

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 perspec-

tive 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