MAXIMISE THE VALUE OF TESTING – FOCUS ON THE FEEDBACK LOOP

In many organisations the independent testers are little more than gatekeepers. They don't see software for test until shortly before it is scheduled for release, and although they might identify various types of defects, there is usually no time for fixing anything other than "showstoppers".

This sort of testing isn't really helping to bring quality to the product – it is doing nothing more than identifying what needs to be done to avoid disaster on release. Working in this environment, the testers come to be seen as little more than bearers of bad news.

For testing to have a real impact on the quality of the product, it must be part of an effective f*eedback loop*. The testers must identify quality issues and feed them back to the developer, and the developers must act on them in a timely manner. With an effective feedback loop the testers become an integral part of improving the quality of the product.

ENABLING THE FEEDBACK LOOP

An effective feedback loop can only exist if it is built into the process and culture of the development project. This requires a number of key things to occur:

- Software must be delivered to the testers early enough for there to be time to act on the test results. As there is rarely enough time "at the end", this means that software must be released to test in an incremental fashion. The increments should ideally provide "vertical slices" of functionality, as this maximises the efficiency of independent testing.
- Testing must start early in the implementation phase. If the developers don't get any feedback until late in the process, there is too much "investment" in the code to make it practical to address pervasive issues.
- Test areas of functionality which are critical, risky, or particularly difficult first. This maximises the time available to the developers to address the difficult issues.
- The test-and-report cycle must be reasonably quick. If the feedback is too slow, the developers are likely to have moved on to other parts of the system, and so are likely to be slow to act on any problems found. It is also easy to fall into a "version disconnect" – if the testers are making reports against software which is several versions old, it is easy for the developers to ignore them as "probably already fixed".
- The developers must be prepared to seriously consider all problems reported by the testers. In some projects the developers adopt a "we know best" attitude that effectively prevents feedback from occurring – the reports are made, but no serious attempt is made to act on them.

MANAGING THE FEEDBACK LOOP

In order to be effective, the feedback loop needs to be managed.

Some projects get by with informal methods such as free-form emails and verbal communication, but this rarely works except for simple developments with very small teams. Beyond that, some sort of Defect Management system becomes vital.

A defect management system doesn't need to be complicated or expensive.

In small projects a simple spreadsheet accessible to all team members can work just fine. If you are producing a custom system, don't get carried away will bells and whistles – you just need something that is:

- Easy to enter new reports – too hard and they won't be reported;
- Makes the status of all problems clear – so that they don't just get forgotten about; and
- Provides simple summary reports – so that the process can be easily monitored.

For larger projects simple spreadsheets or databases become Impractical and an “industrial strength” tool is required.

It is important that the Defect Management system is visible to everyone on the team. A good way to do this is to use a web-enabled tool and put it on your intranet.

MONITORING THE FEEDBACK LOOP

Once you have an effective feedback loop operating, it can become an extremely powerful source of information. With very little overhead it can provide some very useful metrics:

- Cumulative number of problems raised over time. Although this is a difficult metric to interpret on its own (Is a low rate good? It might be due to good software, but it could equally be due to bad testing!) the shape of the curve says a lot. You should be looking for the curve to flatten well before release, indicating that fewer new problems are being found. If its still going up at a steady rate there are still many more problems to be found.
- Cumulative number of problems closed over time. The gap between problems raised and problems closed should be narrowing in the lead up to release and should be close to zero at release. If it is still widening, you are not ready yet!
- Average elapsed time to fix problems. If this is long, then perhaps the developers are not spending enough time on fixing problems (maybe they have been re-assigned to other tasks),

  and the feedback loop will break down. Alternatively, it may indicate that there are serious design problems making fixes very difficult.
- Fix effectiveness (ie proportion of fixes passed on retest). This should be close to 100%. If not, the feedback loop is falling down – maybe more information is required from the testers, or maybe the developers are not paying serious attention to the full impact of the problems that are being seen. This could also indicate serious design problems making bug fixing very difficult.

CONCLUSION

Without a feedback loop, the testers are merely helping to minimise the risk of disaster. With it they can be an integral part of building quality into the product.

Focus on the feedback loop. Enable it, manage it, and monitor it and you will have a powerful mechanism for improving your software quality.

