measurable attributes of people relating to testing needs to be identified. And the attributes need to be less academic in nature.
- I think a formal certification process is a good thing - but the current ISEB course contains too many caveats in the nature of "I know we don't call it this and we know it doesn't work like that in reality but ..." To pass the exam, I regurgitated course content rather than used testing experience.
- Perhaps base it on an NVQ
- ISEB Certificate - lot of good ground work for those new to testing. Does codify well the core skills. However too much emphasis on *developer* testing (e.g. unit testing etc.) which is not needed/useful to testers. Taking this approach the Practitioner could be more tailored to real world things of a Test Manager rather than non-core activities. E.g. more on Project Management.
- The ISEB Practitioner gives a good basis for how one should manage testing. It's a bit of an "ideal world". A test management certification should measure candidates' ability to spot potential problems and avoid them and their ability to identify ways to rescue a project that is in trouble.
- Test Management is about a range of competencies that spans far more than testing. I'm of the opinion that the major difficulty will be defining its boundaries.

- Test Management, like Project Management, has to be tailored to the culture and desires of the client. Having said that there are basics that every Test Manager should be aware of - a course would be viable.
- I think the ISEB practitioner plus a Project Management qualification (Prince 2?) is enough.
- One of the major problems within IT is the lack of structure to qualifications and standards such as those in the accountancy world. It is true that a graduate with 6 months experience can be far more productive than other staff with 5 yrs experience. IT is a far more ability-based career. Many people gain qualifications but find it difficult to perform the role to a satisfactory level.
- ISEB certifications currently looking into only testing skills not management skills. So another test management scheme should be introduced.
- All certification schemes have limited direct applicability in any real-life situation. Further it is always unwise to rely excessively on certifications such as MCSE, SAP etc. In this respect the ISEB Practitioner is no worse than other practitioner certificates such as PRINCE II.
- Test Managers need a formal professional certificate, not only improve the level of competence, but to ensure their relevance and acceptance within the IT community.
- ISEB is all very well but doesn't cover all of the industry's needs as it mainly geared to test automation rather than early production testing which is the area we're moving into. Would be good to have a proper test management course that looks into all aspects - sales/marketing/handling staff/testing/automation/future of testing/QA rather than just testing/handling non-responsive or anti-testing teams/lessening test documentation etc etc etc
- As with most companies we have a skill set definition for test managers based on SFIA. I would be keen to work with the group to assist in defining an industry model.
- There are so many different types of test manager that trying to come up with a single definition will not work. Also, the most useful skills as a test manager are the same as for any other managers, i.e. how to get a team to work together, managing information, being an advisor, managing change. The pure testing element is not that great.
- As with many qualifications, the ISEB have some value for the recruiting manager when the candidate doesn't have a track record in testing. Most Test managers that I have discussed this with would prefer to recruit an appropriately experienced test specialist without ISEB qualifications than an ISEB qualified but inexperienced candidate. No surprise there. Having said that, an experienced candidate with an ISEB qualification may have the edge. I expect that a Test Management certification scheme may be considered in the same way by the hiring manager.

CLOSE

This paper can also be downloaded from the Test Management Forum website:

www.evolutif.co.uk/tmforum