iv&v australia The independent software testing specialists

Welcome to IV&V Australia Testing Newsletter

This e-newsletter provides a practitioner's view of how to manage and perform SOFTWARE TESTING in today's world.

IN THIS ISSUE – September 2010

1. *Filling the Requirements Gap – Use Generic Test Cases*
2. *Upcoming Courses @ IV&V*
3. *Alice's Adventures*
4. *Thought of the day*

FILLING THE REQUIREMENTS GAP – USE GENERIC TEST CASES

One of the trickiest tasks facing software testers is to identify the full extent of the tests that they should do. They are responsible for fully testing the system, however testing against the requirement specification alone is often not sufficient to design effective tests that ensure that all functionality has been correctly implemented and that the integrated system is fit for its intended purpose.

REQUIREMENTS ARE NOT EVERYTHING

Most projects understand that they need to define their product requirements. However, it is extremely common for the requirements to be poorly written and incomplete. Requirements can be "missing" from a specification for a number of reasons:

- They are hard to define and quantify. These include issues such as reliability, availability, and user friendliness.
- They relate to basic system behaviour and failure. These are the "unspoken requirements" that are perceived as so rudimentary that the author does not think to include them in the specification. This would include functionality such as conformance to user interface conventions.
- Uncodified knowledge. This is the functionality that the author knows from experience must be included in the system, however they do not feel that it is an effective use of their time to document. A common example of this is where a series of products are defined by means of "deltas" from a previous core product, but where the original product was never properly specified.

The commonality between these types of "missing" requirements is that they tend to relate to typical patterns of behaviour that are common to systems of a

particular type. They are seen as "generic" issues and so are given less focus when writing the specification.

To design effective tests, testers also need to be aware of a number of technical issues associated with the system under test, such as common failure modes for applications of this type, design constraints for the system, and industry norms. The tests should be designed to exercise the functionality with these issues in mind.

UNCOVER THE FOLKLORE

As a tester, it is (theoretically) not your job to gather the product requirements. In reality, however, most testers will admit to spending a considerable amount of time doing so. We gather the requirements from a variety of sources, in the hope that we will gain sufficient insight into the functionality and system operation to produce good tests.

One of the best ways to uncover this information is to draw upon the domain expertise of the subject matter experts who can be found in most organisations. These "gurus" are the keepers of the folklore, who have extensive product knowledge, having lived through previous development iterations or the development of similar products. They have seen the ways in which similar products have failed during development and in the field, and they have first-hand knowledge of the "gotchas" that lie within.

BUILD A REUSABLE TEST KNOWLEDGEBASE

The information you collect will lead you to an understanding of the general operation of the system and the ways in which it might fail. Much of it will probably also be applicable to other similar systems and application types.

Therefore, having made the effort to capture this information, you should make sure to document it in a reusable form, in say, a set of Test Checklists. By creating a knowledgebase of test design patterns, you will gain two main benefits:

- Increase the effectiveness of testers who may have less subject matter expertise
- Speed up the test design process for subsequent similar projects.

Examples of test types include User Interface Tests, Load Tests, Reliability/robustness Tests, Data Migration Tests, Interface Tests, etc.

For each Test Checklist, define the *Approach* to doing the test and the set of generic *Test Cases*:

- The Approach section should identify:
    - When the test type is applicable
    - The rationale for doing the test
    - Common environment issues and pitfalls associated with the test.
- The Test Cases should be a series of short objectives that represent a grab-bag of issues to consider, places to look, data sets to use, error conditions to try, etc when testing the application.

Once developed, the generic Test Cases will form a valuable part of a tester's toolkit of strategies and techniques. They can help to bridge the gap that is

created by missing requirements, as well as enable testers to effectively target potential problem areas. This testing knowledgebase will mean that the testers are less dependent on the development "gurus" for technical input if this does not occur.

Remember that you can ALWAYS test, even if the requirements are poorly written and incomplete, using this technique.

Upcoming Courses @ IV&V

We have a range of courses of the coming months in the following locations:

Sydney

- Introduction to Scripting: 21st October 2010
- End to End Software Testing: 8th – 9th December 2010
- Test and Governance: TBA

Canberra

- End to End Software Testing: 29th – 30th September 2010
- Introduction to Scripting: 26th October 2010

Melbourne

- End to End Software Testing: 13th - 14th October 2010

For more information on each of these courses feel free to contact us at info@ivvaust.com.au

What is included? All catering (arrival tea/coffee, lunch, morning/afternoon tea), course notes and interactive examples.

Alice's Adventures

Chapter 6 – Testability

Max's Dev Team and Alice's Test Team had partnered up and were working towards testing the first release of the software. Their problem was that the application itself hadn't been finished yet - the functionality was incomplete. How do the testers get in there to test it, if it isn't finished? Alice knew that they needed to test the system in a controllable and compartmentalized way as early as possible, to reduce the risks of finding tricky bugs later on. Unfortunately, the system design hadn't taken this into consideration.

That meant they had to somehow test the incomplete software with test harnesses at this first pass. The tricky part was that to be really effective, the test harnesses needed to be a close enough replica of the final version of the real software. As their analysis continued, the testers realised that they also needed some "test points" in the software to allow them to test it properly and see the results. None of them had thought of this aspect when finishing off the requirements! "What we really need, is a diagnostic plug", thought Alice.

The developers were a bit steamed up that, once again, they would need to change the design. Alice and Max realised they had a mini revolution to calm down! They both knew that changes needed to be made, otherwise the software could be potentially left with undetected bugs deep down. They also agreed it was too late to change the design, and so another way had to be devised.

Working with the buddy system, they decided that the only way to go was to have the developers cover off some of the requirements in developer level testing, where they were visible and testable, with the testers independently verifying that everything was correct as an observer. Effectively, the developers would act as an impromptu "window to the system internals".

Alice and Max made a note in their planning to ensure that they didn't forget this aspect on their next project – to be most effective, system testability needs to be considered during the initial design phase, not added in later. That said, they felt justifiably pleased that they'd dodged the bullet this time and worked together to find a solution!

Thought of the day

If your software works, thank a tester.

FEEDBACK

Have you found this issue useful? We want to hear your comments and suggestions. Email us at info@ivvaust.com.au.
For more information about IV&V Australia, visit our web site at http://www.ivvaust.com.au.

If you do not want to receive further correspondence, please respond to subscribe@ivvaust.com.au with "unsubscribe" in the subject line.