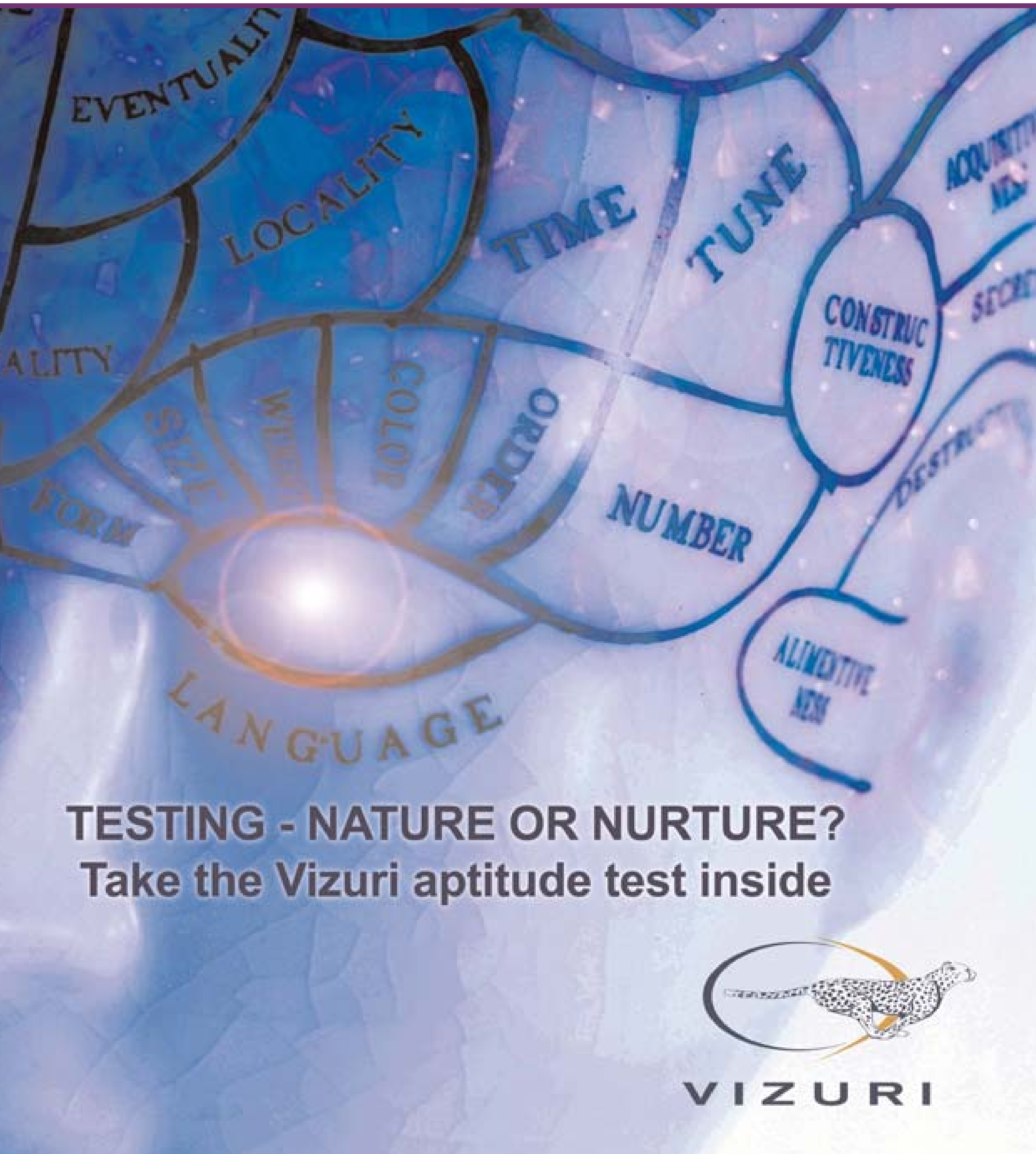
Professional TESTER

Number 16
October 2003

THE MAGAZINE FOR TESTING PROFESSIONALS £4



Having problems solving the software testing puzzle?

Whether it's Weblogic or Websphere, Tibco or MQ, Oracle or DB2**, nmqa** have the skills and experience to solve your testing problems.

For the most flexible, innovative and high quality software testing solutions, think:

n m q a

intelligent testing

For more information on this and other services, please visit www.nmqa.com

11-15 betterton street
covent garden
london
WC2H 9BP

tel: 0207 470 8818

email: info@nmqa.com

intelligent testing

We set the standard. But who noticed?

Because standards can be both disabling and enabling depending upon a number of factors, the fledgling specialist interest group in software testing approached the challenge of creating a practical standard for use by testers, in the late 1980s, with great care. The N19 standard had already been created, but this lacked credibility since it required only statement coverage.

Soon after, the group was incorporated into the British Computer Society and as founder chairman of the BCS SIGiST I was very keen to see a practical standard created. The team laboured long and hard. The standard was much modified, both in committee and later in conversion to what is now BS 7925. The contents are known, at least, by the many thousands that have taken the ISEB Foundation Certificate examination; but the really big question is "*is the standard used?*".

Although it is rarely cited, IEEE 829 is widely used as the foundation for test documentation which is then customised to meet the specific needs of the users. Similarly, testing standards for specific disciplines and industries are well known. But what about the highly practical BS 7925? If you are using it, we'd be very interested to know which sections you use, and the types of application being tested. Please send a brief email to **bs7925@professionaltester.com**.

If you are not familiar with all the available standards, I strongly recommend you explore them and the possibilities of using them on your projects; there is much wisdom and hard-earned experience to be found. I hope you will find this issue of PT to be a valuable starting point.

Geoff Quentin
Technical Editor

SECURITY BRIEFING: WHY VIRUSES HAPPEN

"Hey, Outlook! Run this program!"

"Okay!"

© every security site on the web, September 2003

Professional TESTER

Number 16 • October 2003

We apologise that Sauli Karhu and Eero Karejoa's article on testing with Perl does not appear in this issue due to the birth of Sauli's daughter; congratulations Sauli. If he can get any sleep at all, the article will conclude next issue.

The theme for the next issue, in January 2004, will be "Skills, training and certification". ***Please contribute your opinions, especially if you have attended training recently.*** *Can you do your job better as a result? What were the best and worst aspects? What do you expect or wish trainers to provide? Please state the topic of your training; it is not necessary to name the training provider.* ***Email training@professionaltester.com.***

Professional Tester is published quarterly by Test Publishing, 73 West Road, Shoeburyness, Essex SS3 9DT. Tel 01702 294491; Fax 01702 299223. Email editor@professionaltester.com. Views expressed by contributors are not necessarily those of the editor or proprietors. ©Test Publishing Ltd 2003. All rights reserved. No part of this publication may be reproduced in any form without written permission. Professional Tester *is a trademark of Test Publishing Ltd.*

News: developments

Industry generally busier • brand new products from Embarcadero and Original • major new versions from Segue, Seapine, Merant, Axway • Mercury busy doing something

Industry hotting up as weather cools down The end of the summer brought very encouraging signs of an upturn in IT and testing activity. Recruitment agencies tell us they are actively looking for candidates, rather than vacancies, again; educators report more people looking to update team skills ready for new projects and challenges. And perhaps the most reliable indicator is the increased number of people attending testing conferences and exhibitions. TestExpo in September was the busiest ever; highlights included a very well-received presentation on *Implementing quick wins that keep their value* by Neill McCarthy of event sponsors Quality Assured Testing, the launch of Embarcadero's brand new goals-based performance testing product *Extreme Test*, and a fascinating closing keynote on risk by PT's regular columnist Felix Redmill.
www.testexpo.co.uk • +44 (0)1702 290558
www.qa-testing.com • +44 (0)207 173 6202
www.embarcadero.com • +44 (0)1628 684443

In its second year ICSTEST-UK more than doubled its attendance; see the full report in this issue.
www.icstest.com • +44 (0)1483 733109

Finally, very high interest is reported in the last major event of the year, the EuroSTAR conference in Amsterdam on 1st-5th December. As always, we at PT are looking forward to attending very much and we hope to meet many readers there.
www.testingconferences.com
+353 (0)91 514470

Nice work if you can get it Andy Redwood, associate director of test consultancy Cresta, was sponsored to visit Wellington and Auckland by the New Zealand government recently as part of its scheme to introduce and promote IT specialisms. He presented structured workshops on various testing topics for several groups including testers, developers and clients from Cresta's NZ partner, business solutions provider Synergy.
www.cresta.net
+44 (0)207 448 4688

Up and running StarBase has completed the first presentations of its brand new training course teaching advanced skills in Mercury's LoadRunner load testing tool, announced in the last issue of PT. The course has been developed to answer the questions and address the challenges that arise when trying to use LoadRunner for complex, real world load testing scenarios.
www.loadrunnertraining.co.uk
+44 (0)208 905 1120

Multitracking Seapine Software's issue management application *TestTrack Pro* is now at version 6.0. The new release adds fully customizable workflows which let users define states, events, and state transition rules, allowing them to model workflows after their organisation's issue management process, workflow diagramming and multiple user assignments. Later this year, Seapine intends to offer pre-built workflows to address a variety of business problems and industry standards including change management, issue management, help desk processing, ISO 9000, CMM, CMM II, 21CFR11, and HIPAA. Users can now define up to 100 custom fields for their issues and these can be accessed via an enhanced SOAP API enabling integration with SOAP-compliant development tools.
www.seapine.co.uk • +44 (0)1344 297613

Flexible planning Original Software has launched *TestPLAN*, a project management and defect tracking utility which allows large elements of the testing process such as test scripts, test data cases, data comparison processes etc, from any combination of Original's other testing products, to be launched and controlled from within the test plan. It also assists communication between QA, UAT and development teams by enabling testers to add documents, images (eg screenshots), data files, test scripts and test data cases to tasks or defects and all authorised persons to access and update the plan. TestPLAN is compatible with MS SQL Server, Oracle, DB2 and the IBM iSeries.
www.origsoft.com
+44 (0)1256 338666

Remember, remember SCM suite *Merant Professional* (formerly known as PVCS Professional) version 8 will be launched in November. It includes significant improvements in security, performance, scalability, and support for distributed development, and fully integrated build management.
www.merant.com
+44 (0)1727 812812

Centralization… In what it describes as the biggest launch in its history, Mercury Interactive has announced its business technology optimization (BTO) blueprint, describing how its product range is being integrated into a strategy designed to help organisations maximize the business value of their IT systems. The products are grouped into 'optimization centres'; eg the products best known to testers, such as *TestDirector, WinRunner* etc are part of the Quality Centre. There is also a Performance Centre which includes *LoadRunner* etc. It appears the full significance of all this may be yet to become clear.
www.mercuryinteractive.com
+44 (0)1276 808200

…integration… Princeton Softech has integrated its *Relational Tools* test data preparation and analysis suite with suite with Mercury's BTO solutions. The included Move, Access/Edit and Compare programs provide advanced capabilities for migrating, editing, and comparing complex sets of related data and support the leading client/server and mainframe DBMSs; Mercury has certified the suite for use with its products.
www.princetonsoftech.com
+44 (0)161 266 1037

…and competition Axway describes its new *Sentinel 2.1* as 'the complete solution for efficient supervision and monitoring of business process flows'. Its user interface is accessed via a web browser, features intuitive display objects such as speedometers, and can be fully customised to construct a "cockpit" for overseeing activity.
www.axway.com
+44 (0)1895 202780

Silk smoother Segue has released version 6.0 of its *SilkPerformer* load testing tool. It now includes better support for Oracle, J2EE, and web services; the TrueLog feature is a visual display of exactly what has been sent to and received from the database server via interfaces including ODBC, JDBC, ADO and IBM DB2 CLI. Instructor-led online training is available.
www.segue.com
+44 (0)118 965 7721

Please send press releases, news etc to press@professionaltester.com

Nature or nurture?

We all know it takes an unusual personality to be a good tester. But do you ever ask yourself 'how did I get here'? Vizuri's lifestyle quiz will reveal all

Thoroughness, diligence, caution and pragmatism are all core qualities which we must possess to succeed in a profession which can be highly complicated and pressurised and which can appear dull to outsiders/those who don't 'get it' (ie normal people). After all it's rare for any child, when asked in junior school, to profess a burning ambition to be a software tester.

So what made us take this route, rather than becoming the traditional choices of the young, aspirant pre-teens: lawyer, sportsperson or fighter pilot? *[What kind of pre-teen wants to be a lawyer? Ally McBeal's dancing baby? -Ed]* Or even developers, engineers or IT managers? Were we born to it, or is it something we've drifted into?

It's the classic psychological 'nature or nurture' question. So that's why, in line with our intelligent approach to testing, Vizuri has persuaded the editor to turn a page of *Professional Tester* into *Cosmopolitan* magazine. Not something that happens every day, so make the most of it.

In order to see on which side of the divide you fall, answer the following set of questions, match your score to the appropriate analysis, and see whether you're a natural born tester, or just ended up there somehow.

Q1: What's your ideal way to relax?
- a) A couple of pints in the local
- b) On your PlayStation trying to figure out how to beat the system on *Grand Theft Auto*
- c) At your machine finding flaws in the latest edition of Windows

Q2: You're off to the races. How do you decide which horses to back?
- a) Gut feel, or choose a name you like
- b) Study the recent form in the programme then make an educated guess
- c) Spend the previous evening poring over your files of stats and clippings and arrive with your decisions already fixed

Q3: Derren Brown or David Blaine?
- a) Blaine – he's more of a showman
- b) Brown – he takes a more scientific approach to magic
- c) Neither – I can see straight through their little act

Q4: What does your CD collection look like?
- a) A mix of everything – dance, pop, rock, rap
- b) Retro and rocky, with the tee shirts to match
- c) Dangerously close to heavy metal and arranged in strict alphabetical order

Q5: What is testing for you?
- a) A job, a means to an end which supports my lifestyle
- b) A career, and it's good to have one that's enjoyable and challenging
- c) A real specialism which should be held in the same esteem as professional vocations like accountancy

Q6: How do you approach testing challenges?
- a) Do what I can and get support from specialists when I need it
- b) Follow defined processes and methodologies to reach the desired result
- c) I don't need to think and plan too much – it all just comes together

Q7: A light bulb needs changing. You say:
- a) That's trivial - I'll do it. Then we can get on with things
- b) It's not a showstopper. Let's do what we can without it
- c) That's not my job - I only turn the switch on and off. Also, I now need extra budget for regression testing to check that the change of light bulb won't affect other systems in the building, the street and the rest of the world adversely. And anyway, if I'd been involved earlier, a better light bulb that wouldn't have blown would already have been designed so we wouldn't have this problem…

Q8: You arrive on a blind date. What do you do?
- a) Turn up without preconceptions and try not to get nervous
- b) Give her ten minutes and then decide whether to get out fast
- c) Blind date? I'd never meet somebody without vetting them first

Q9: For women: My ideal man is…
- a) George Clooney – cool, confident and with eyes to die for
- b) Michael Moore – smart, cuddly and opinionated
- c) Anyone with code in their soul and a goatee on their chin

Q10: For men: My ideal woman is…
- a) Cameron Diaz – tall, blonde, beautiful
- b) Carol Vorderman – smart and sassy
- c) Anyone with code in their soul and a goatee on their chin

Q11: Which film personality do you identify with?
- a) John McClane in the *Die Hard* series
- b) Winston Wolf from *Pulp Fiction*
- c) Tom Ripley in *The Talented Mr Ripley*

So, are you a tester by nature?

If you answered mostly A: Are you really involved in software testing? Or did you think *Professional Tester* was some kind of consumer advice mag? Testing is not so much your pre-ordained destiny as a destination you've ended up in by mistake – and you can't even remember the route you took. Stop browsing this magazine now and pick up a copy of *FHM/Glamour* instead.

Mostly B: A bit of both. You have many of the inherent attributes of a natural tester, but you still have to work hard to grasp many parts of the job. Maybe it wasn't always your intention to be where you are today, but now you're there you've embraced it and see it as a logical vocation. Nevertheless, you have found a good, healthy balance - you still don't let it invade your personal life too much and are willing to take the odd risk.

Mostly C: Obsessively, compulsively, living, sleeping and working the job. You're a born tester, with the typical dogged, meticulous nature which pervades all areas of your life. If you see a risk, you deal with it or you leave well alone, and you analyse everything to the point of mania. Unfortunately, those around you don't always share the same outlook. Well done, you've found your natural role in life – which is lucky for the rest of us too because it keeps you off the streets.

Every member of Vizuri's staff, not just testers, has scored at least 90% Cs on this test. They are also all qualified to at least ISEB Foundation Certificate level, and have an enormous wealth of experience in finding and implementing appropriate, pragmatic and highly effective solutions to testing challenges faced by major organisations. If you fit the profile – or if you don't, and you want someone who does to do the worrying for you – contact Paul Dixon (10 Cs), Matt Arpino (9) or Gary Kingsmill (9) on +44 020 7297 2030 or visit www.vizuri.ws

Over 100% growth for ICSTEST®-UK

The recent 2003 ICSTEST®-UK, organised by SIM/SQS and held at London's Café Royal, bucked the recent downward trend for conferences and exhibitions by doubling its attendance and number of exhibitors. Dave Sudworth of SIM/SQS reports on proceedings

The number of exhibitors grew from 13 to 29 and the number of attendees grew from 125 to 250. Sponsors, supporting organisations and media sponsors saw the value of being associated with the largest conference on Software Testing in the UK - with 12 supporting organisations in 2003 compared to nine in 2002. They included BCS SIGIST, Computing, Intellect, ISEB, Mercury Interactive, Munro Datacom, MFESD, Newell & Budge, Professional Tester, Austrian Computer Society, SIGIST Israel and Testing Services Group.

The conference took place over two days on September 9th and 10th with three separate and distinct tracks each day - addressing topics of major interest to all professionals involved in Testing and Quality Management.

For the first time in 2003, a Tutorial Day was organised on Monday September 8th enabling delegates to opt to spend a whole day covering a topic in depth. These tutorials were presented by experts in the field and provided practical advice and real world solutions to those attending. These tutorials proved to be a real draw – with a significant number of delegates opting to spend three complete days attending the conference testimony to the importance placed within the Testing Community on the effective transfer of knowledge delivered by the conference.

The topics to be covered by the conference were researched a year in advance to ensure that the very latest issues concerning the industry would be covered. Keynotes on the first day covered 'Testing in Convergent Technologies' by John Noakes, .NET advisor from Microsoft and Rudolf van Megen from SQS followed with a presentation on 'Advances in Test Case Specification and Design'.

After the first keynote, the conference then split into three tracks. The first track entitled 'Testing Mobile Applications' included presenters from Nokia, Symbian, Sony Ericsson and NSTL. The second track 'Testing in the Real World' saw presentations from Vodafone, EDS and University of Hamburg. The world's leading suppliers presented a third track, covering the latest in tools and services.

Keynotes on the second day started with 'Testing within Smart Sourcing' by Jan-Gerold Winter of Deutsche Bank and was followed by 'Cognitive Illusions in Development and Testing' from Dorothy Graham of Grove Consultants. Again the conference split into three tracks.

The first track entitled 'Test Management' had real-world project examples from Microsoft, npower, IBM and Tite & Lewis. The second track 'Non-Functional Testing' had examples from ABN Amro Bank, Vanguard Technologies, Marks and Spencer and Bank of Scotland. An extension of the Tools and Services track from day one completed the second day.

Feedback gained from delegates rated the conference between 'good' and 'excellent' in all areas. The best-attended tracks were, unsurprisingly, 'Testing in the Real World' and 'Test Management'. Topics such as, 'The Past, Present and Future of Software Testing and 'An Analysis of Solutions Testing' attracted high levels of interest.

Some comments from those attending

"The variety of products and services on display in the exhibition was very interesting. We had the usual comparison of features from testing tool vendors, but we also saw products that were unique in their offerings and surprisingly fresh in the testing and quality issues they address. It is good to see new ideas coming from vendors, and it is specially good to see established vendors realising they have a value to add in testing and quality management."

"The thing I really like about ICSTEST is the feeling of community and sharing. People that are new to testing have a chance to meet and hear the more experienced practitioners in testing. Talking to people that have tackled some of the issues we all face is invaluable in getting tips as well as instilling confidence to persevere. For experienced people it is remarkably easy to approach anyone and talk about testing. When even the vendors are sharing information and experiences with each other, I realise how special the whole experience is. I am certain everyone left the conference with new ideas and renewed enthusiasm."

• 2003 ICSTEST®-UK is one of a series of International conferences arranged by SQS. Other conferences include 2003 ICSTEST®-NL to be held in Noordwijk, The Netherlands from October 21st to 22nd 2003, ICSTEST®-E to be held in Bilbao, Spain from November 24th to 25th 2003. 5th ICSTEST–International Conference will be held in Düsseldorf, Germany from April 21st to 23rd 2004. Details of all the conferences can be found at **www.icstest.com**.

Gratuitous violence and creative destruction, over a nice cuppa.

Using a new system in anger without testing it in advance could put your business on the ropes. With inadequate software testing costing businesses literally billions of pounds a year in lost productivity and disrupted operations, Cresta's new Enterprise Testing Education Centre (ETEC) allows you to test applications to destruction without beating up your organisation.

Through simulations, courses, training and consultancy provided in an informal eCafé, the ETEC provides a single source of education and training to allow organisations and individuals to improve their understanding of testing in major IT environments. Based at our eCafé in the City of London and backed by Mercury Interactive, the global market leader in business technology optimisation (BTO), Cresta, through the ETEC helps you protect your investment in applications which include but are not limited to SAP, Siebel, Oracle Apps and PeopleSoft.

The ETEC is open 12/7 and offers both supervised learning and instructor-led training, not only during the working week but also in the evenings and at weekends. All our staff are professionally qualified and field-experienced, so we can help you across the whole range of testing disciplines – from exploring specific scenarios and applications to understanding and implementing best practice.

For testing professionals, we offer a career development path leading to industry-recognised qualifications.

So drop in for coffee. We'd love to show you how to put an armlock on your systems.

For more details call Mike Stevenson on 020 7448 4688
email: training@cresta-group.com
visit: www.cresta.net/testing_training.php

cresta
Proving the difference

www.cresta-group.com

CHANNEL ALLIANCE PROGRAM
MERCURY INTERACTIVE

Dutch encouragement

Erik van Veenendaal and his fellow consultants Rogier Ammerlaan, Rob Hendriks, Vivian van Gansewinkel, Ron Swinkels and Mark van der Zwan of Improve Quality Services in Holland discuss their practical experiences of using testing standards

It is important to understand that standards are not just theory, but more often reflect industry best practices. Over the years a number of useful testing standards have been produced. They can support you in doing more effective and more efficient testing. We would like to encourage you to look at some of the standards discussed in this paper, and study them to see where and how they can support you. Remember, this paper has been written from a practical background. The standards have made our 'testing life' a little bit easier; now it's up to you!

BS7925-2 Software Component Testing

The BS 7925-2 standard focuses on software component testing. It contains a short guideline for a component test process. A large part of the standard is dedicated to test case design techniques, both black-box (eg equivalence partitioning, state transition testing) and white-box (statement coverage, decision coverage). In the annex, each technique is explained in detail using an elaborate example. Related coverage measures are also provided. Although the standard's name indicates that it can only be used for component testing, the black-box test case design techniques can be applied at almost any test level.

We have often used the standard as an aid to specifying test cases on system level. In particular, the descriptions of equivalence partitioning, state transition testing, boundary value analysis and syntax testing are very useful. The elaborate examples provided by the standard support the implementation process. Many test engineers have by now attended an ISEB Foundation Certificate course, which covers the BS7925-2 standard. This facilitates communication and implementation.

According to BS 7925-2, a model derived from the requirements specification is needed for each item you want to test. A model can be, for example, a state transition diagram, a cause-effect diagram or a table of inputs and equivalence classes. Having a good and complete model is essential for quality testing. To derive test cases from the model, BS7925-2 has often been used. However, in most projects, the test techniques are used slightly differently as described (eg the switch coverage rules of state testing were sometimes not explicitly followed). The BS 7925-2 techniques are often combined with other test design techniques, eg test use case technique. Use case testing is typically a system test technique and currently not described in BS 7925-2. The standard could be improved by extending it to some informal and system level test design techniques. Overall, the BS 7925-2 standard has proved very helpful during the test design phase. It supports the testers in establishing thorough test designs.

IEEE 829 Test Documentation

IEEE 829 is a standard that describes a set of basic test documents, eg test plan, test case specification, test summary report, applicable for software testing in any environment. For each individual document the standard describes the structure and content. Additionally, an example of each type of test document, including implementation and usage guidelines are provided in the annex. This helps to understand the standard and supports the application.

The standard can be regarded as a useful guideline for test documentation. Since for each type of test document, the purpose, outline and content is provided, applying and implementing the IEEE 829 standard in a particular project and/or test phase is not too difficult. We've also found that the descriptions (purpose and outline) of the documents are highly usable in practice. The tester is still free to decide whether or not to implement all of the documents described in the IEEE 829 standard or even only a part of the document, which is considered relevant for the particular project. The tester can also determine how to apply the documents. The standard does not require a specific test design technique or test tool, and is therefore easy to use without many pre-conditions or environmental needs. Often companies have already created their own set of test documentation standards. In these situations the IEEE 829 standard can be used as a checklist to identify improvements in test documentation. An advantage of applying this IEEE 829 standard is that it covers a broad range of test documents, from test plan and test design specification to the test summary report. When testers within a specific project are all working according to this standard a consistent structure and terminology for all documents will be achieved. No unnecessary overlap in documentation or inconsistent terminology between the different documents is to be expected, even if the testers have different background and experience, which is often the case within real-life projects. The only test documents missing in this standard are the test policy and test strategy documents. For this we currently used the descriptions provided by the ISEB Practitioner syllabus. So far, we have used the IEEE standard to create useful test plans, test design specifications, test case specifications, test logs and test incident reports in our projects. The common reference has facilitated communication, thus saving time and money.

IEEE 1028 Reviews and Inspections

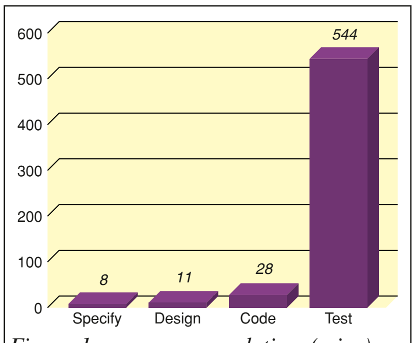


Figure 1: average rework time (mins)

The IEEE 1028 standard for software reviews describes a set of review types and processes applicable throughout the software life cycle. The standard distinguishes three main types of reviews: the technical review, the inspection, and the walkthrough. Per review type the process steps are described on a detailed level. The standard is completed with the description of the software audit process. The annex provides a comparison of the different review types to support the selection process of the appropriate type of review. The use of informal reviews seems to be common practice, but few organisations are applying reviews and inspections to their full extent. In practice review types are often mixed and simply named 'reviews' or 'inspections', resulting in a very time-consuming and ineffective process. A walkthrough, focused on achieving consensus, is something very different from an inspection, focused on formal defect finding. The major advantages of using IEEE 1028 are the clear definitions and straightforward process steps it provides for the different review types based upon different objectives.

We have implemented reviews in many organisations based upon IEEE 1028. It provided a thorough reference framework and common terminology. It just doesn't make sense to define 'new' review types, where the standard already provides them for you. A disadvantage of the standard is that it does not explicitly address and explain a number of critical success factors for reviews, such as optimal 'checking rate', 'logging rate' and 'number of pages per review'. If more information on implementation is needed, one might refer to books like *Software Inspection* by Tom Gilb and Dorothy Graham and *Walkthroughs, Inspections and Technical Reviews* by Freedman and Weinberg. These books provide numerous tips on how to make IEEE 1028 even more applicable. As stated, we have used the standard to check and complete review and process descriptions within different environments. The common reference has helped us to create guidelines, untangle existing review processes and get the message across that different objectives ask for different review types. Reviewing, when applied correctly and early in the process, remains the most effective and efficient defect detection technique. (See also figure 1 'Average rework time' that shows recent data from one of our clients.)

ISO 9126 Software Quality

Without a solid definition on quality, communicating on this very broad term is useless. The most practical definition on software quality is provided by ISO 9126. The standard describes how software quality can be expressed by means of quality characteristics. This standard distinguishes six main characteristics (functionality, efficiency, usability, reliability, maintainability and portability) and 26 sub-characteristics. In addition to providing a definition framework for software quality, the ISO 9126 standard also provides a number of useful internal and external metrics that can be used by the tester as a source when defining completion criteria.

In practice non-functional quality is often not or only briefly specified in the requirements. This causes a problem for testing, as it has to make a judgement on product quality. ISO 9126 is an excellent checklist to use throughout the risk analysis phase, in order to identify the most important quality characteristics. Subsequently the standard is useful for specifying measurable completion criteria. This makes answering the question "Is the product good enough to release?" more straightforward and less subjective. Once the completion criteria have been defined for the various relevant quality characteristics, it appears to be easy to report on quality progress. The completion criteria quantify quality in terms of metrics, and can be used during testing to report progress and status, eg a completion criteria for reliability is Mean Time Between Failure (MTBF) at 40 hours and the current MTBF is at 23 hours, thus the product cannot be released. This makes quality tangible and visible. The metrics provided by the standard can be customised for the project. This makes the standard flexible: one doesn't have to use all metrics and can easily add one's own metrics.

We have found the ISO 9126 standard very useful in our test projects. However, this standard should be used during the entire development process, because in each phase of the process quality related actions must be taken. In the requirement phase, quality must be defined. In the subsequent phases, activities must be executed to design and/or measure quality to assure that the product fulfils the targets. At the International Organisation for Standardization (ISO) research is continuing to improve the ISO 9126 standard. Current work is carried out towards improvements on specifications of quality requirements.

TMap (Test Management APproach)

TMap, the Test Management Approach to structured testing is applicable to both lower and higher level testing of software products. TMap provides detailed answers to the what, when, how, where and who questions of testing. To structure the organisation and execution of the test processes, TMap is based on four cornerstones:

- a development process-related life cycle model for the testing activities;
- solid organisational embedding;
- the right resources and infrastructure;
- usable techniques for various testing activities.

In recent years TMap has evolved and is now recognised as the standard for software testing in The Netherlands and Belgium. Most Dutch organisations, especially banks, insurance companies, pensions funds and government departments use TMap. To support the approach, books are available in various languages. TMap is applied most frequently as an approach to high-level testing (system and acceptance testing), and is probably less strong on component, non-functional and static testing. TMap is a generic, yet highly practical approach and consists of an extensive number of supporting techniques (together with BS7925-2 'all' techniques are covered), tools and procedures which may be selected for a specific test project. In the numerous projects where we have applied TMap, it has always been a methodology that fulfilled the needs of these projects, not only because of the availability of a well-documented approach, but also for the great number of templates and useful tips that are available. On several projects it has been helpful during the definition phase for structured testing. The TMap approach has more or less served as our 'complete guide to high-level software testing' during many assignments at customers' sites.

Testing Maturity Model

The Testing Maturity Model (TMM) framework has been developed by the Illinois Institute of Technology as a guideline for test process improvement and is positioned as a complementary model to the CMM. Just like the CMM, the TMM also uses the concept of maturity levels for process evaluation and improvement. The TMM consists of five maturity levels that reflect a degree of test process maturity. For each maturity level, a number of process areas are defined. A process area is a cluster of related activities within the test process, eg test planning or test training. TMM addresses both static and dynamic testing, and with respect to dynamic testing both low-level and high-level testing are within its scope. The structure of the TMM is partly based on the CMM and the staged version of its successor: the Capability Maturity Model-Integrated (CMM-I). This is a major benefit for organisations that are already familiar with the CMM(I).

Our recent projects have involved supporting test organisations, working on embedded software and technical automation, in achieving TMM level 2. We found that the model is highly usable and focuses on the practical needs of most test projects, eg test planning, risk analysis, test design and incident management. (TMM makes use and references the IEEE testing standards.) Since TMM level 2 also addresses the test policy and test strategy, it helps to involve management in test process improvement. Having test performance indicators derived from business goals is necessary to show the added value of implementing TMM. The papers and book available through the Illinois Institute of Technology, but also the guidelines that we, together with several partners, developed, supported the implementation process and provided concrete and practical guidance for test improvement. Figure 2 shows a major reduction in system test lead time, reported by one of our clients who achieved TMM level 2 shortly after completing the final project shown on the graph. PT

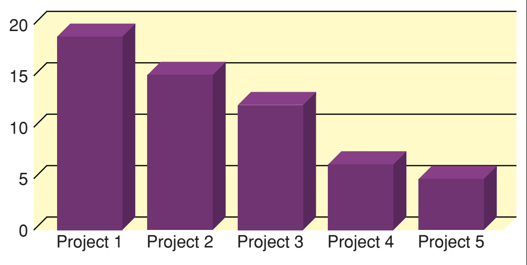


Figure 2: test lead time (weeks)

Can we ever finish the paperwork?

Documenting software development can be a time and cost-consuming process.

Brian Hambling, technical director of software acceptance at ImagoQA, challenges the role of software documentation standards and asks whether they are really needed

Software engineering, by its very nature, is a confused animal. A long and painful history of high profile failures has left the community in a schizophrenic state of mind: one persona wants to wrap itself in process while the other wants to stride out into a Brave New World, armed only with the latest method that will enable more rapid solutions to more complex problems than ever before. It is the difficult function of the quality practitioner to resolve these conflicts and while they naturally align themselves with the pro-process group, the daily challenges they face have encouraged them to take a more pragmatic view. Software documentation, therefore, represents the fulcrum of two schools of thought; the ultimate symbol of process for those who see it as pure bureaucracy and the most tangible evidence of due process for those who see it as essential to success.

The debate is further complicated by the uncomfortable reality that the production of software documentation is usually an uninspiring part of a project that you have already given your heart and soul to deliver. Indeed, documentation is probably the only part of the software engineering discipline that is less popular than testing, because it can be an expensive and time-consuming activity. However, one of the main purposes of software documentation is to enable effective testing to take place and it remains an important part of effective testing practice, with new standards appearing regularly.

So, what is software documentation? It can be defined as anything that records information about a software product or service. The areas that are most focused on are requirements documentation, design documentation, test documentation and user manuals but there are a number of other areas that might need documentation for a particular system. All of the key areas need to be documented and, to be effective, all of this documentation needs to be consistent and accurate.

Software documentation has three important roles in the software development process:

1. To capture the requirements and the emerging design so that the development process can be controlled
2. To provide a basis for quality assurance and, in particular, for the design of the testing activities
3. As a baseline for future maintenance activities.

If this approach is implemented in detail it will be expensive in both time and resources. However, good documentation can reduce development costs and timescales by reducing the number of mistakes, omissions and repetitions that lead to rework. It can also reduce the cost of maintenance by making it easier to identify the impact of changes and easier to implement and test the changes. Furthermore, it can enhance quality by reducing the number of errors and the impact of changes made to correct those errors.

Documentation is also essential to enable the effective use of reviews and inspections which, ironically, are used to improve the quality of documentation. This demonstrates that there is a built-in assumption that better documentation means better code and better software products, but is that justified? Does good process make good product? If we get all the documentation right, will we get the right end product? Not necessarily so. In fact there is a long chain of logic that leads inexorably from the decision to deploy a structured development life cycle. An approach like the V life cycle, for example, demands that a sequence of development and test phases be defined, leading to the necessity to verify each phase, and thus the need of a baseline for the verification activities. This baseline is software documentation. The real irony is that few organisations that adopt this stance actually implement the life cycle effectively enough to reap any real rewards.

Of course there is a counter argument to this. This challenges whether the right end product could be achieved without going through the exercises of collating information, checking assumptions and demonstrating traceability and conformance to requirements. Put simply, does the process we go through to create documentation actually add any value? Yes, it certainly does, and the thought processes are essential even if the documentation is not produced, with the proviso that the thinking is done at the right time and by the right people. For example, if software documentation is written after the event it will have no real value. Equally, a technical writer who was not involved in the development process at all cannot be expected to write accurate documentation. It is true that using a technical writer to produce documentation after the event might minimise the cost of producing the software documentation in the short term, but there is a real risk that what is being produced has no practical value in the long term.

Standards are, therefore, necessary to avoid ‘reinventing the wheel’. If we know what needs to be documented and we have identified a way of making documentation clear and useful, we should use that knowledge wherever we can. This not only saves time in production, but it also makes it easier to check that documentation is complete and at the right level of detail. A standard can also provide guidelines or templates to simplify the task for anyone new to the project.

It is also clear that there are several practical problems associated with using standards. However well written, standards cannot be ‘all things to all people’. If they address one area particularly well they will inevitably be less well suited to other areas. A life cycle process standard, for example, cannot deal with development or testing issues in enough depth to satisfy the specialists. Similarly, a standard on structured testing may not offer much help to a tester operating in an Extreme Programming (XP) environment. Often a standard can imply a particular world view that is not relevant to all its users; the author may be an advocate of a particular structured method or an evangelist for iterative development, and that flavour will be visible in the standard. As a result, many users find standards hard to work with because they do not fit either their technical culture, their business environment or their specific requirements. Conversely, the standard may require a level of detail in the documentation that is simply inappropriate in a particular case and the result is often that critically important information gets only superficial coverage because the documentation author tries to 'fill in all the boxes'. Quality checks based on standards, too, can be superficial because the checker is drawn into checking the 'fit' of the document rather than evaluating the content.

In the testing sphere, IEEE 829 provides an example of the issues that standards can raise. To begin with, IEEE 829 is entitled 'Standard for Test Documentation' but mostly addresses the specific content of a test plan. It is generally accepted that test plans are driven by test strategies and test strategies are driven by test policies. That is certainly the view of the BCS, as documented in both the ISEB Foundation and ISEB Practitioner Certificate syllabuses. Yet IEEE 829 embeds the heading 'approach' within its test plan outline. Does that mean test strategy? If not, what does it mean? Are the authors of IEEE 829 really sending a message about the relative significance of test strategies and test plans? The point here is not so much to question IEEE 829, a standard that many have found extremely helpful, but to recognise that standards can convey unintended messages that generate uncertainty.

Then there is the issue of unit testing. There are a variety of unit (component) test techniques with associated measures for test coverage documented in BS7925. Are developers expected to use these techniques and deliver code tested with these techniques to a defined level of coverage? While this might be an ideal scenario, it is not reality and even if it were so, would our products be better? Not necessarily. What purpose is the standard serving here? Is it setting a standard to which we should seek to aspire? Is it providing an arbitrary or academic baseline for good practice? Is it documenting impractical techniques for the sake of completeness? Like IEEE 829, BS 7925 is a very valuable standard, but its very completeness and comprehensiveness can intimidate the newcomer to testing and mislead the unwary into believing that the practices it documents are typical of the real world.

Another powerful example is the ubiquitous V life cycle. For decades, the testing community has based much of its theory, reflected into a variety of standards, on the principle that good development is based on good requirements and that testing based on those requirements should begin in parallel with the requirements documentation activity. Yet practitioners will be aware that 'good' requirements are as rare as rocking horse manure and requirements invariably change during the life cycle, causing massive rework to any test design based on them. They will also recognise that any life cycle that depends totally on the completeness, correctness and stability of initial requirements is fundamentally and fatally flawed. Given that the main requirement for software documentation and the standards that support it is the nature of this flawed life cycle (or rather the nature of any sequential life cycle), why do we continue to build standards around that model? Meanwhile, the really valuable lessons we have learned from the model – the approach we call risk-based testing – appears nowhere in a standard.

Perhaps the approach to software documentation just needs to be more creative. For example, writing a user guide before we write the application code could help to ensure that developers focus on what the user expects at a practical level. We could, perhaps, rely on software development tools to generate a database from which software documentation could be extracted on demand rather than writing it in case someone may wish to read it. Given that a professional system testing team will discover how a software product actually functions and will also know how its behaviour differs from the behaviour anticipated by the requirements, it seems sensible that the outcome of system testing should be used as the basis for system documentation. Perhaps system testers should be the generators of the 'as built' documentation. Perhaps the test suite itself is the best representation of what the system does and the standard we need is a standard for generating a systematic test suite; that, at least, would have a value after the system is delivered.

Even more radical is the notion of testing before development, as advocated by adherents of XP and other Agile development methods. Agile development asks hard questions of conventional software documentation. Indeed, the Agile community disagrees with some of the fundamental principles that underlie the strong process approach of conventional methods, and one of their justifications is the large volume of documentation that gets created but never gets used.

Whatever path documentation standards take in the future, there is no question that they will remain integral to software development in order to maintain system quality. Test practitioners are faced with the balancing act of adopting best practice procedure for current standards while constantly questioning their relevance and benefit to the project in hand. This ensures that documentation supports practical development without becoming a theoretical burden. PT

Automated Testing Just Got Easier

Seapine **QA Wizard**™

REGRESSION TESTING TOOL

- ▶ Powerful capturing capabilities
- ▶ Easy database connectivity
- ▶ Powerful checkpoint engine
- ▶ Browser-based toolbars
- ▶ User Friendly Graphical Interface
- ▶ Multiple filtration systems
- ▶ Organisational tree
- ▶ Highlight Indicators

Rapidly create automated test scripts for Windows and Web applications

CONTEMPORARY
Contemporary plc

Seapine Software™

Call now on 01344 297 613

Download evaluation at: www.seapine.co.uk

Testing needs to get into the real world

Paul Down, a senior consultant at Embarcadero Technologies, says goals-based testing eliminates cost and time burdens common to traditional performance testing methodologies

Almost every application suffers from performance problems at some time. It is difficult to find a well-tuned application and even harder to find one that stays that way in the face of growing user loads, network traffic, and data volumes. Neglecting these performance problems can lead to not only poor end-user experiences, but also application outages and that means real business costs.

The goal of a performance test is, therefore, to determine how a new or existing database, system or application might respond under real world stress and how well it can scale. It is also critical that testers discover whether resources are used effectively. Finally, before any new system can be deployed, organisations must be confident that their existing infrastructure can cope with the new workload, either by harnessing the resources they have or acquiring new ones, such as additional CPUs, and so on.

Today, given the complexity of the challenge, performance testing is often an incredibly painstaking, expensive and time-consuming burden.

Testing times

Every engineer understands the frustrations of traditional approaches to performance-based testing. The tester reaches the end of a day's test cycle only to discover that many hours ago, the test failed because a critical condition was not included in the original test criteria. While this will be annoying for individual engineers, it is the greater business cost in terms of time and effort that is a more pressing concern.

The problem is that typical performance-based testing routines only really provide engineers with basic load-testing functionality; they provide a hypothetical evaluation of systems under test. Provided systems can handle a given workload the test will continue, regardless of any other conditions that might in reality justify an early test termination. It would be much better if performance tests reflected the complexity of real-world production activities by systematically evaluating the performance of applications under stress to identify potential bottlenecks before these systems went into production.

However, the over-simplistic nature of current approaches to performance-based testing means that an engineer might have to run the same test tens of times to achieve such a result and as any senior test manager will tell you, this is a very inefficient method of testing.

Today, the pressure on testers to deliver enhanced business performance has brought the inadequacies of traditional load-based, performance testing to the fore. More than ever before, organisations are completely dependent upon an ever-growing enterprise IT infrastructure to perform basic operational functions. Any system outages or performance drops have a dramatic and costly knock-on effect across the enterprise. Now, because of the time-consuming and expensive nature of current methodologies, engineers are calling for a testing revolution.

Stakeholder pressure for change

The business user of a proposed application or system and the technical team responsible for its ultimate deployment both face difficult challenges today.

Business sponsors must specify performance parameters from the end user's perspective. For example, considerations such as ensuring fast transaction response times, as well as the number of users the system can support must be incorporated into test procedures.

Once business sponsors have defined their requirements, the technical team can begin to calculate the impact of new development projects on overall system performance. System performance is the foundation for a positive end-user experience and refers to acceptable levels of resource utilisation when under stress. Since performance testing is the process of applying load to the system to determine how it reacts, these parameters must be included in the performance testing process. Issues such as CPU utilisation become critical when safe thresholds are exceeded, because if the CPU cannot cope with its workload the end-user experience will be poor.

Ultimately, the business sponsor and the technical team must find a way to achieve positive end-user experience without taxing the infrastructure. This places a heavy burden on performance test engineers. They are tasked with determining if the application infrastructure and the performance test infrastructure are performing well, which is challenging even under normal circumstances. It can become almost impossible when there are many performance thresholds, including many different transaction response times and system performance statistics. Goals-based performance testing enables test engineers to overcome these difficult obstacles.

The goals-based testing difference

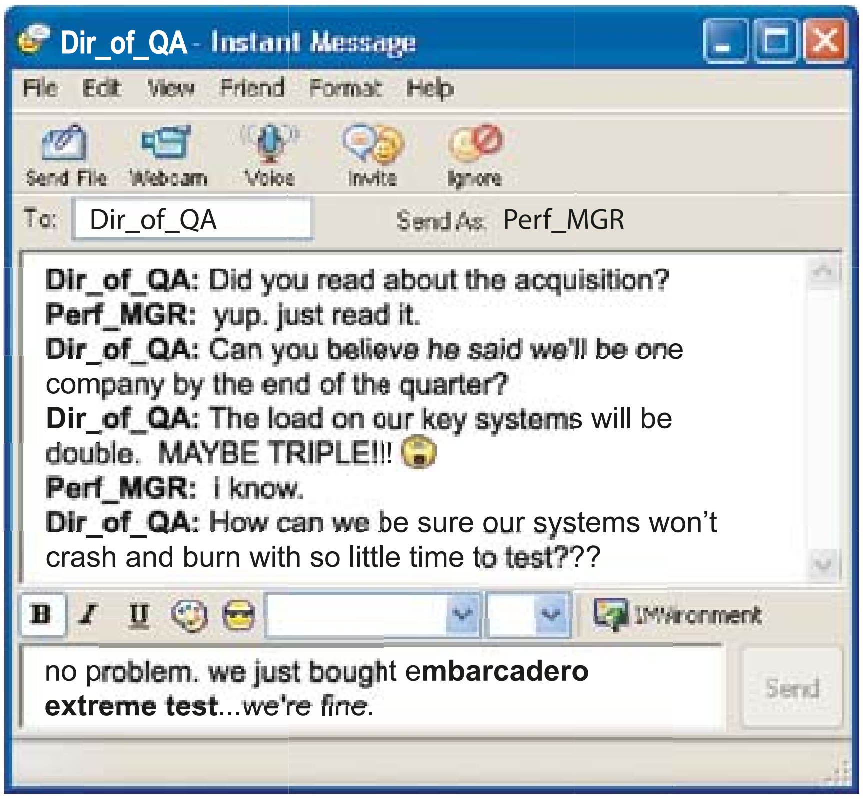
Whereas traditional performance testing can evaluate test systems for simple criteria such as load, the ideal would be that test routines should reflect the challenges of the production environment. In other words, tests should be able to evaluate systems for multiple criteria, including users, activities and data. Traditional testing methods do not offer this multi-dimensional test functionality.

The goals-based testing approach to performance testing delivers the multi-layered capabilities that test engineers are calling for today. This approach does not just operate on the principal of a simple load-based threshold; it allows testers to specify a number of critical conditions, which if attained, will facilitate a subsequent action, such as the premature conclusion of a test, which saves time.

The key difference between established performance testing and goals-based performance testing approaches is that the goals-based approach enables users to ask critical performance questions before the test begins. This pro-active test methodology enables testers to set thresholds and parameters during test configuration to ensure the test results meet all the defined requirements. This dramatically accelerates the testing process because tests can be aborted should thresholds be exceeded.

A second major benefit is that goals-based performance testing gets much closer to simulating real production environment conditions because it supports complex, interlinked test

With Embarcadero Extreme Test, the impossible becomes *possible.*


Dir_of_QA - Instant Message
File Edit View Friend Format Help
Send File Webcam Voice Invite Ignore
To: Dir_of_QA
Send As: Perf_MGR
IMVironment
Send

Dir_of_QA: Did you read about the acquisition?
Perf_MGR: yup. just read it.
Dir_of_QA: Can you believe he said we'll be one company by the end of the quarter?
Dir_of_QA: The load on our key systems will be double. MAYBE TRIPLE!!!
Perf_MGR: i know.
Dir_of_QA: How can we be sure our systems won't crash and burn with so little time to test???

no problem. we just bought **embarcadero extreme test**...we're fine.

Extreme Test allows QA teams to use a goals-based approach to create realistic user scenarios that produce more accurate results in a fraction of the time. For more information or to evaluate Embarcadero Extreme Test, please visit **www.embarcadero.com/proftester.asp or contact Chris Birks on +44 1628 684443** (chris.birks@embarcadero.co.uk)

EMBARCADERO TECHNOLOGIES®

Goals-based testing in action – a scenario

Test criteria

- 1,000 concurrent users
- 8-second response time 90% of the time (90th percentile)
- Less than 85% CPU utilisation
- Monitor complete application infrastructure (database, Web server, application server, etc.)

Three phase test

Phase 1: Ramp Up until we exceed eight-second response time or 85% CPU utilisation. If response times reach eight seconds with fewer than 1,000 users, exit the test. If CPU utilisation exceeds 85%, exit the test. If the test exceeds eight seconds with more than 1,000 consurrent users, exit the phase.

Phase 2: Test the system at a constant load. If CPU utilisation exceeds 85%, exit the test. When the phase runs for 15 minutes, exit the phase.

Phase 3: Ramp down until there are no more users.

Results

The engineer ran the test and it exited at Phase 1 because CPU utilisation exceeded 85%. This was determined quickly – the engineer did not have to wait until the test was finished. Moreover, the performance engineer configured database and application server monitoring as one of the test parameters, which helped immediately identify bottlenecks in the database layer and corrected it. The next time, the test ran to completion.

The goals-based test indicated the page delivery time for 1,000 concurrent users, and it indicated the number of concurrent users required to deliver pages inside eight seconds.

By taking a goals-based approach to this scenario, the engineer could accurately answer the question “How many concurrent users can the system support with acceptable performance?” The goals-based approach also minimised the time required to adjust and reconfigure time-based performance tests to answer the same question.

For example, in a time-based performance test, a test schedule is configured. In a goals-based performance test, a load model is configured and thresholds are defined so that instead of applying a time-based transition from increasing users to a constant load, the ramp-up continues until a threshold is exceeded. The result is that either the thresholds are exceeded, the phase ends and the next phase starts, or the whole test terminates because performance is unacceptable and there is no reason to continue the test.

The benefits

Overall, goals-based testing as in the scenario above reduces the total time required to conduct a system evaluation. Moreover, test engineers no longer have to wait around for tests to abort. Most importantly of all, tests can be carried out in a way that reflects how systems are used in the production environment – a first for any kind of performance testing.

criteria. Unlike load-based testing, users can set a range of thresholds and realistic user scenarios that provide the criteria for success (or failure), before the test begins.

Testers can identify key test goals then set criteria accordingly — defining acceptable thresholds for the various parts of the system, such as response times, CPU utilisation and so on. This approach removes the typical try/repeat cycle that is so common with most load-testing processes. It saves test professionals significant amounts of time they would otherwise have spent waiting for a standard test to finish, only to discover they have an invalid result.

More valuable results

Generally, the goal of performance testing is to determine how many concurrent users your system can handle while continuing to deliver a positive user experience. To meet this goal, the test must answer some important questions:

1. What are acceptable user performance levels?
2. How many concurrent users are required?
3. How fast should the site respond?
4. What are acceptable levels of system utilisation? (CPU, memory, network, disk and so on).

Traditional performance testing makes it difficult to answer these questions with any depth of detail. To reach the required degree of testing ‘granularity’, organisations must specify and program these goals into the performance test itself. With traditional tests, engineers ‘guess’ the number of users the system can support. The test is then repeated until the four questions (above) are answered. Unfortunately, it can often take much iteration before traditional performance tests achieve their goals.

In stark contrast, goals-based testing enables the technical team to determine the exact requirements for the system before testing begins, and provides more sophisticated answers to the four critical questions above. Effectively, this means testers can ask questions such as: ‘How many users can my application infrastructure support with X second transaction response times without risk of system overload?’ and ‘How can I feel confident that the results returned are accurate?’

Goals-based tests include a series of phases and exit conditions. Within a phase, if the test exceeds any of the designated thresholds, the pre-specified exit actions are initiated. In addition, users can set thresholds that will simply stop the test if exceeded. The user can configure the test with many different thresholds.

For more information about how goals-based testing works in reality, see *Goals-based testing in action – a scenario* above.

Time to get in the real world

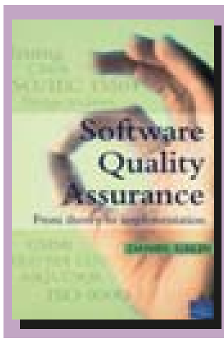
Put simply, goals-based testing is a more powerful approach to performance testing because it focuses on modelling the real world. Practically, every system will suffer some kind of performance deterioration or outage at some time.

Yet, today’s end user has zero tolerance for such outages and slow performance. Explosive growth in technology means increasing system stress, and this leads to potential performance problems across all system components, including the database, web servers, application servers and so on. Neglecting performance problems is costly and laden with risk. Poor performance leads to downtime, outages, and ultimately, a poor end user experience. To avoid this, companies must adopt a broader performance assurance programme in which the performance and stability of critical applications can be constantly evaluated and optimised after they have reached production.

Goals-based testing provides a powerful framework for these broader performance testing and evaluation processes. It enables test engineers to create more sophisticated test environments that reflect the key challenges of today’s highly complex and disparate database, application and system heavy architectures. So much time and money is wasted because of the high volume of repeated tests that result from the inadequacies of established performance-testing methods. With a goals-based approach to testing, organisations can test smarter and more efficiently. They can be pro-active and make allowances for many possible test failures even before testing has begun. Technicians and test engineers can also use the approach to play a fundamental role in overall capacity planning. Finally, with goals-based testing, you can bring your activities into the real world. PT

Test Library

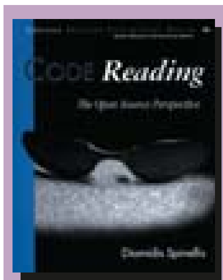
This quarter's new books - and a few which have been around longer - of interest to testers



Software Quality Assurance: From theory to implementation
Daniel Galin
Pearson, ISBN 0-201-70945-7

Extensive and comprehensive; clear writing style, and very dense in terms of amount of information per page. We particularly liked the assessments of standards and the review questions at the end of each chapter.

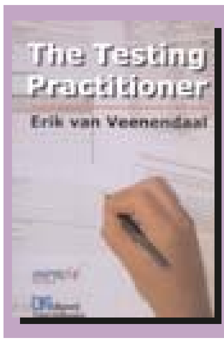
Relevance to testing: very high



Code Reading: The Open Source Perspective
Diomidis Spinellis
Pearson Education, ISBN 0-201-79940-5

Fascinating information on analysing, understanding and inspecting code written by others in C and similar languages popular on Unix/Linux platforms. Includes instructions on use of searching/analysing/unit testing tools.

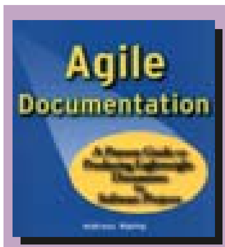
Relevance to testing: high, for technical testers doing code inspection and unit testing



The Testing Practitioner
Erik van Veenendaal (ed)
UTN, ISBN 90-72194-65-9

Seven sections, named after and following the sections of the new ISEB Practitioner Certificate syllabus, with self-contained chapters by 24 different authors; this actually gives the book variety, and makes it easier to 'dip into' individual topics. Not everything in the syllabus is covered, but a good deal is.

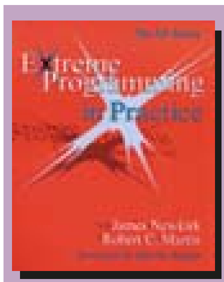
Relevance to testing: very high



Agile Documentation
Andreas Rüping
Wiley, ISBN 0-470-85617-3

Highly prescriptive guidance not only on the content of simplified specification, design and project management documentation, but also style, typography, layout and workflow.

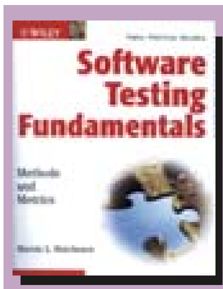
Relevance to testing: medium. Test documents are mentioned but not described. However many would definitely benefit from this treatment



Extreme Programming in Practice
James Newkirk and Robert C Martin
Addison-Wesley, ISBN 0-201-70937-6

There is a whole shelf of books on this controversial subject, but this one will appeal to testers, especially those interested in unit testing, requirements specification, development lifecycles and good communication with project sponsors.

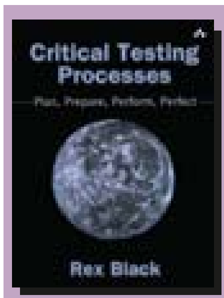
Relevance to testing: high



Software Testing Fundamentals
Marnie L Hutcheson
Wiley, ISBN 0-471-43020-X

Not the textbook the title suggests, but a conversational (but detailed), thought-provoking and refreshing description of the author's own approach to testing. Traditional testers may think this somewhat lightweight; it is clearly influenced by the rapid/agile movement.

Relevance to testing: very high, although not everyone will want to apply this advice



Critical Testing Processes
Rex Black
Addison Wesley, ISBN 0-201-74868-1

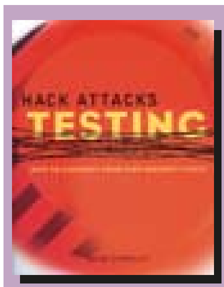
A collection of articles on a wide range of processes and activities to be carried out by test managers, including team recruitment and development, estimation, communication of results and process improvement. Readable and authoritative style with many case studies and fictitious scenarios.

Relevance to testing: very high

Software Design
David Budgen
Pearson, ISBN 0-201-72219

Ambitious, and very densely packed; contains good explanations and examples of the important structured design methods including DSDM, SSA/SD, JSP etc as well a detailed general view of the issues and challenges.

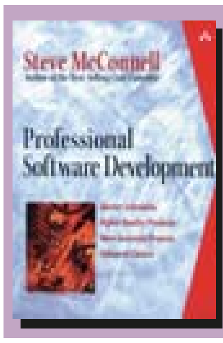
Relevance to testing: medium. Advanced testers wanting to understand and thus influence software design and architecture will benefit



Hack Attacks Testing
John Chirillo
Wiley, ISBN 0-471-22946-6

Technical instructions on how to install and configure tools on a range of platforms to carry out security audits. Focused on server configuration and known system software loopholes and exploits rather than security requirements and generic processes.

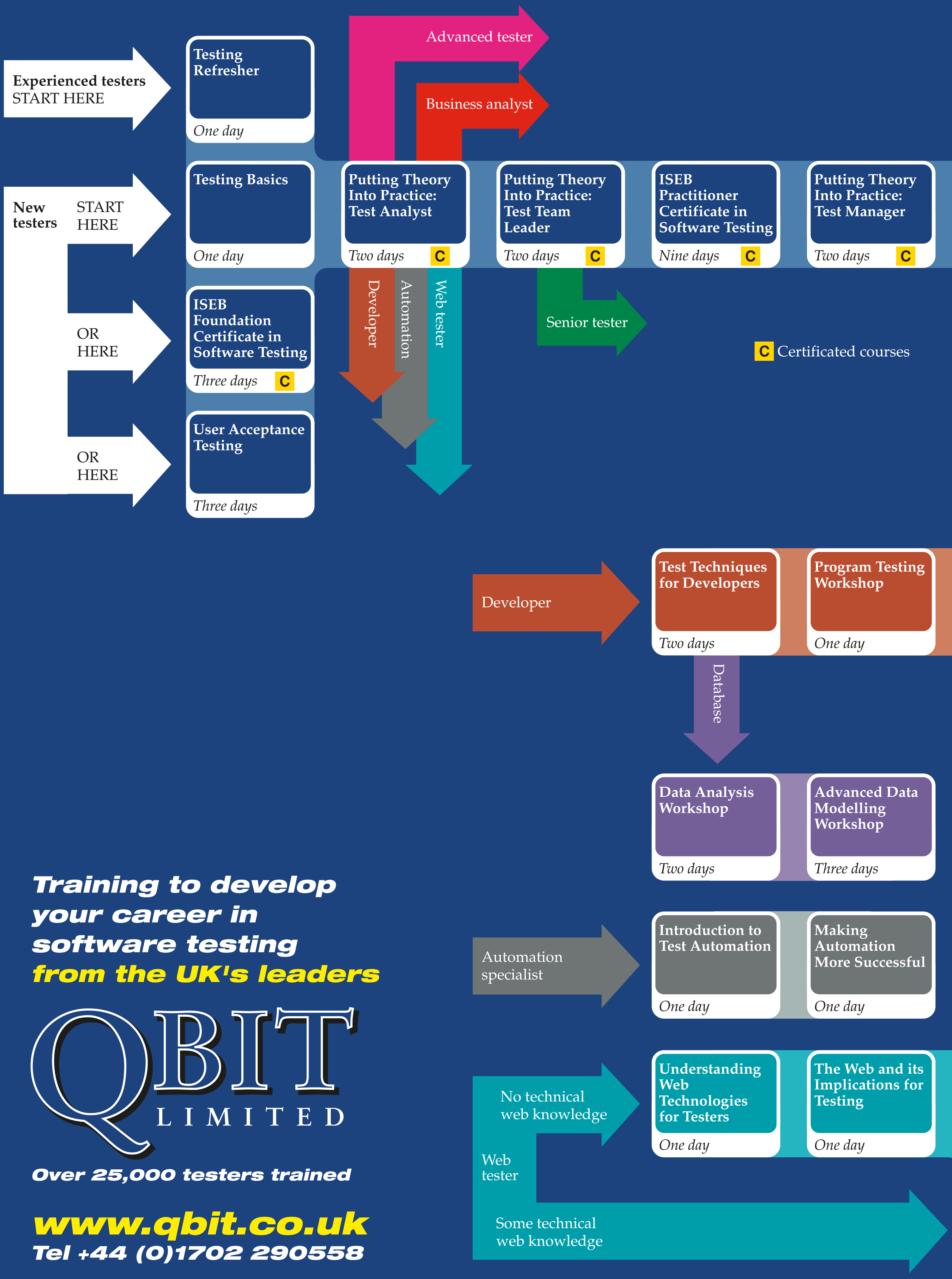
Relevance to testing: low. Auditing servers rather than testing applications



Professional Software Development
Steve McConnell
Pearson Education, ISBN 0-321-19367-9

Steve McConnell is well-qualified to opinionate on software engineering and programming, and he doesn't pull any punches here. Productivity and efficiency are his central concerns; this means avoiding errors rather than finding and fixing them.

Relevance to testing: low, but still of interest; full of ammunition to fire at developers


Experienced testers START HERE
New testers START HERE
OR HERE
OR HERE
Testing Refresher — One day
Testing Basics — One day
ISEB Foundation Certificate in Software Testing — Three days C
User Acceptance Testing — Three days
Advanced tester
Business analyst
Putting Theory Into Practice: Test Analyst — Two days C
Putting Theory Into Practice: Test Team Leader — Two days C
ISEB Practitioner Certificate in Software Testing — Nine days C
Putting Theory Into Practice: Test Manager — Two days C
Developer
Automation
Web tester
Senior tester
C Certificated courses
Developer
Test Techniques for Developers — Two days
Program Testing Workshop — One day
Database
Data Analysis Workshop — Two days
Advanced Data Modelling Workshop — Three days
Automation specialist
Introduction to Test Automation — One day
Making Automation More Successful — One day
Web tester
No technical web knowledge
Understanding Web Technologies for Testers — One day
The Web and its Implications for Testing — One day
Some technical web knowledge

Training to develop your career in software testing from the UK's leaders

QBIT LIMITED

Over 25,000 testers trained

www.qbit.co.uk
Tel +44 (0)1702 290558

Residential Masterclasses
Various

The future...

Senior tester

Test Management
Two days

Making Reviews Work
One day

Understanding Risk and its Application to Testing
One day

Test Process Improvement
One day

Advanced tester

Advanced Testing Techniques
Two days

Improving Communication Skills for Testers
Two days

Planning for Testers
One day

Systematic System Testing
Two days

The Theory and Practice of Reviews
Two days

Designing Tests using BS7925
Two days

Object Oriented Testing
Two days

Testing Software Updates
Two days

Business analyst

Specifying User Requirements
One day

Testing User Procedures
One day

Auditing Systems Under Development
Two days

Testing Web Functionality, Usability and Compatibility
Two days

Testing Web Implementation
One day

Test planning to derive confidence

Independent consultant and trainer in risk and management Felix Redmill believes that test effectiveness and test efficiency should be sought separately

It's possible to write software that never fails. But it's not possible to prove in advance that it will never fail. We cannot prove that software is correct (except when mathematically formal languages are used) or that operation will avoid the faults that exist in it. So, if not proof, what do we seek from testing? We seek confidence. Yet, questioning a small sample of testers suggested that the derivation of confidence was not something that entered the majority of their minds.

What did enter their minds was doing their job of testing, executing the specified tests in the time allowed, finding as many bugs as possible, and frustration at the difficulties put in their way by the software developers. In other words, being efficient. The trouble is that efficiency rarely provides evidence to underpin confidence - except in the testers themselves. We may find many faults, or many faults per unit of testing time, but these general statistics tell nothing of what is gained by doing so. Indeed, if the bulk of the faults were in parts of the software destined to be used only rarely, or for trivial purposes, knowing that we found them could generate a false sense of confidence in the value of our testing and in the quality of the software.

Being effective - that is, finding the faults that matter most - could provide the basis of confidence. But focusing testing at them is the responsibility of the test planners rather than the testers. Moreover, it's not easy to achieve, for it depends on being able to determine what really matters and how to target it with testing - and what matters depends on perspective. Perhaps the testers are right: get on with the job of doing the specified testing and leave the question of whether it is the right testing to the planners.

But in what should the test planners seek confidence? In the likelihood of a given undesirable outcome being below a certain value. In other words, that some defined risk is tolerably low. An outcome of interest might be the occurrence of a certain type of failure, or of failure with certain consequences, within a defined time period. But determining exactly what type of failure, or from whose perspective we should judge the consequences, requires not only high-level planning but also, importantly, detailed consideration. In one system, a safety-critical function may be singled out, while in another it may be a communication hub or the controller of customer interfaces. Further, almost never will one element of a system be so pre-eminent that testing of others can be considered unimportant. There needs to be prioritisation, and this requires the evaluation of the entire system and not simply the identification of a single significant component.

The outcomes in which we want confidence will vary depending on the risks that are judged to be most significant in the circumstances, so the test planners need to consider carefully what are most important. But it appears that it is not usual for them to plan in terms of levels of confidence. And even if they did, it would not be easy, in the present state of the art, for them to know when they could justifiably claim to have achieved the desired level of confidence. A major impediment is that software is a discrete system and not continuous. It is not homogeneous. Correct performance in one part of its structure does not guarantee, or even imply, correct performance in another part. So using test results as the basis of an argument for the 'goodness' of a system requires assumptions and carries dangers.

For continuous systems, the 'less than principle' can offer a high level of confidence. Imagine that you are driving your small car along a narrow road in a remote rural area. You arrive at a stream over which there is a small bridge. Its cut-stone arch is a thing of beauty, but it strikes you more as an antique curiosity than a functional bridge, and you wonder if it can bear the weight of your car. So you pull off the road and ponder your situation. While you are there, a local farmer approaches in his tractor and crosses the bridge. Immediately your doubts are dispelled. You know that if the bridge can carry the heavy tractor, it can carry your little car. The 'less-than principle' applies. It applies because the response of the bridge to loads is a continuous function.

Similarly, justifiable assumptions about goodness of the whole may be based on tests of a part. In the construction industry, confidence in a batch of concrete is derived from tests on samples. Also for continuous systems, knowledge of intermediate points can be determined by interpolation between measured values. In a school physics experiment we monitor a spring's extensions in response to increases in the load that is hung on it, and we thus confirm what Hooke's Law predicts - that the extension is proportional to the load (up to the elastic limit). Once measurements are taken at a number of points and a graph of extension versus load has been plotted, interpolation identifies the exact extension that would result from a given load (or the precise load that would be required to produce a given extension). Up to their elastic limits, the behaviours of the concrete and the spring are consistent and predictable.

But his is not the case for discrete functions, and software-based systems, being digital, are discrete. Their behaviour is not easily predictable. A small difference in input can lead to a huge difference in response, for the system's behaviour at a point cannot reliably be determined from knowledge of its behaviour at an adjacent point. Each input results in a discrete output. The less-than principle does not apply. Interpolation between, or extrapolation from, experimentally determined points would necessarily involve assumptions. Such assumptions may be valid, but to have confidence in their validity we need to recognise them and understand them, for they might also be dangerously wrong.

How and in what can we derive confidence from testing? We know that the number of combinations of the logical paths through an item of software, and the values of its variables, can be astronomically large, rendering exhaustive testing non-cost-effective if not impossible. So we are forced to test at points and to make assumptions about the goodness of the software at other points. Are we aware of our assumptions? It is one thing if we recognise both the assumptions and the risks attached to making them. Then we could begin to assess the level of confidence that is justifiable in the circumstances. But it is quite another thing if our assumptions are unrecognised. Then, any belief in a level of confidence is likely to introduce new risks.

The problems are often avoided by not giving thought to the question of confidence, but rather, simply planning a wide-ranging test programme. Yet, while testers might beneficially pursue efficiency, the task of test planners is to make testing effective, and addressing the issue of confidence would facilitate this. The importance of thinking in terms of confidence increases in proportion to the risks attached to the system.

Questions such as the following could usefully be asked:

- In what is it important to gain confidence?
- What levels of confidence do we require?
- What do we need to do in order to increase confidence?
- How might our requirements for confidence inform testing?
- What criteria need to be satisfied in order that we might claim to have achieved the required levels of confidence?
- What assumptions will our claims imply, and will they be valid?
- What risks might our assumptions create, and how might we reduce them?

Such questions are always difficult to answer. But posing them helps to define the difficulties, and trying to answer them improves our understanding of the problems. If we don't pose the questions and don't understand the difficulties, we are likely to focus testing less effectively than we should do, while, in ignorance, believing that we have done the best job possible. If test planners understand how to seek effectiveness - and do it - and the testers execute the plans efficiently, our testing is likely to serve us well. PT

Safety-Critical Systems Club

Raising awareness of safety issues and good development and testing practice

regular events • newsletter three times per year • co-ordinator: Felix Redmill

Annual Safety-Critical Systems Symposium

with 1-day tutorial on Software Metrics
by **Professor Norman Fenton**

17-19 February 2004

To join, or for information on the Club or the Symposium, contact
Mrs Joan Atkinson, tel 0191 221 2222, email csr@newcastle.ac.uk

SCSC

21st century schizoid plans

Technical consultant Gordon Alexander of Compuware believes resistance to documentation and procedural standards is rooted in lack of sufficient training, resources and time to implement them properly

I talk to a lot of people about testing. When you talk testing you talk documents; lots of them. There are different thoughts on handling this.

Sometimes I'm told that things aren't being done correctly, a feeling that procedures and documentation are a bit flimsy, not as formal or organised as they should be. These are the guilty, furtive testers muddling through in fear that they are about to be found out.

I hear the contrary view too. Heartfelt complaints about documentation and procedural standards foisted upon an already overworked organisation. Restrictive and pedantic without adding anything of value to the development process. These are the wild free spirits held back by the chains of bureaucracy.

Of course, both these views are perfectly reasonable but the bizarre thing about this is that very often *it is the same people who hold both these views simultaneously*. Analysing why these multiple personalities exist within the testing practitioner community is actually quite illuminating.

What exactly is the testing problem?

Without documentation and standards you discover a whole set of issues:

- Everyone has to re-invent the wheel;
- People may or may not know what to do or how to do it;
- You duplicate work that other people have done;
- You repeat work that you have done;
- You don't know what you have done. Eg, you can't tell people what you have and haven't tested. "Is the warp drive going to work up to warp factor 10?" – "Erm, don't know, let me ask Scotty if he checked that."

Unless you can consume a lot of their time it is difficult to capture input from other groups to assist and inspire your testing.

The purpose of CMMI, SPICE, TMM, TPI, Six Sigma and all the rest of the many and varied methods that exist today, is to deal with these issues (amongst other things). However, introducing documentation and standards can initiate a whole set of new problems:

- You may not know where to start to create the documents and standards from scratch;
- You don't have time to set it up during this release "We'll do it next release when the fires are out";
- It would take more time than the whole project to design and develop a decent set of procedures;
- Using standard methods you find that many of the steps in the procedures and much of the documentation does not add anything of value to the project;
- It takes an incredibly long time to produce all the documentation even though it is of some value;
- You spend all your time doing documentation instead of testing.

So, what is the answer?

What is the best way of organising your testing? I've seen a lot of good testing being carried out by simply using common sense and a high degree of individual skill. This can work well, but it's not optimal.

Sometimes I've seen people attempt to move towards a more formal approach but have encountered difficulties in doing that. Implementation often founders because efforts are made to implant an entire process onto overworked people who aren't adequately trained or resourced to implement it. This is why you can sometimes find our testers simultaneously expressing seemingly contradictory views.

In my opinion, the best way to test is to implement a tailored and resourced standard method using well-trained staff given time to get through the learning curve.

So what do we need? Documents.

Let's have a quick look at what sort of documents we are talking about. Broadly speaking, there are three main categories of documentation.

1 Documents to manage the process of testing: *test plans*

2 Documents that describe the tests you wish to run: *test assets*

3 Documents to record testing results and faults: *test logs*

Test plans

Here I mean things like test mission, test policy, test strategy, test plans, test schedule etc. A fundamental tenet of testing is that possible activities can consume an infinite amount of resources. You use documentation to manage this effectively. You also use your test planning documentation to write down what you mean by testing, thereby limiting the scope, and to say how and when you are going to do it. The reason you write it down is so that you can get it clear in your own head and get everyone else in the organisation to agree it. This is particularly important when looking at risk.

It is interesting to examine all the different documents that exist in this category. Sometimes it seems that they are duplication. They all purport to explain what, when and how you are going to do your testing.

Most times you need to design a hierarchy of documentation, which is why you need more than one type of document in your 'test plan' set.

You have two different kinds of information that you need to write down.

Firstly, there is information about testing which is applicable to your whole organisation. This is general information. This includes things like the purpose and definition of testing, the methods that are available for use, the measures that will be made and the tools that will be used.

Are your applications leaking money?

64%

of IT Executives say Yes...

*having experienced material revenue loss as a direct result of application failure**

Compuware's ASQ solution can prevent application failure by building quality in from the start. To learn more about how we can help save you time and money simply visit

www.compuware.co.uk/products/qa and request your **FREE Patricia Seybold Group White Paper - 'Your Applications are Leaking Money'.**

Alternatively you can call us on 01753 444 444

In addition you can view our latest webinar **'Quality First, Strengthening the Quality Process for Successful Application Deployment'** by visiting **www.compuware.co.uk/webinars/asq2**

COMPUWARE®

www.compuware.co.uk

*Forrester Research – June 2003

These can be very different for different organisations. So for one company the goal of testing will be to find bugs, for another to improve quality. The communication of this information across your organisation can have a profound effect on the testing activities, for example the goal of "proving the software works" is vastly different from "reduce the number of failures that occur in released software".

The other type of information that these documents contain is project-specific information such as who will do the testing, when it will be finished etc. It can also contain information about variations from the standard rules of the organisation. You should keep clear in your mind these two different purposes when writing your documents.

So how do you decide what documents to have in your hierarchy and what to put in them? For the general (non-project-specific) information each higher level document contains a summary and more a strategic view of the information. So for example, a test strategy document may say "we will proactively investigate the use of test tools provided there is a demonstrable return on investment in six months," whereas a test plan may describe specific tools to be used.

The decisions you have to make are whether to have a particular document in your hierarchy set and what information to put there. Use the standards to get a view of everything that anyone could possibly want in documentation. Then the key is to be really honest about the audience and the purpose of the document.

So for example, if you want to provide guidelines to your team of four testers then you don't need a test strategy, you need something more specific, "use state transition testing, equivalence partitioning in your testing" etc. If, ultimately, you can't influence other testers in your organisation, then there is little point in spending your time simplifying, generalising and explaining what you do in a document that no-one except you and your team will read.

If however, you want to ensure a certain level of quality across all key technologies in your company, then get your best people to write a strategy in terms that you, the board and all stakeholders can understand. This strategy should be both a condensation of your current best practices and your best vision of the way forward.

You can give access to all levels via a simple but high-level interface which details the test status – with red lights representing significant issue areas, yellow lights representing areas of caution and green being fine.

Similarly, if you look at project information each level of the hierarchy contains more detailed information about the project as it is developed and more details are known or required.

When defining your test planning documentation keep these principles in mind and you won't go far wrong.

Test assets

Generally tests;

- consist of actions and expected results
- are grouped together
- are derived from either source documentation (black) or source code (white).

The documents you need are things like test conditions, test scripts, links to source documentation. You use this documentation to capture the design effort so you don't have to repeat it.

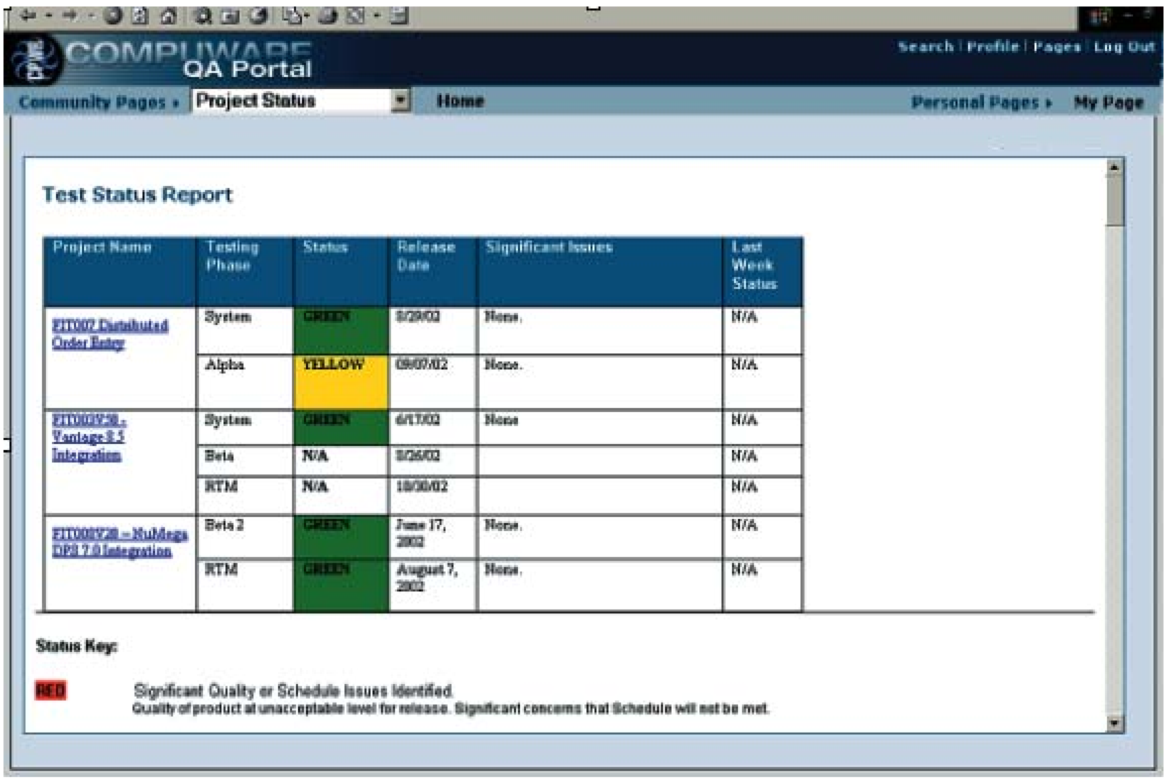
A key element of this process is to design an infrastructure that can provide traceability, impact analysis and crucially allow all parts of the business to participate in meaningful risk assessment.

Test logs

- record what happened
- capture fault information to communicate to others
- give you information on risk; what has and hasn't been tested/passed.

There are various standards you can use to get advice on documentation, for example, IEEE 829 and BS 7925-2. *[See elsewhere in this issue - Ed]*

Ultimately however, the most important thing is to keep focused on the benefits; ie the reasons why you are doing all this. PT



Test Status Report

Project Name	Testing Phase	Status	Release Date	Significant Issues	Last Week Status
FIT007 Distributed Order Entry	System	GREEN	8/29/02	None.	N/A
	Alpha	YELLOW	09/07/02	None.	N/A
FIT003V30 - Vantage 8.5 Integration	System	GREEN	6/17/02	None	N/A
	Beta	N/A	8/26/02		N/A
	RTM	N/A	10/30/02		N/A
FIT008V20 – NuMega DPS 7.0 Integration	Beta 2	GREEN	June 17, 2002	None.	N/A
	RTM	GREEN	August 7, 2002	None.	N/A

Status Key:

RED Significant Quality or Schedule Issues Identified.
Quality of product at unacceptable level for release. Significant concerns that Schedule will not be met.

Figure 1. Example high level test status report

Nobody ever writes to me

Actually a couple of people have, this time, but we want more. We know you're busy but instead of moaning to whoever will listen to you in the canteen, why not point your email client at editor@professionaltester.com and air your grievances to a wider audience? Cheerful letters are allowed as well

Take testing to the dark side

The essential problem with a journal such as Professional Tester is that you're preaching to the converted. I think the readers need to be encouraged to circulate it amongst fellow project members – particularly the project managers.

There always seems to be mild amusement amongst any developers who have seen the magazine on my desk. "Is there such a thing? "they ask. "You bet yer" I reply and invite them to read some of the articles. Some of them come back later in a more serious mood.

Two articles in the July issue, *Don't let testing become part of the furniture* and *Tests without specs* rang many familiar bells for me.

Adam Ripley is so right in his attempt to not let testing become part of the furniture. However, in many projects it already has – its the small broken down old chair in the spare room that is brought out for the newly arrived tester to sit on long after all the other project members have arrived, been introduced and are busy in conversation. Of course, when this happens, its too late if the requirements spec is poor, or worse, has not been provided at all. The testers have to fly by the seat of the pants, always behind the curve. This invariably ends up with incomplete testing and we then enter the scenario 'Tests without specs' as George Wilkinson describes in his article. I would avoid this situation at all costs.

From my experience, a good User Requirements Specification (URS) along with other documentation (detailed design documents, design review minutes, unit test logs) are absolutely essential to the success of a software development project. I would refuse to test if there was no URS of some sort available. I would also be very wary about testing any software where the developer had no unit test log. And, by the way, if there is no URS what are development using to design and build the software?

Adam Ripley asks "Why do we fail?" Of the many software development projects that I have been involved with, none could have been called total failures. There have been some which have been late, by some years in one case, and others in which the user (customer) was obviously disappointed at what has been delivered. This resulted in a breakdown of trust between the customer and his provider. Dark mutterings about court cases arose! Unsuccessful projects only seemed to bring in the test team at a late stage. The successful ones? – well, there is no doubt in my mind the test manager, or a senior tester must assume their role at the earliest opportunity in any project, large or small. To do this they must be part of the project team from day one. Moreover, the testers should be almost the last to leave the project after its delivery to the customer. This is because there's much more in the good tester's remit than just testing the software. In successful projects the testers are also technical authors, quality auditors, diplomats, trainers, configuration managers. They can and frequently do become the bridge between the customer and the contractor. They can interpret the URS for the development team and become the in house 'experts' on how the new software should work, so they then end up training the customer's personnel when the time comes. But – only if they start early enough.

Test differently? No, I don't believe we need to test differently per se. Yes, we do need to take a more continuous approach to accepting a system. This means the involvement of the test team from the earliest point (see previous paragraph) and, most importantly, means testing the documentation before we even get near the software - as Adam Ripley suggests. But this is not new! Many of the books written about software testing that have been around for years always emphasise the importance of starting early and of testing the documentation as well as the software. As an example see *Software Testing in the Real World* by Edward Kit.

Some of the companies I've been contracted to accepted the need for getting the test team in early. This invariably led to success. Others have tried to reduce costs to the bone and have not considered testing until it is (almost) too late. It appears to be a problem of educating the higher management, the sales team, the project managers, everyone but the testers – we know this already!

–Bob Corfield

Remember when *you* were innocent?

As a newcomer to software testing I have read some amazingly ridiculous things such as "The requirement is for 80% statement coverage." and I need guidance. So for example what 20% of the statements do not need to be tested? Can I throw away this 20% and get a 20% discount on the cost of the application?

Another example of an amazing statement I came across in a testing book was "a package could make do with a medium level software quality". This I found utterly confusing and many questions follow such as was this a user request for medium level quality and what is medium level and how much extra does high level cost? What are the cost savings to be made on issuing medium level quality as opposed to high level quality and how much cheaper would it be if the level of quality was low? What are the cost increases in user frustration so what is the return on investment (or lack of)?

Surely the requirement is for the job to be done properly? I think I now understand why testing is a difficult concept to sell.

–name and address supplied

Dig the SPACEDIRT

Peter Morgan, senior practitioner with e-testing Consultancy, on IEEE 829: arguably still the most used testing standard

*"**Why standards?** The use of standards simplifies communication, promotes consistency and uniformity, and eliminates the need to invent yet another (often different and even incompatible) solution to the same problem. Standards, whether 'official' or merely agreed upon, are especially important when we're talking to customers and suppliers, but it's easy to underestimate their importance when dealing with different departments and disciplines within our own organisation. They also provide vital continuity so that we are not forever reinventing the wheel. They are a way of preserving proven practices above and beyond the inevitable staff changes within organisations."*
– Ed Kit, *Software Testing in the Real World*

That paragraph neatly and (quite) succinctly describes why standards exist. But how does that affect testing practitioners who live, as in the title of Ed Kit's book, in the real world? Anything that promotes better project communication has to be good for testers. So standards have to be effective, and produce recognisable (and measurable?) gains, while not adding disproportionate overheads. I once worked for a large organisation that had an internal (and mandatory) standard for almost all documents. This was such that a document of 200 real words was turned into 18 pages, when all the necessary parts ('glossary', 'associated documents') were added. Perhaps this was counterproductive and unnecessary.

IEEE 829 in overview

There have been diverse document types used in software testing, developed in many cases for the needs of a particular organisation. IEEE 829 (1983) is the *Standard for Software Test Documentation*, and this was an attempt to pull sources together and present some best practice ideas. The standard was revisited and revised in 1998. Note that the standard applies to any level of testing that may take place (including acceptance testing), although application in agile development methodologies may be less obvious. It is common to have 'a full set' of IEEE 829 documents for each testing stage that is being undertaken.

IEEE 829 is often thought of as being the standard for a High Level Test Plan or Master Test Plan (HLTP or MTP). It is more than this, as the standard describes eight documents that can be produced as part of the testing effort. These documents are sometimes distributed between different categories, although there is no consensus on the subdivisions. I find the following partitioning helpful:

- Test planning
- Test plan
- Test specification
  - Test design specification
  - Test case specification
  - Test procedure specification
- Test reporting
  - Test item transmittal report
  - Test log
  - Test incident summary
  - Test summary

The eight parts

Most of these eight document types are well known; I will provide a very brief summary, before returning to the test plan.

Test plan: a high level view of *how* testing will proceed; *what* is to be tested, by *whom*, how, in what *time* frame and to what *quality* level.

Test design specification: details the test conditions to be exercised, with the expected outcome (in general terms).

Test case specification: Specific data requirements to run tests, based upon the test conditions identified.

Test procedure specification: Describes how the tester will physically run the test, including set up procedures. The standard defines ten procedure steps that may be applied when running a test.

Test item transmittal report: records when individual items to be tested have been passed from one stage of testing to another. This includes where to find such items, what is new about them, and is in effect a warranty of 'fit for test'.

Test log: details of what tests were run, by whom, and whether individual tests passed or failed.

Test incident summary: details of instances where a test 'failed' for a specific reason.

Test summary: brings together all pertinent information about the testing, including the number of incidents raised and outstanding, and crucially an assessment about the quality of the system. Also recorded, for use in future project planning, are details of what was done, and how long it took.

This document is important in deciding whether the quality of the system is good enough to allow it to proceed to another stage. This assessment is based upon detailed information that was documented in the test plan.

Test planning revisited

Test planning is a key activity in any software testing project, and for that reason many people associate IEEE 829 *only* with test planning. The standard defines 16 items that should be considered for an MTP. This includes the key activities of estimation (as 'schedule' is one of the 16) and risk, both of which are large topics in their own right.

The 16 are given below with a well-known mnemonic for remembering the list; much more detail on each can be found in textbooks on the subject.

S	**Scope** *test items, what to test, what not to test*
P	**People** *training, responsibilities, schedule*
A	**Approach** *the approach that will be taken to the testing*
C	**Criteria** *entry/exit criteria, suspension/resumption criteria*
E	**Environment** *test environment needs*
D	**Deliverables** *what is being delivered as part of the test process*
I	**Incidentals** *introduction, identification (of the document), approval authorities*
R	**Risks** *risks and contingencies*
T	**Tasks** *the test tasks that are involved in the testing process*

It is worth noting at this point that the standard lists as 'deliverables' the seven other document types that perform part of the standard. Some organisations add to this basic list, including key items such as 'glossary', and 'references to other documents'. I usually keep MTP documents from previous projects, and from projects I worked on for previous organisations, so that I can look back and see the specific details that were included.

MTP is a *living* document

The document specifies what is going to be done, and how it is going to be done. It needs to be published, to appropriate people, to make others aware of what is going to be tested, and what is not going to be tested. However, don't wait for everything to be completed before the document is circulated for comment and/or review. The MTP will change during the life of the project. This does not mean that it is not necessary to get individual and departmental sign-off; sign-off is achieved on the basis of what is known at a point in time. In one organisation I know of, sign-off is achieved by stating that unless this is received by a specified (and realistic) date, it will be assumed. It is remarkable how that concentrates the minds of those concerned.

Two areas that indicate the dynamic nature of the MTP concern schedules, and risks. During the testing phase, good news and bad news can occur, and this can change priorities. Does that mean that the original MTP was wrong? No; the MTP is what its name suggests, just a plan. At the time, it was based on the best available information, incomplete though this was. Information will be improved as testing progresses; for example what was at one time a critical risk may have now been addressed (eg by third-party security testing). The risk is now answered, and will possibly require no further action.

Review the document

The MTP needs to be reviewed, with review taking place face-to-face. If it is contentious, points of conflict need to be talked through. The MTP is not just "owned" by the testing team(s), but development groups and users can contribute significantly to clarification and suggest the addition of new items. What is to be tested and is not to be tested are two key elements in the MTP. In October 2002 I worked on a project where testing (as always) was pushed for time. The MTP specified that significant testing would concentrate on the retail system, with respect to 53-week year processing (2002–2003 is a 53-week year). The development team had not realised the significance of 53-week years (that it was *this* year), and *merely the insertion of the testing intention* resulted in better code (development extended unit test coverage, found some problems and implemented fixes).

It is usual for the detail listed in the MTP to be used as a basis for deciding whether the software under test is suitable for the next stage of testing, deployment to production, etc. Therefore key individuals need to see this detail, and agree, before the crunch implementation meeting!

Face reality

The MTP is one place where testing faces reality. The MTP is not free-standing, but fits into the overall test strategy. In some ways, it is not a prescriptive approach, but more of a check list, to remind those responsible what should be *considered* to go into the MTP. The only prescriptive thing about it is the use of the 16 point "checklist". It is perfectly OK to exclude one of the 16 points – as long as the reasons why that has been excluded are listed (and agreed by the reviewers of the MTP). Risks and assumptions are also included in the MTP; sometimes the explicit stating of a risk or assumption can promote lively discussion, and even resolution!

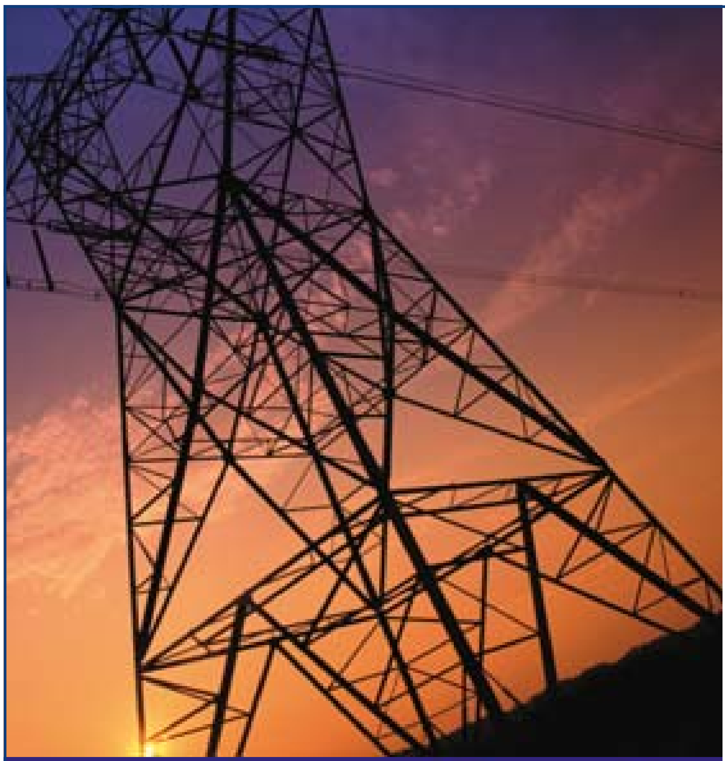
Relationship to other standards

IEEE 829 as a standard is not so much about how to test, but how to document that you have tested. These are some of the other standards that may be referred to when documenting according to IEEE 829:

- IEEE 1008 Standard for Unit testing,
- IEEE 1028 Standard for Software Reviews
- IEEE 1044 Standard Classification for Software Anomalies
- IEEE 1044-1 Guide to Classification for Software Anomalies
- BS 7925 Standard for Software Component Testing

Conclusion

IEEE 829 should be used as a standard appropriately, not blindly. In themselves, testers add nothing to the output of the project team; a tester does not make better software. Therefore, we need to slay the "documentation for documentation's sake" myth and ask ourselves "is this output enabling the test and/or development teams to do a better job, or helping the users understand what is being developed and whether it meets their needs?" IEEE 829 can help to make this the case by giving useful guidelines; it points the way to truly useful documentation. PT



e-testing®
consultancy

iseb

Special ISEB Training Offer!

3 days Foundation Course – only £725*
10 days Practitioner Course - £2000*
delivered in 3x3x2x2 sessions

***Course price includes course notes, exam fees & lunches.**

To book a place on any of the above courses or for further information,

Please email training@etesting.com or contact Caroline Barlow on 020-8387 1701

e-testing only operate in the Testing sector of IT and are dedicated to providing testing services to its clients. We are, therefore, always looking for testing resources to join our firm on permanent or contract basis to work on new projects. So to register your interest with us please complete our on-line form on www.etesing.com or send your current CV to jobs@etesting.com

e-testing Consultancy Limited
Kinetic Centre, Borehamwood,
Hertfordshire WD6 4PJ

Best practices: what testers need to know

Sources of useful, established methodologies and advice are not limited to standards organisations. Dr Adam Kolawa of Parasoft introduces some you might not have seen

Testers can benefit from understanding both development and testing best practices.

Development best practices

When developers follow these, testers can spend more time focusing on high-level testing tasks and less time chasing after problems that developers should have prevented or fixed.

1. Defensive programming

Defensive programming is the practice of anticipating where failures can occur and then creating an infrastructure that tests for errors, reports when anticipated failures occur, and performs specified damage-control actions -- such as stopping program execution, redirecting users to a backup server, enabling debugging information that can be used to diagnose the problem, and so on. These defensive programming infrastructures are typically built by adding assertions to the code, implementing Design by Contract, developing software defensive firewalls, or simply adding code that validates user inputs. By applying defensive programming techniques, developers can detect problems that might otherwise go unnoticed, prevent minor problems from growing into disasters, and save themselves a lot of debugging and maintenance time in the long run.

Hint: If the developers are performing defensive programming, they will be able to compile the code in two ways - with or without the defensive programming infrastructure.

2. Code review

A code review is the process where the developers and architects meet and discuss code. Its purpose is to exchange ideas about how code is written, and to establish a consistent interpretation of code throughout the group. During these reviews, developers should be given the opportunity to explain their code to one another. Often, simply explaining the code helps developers identify problems and envision new solutions for previously troubling dilemmas. When the group members discuss the code, their discussion should focus on important issues such as algorithms, object-oriented programming, and class design.

Hint: If developers are not complaining about code reviews, they probably are not performing them.

3. Coding standard compliance

Coding standards are language-specific programming rules that greatly reduce the probability of introducing errors into applications. Coding standards originated from the intensive study of industry experts who analyzed how bugs were generated when code was written and correlated these bugs to specific coding practices; they took these correlations between bugs and coding practices and came up with a set of rules that prevent coding errors from occurring. In a team environment or group collaboration, coding standards ensure uniform coding practices, reducing oversight errors and the time spent in code reviews. When work is outsourced to a third-party contractor, having a set of coding standards in place ensures that the code produced by the contractor meets all quality guidelines mandated by the client company.

Hint: If there is no system set up to scan code on a regular basis, the developers probably are not following coding standards.

Development best practices reading list

- Hunt, Andrew and David Thomas, *The Pragmatic Programmer*, 1999 (defensive programming, code review)
- Eldridge, Geoff. Java and Design by Contract (defensive programming). www.elj.com/eiffel/feature/dbc/java/ge/
- Kolawa, Adam, Wendell Hicken, and Cynthia Dunlop, Bulletproofing Web Applications. 2001 (defensive programming, coding standards)
- Maguire, Steve, *Writing Solid Code*, 1993 (defensive programming)
- McConnell, Steve. *Code Complete*, 1993 (defensive programming, code review, coding standards)
- Meyer, Bertrand. *Object-Oriented Software Construction*, 2000 (defensive programming)
- Payne, Jeffrey E., Michael A. Schatz, and Matthew N. Schmid. *Implementing Assertions for Java.* Dr. Dobb's Journal January 1998 (defensive programming)
- Plessel, Todd. *Design by Contract: A Missing Link in the Quest for Quality Software:* www.elj.com/eiffel/dbc/ (defensive programming)
- Meyers, Glenford J., *The Art of Software Testing,* 1979 (code review)
- Kernighan, Brian and P.J. Plauger. *The Elements of Programming Style,* 1988 (coding standards)

Testing best practices

These are designed to help verify that the product really works and is solid.

1. Understand the product architecture before you start testing the product

If you do not understand the architecture and inner workings of the product you are testing, you will not be able to anticipate where it is most error prone. As a result, you could overlook easy opportunities to uncover a large amount of errors in a small amount of time.

As testers gain experience, they learn that there are some parts of programs that are more error-prone than other. Generally, the most error-prone parts of a program are the interfaces between different modules. Why? Because different groups of developers work on different modules. These groups often misunderstand one another's intentions and assumptions, and errors typically result when their code interacts. Consequently, a lot of bugs are usually hidden in program interfaces.

Another trick is to learn which development groups worked on which program segments, and use those development groups' track records to anticipate which parts of the program are most error-prone. For example, if you know that Group C worked on a certain part of the program and that Group C usually produces code with a lot of errors, you might want to focus a large percentage of your testing efforts on the part of the program created by Group C. Likewise, if you know that Group A almost always delivers clean, functional code, it is probably safe to spend a smaller percentage of your time testing the parts of the program that Group A worked on.

Just like a police detective needs to understand the entire situation before he can solve a murder mystery, you need to understand the entire situation to solve the mystery of "does this software really work"?

2. Anticipate potential misuses and verify how the software responds in those cases

Don't think that your job is done once you have verified that the software does what it is supposed to do. Users inevitably try to use software in unexpected ways—sometimes because they misunderstand how to use the product, sometimes because they see additional usages for the product, and sometimes because they want to launch security attacks through the product.

The specification is the best starting point for testing unexpected usages. If you don't have a specification, find one or write one yourself if needed. For each feature in the specification, try to imagine what unexpected paths could be taken by a new user exploring the program, an experienced user trying to maximize the program, and a hacker trying to manipulate the program. For example, what happens if the user tries to apply a tool to an unexpected type of source? If the user does not provide critical information? If the user designs and sends unexpected inputs in an attempt to gain access to privileged data or to gain control of a program?

The appropriate response to these unexpected situations depends on the program and the situation. In all cases, the response should be intelligent. For example, if the user does not provide critical information, it is better to have the program display a helpful dialog explaining the problem than simply to fail to perform the requested action.

3. Record clearly the procedure to reproduce each error found

For each problem that you detect, be sure to record detailed, unambiguous instructions for reproducing that problem, as well as a detailed description of the environment and context in which the problem occurred. If your bug report documentation is incomplete or confusing, two problems could occur.

One problem is that developers might not be able to reproduce the error and thus probably will not be able to fix the error. If the developer does manage to reproduce the error without proper instructions, he or she will have probably wasted a lot of time in the process.

Another problem is that you will not be able to verify effectively whether the developer's modification corrected the problem. If you don't test the repair using the exact same environment and procedures that produced the error in the first place, a passed test will not necessarily mean that the error was corrected.

4. Help the team prevent errors

Typically, when testers find errors, they add a report to the bug tracking system, the responsible developer tries to reproduce and repair the problem, then the tester must verify that the modification corrected the reported problem and did not introduce any new problems. This approach is not only time-consuming and costly, but also inefficient because it doesn't help prevent the same types of errors from recurring. Moreover, it causes the team to waste a significant amount of time, effort, and resources on the same types of errors thousands of times over.

Various methods have been proposed to help ensure that once an error is discovered it, and often other, similar errors, are prevented from recurring by an improvement to the development and/or testing process. The key to these is that developers or testers should find each type of error only once. The knowledge the team gains from finding and analyzing errors should be used to improve the process so that you never encounter repeat occurrences of errors similar to those you have already found.

This process benefits the entire team by improving quality and reducing costs, but it is especially beneficial to testers because if developers are performing the required error prevention practices, testers won't need to repeatedly chase after errors that developers could have easily found or prevented and will have more time to dedicate to higher level verification tasks.

You can further error prevention not only by finding bugs, but also by helping the team architect, manager, and developers pinpoint the source of each error you uncover and by suggesting ways to prevent recurrences of that error.

For example, suppose that some of your load tests reveal that a heavy load stops the system. One course of action is to report the problem in the bug tracking system and hope that someone else figures out why it was caused and how to prevent it. A better course of action is to work with the development team to pinpoint the source of the error and reach a group consensus on how to prevent the error from recurring.

After the problem and resolution are identified, the architect and manager should determine how to implement and enforce the error prevention measure; this should not be responsibility of the tester.

However, if your testing indicates that a required error prevention measure is not being followed correctly, it's important to notify the architect or manager so that he or she can address the problem. PT

The case for a central support group

The takeover of NatWest by the Royal Bank of Scotland led to perhaps the biggest and best-known testing project of the decade so far. Alan Bowers of RBS Group Technology Migration Testing Support explains how the process was run smoothly and completed early

One of the key benefits of integration is to be able to realise the economies of scale. This is certainly true of business activities but is more eminently evident when combining IT systems. The cost of the integration of differing application systems is very quickly outweighed by the savings that can be realised by moving to a common IT platform.

Of course, this IT integration activity is not a simple task. It involves vast amounts of development work to application systems to provide the target and creation of software to extract customer data and convert it to the format required. All of this requires testing and this is where I became involved with the project.

The scale and variety of the testing required was so huge that a separate team was set up to undertake the testing activities that were needed. Application development teams would perform the usual unit and link testing activities themselves and then hand over their systems to the testing programme. There then followed a series of test streams that were designed to test the components from a number of differing viewpoints:

- system tests to test the functionality of the components
- tests to prove that the systems would meet performance requirements
- tests focusing on the business processes that must operate correctly following the integration activity
- tests to prove the migration process.

What was happening

This multitude of testing teams meant that we were faced with a number of challenges:

- review and approval of documentation was performed in different ways
- testing documents were created with a variety of templates/formats
- test teams recognised the need to be compliant with policies and standards but needed clarification on detail
- test streams had multiple dependencies with one another as well as with the development teams
- it was important to ensure that each stage of testing did not start until all the pre-requisites were in place
- There was a risk that some functions could be duplicated across testing teams.

What we did about it

To overcome the challenges outlined above it was decided to centralise a number of functions in a single, dedicated team called Migration Testing Support. MTS provides an end-to-end service to the test teams

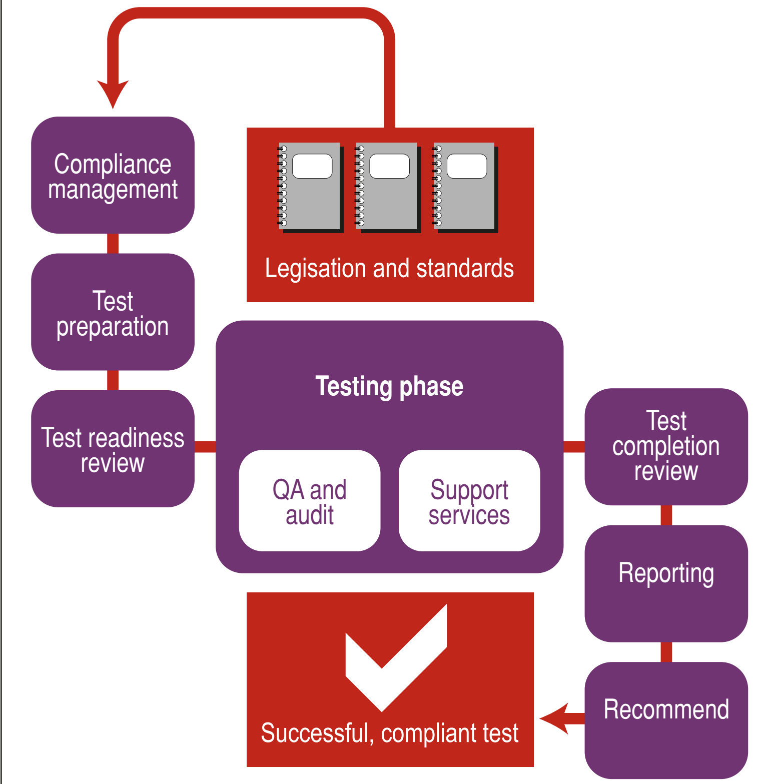


Figure 1. Scope of the Migration Testing Support group

Figure 1 shows the scope of MTS. The process flow consists of the following activities:

- legislation and standards are passed down from both internal and external bodies, such as the internal auditors, group risk and security functions
- TMS then takes this information and defines what is needed to make the test compliant. Guidance is then given to the testing teams so that they can properly prepare for a compliant test. Then a review is carried out to ensure that the test is ready to start
- the testing itself is then carried out. The test team is supported by a number of services provided by the central group – fault management, dependency management, document reviews and test tool support
- audits are carried out on a regular basis to determine compliance with agreed standards
- at the end of testing a review is carried out to check that the test has been completely executed and any relevant reports are created. These reports are used to improve test planning, management information or as inputs into other test readiness and completion reviews and compliance assessments.

The services in detail: compliance management

All testing must comply with applicable legal and external regulatory policies and with additional company policies and standards. Compliance also has a wider definition, however, covering the adherence to other relevant rules and regulations, whether derived from statute or from voluntary codes of industry best practice, covering any aspect of the businesses carried out by the organisation.

The role of the MTS in this is to:

- determine what these policies and standards are
- ensure that the latest versions of policies and standards are available to the testing community through use of the agreed cascade process
- develop policies and guidelines where these do not already exist.
- act as an interface to/from regulatory bodies
- act as a central point of reference in respect of all compliance matters emanating from the testing teams.

Test preparation

Before testing can start a number of activities have to be performed. The environments have to be built, prepared and "shaken down" to ensure that they are ready for the serious business of testing. Documentation has to be obtained from the development teams defining the requirements against which the tests are to be run so that test cases and conditions can be defined. MTS had little involvement in this but it was able to provide a support role through the definition of a *test calendar*. This is an Excel spreadsheet, colour coded for ease of reference, that gives a view of the tests planned for a particular test iteration. The calendar is updated in accordance with a timetable to ensure that it is available to the Test Authority for approval on a weekly basis.

Test readiness review

There is always a great desire to start testing as quickly as possible. Often this is before all the required elements are in place and this can lead to problems later on. The Test Readiness Review (TRR) process is designed to overcome these problems. It is essentially a facilitated meeting that uses a checklist to ensure that all components have achieved a specified level of 'readiness' before testing is permitted to start. MTS's role in this is:

- Assist the test stream in the preparation required to undertake the TRR
- Facilitate the review by organising the meeting, chairing the meeting and taking notes against the checklist
- Agree any actions that need to be taken before the testing can commence
- Produce a report documenting the current state of readiness
- Work with the test team to resolve any actions so that all components reach the desired state of readiness.

Quality assurance

QA means different things to different people. For the MTS team our main objective with QA was to ensure that documents delivered by the test streams were "fit for purpose", accurate and timely. We did this by adopting a typical quality review process based on existing company standards and in line with best industry practice. This process covers both documents produced by the testing teams and documents produced outside of testing that are used by the testers to determine what test cases and conditions need to be developed.

The sheer number of documents (over a thousand) that required a review gave rise to a number of problems:

- it was very labour intensive. This was resolved by allocating a QA analyst to work alongside each of the test streams, requiring seven people in all. This was a big overhead but easily offset by the benefits that are well known to be provided by formal quality reviews. In any case, it was not always a full-time role for the analyst and they were also able to provide a support role to the test team giving advice on what standards needed to be followed, and developing new templates where these were not available
- it was hard to control. This tricky problem was solved through setting up a QA matrix based on an Excel spreadsheet that identified all the documents and tracked them through the stages of the document review and approval process. Colour coding was used on this to enable easy identification of issues, and potential issues, that needed some corrective action – which usually meant chasing up of document authors/reviewers/ approvers
- keeping track of the latest version of documents was a challenge. This was addressed by defining a standard file structure for each of the test streams to follow, with specific areas set aside for draft, current and archive versions of documents.

Audit

The company has an internal audit department that carries out audits across all areas of the group from time to time. The sheer scale of the Integration Testing Programme, and the need to ensure compliance was maintained, meant that the level of audit activity needed to be increased. To achieve this, MTS formed its own audit team whose role was to:

- determine whether the agreed processes were being applied consistently across the programme
- to highlight any deficiencies and omissions
- to agree corrective action to be taken by the test team
- offer a degree of protection for the test teams from both group and external auditors. This was aided if required by allowing the QA analyst to work with the test team to resolve any findings.

Support services (test tool support)

TestDirector and *WinRunner* were already used, but knowledge of the tools varied amongst the vast number of staff that were brought in to this testing programme. In addition, few standards were available for how the tools could be used for a programme of this style and scale. MTS came to the rescue again by providing the following services:

- ensuring that the TestDirector environment was set up appropriately
- defining operating standards and processes for TestDirector
- assisting in setting up WinRunner for specific test streams
- providing or facilitating TestDirector training
- giving support and solving problems for TestDirector and WinRunner

Fault management

It would be nice to think that testing did not uncover any faults, but of course the reality is that in such a large programme, a large number of faults were identified. It became a major exercise for the test teams to follow up resolution of the faults raised on PVCS Tracker at the same time as they were trying to run more of the identified test scripts. It was clear that they needed help with this and MTS stepped in to offer support by:

- monitoring PVCS Tracker to ensure faults are dealt with in line with the agreed service levels
- preparing and distributing fault reports
- arranging control meetings to discuss outstanding faults and preparing reporting packs for these meetings.

Dependency management

The nature and complexity of this testing programme inevitably meant that there were dependencies not only between individual test streams but also with other groups such as Application Development. It became important to identify these, agree them and log them in a central database. MTS managed the process, providing the following role:

- facilitate process of agreeing dependency deliverables
- assist in completing dependency documentation
- ensure agreements are documented
- ensure logging on Testing Group dependency database
- notify parties of dependency identifier
- monitor and maintain database entries.

Test completion review

This process is the complementary checkpoint to the Test Readiness Review described earlier. It is designed to overcome the problems that can arise when a test stream says it has finished but in fact there are loose ends lying around that need to be tied up before it can really be said to be complete. As with the TRR it is essentially a facilitated meeting that uses a checklist to ensure that all components have achieved a specified level of 'completion' before testing is permitted to close down. MTS's role in this is:

- determine who should attend the TCR meeting from the project team and other areas.
- chair the review based on the use of the TCR Checklist. Complete the details for the Test Completion Review Checklist template.
- issue TCR Report, stating the conclusion reached by the TCR meeting.
- work with the test team to resolve any actions so that all components reach the desired state of completion.

Benefits achieved

The benefits that were seen from adopting this centralised approach fall into three categories:

Cost reduction

- minimise duplication of effort
- reduce the learning curve
- reduce costs through process repeatability
- improve getting it right first time
- risk mitigation

Risk reduction

- providing consistent interpretation and application of standards
- effectively adopting new standards and identifying missing standards
- providing synergy between dependant projects
- independent in action and judgement
- protecting against litigation.

Time optimisation

- focusing on relevant standards for the project
- providing guidelines for the adoption of these standards
- identifying improvements in training and processes
- acting as a central reference for "compliance" guidance
- providing the effective gateway for corporate governance.

Conclusion

Setting up the central support group for testing has proved to be a big success. There have been a number of benefits, feedback from our customers has been good and we have gained the backing of Group functions that are responsible for ensuring that compliance is achieved. Some lessons have been learned and we will be using these to make improvements to our processes before we are called upon to support the next big testing programme. PT

TURKEY • GREECE • CROATIA • ITALY • FRANCE • BALEARICS • CARIBBEAN • SEYCHELLES • WHITSUNDAYS • FRENCH POLYNESIA

Top Yacht

Specialists in top quality yacht charter

BAREBOAT, SKIPPERED, FULLY CREWED AND GULET HOLIDAYS

Tel 01243 520 950
or email sailing@top-yacht.com
www.top-yacht.com

MENTION Professional Tester FOR A SPECIAL DISCOUNT!

StarBase's unique Advanced LoadRunner Training Courses

StarBase's Advanced LoadRunner Training Course has been developed to help you make the most of your investment in LoadRunner. The in-depth 4 day course provides answers to questions that people have when using LoadRunner with real world testing scenarios.

Course dates:
24 – 27 November 2003
26 – 29 January 2004
29 March - 01 April 2004
14 - 17 June 2004

For further details contact Karen Yadav on 020 8905 1120 or go to www.loadrunnertraining.co.uk

CHANNEL ALLIANCE PARTNER
MERCURY INTERACTIVE

StarBase are a Mercury Interactive top tier business partner

"Professionally presented, well organised and full of excellent tips and tricks to utilise the full potential of LoadRunner."

John Brennan, Abbey National

StarBase

Automated Testing Specialists

Incisive solutions with integrity

www.starbase.co.uk

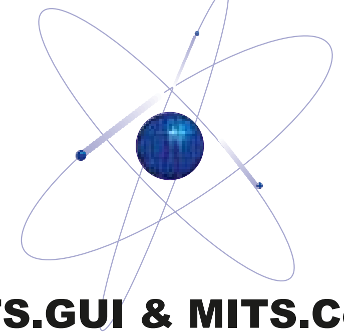
The perfect holiday studio flat in the heart of **Nice** (old town)

Light and spacious • sunny terrace • double sofabed
fully equipped kitchen • modern shower room
exposed beams • lots of charm • 5 mins to beach
easy access to airport, the famous restaurants, markets and galleries of Nice, the fabulous Cote d'Azur and the 'perched villages'
ideal all the year round • car not necessary

£235-£335 pw
depending on season.
For details please call Jennifer Murray on 01273 888033 or email jenmurray@20powis.fsnet.co.uk

GUI testing with no scripts.

None.Zero.Zip.



MITS.GUI & MITS.Comm

Technology totally unique on earth, in the universe, and . . . well actually we're making assumptions beyond our solar system, but you get the idea.

OMSPHERE, for the inside track in an upside down world.

OMSPHERE

San Francisco CA 94013 USA **www.omsphere.com**
phone (+US) 415 439 5272 **fax (+US) 360 397 7221**

Mission Testing

Leading the way in specialist testing

Consulting

Through early involvement in testing programmes our experienced consultants have a proven track record in delivering significant business benefits to our clients.

Recruitment

A tailored flexible service for all your test recruitment needs. Our expert team take care to understand your exact requirements to ensure we provide the right people at the right time.

Education

We provide ISEB accredited courses, specialist industry workshops and seminars to motivate, retain and develop testing staff, thereby enhancing skills and productivity.

Managed Services

By combining our capabilities we offer a unique solution to the requirement for flexible levels of skilled testing resource. This enables improved planning and cost reduction.

For more information please contact:

T: 01293 44 00 22 E: info@missiontesting.com
W: www.missiontesting.com

Mission Testing is part of The Capita Group Plc.