

The Things People Say About Testing

Fabian Musci

EuroCyber Air Technologies Pty/Ltd

ABSTRACT

The activities and objectives of Test are often, though unconsciously, regarded as both natural and intuitive! This genuine, though mistaken, feeling of familiarity with Test leads many organisations and individuals to hold beliefs about Test, which are neither true nor logical. Generally these beliefs are of no consequence to the responsible Test manager. However, in a time of crisis important, but "test naive", stakeholders may begin to question the function, scope, and methods of Test. Under these circumstances, the Test manager will need to reply satisfactorily to "the things people say about testing".

INTRODUCTION

Test activities are regarded as intuitive by many of its non-practitioners! This is probably because, unlike the development activities, our everyday lives are laced with a kind of "test phase". We unwittingly draw parallels to the function and method for Test in an R&D environment with such common activities as, testing the brakes on the car, or testing the video recorder works properly. The perceived familiarity encourages all manner belief and conjecture. For the most part, the mistaken beliefs are of no consequence, it is sorted out during the creation and review of the Test Plan. However, we often enter the test phase with deadlines fast approaching (if not already passed), the client's goodwill gone, and most the money spent. At this time of desperation and urgency, there will be a desire to save time in anyway possible. Stake holders such as; executives, accountants, and non-technical managers will be drawn into the discussions. Test will not escape scrutiny. Test Plans prepared long ago and in tranquil times will be questioned, and there will be pressure to revise them in accordance with the contingent need of each of the stakeholders. Under these circumstances the Test manager will encounter the misplaced beliefs of many important but "Test ingenuous" people. The Test manager will need to defend the Test process from many and wide ranging perceptions and misbeliefs. The Test manager will need all his aplomb and diplomatic skill to, soothe, assert and see it through.

SUBMISSION

"Our testing should guarantee quality!"

Why it's said: The Test manager will hear this said by some of the stakeholders as a kind of guarantee they make, or have made, to the client.

Why it should not be said: The truth is, you can only build quality into a product, and you can not test quality into a product! At the end of a Test iteration all Test can do is to tell you how good your process was at implementing requirements correctly? The output of the test phase can be used to "fix" the product assuming the process is of sufficient quality to introduce fewer errors that are "fixed" (this is by no means always so). So, unless there is a process in place that will transform the Test results into revision and maintenance of the product, no amount of testing will increase the quality of your product.

"Test will help the development "

Why it's said: When the implementation of some functionality is proving difficult, there is a belief that Test can assist in the development by providing detailed analysis or collection of data logs.

Why it should not be said: While collection and feed back can certainly be performed, it is an inefficient use of the Test resources. The "debug activity" is better suited to the developers. The developers are better able to improvise the " debug code and private tests " and get to the root of the problem quicker and cheaper. The bonus is that in the mean time Test has continued as scheduled.

Occasionally, there exists a general belief that Test is part of the "debug" effort. In such circumstances, the Test Manager should explain that Test is an audit of the development processes not its assistant.

"We can deliver once testing is over"

Why it's said: The frustration of being stuck in a long test phase often cause people to focus on Test as being the impediment to progress. The expanded version of the statement would be "We can deliver when test verifies the product is working to specification"

Why it should not be said: The reality is that unless you are otherwise contractually bound you can deliver at any time. Delivery is a management (commercial) decision. The Test department's role is to quantify if the item under test has reached an acceptable level of compliance as stated in the Test Plan. Pre-planned, well-defined gates, levels, and triggers should exist and must be attained before Test can recommend delivery of the item to the next phase. Change to acceptance criteria is permissible. However all variation should be well documented.

"We can't test everything, we don't have the time! But just to be sure, test this again...!"

Why it's said: People who are directly in contact with the client are likely to insist on quick delivery. They will argue that the time to market is more important than thorough testing. On the other hand they also realise how disastrous a bad day can be for the product. Their compromise view becomes one of testing only that which they fear will fail. Usually, that "something or other" is a functionality that failed in the presence of the client or the company director! As a result the test department is asked to verify that item of functionality that had the high profile error.

Why it should not be said: This is not a good strategy for test but it is often the one adopted. It is again an inefficient use of Test time. High profile fails will attract a lot of Test, usually various stakeholders will appear in the lab and ask for ad-hoc and undocumented tests to be performed. These low quality tests will invariably take up a lot of time produce unclear results that will encourage more ad-hoc and informal tests.

Criteria should be established at an early part of the phase. Senior managers should keep their cool and believe in their own judgment. Regression test should be scheduled and executed. An attempt should be made to improve the process that created the error rather than spending inordinate effort trying to detect it.

"That's a lot of work!"

Why it's said: At some stage the Test manager will be asked what is required to ensure Tests success. The test manager (an intrepid soul) will detail the "minimum success factors". High on the agenda will be clear, explicit and controlled requirements. What the Test manager will hear said is "If we do all that, the documents will be huge ... that's a lot of work!"

Why it should not be said: Test is often the first to be exposed to the uncertainty of the development phase. As Test planning and design begin the inconsistencies of requirements, the shortcomings of configurations management (usually via the poor state of the requirements and

design documents) become exposed. Test is the first to see if requirements are ambiguous or contradictory, if the standard definitions are inadequate or non-existent.

The Test department needs to be involved in the definition and codification of requirements. Test is a stakeholder in the process it can provide early warning. The fact is that "a lot of work" needs to be done. And, one way or another it will be done! If system analysts fail to make explicit the requirements then Test will make assumptions. Test may informally ask developers and managers to "please explain" but the informal verbiage may just be assumptions. If the assumptions are correct, fine! But if the assumptions are wrong no one will know until a test fails. Sorting out which of the requirements, the tests, the assumptions or the implementation was incorrect will also be a lot of work. The "lot of work" shall be done, you can do it at the start, when the cost is low, or you can wait to fix it after implementation, when the costs are high!

"We have been stuck in the test cycle for a while now!"

Why it's said: The CEO of a company implementing an automatic toll way billing system was asked why the opening of the toll way had been postponed for the fifth time? He replied that everything was fine but that the delays were because of some problems in Testing!

Why it should not be said: The fact that testing was discovering the system didn't meet requirements means that test had done its job well! The real problem is that something had gone wrong either during requirement capture or in implementation of those requirements.

The possible solution is to identify and prioritise the critical areas of deliverable functionality.

Cut the list of test such that, only the identified functionality is tested. Test it! When it passes deliver it! Then promise you'll deliver the rest later!

"It's a small change and won't effect testing."

Why it's said: During field trials, customer demonstrations, or as part of staggered deliveries, the organisation is keen to show off the product's functionality. Development comes under pressure to include highly visible functionality that is either missing or has not worked correctly in previous demonstrations. From the marketing perspective the inclusion and successful demonstration of "small embellishments" appears worthwhile and of great benefit to the product's position in the marketplace. The desire for timely release is propagated to the various project managers who are often reminded of the cut throat nature of the marketplace. Invocations of pragmatism and commercial reality are hard to ignore. The risk involved with the addition of a few small items appears out weighed by the benefits.

Why it should not be said: The decision not to Test is ultimately a business management decision. However, technical manager's best make the assessment of that risk. Test is in the position to quantitatively measure risk. The pragmatic engineering reality is that the alteration of any component will require the retesting of that component and all those functions that make use of, or traverse the component. Any change to the system incurs a testing cost!

"Why wasn't it found before?"

Why it's said: In the emergency meeting to discuss the appearance, or reappearance, of a "bug", the phrase "Why wasn't it found before?" is often heard. The disbelief may be particularly stinging when that component is a reuse module or found in the released product.

Why it should not be said: With the growing complexity of most commercial systems it has become impossible to test the entire input range let alone the combinatorial aspect of those inputs.

A bug found is a bug defeated. The discovery of an unforeseen error (unforeseen by test scenarios) should result in the revision and inclusions of new Test case(s) at the level at which the error is detected. A proper process should allow the continuous improvement of Test procedures and should include reviews of those tests by developers and designers.

"We need to test that 'such and such' wont happen"

Why it's said: When a particularly disastrous scenario is brought to light there will be some pressure to ensure that such a disaster cannot happen in the system under development. To that end Test is asked to produce and execute a test or tests.

Why it should not be said: Notwithstanding the comments in a previous section, great caution should be exercised when attempting to verify the absence of disaster.

The requirements pertaining to the functionality in question should explicitly state the method by which events are handled. If such requirements do not exist it is highly likely that safeguards have not been included in either design or implementation. Whatever the case, if developers or designers can provide specific states and values that will elicit the error condition then, test(s) can certainly be devised, and should be included. However, in the absence of firm data, the design of tests will be extremely difficult. The natural tendency may be to overcompensate and create a large suite of tests based on conjecture and speculation that are aimed at the impractical, if not impossible verification of absence of error.

Glossary

Bugs: Non conformances. "Bug(s)" is not to be confused with "unexpected outcome".

Codification: Putting good ideas down on paper.

Debug activity: Informal and ad hoc activity carried out by developers in an effort to collect data on implementation non-conformance.

Debug Code: Informal and ad hoc code stubs and harness used to assist developers to locate implementation non-conformance.

Private Tests: Informal testing carried out by developers to search and investigate implementation and design ideas.

Regression: The Tests re-executed to verify the changes made to item under test have fixed any previously reported problem or verify the addition of functionality has not introduced new errors.

Requirement: Single imperative and explicit statement that will have a direct trace to a point of verification in a test procedure.

Specification: The collation of requirements into a cohesive and non-contradictory document.

ABOUT THE AUTHOR

Author's name:	Fabian Musci
Business affiliation:	EuroCyber Air Technologies Pty/Ltd
Address:	PoBox 443, KEW, Victoria 3101
Phone number and fax:	Phone: +61 3 0419137132 Fax: +61 3 98539502
e-mail:	fabian@eurocyber.com.au

Fabian Musci is a Senior Consultant with EuroCyber Air Technologies. Fabian received his training in Melbourne Australia attaining BAppSc in 85 and going on to complete Graduate diplomas in Computer Science and Digital Systems Engineering. Fabian began his professional career working in medical research. He is a published scientist in the area of diagnostic neurology. He turned his attention to Software in the late eighties. Since then he has worked in a variety of large engineering projects, including safety critical, aerospace and military application.

Fabian is a member of the Systems Engineering Society Of Australia (SESA) which is a chapter of the International Council of Systems Engineering (INCOSE).