Testing new technologies

Including articles by:
Allen Fisher TestPlant  Sami Petäjäsoja - Jarkko Lämsä - Anna-Maija Juuso Codenomicon
Ivan Ericsson SQS  Patrick Heymans PROLIN  Geoff Quentin
iqnite is the leading software quality management and testing conference. A series of annual events worldwide are the meeting place for the testing and quality community where you meet experienced industry insiders and eminent experts sharing their knowledge and presenting the latest industry trends.

www.iqnite-conferences.com
Testing new technologies

Innovation in testing often reacts to that in development, to testing’s detriment. In this issue our contributors present three ways of working that might help us keep up with or even get ahead of emerging technologies, by remaining applicable and effective even in the face of future change. That powerful idea of truly generic testing is continued in the first instalment of Geoff Quentin’s Consolidated Testing Process.

In the next issue: where next for testing standards?

To many testers, standards are something they hear of on training courses and then forget about. Others believe that trying to impose predefined practices on what they consider to be a creative activity is a waste of time. There do appear to be some large gaps between standards and their practical application. But closing those gaps might lead to great benefit: the elimination of variation and uncertainty in the meaning and effectiveness of testing. The next issue will survey the standards currently in use and disuse, examine the work so far on some emerging ones, and consider what might be needed to make them truly and universally useful.

Share your knowledge with fellow testers

As always, we invite contributions, ideas and opinions in any form, emailed to editor@professionaltester.com. Sincere thanks to everyone who gave feedback on the last issue: you are helping us to deliver the magazine you want.

Edward Bishop
Editor
Test automation ready for innovation

By Allen Fisher

Pure black box testing is by definition immune to technology change. But can black box test execution be automated?

Allen Fisher explains how image recognition can solve long-standing problems of test automation and enable its application to new and future platforms.

The best-known test execution tools run on Windows and are for testing Windows applications, or Web 1.0 sites optimized for Internet Explorer. But what about:

- other OSs such as Mac OSX and various Linux distributions which are gaining market share?
- smartphones, netbooks and tablets?
- automatic online updating of third-party software causing change?
- cloud applications, and users who want to use them with their choice of browser, a choice that’s growing fast?
- Web 2.0 sites including Flash, Flex, Silverlight, Java, Rails, PHP, .NET, etc applications, often alongside one another on the same page? (see figure 1)
- highly configurable and skinnable user interfaces?
- closed systems which will not run alongside other applications or would then behave differently due to time dependencies? (see figure 2)

The solution to these challenges is a truly black box approach, which most test execution tools cannot achieve. To work they have to (i) identify user interface objects, or (ii) record events in terms of screen coordinates. In case (i) the tool is highly implementation-specific: testing a version of the same product for a different platform requires either a different tool which cannot use the tests already created, or reversion to manual execution.

In case (ii), tests become invalid when the UI changes, whether as a result of development of the system under test or external factors: they fail to complete and/or return false incidents. Finally, traditional test tools run on the same computer as the SUT, requiring the OS to switch control continually between the two, so are often unstable and/or intrusive.

Achieving black-box automation: run the tool on another computer.

The test automation tool eggplant runs on Mac OSX and Linux. But it can test applications on a vast range of other platforms. That’s because it connects to and controls them using VNC (Virtual Network Computing). VNC is a remote control protocol similar to Apple/Windows.

Figure 1: iGoogle page using multiple technologies

Figure 2: Image recognition in test automation

Testing new technologies

PT - March 2010 - professionaltester.com

4
Remote Desktop, used for example by IT departments to manage server configuration or support users by temporarily “taking over” their workstation. eggPlant takes the role of a human operator, so can “see” the display, control the pointing device, and send keystrokes on the SUT device. VNC servers are available for virtually all important platforms including Symbian, Windows Mobile, Android and iPhone. An iPad version is sure to emerge soon.

Achieving test robustness: image recognition

So, eggPlant works by “seeing” the screen, and does not care what any of the items on it are called or what is their underlying technology. In that way, it’s like a screen coordinate based tool. But it doesn’t share their disadvantages because it records the UI objects as bitmap images of sectors of the screen rather than simply geometric points.

When scripting a test that includes, for example, a click on a button, the tester captures an image of that button (see figure 3). The image is given an arbitrary name which is used to refer to it in eggPlant’s scripting language, SenseTalk (figure 4). It is necessary to capture it only once; subsequent interactions with the object are scripted by selecting it by name. As well as text, images can form part of the expected outcome for the test, i.e. the screen output: they are captured and named in the same way.

When the script is executed, eggPlant finds the image wherever it is on screen using image recognition algorithms and performs the scripted action. So the test runs even if the UI objects and outputs are in different locations on the screen, for any reason. If an image cannot be found, SenseTalk’s ImageFound() function returns FALSE, so the script can raise an incident.

Other than if navigational structure of the SUT changes, the only time an eggPlant script needs maintenance is if the appearance of an object changes. It’s achieved by executing the script with the Image Doctor feature. When eggPlant cannot find a required object, it pauses. The tester then selects the object as it now looks. Execution continues, pausing again if another object cannot be found so that the operation can be found. When the script finishes, the test has been repaired; it is now correct for the current build of the SUT.

Achieving test reuse: image collections

When an image is captured using Image Doctor, it does not by default replace any previous image of that object, but is added to a “collection” of valid images for that object. During subsequent execution of the script, eggPlant searches for all of them.

This means the script can be enabled to test any other version of the SUT, including those running on other platforms, by exactly the same procedure: eggPlant is connected to the new test item and the script is executed. Images of how the objects now look are added to their respective collections (images in embedded objects such as Flash and Java applets will often look similar enough to be found without user intervention).

Abstracting the UI from the test procedure in this way means the same script can be used for multiple builds, versions (including language translations) and ports to new platforms (figure 5), multiplying the return on investment in automating the tests.

Allen J Fisher is a Senior Systems Engineer with TestPlant, the producer of eggPlant.

Testing new technologies
Introducing TPI® NEXT
Sogeti’s Business Driven Test Process Improvement

The world’s leading approach for improving test processes... has now been enhanced!

Sogeti’s new book, TPI® NEXT, Business Driven Test Process Improvement written by Sogeti test experts and validated by extensive customer field tests, builds on the strengths of the successful original model and provides invaluable insight and recommendations on the maturity of an organization’s test processes.

What’s new about TPI® NEXT?
— Key focus on alignment with business drivers is at the heart of the model
— Reflects Agile and Outsourcing changes in testing environment
— New ‘Enablers’ to identify the impact on broader Software Development Lifecycle
— Clear maturity levels – Controlled, Efficient and Optimizing
— Easy-to-view representation for business management.

How TPI® NEXT can help your test organization
— Develop a clear improvement path to improve quality, reduce time and save costs
— Ensure processes support the most important business drivers
— Gain a better understanding of the correlation between testing and other software development processes.

Order your copy now!
Available from 17 November 2009 from specialist publisher UTN at www.utn.nl or contact Sogeti at tpi@sogeti.nl.

TPI® NEXT an indispensable step-by-step guide to improving your organization’s testing processes.

www.sogeti.com
Hot fuzz

By Sami Petäjäsoja, Jarkko Lämsä and Anna-Maija Juuso

New access technologies make 4G, the new generation of wireless networks, more vulnerable but also more testable.

Sami Petäjäsoja, Jarkko Lämsä and Anna-Maija Juuso explain fuzzing

Long Term Evolution (LTE) is the emerging network standard for wireless communications. Networks based on it are also referred to as “4G”.

LTE differs from earlier standards in that it is completely IP-based. This will improve capacity, speed and profits, but it brings Internet security risks into telecom networks which until now have been closed. The new and more powerful access technologies it enables make LTE networks more vulnerable.

One approach to security is to try to stop malicious data entering the network. However that adds to complexity, expanding the attack surface of the network system. As long as software defects exist, so do potential exploits, and adding more software can increase both. It is better therefore to concentrate on detecting defects for removal and one way to do this is to test robustness of software using the underlying protocols. The fact that IP is an open standard makes this relatively easy, leading to novel test automation techniques such as “fuzzing” which can be used to assure reliability and security in applications of emerging technologies including LTE.

Why and where LTE is vulnerable

The first step in assessing security risks is identifying critical interfaces. In mobile telecoms, we must study the network boundaries. For example, when a mobile phone user accesses different carriers while travelling abroad (“roaming”), core functions are performed by components in those carriers’ networks. In order to charge the user, carriers must allow the roaming network provider access to their subscriber data, creating a trust boundary. This requires a high degree of robustness from the network components involved.

Also, LTE is backwards compatible, supporting 2G and 3G access to enable smooth transition from earlier technologies but exposing networks to security threats. Previous technologies contain legacy features such as Wireless Application Protocol (WAP) which are dangerous to LTE because they are rarely used and require their own safety features, adding complexity.

The most critical interfaces in the LTE network are the interface between the off-site access point base station (eNB) and the core network, and the interface between the core network and the Internet (see figure 1). These are the two interfaces carriers can least control. High-level protocols, in particular IPv4 and IPv6, pose the biggest risks as they are the most open protocol layer used and the easiest one to attack due to the large number of ready tools.

In 2G/3G networks all traffic passes through a radio interface and is controlled. Unlicensed Mobile Access (UMA) enables handsets equipped with Wi-Fi or Bluetooth to access GSM or GPRS core networks through unlicensed public and private...
wireless networks, but UMA traffic is still controlled by the UMA Network Controller (UNC).

LTE however provides for the use of femtocells, which are like home routers and connect in-home cellular access points via a home broadband connection to a carrier’s switching system network to establish a localized cellphone service. Femtocells can access the carriers’ core networks directly via the internet (figure 2).

Earlier technologies did not enable users to program the handset radio features, but LTE now makes this possible. For example malicious data packages designed to open within the core network can be created.

Using fuzzing to detect vulnerabilities

In robustness testing invalid, unusual or unexpected inputs are made to the system under test to establish how it behaves, externally and/or internally, when it receives them. If an incident occurs then there is a vulnerability that could also be exposed in production.

Fuzzing is a form of robustness testing which focuses on communication interfaces and the discovery of potential security-impacting issues such as overflows and untrapped runtime errors. Any defect is a potential threat to both security and functionality, because attackers search for vulnerabilities in a similar way, looking for an input that produces an abnormal response. They then refine the input, trying to find a combination that causes their desired behaviour. However, not every failure is caused by illicit activity of this kind. Coincidental combinations of inputs and circumstances can also trigger failure: usage peaks are one common contributor. The most effective way to reduce risk of failure is to improve code quality, and using fuzzing throughout the development process is an easy method of doing so.

The strength of fuzzing is its unparalleled ability to find unexpected vulnerabilities. It can be used at all test levels, including on individual software components, enabling defects to be detected and repaired and so software to be retested and regression tested as early as possible. Fuzzing is a very representative test technique because it targets the problems attackers would find more systematically and far more thoroughly than manual exploratory testing. It uses automation to create and execute thousands or even millions of misuse cases for every use case.

Mutation-based and model-based fuzzing

Mutation-based fuzzers are created by capturing network traffic, breaking down the structure of the message exchanges within it, and tagging each element with metadata, which can then be used to automate their recombination to form new structures. The weakness in this approach is that the traffic captured is only a sample; how representative it is is unknown and it tends not to include patterns caused by rarely-used legacy technologies and features such as WAP. Mutation-based fuzzing is often used for very new or obscure protocols and other technologies which are not well specified.

Good specifications allow the use of model-based fuzzing. The specification is used to identify the data elements and also provides other protocol- or format-specific information, for example the boundary values of each element. Model-based is more systematic than mutation-based test generation: the specification allows the protocol to be understood rather than just sampled. Thus the model-based approach produces fewer test cases, because the specific information enables redundant or impossible test cases to be eliminated. The resulting shorter test execution times allow tests with high defect-finding potential to be run more frequently during development, for example on every build, shortening the time between defect introduction and detection and therefore reducing the cost of repair and retest.

Sami Petäjäsoja is senior product manager, Jarkko Lämsä is a robustness and security testing specialist, and Anna-Maija Juuso is a communication and marketing specialist at Codenomicon, which produces a range of protocol-level security testing tools.

Figure 1: LTE network boundaries

Figure 2: access networks
Testing centralization means business optimization

Separating testing from the business and development entities it serves can make it more adaptable when called upon to test systems involving new technologies.

Ivan Ericsson of SQS explains why and how to go about centralizing testing.

To enable their parent organizations to benefit from and compete using emerging and developing technologies such as web 2.0, cloud and devices like the iPad, test organizations need to grow and adapt their capabilities rapidly; but to manage the risk associated with change, they need processes to mature faster too. That can be achieved by centralizing testing functions.

A test process used by a particular business unit or function to test its own systems will mature as lessons are learned, but too slowly. Concentrating testers and their knowledge, and supplying testing as a service to business and development, accelerates this. The more often a process is used, and adapted to deal with different challenges, the faster it improves. Practice, after all, makes perfect.

The concept of an effective centralized testing function is not new. It has been discussed for years and attempted repeatedly, often with vague goals and rarely with measured benefit. That must change, and we at SQS study continually the factors that lead to success in building a modern test centre. This article will explain what we believe are the five most important, and the advantage attention to them will bring to organizations that want to gain the maximum from new technologies.

**Factor 1: strong and deep foundations**

In our experience a leading cause of failure is rushing into building the test centre and delivering its first projects prior to establishing organizational foundations. That weakens its integration, disconnects it from business value and can block process improvement. It is essential to agree defined success criteria and manage relationships with all stakeholders, to ensure that the test centre is properly involved in deployment of new technologies. Using the test centre must not be seen as optional: active senior sponsorship is the one of the hallmarks of a successful test centre.

For example, consider an organization which is beginning, as are so many at this time, to convert its applications into cloud services. That needs engagement of many organizational divisions and their functions, including architecture, support, business, development and governance. To deliver on-demand testing successfully requires that the designs have testability built in. Distributed, isolated test teams rarely have the influence to achieve that, but a test centre must.

Integration with other processes is also key. The test model needs to be supportive of and supported by the development...
model, project model, design model etc. A major benefit of cloud is its elasticity, and that means testing must be flexible and scalable. That is impeded if the time, logical, informational and resource dependencies between testing and other activities are not present in all relevant plans, impacting upon delivery.

**Factor 2: focus on organizational benefit**
An organization with very expert testers is most likely to make the decision to centralize. However these experts are not necessarily the best people to make it succeed, if their understanding of testing exceeds that of business reality. The test centre must not become an academy or standards organization. Rather, it must deliver the effective testing needed to support organizational success, pragmatically.

**Factor 3: measurement**
Every test manager should be able to answer these three questions:
- What is your current test effectiveness level?
- What is your current test efficiency level?
- What is your test process improvement plan?

Test effectiveness is measured some time after delivery and deployment of a product by quantifying:
- Revenue loss due to downtime and/or incorrect operation
- Reputation loss due to downtime and/or incorrect operation
- Cost of supporting, fixing, retesting and redelivering to reduce downtime and/or incorrect operation

Test efficiency is measured at the time of delivery. There are many ways to do this, including quantifying:
- Cost per test
- Cost per defect
- Number of tests executed per defect found
- Proportion of tests automated
- Test phase containment (or testing overrun)

The measurements and how to take them should be defined and agreed, and then the measures taken, before starting work on building a test centre. Thus its success or otherwise can be demonstrated by repeating them once it is operating, eg for each project. The measurements are also essential input to test process improvement.

**Factor 4: legacy**
To sustain success a test centre must be built by permanent, internal staff from the outset. Using external specialists additionally is often necessary, but using them exclusively causes failure, as too much of the essential knowledge and experience gained leaves with them.

A good way to retain it is to appoint permanent staff members to head special interest groups, whose members are colleagues and external consultants. The SIG head is responsible for research, attending events, evaluating tools etc and communicating what is learned to members. For example, an organization interested in the iPad should create and resource an iPad app testing SIG now.

If/when the organization’s first iPad project is initiated, this small investment will give it an immediate competitive advantage - the ability to provide the right and best testing on demand, without the pain and delay of discovery. Even more importantly, that ability will remain integrated when external consultants are no longer needed.

**Factor 5: think centrally, act locally**
Every test centre has strategic goals: general, widespread quality improvements and cost savings. But pursuing these alone can cause a loss of empathy with individual parts of the organization and understanding of their localized changing challenges and needs - “ivory tower syndrome”. The centralization and ongoing improvement plans should therefore include also tactical goals: defined, measurable localized benefits. After these are delivered, future widespread adoption of new technologies will be far easier, because all the entities, divisions and functions required to support it have been enabled to do so using the services provided by the test centre.
The Consolidated Testing Process

by Geoff Quentin

The implementation of a formal test management process using the Generic Test Process in a consolidated approach

Part one - objectives, structure and the principles of testing

Geoff Quentin explains what testers should do and the deliverables they should produce at every stage of the software life cycle

Objective of the Consolidated Testing Process

This Consolidated Testing Process ("CTP") is intended for use by anyone involved in software testing, from providing high-level strategic advice to executing component tests. It aims to define best practice so that testers and their organizations can concentrate on improving the delivery process, rather than its products, in order to help prevent loss caused by producing (i) wrong products and (ii) right products, wrongly.

Structure of the Consolidated Testing Process

The CTP is based on the software life cycle described by ISO/IEC 12207 Software Life Cycle Processes (“12207”) and describes a Generic Test Process (“GTP”) and Test Management Process (“TMP”) within that framework. Because these generic processes are applicable to any testing work, they can be used to integrate all the testing activities, achieving complete traceability. This integration and traceability enables full management control and is the reason the process is called “consolidated”.

Section 1

The principles of testing

Verification

The verification process is the process by which the product of a development process is checked for compliance with the specification for that product. It consists of checking that the product is each of the following: complete; consistent; accurate; unambiguous; compliant with prevailing standards.

The verification process is carried out using one or more of the following formal checking processes: review; prototyping; simulation; modelling; empirical testing; mechanization.

The verification process is further defined in 12207 section 6.4.

Validation

The validation process is the process by which the product of a development process is checked for compliance with the original requirement for the system as a whole. It consists of checking that the product is each of the following: required (i.e. it has an owner); of the correct quality; built to the correct specification; compliant with prevailing standards.

and that the product has been produced
using the correct standards and processes.

The validation process is carried out using one of the following formal checking processes: review; empirical testing; model office or another appropriate method.

The validation process is further defined in 12207 section 6.5.

Certification and completion criteria

Certification is the process of checking that the product is fit for purpose. The certification or quality control process is a vital assessment task for ensuring the correct quality of all deliverables and of the delivered system as a whole. The certification work establishes completion criteria and meeting of these completion criteria means that the product under test is ready to release to the next stage of development or for implementation. Completion criteria are individual conditions that must be met: the set of all conditions that must be met can be called the “requirements” or “exit condition”.

The certification process is the process by which the product of a development process is demonstrated as compliant with the original requirement for the product. That product could be software at any level from a component (program) to a complete system, or other deliverables such as specifications and design documents.

Certification consists of the following planned tasks: establishing and maintaining status reports; defining the testing strategy identifying test objectives and commencement and completion criteria; planning the tests; analysing what is to be tested; designing the tests, scheduling the tests; executing the tests; analysing the results

These seven tasks comprise the GTP. They will be defined in section 2 and specific examples will be given in sections 7, 8 and 9. They may occur as a single block of tasks but are far more likely to be spread across several activities. All must be performed.

The certification process is carried out using one of the following formal checking processes: analytical testing; empirical testing; model office

The certification process occurs at specific planned points during the development process: acceptance test; system qualification test; software qualification test; unit test

For especially large or complex systems, it may be useful to perform the certification process at additional points: component (program) integration test; application (system) integration test

The views

There are many roles and responsibilities that need to be covered with correct and adequate resources if a development project is to succeed. They can be grouped in various ways. The simplest is to use just two: the acquirer and the supplier. These are sufficient for many purposes but for producing software which operates core business processes it has been found essential to add further views and roles. 12207 defines the following views: contract; management, operation; engineering; support

The contract

At contractual level the acquirer to supplier relationship needs to be defined. Documents will be exchanged to formalize this relationship. There must be a formal document recording the decision to acquire a system, the time scales, budget, the risks and the benefits, and the acquisition process to be followed.

The CTP requires that there is a formal record of identified risks; this information will be used in both the establishment of the testing strategy and the prioritization of planned tests.

The CTP requires that there is a formal record of how the acquired system will be part of the existing business system, be part of planned changes to existing systems or replace systems within the acquirer organization, expressed as critical success indicators (CSI). Critical success factors (CSF) are also identified. The CSIs are used in formal planning, test prioritization and test design and in the management of change. The CSFs, the factors influencing success and failure, are critical to the successful management of the project.

So, to support the CTP the following existing documents are needed: acquirer-supplier contract; project objectives; milestones; budgets; roles and responsibilities

and the following new documents are created: critical success indicators; critical success factors; business objectives and quantified benefits; risk matrix and quantified risks; testing policy; quality control points; quality requirements; acquirer acceptance criteria

The following existing documents may be useful but are not essential: company structure; existing system catalogue; usage information

The management

All the management of the work needs to be defined. This management definition covers the management activities within both the acquirer and supplier processes.

Documents will be created to formalize the responsibilities. Roles and responsibilities must be defined.

Management objectives must be clearly identified and the choice of development process formally recorded.

The process requires that there is a formal record of all quality control (QC) points throughout the chosen development and acquisition process. The CTP requires that these QC points are directly linked and hence traceable to both the testing activity to be done at other QC points and the business objectives recorded as CSIs.
The CTP requires that there is a formal record of identified responsibilities for all the quality control tasks; this will be used in establishing the testing strategy. The process requires that there are statements of management policy supporting the testing tasks needed to confirm that the QC points have been passed successfully.

It may also be useful to link CSFs to specific roles and responsibilities. Thus the success of the project as a whole is made the responsibility of the whole management team and the specific contributions of individual managers are clearly defined.

In order to achieve process improvement, formal standards and procedures and metrics are required. The CTP meets all the requirements for process improvement defined in the ISO 9000 quality management standards.

To support the CTP the following existing documents may be useful but are not essential: acquire/supplier contract; project objectives; milestones; budgets; roles and responsibilities; critical success indicators; critical success factors; business objectives and quantified benefits; risk matrix and quantified risks; testing policy; quality control points; quality requirements; acquire acceptance criteria

and the following new documents are created: specific testing roles and responsibilities; specific acceptance criteria; function catalogue; issue grading guidelines; issue tracking system

The following existing documents may be useful but are not essential: training schedule; company structure; existing system catalogue; usage information

The operation

The management process is further defined in 12207 section 7.1.

The Consolidated Testing Process

The operation

It is crucial to the successful implementation of a system that those who will have the responsibility for the operation of the system are involved in the development of those aspects of the system that have a bearing on its operation. The CTP requires that operation test plans are created, by operations staff, at system design stage. These tests are to check the specific operation of the system and can be conducted using role playing, modelling and other techniques. Once tested the tests are then used during system qualification testing.

To support the CTP the following new documents are necessary as for the management view (section 1.4.2 above), and the following new documents are created: operations procedures; specific testing roles and responsibilities; specific operational acceptance criteria; issue grading guidelines; issue tracking system

The operation process is further defined in 12207 section 5.4.

The engineering

The development process is the process by which a system is specified, designed, built and implemented. The development process is instantiated by management, supported by the support processes and quality controlled by all the various testing tasks. It is a requirement of the CTP that all involved at the engineering level understand and follow the testing strategy as created by management at the start of the project.

The CTP defines QC tasks to be used at the various stages from process implementation to post acceptance support. It is a requirement of the CTP that these quality control points are used and the established completion criteria are met.

It is a requirement of the CTP that full traceability is maintained throughout the development process.

To support the CTP the same existing documents are necessary as for the management and operation views, the same new documents are created as for the operation view, and the same existing documents as for the operation view may be useful but are not essential (section 1.4.3 above).

The development process is further defined in 12207 section 5.3.

The support

Success is dependent on the availability of suitable, essential support processes: documentation; configuration management (including change management); quality assurance (including process improvement); verification; validation; joint reviews; problem resolution; audit

The CTP defines how to carry out verification, validation, review and audit.

The support processes are further defined in 12207 section 6.

All documents identified in sections 1.4.x should be subject to configuration and change control.

In the next issue The Consolidated Testing Process part 2, including: The V model according to 12207, issue tracking, business-driven testing, the Generic Test Process

Geoff Quentin was chairman of the British Computer Society Special Interest Group in Software Testing at its foundation in 1989, is author of The Tester’s Handbook and many seminal training courses, won the European Testing Excellence Award in 2006 and founded, with his wife Caroline, this magazine.
The growth of “agile”, other non-sequential lifecycles, and low-level testing tools has tended to blur the distinction between developers and testers. Increasingly, it seems, developers believe they can perform adequate testing themselves, or at best just need someone to find bugs after coding, and fail to see the benefit of integrated, truly independent testing. Finding the right role in this environment is one of testing’s key challenges. It’s time to reappraise how we can best work together.

As a starting point, PT asked Patrick Heymans, the most insular, code-focused, “heroic” architect/designer/developer we could find, for his views on the matter. They are provocative and we present them here without further comment, but invite readers to join a discussion PT believes is very important.

Professional Tester: You work on a commercial product with an advanced user interface aimed at the technically very able (support technicians). What is your starting point for a development cycle? Do you use anything testers would recognize as requirements or their equivalent?

Patrick Heymans: The product we’re building replaces an existing one and more than 95% of its functionality is similar. Testers already have sufficient knowledge about that. The new functionality is described on paper.

Do you think early work on detecting defects in that description could help you work more efficiently?

Patrick Heymans: They are high-level requirements that serve mainly as input for a more in-depth software design document containing both a functional and technical specification. However, from a technical perspective it can often be difficult to write things down first, especially when skill levels are low. So we’ve adopted Test Driven Development [ie test-first coding using an automated unit test framework and code coverage measurement tool – Ed] which requires that you write tests before you write the actual implementation. If used right, it leads to clean and highly maintainable code which is validated by an automated test suite on a daily basis. Regression is significantly reduced and this saves a lot of time later in the project and on future releases.

How do you think subsequent change to requirements should be handled, by business, developers and testers?

Patrick Heymans: Another benefit of TDD is that it leaves room for changes. This product has undergone large ones, but none had a great impact on overall stability. In traditional development methodologies like waterfall such changes often cannot be made because more and more time will be spent fixing bugs when the project is reaching its deadline. Changes to requirements are inevitable as it is the business that drives them. People and
Edgewords provides training in the full suite of HP test tools; LoadRunner, QuickTest Professional (QTP), and Quality Center, as well as ISTQB, ISEB and testing theory courses.

Specialists in on-site training. Full public course schedule also available.

www.edgewordstraining.co.uk
Tel: +44 (0) 1746 766246 or Email: info@edgewordstraining.co.uk
processes should be open enough to deal with this.

**What tools are you using to perform TDD?**

— The most important is NUnit. We also use TestDriven.NET to interface with the Visual Studio 2008 IDE, Hudson for running nightly and integration builds, NCover for code coverage metrics, and a tool we developed in-house that produces fancy reports from the test results.

How do you know that you are selecting the best tests? Do you use test case design techniques? Could having an expert tester sitting alongside to create more thorough tests, before you start coding, make TDD even more effective?

— Knowing that your tests are good is a matter of experience, but you definitely need a good code coverage tool to support you. To me expert testers are best used at a higher level, doing other work which complements the testing done by developers. That is crucial to bring the product to a higher quality level. However with the developers doing TDD the testers can concentrate more on the really hard-to-find bugs.

**Advocates of agile etc - and many testers with more traditional views - believe it's good for testers and developers to talk a lot. Others think they should keep apart and communicate only by formal means such as incident reports, to avoid risk of influencing one another wrongly. Where do you stand on this?**

— I don’t see anything wrong with a debate about a feature whose design could be interpreted in different ways. That should be noticed by the developer, but not all are capable of doing so. So input from a tester early in the process can be valuable. However a software architect should be involved in any discussion.

Many testers have little or no development ability. In fact some see this as a positive advantage, helping them to focus on the users’ interests and point of view rather than “going native”. Would you prefer to work with testers with a development, business or user background?

— I prefer testers that incorporate all three, but I think a development background is the least important. In TDD there is an overlap between developers focusing on low level (unit, integration) testing and testers focusing on high level (functional, system, acceptance) testing.

A common complaint of business is that testers’ pedantry tends to delay and de-energize projects early on. Testers on the other hand consider that by insisting on correct detail they are reducing the risk of wasting development effort. Do you see any common ground here?

— Having a clear and correct design before development starts would be good but in practice this is not how things work. By the time the design is finally ready, requirements may have changed. In my experience most development effort is wasted later in the project on bug fixing and reworking implementations that didn’t follow good design principles.

**Excellent, intuitive usability is a key sales point for your product. Modesty aside, how do you think it has been achieved? Has testing helped, or could it have?**

— There was a previous attempt to develop the same product, not involving me. The team was 20 times larger than ours with multiple developers, testers, human factor engineers, technical writers, project managers and so on, using the waterfall approach. They took more than six years to deliver the first release which had very poor stability, performance and user experience.

We have released our first beta after just two years of development with a tiny team (ie just me for the first year). The product is not 100% stable yet, but because it is backed by almost 8,000 automated tests the number of defects is fairly low. I think this proves that TDD works.

**Finally, as a successful developer with a record of achievement would you ever consider bringing your skills to testing? And if you did, do you consider that your experience would help you influence design/development in a positive way or that, as many developers-turned-testers claim, to be an effective tester you’d have to change your focus radically?**

— TDD already lets me bring my skills to testing. Doing that requires discipline and may take some time depending on the individual. However, if they give it a fair chance and understand the benefits they may well become hooked.

Patrick Heymans is chief software architect at PROLIN. The product to which he refers in this interview is PROLIN Smart Client 7, an add-on to provide an enhanced, high-performance user interface for HP OpenView Service Desk 4.5. An evaluation download is available free at prolin.com.

**Do you agree?** Should TDD change the role of testers in this way? If not, how else can we as testers contribute to developer-defined processes like this? Or should we continue to argue for other processes? Please send comments, opinions and further questions to editor@professionaltester.com. They will be put to Patrick and other developers in future issues.
VELT for Web  [part two of two]
Example of creating test objectives

The conclusion of our tutorial on the key challenge of very early lifecycle testing: making something testable from nothing

Business requirements for change to public-facing websites are usually incomplete. Testers can add important properties to the test path by considering the requirements systematically with reference to the fundamental functions of any public-facing website (see panel). Most design decisions will already be known, or easily predictable. When that is not the case, they are written generically; specific information can be added when available and if necessary. They are then used as basis for creating and detailing test cases, which should reveal if anything important is still missing.

Suppose the business requirements document or other test basis information for change to an insurance company’s website contains the following requirement:

The user should be able to discover the current status of an existing claim

Before starting to plan testing to assure this, consider first the fundamental functions. This will give rise to questions that should be asked. If these questions can be answered, the test objectives can be made more detailed. If information is not yet available (because some design decisions have not yet been taken), the test objective should still refer to it.

Load: should this function be accessible directly, ie by navigating directly to a URL rather than navigating from other locations within the site? By what name(s) will the function be referred to in navigation mechanisms, eg menus and breadcrumb trails? What should happen if required browser features are not available or are disabled, eg JavaScript, pop-up windows, Flash?

Confirm: are all pages visited when using the function compliant with prevailing content and style guidelines/constraints?

Navigate: can all pages visited when using the function be bookmarked (added to favourites) conveniently? Can the pages be accessed via the browser history? Can the pages be navigated between using the browser's forward and back commands/buttons?

Generate message: is it required to email the user or any other parties to notify them that someone has attempted to discover the claim status, or system administrators if the status information cannot be retrieved?

Accept and forward order: is it required to record that the status of the claim has been discovered by a user?

A careful consideration of the possible meaning of each of the fundamental functions to this requirement reduces the risk of important features and properties not being tested.

A set of test objectives is then written, which depending upon the answers to the questions above may include the following:

TO1: To show that a page enabling the user to discover the status of an existing claim loads when the user directs the browser to the URIs:

http://protesterinsurance.com
/claimsTracking/claimsTracking.htm
http://www.protesterinsurance.com
/claimsTracking/claimsTracking.htm
http://protesterinsurance.com
/claimsTracking
http://www.protesterinsurance.com
/claimsTracking

PT editor
Edward Bishop
derives test objectives
Fundamental functions of virtually any public-facing website

These are derived directly from the fundamental business objectives: presence, advertising, providing information, facilitating communication and facilitating operations respectively. During very early lifecycle test design, they should be considered in addition to the available business requirements.

Load: the most fundamental function of any application. In web terms, it means what the user must do in order to see the first document needed to access content or a specific function, and what should happen under the various circumstances that might impact that initial access.

Confirm: relates to the content and design aspects of the site that help the user to identify it and/or its parent organization.

Navigate: the function provided by the mechanisms and choices offered to enable the user to access desired content and specific functionality.

Generate message: the function that operates whenever the site sends output to anywhere other than the client workstation or its own components (including physically or commercially external components such as web services). For example, an email to an appropriate person when the user submits an inquiry or feedback form, or a warning message to system operators when unusual user activity (e.g., three failed login attempts) is detected.

Accept and forward order: the function that operates whenever a user event is recorded in persistent data. The term “order” in this context is very broad; it could be a sales transaction on a commercial site, or a user downloading a file for free, or just accessing information. Any user event is called an order if it should be recorded.

Of course when this activity is being carried out, these URIs many not yet be decided or known. In that case, a generic phrase such as “the applicable URIs” is used. Thus, this fundamental function is captured for testing - that is our most important aim. The testing itself will be developed and detailed with the required specific information when available.

TO2: To show that a page enabling the user to discover the status of an existing claim loads when the user chooses the navigation option “Track a claim” or “how to track your claim”

If the specific navigation is not yet decided/known, the phrase “when the user navigates to it” can be used.

TO3: To show that all pages displayed when a user attempts to discover the status of a claim are compliant with prevailing content and style guidelines

Note that this differs from the requirement statement which refers to “an existing claim”. The test objective is worded so that it applies when the user attempts to discover the status of any claim including one which does not exist. This adds the “error” pages and messages to the test path: those are a very common source of failure because they are often omitted from testing.

What the guidelines are and how they are defined varies between organizations; some include them in the requirements document, in a section sometimes called “confirm”, meaning that the identity of the site or its parent organization is confirmed and reinforced sufficiently in the mind of the user as they use the site. This tends to give rise to test objectives defining properties the site should have, e.g.

TO4: To show that the site logo (full definition in appendix) is displayed consistently during navigation

Other organizations refer to a set of constraints, sometimes called a “stylebook” etc, defining properties the site should not
have. This gives rise to “negative” test objectives such as:

**TO5:** To show that no font other than Arial or Helvetica appears on the site

Now we come to the navigation fundamental function.

**TO6:** To show that all pages displayed when a user attempts to discover the status of a claim create a unique bookmark/favourite

This could also be written “To show that all pages … have a unique title” since this is used by the browser to create the bookmark. It also raises a question whether error pages (e.g., in this case a “claim not found” page) should be bookmarkable in the same way.

Now the “accept and forward order” fundamental function. The “order” in this case is the user’s request for information on a claim. If details are known of what information will be captured and how it will be stored this should be included in the test objective, e.g:

**TO7:** To show that when a user discovers the status of an existing claim, the date and time of discovery and the user’s ID are added to the claim’s audit trail file.

If these details are yet to emerge from the design and development work, a less detailed TO can be written. But this should still ensure that the feature remains in the test scope so a test for it will be created when possible, e.g.

**TO8:** To show that when a user discovers the status of an existing claim, details of that discovery are recorded appropriately

What is “appropriate” can be decided later when more design information is available.

Having made every possible effort not to miss out any important considerations that should be tested, we can now state the obvious test objective(s) arising from the original requirements statement:

**TO9:** To show that a user can discover the correct current status of an existing claim

A firm basis for confidence-increasing testing

Analysis of a typical customer-facing business site generates 100-200 test objectives of this type (sites that generate more than this have very many functions and might be better considered as multiple applications with navigation or a “portal” in common). When the method is applied - as is more often the case - to a change to an existing application, the number is typically lower.

The special circumstances and contexts under which web applications are created, operated and used have tended to hamper the efforts of testers to apply skills appropriate to other development types and so make a good contribution to web projects. As with so many other testing issues, the answer is to do the hardest and most critical work as early as possible. Using this method unlocks the benefit of VELT for web, without needing good or detailed requirements. It:

- helps to verify the requirements information that is available and detect potentially severe defects before they cause damage

- completes understanding and documentation of the true requirements, so that important properties of the application or change are not omitted from testing

- focuses effort throughout the rest of the lifecycle on those properties of the site that relate directly to its business objectives and are most likely to impact on its business success.

Edward Bishop is the current editor of Professional Tester and author of many training courses in web testing.
DUTCH TESTING CONFERENCE
www.DutchTestingConference.nl

21 April 2010 - Spant!, Bussum
Details available on the website www.DutchTestingConference.nl