RISKS TO RELEASE

The decision to "go live" with software is often a difficult one, particularly for applications that are time-to-market critical. On one hand, organisations want to deliver a product that has a level of quality that will satisfy their customers; on the other hand they may need to beat their competitors to the market or meet an immovable deadline.

In these instances it is often best to provide senior management with a report on the technical "Risks to Release", and allow them to decide if these risks are acceptable within the context of the business risks.

What the senior management *really* want to know is what problems will occur after release if released now. Unfortunately no amount of prior testing can totally answer this question. However some metrics are good predictors of it -- of particular value are measures in three key areas:

- Quality,
- Coverage, and
- Stability.

QUALITY MEASURES

The quality of software products can be quantified in terms of the number of defects that have been identified. To assist in painting a picture of the risks to release, the following should be reported:

- **The total number of open defects.** Open defects are known defects that have been discovered but have not yet been corrected.
- **The number of open defects by severity level**. Senior management are always most interested in the number of open *critical* defects in a system, particularly when they are considering their risk to release.

These quality measures need to be considered together with coverage measures to get a complete picture of the risk. You may appear to have a good quality product (ie few open defects), but if you have a low test coverage of the requirements, there are likely to be significant numbers of hidden defects in the untested functionality. On the other hand, if the coverage is high, then the quality seen in the tests done so far is probably a good predictor of the overall quality – with few surprises.

These metrics can be normalised to the size of the application and the test coverage, so that comparisons can be made with other applications. Normalised metrics are often presented as defects per function point or defects per SLOC (Source Line of Code).

COVERAGE MEASURES

Coverage metrics relate to how completely the functionality is being tested. To establish coverage metrics, you first need to do a gap analysis between the test cases and the requirements, to ensure that all functionality is adequately covered by tests (assuming at this point that you will have enough time to execute all tests).

During the test execution activity, you can report:

- **% of functionality that has not been tested**. To achieve this you must have a good understanding of the total number of test cases that would be required to get coverage of all functionality.
- **% of test cases executed per priority level**. Test cases should be prioritised by at least three levels: high, medium and low.

  In your test plan, you can establish minimum acceptable levels of test execution coverage for each priority level. For example, you may require that 100% of high priority test cases are executed, but only 75% of medium priority test cases, and so on.

For each priority of test cases, calculate the % of test cases executed as the number of test cases executed divided by the total number of test cases. Note that test cases executed includes both test cases that pass and test cases that fail.

If these figures and priority levels are agreed by senior management, then the coverage risk to release is easily reported. Providing a report of the progression through these priority levels will highlight the risk to release at any point during the test execution activity.

It is important to also remember to test the more critical (higher priority) functionality before the less critical (lower priority) functionality just in case you run out of time and can't run them all.

As a first assumption, a test coverage of n% of functionality means that you have probably only seen n% of the defects - so be aware that a low coverage means you are probably significantly underestimating the total number of defects.

You also need to consider stability measures when assessing overall quality. You might have achieved good test coverage, but if each new software build introduces more defects than it closes, the software is unstable and there is a high risk that you will eventually release a defect-ridden, unmaintainable product.

STABILITY MEASURES

Stability metrics relate to how well the product has converged toward a well-defined release candidate. An unstable product is one where the correction of software defects introduces more problems than it fixes, and/or where there is a high rate of rejection of defects corrections in subsequent regression testing. To measure stability risk to release, defect-related trends need to be identified:

- **The number of outstanding defects over time**. A stable product is one where the number of outstanding defects is under control and reducing over time. That is, the rate of defects being closed exceeds the rate at which defects are being raised.

  This information should be reported in graphical form with the vertical axis identifying the number of defects and horizontal axis showing time. Two plots should be made on the graph - one for the total number of **defects raised** (over time), and one for the total number of **defects closed** (over time). The aim is to see the defects raised plot levelling off as regression tests are run but no more defects are being found, and the defects closed plot rising up to meet the defects raised plot.

  These can be further detailed to show the number of critical defects (priority 1 and 2 defects) raised and closed over time.

- **Defect closure efficiency**. Defects closed are those that have been confirmed as being fixed by re-testing. This information has two elements – how efficient are the developers in actually correcting the defects (ie not rejected during retest as not really fixed) and how efficient are the testers in retesting the fix.

  The **defect correcting effectiveness** can be calculated using the formula:

  Defect correcting effectiveness = (no. defects confirmed closed)/(no. defects confirmed closed + no. defect corrections rejected)

  A stable product is where this ratio approaches 1 (i.e. the number of defect corrections that are rejected is very low).

  There is sometimes a delay in verifying closure of defects – the fix has been made but has not yet been retested. The **defect retest efficiency** is the number of defects that the developers claimed that they've fixed but the testers have not retested yet, as well as the average length of time between receiving notice of the fix and rerunning the test.

CONCLUSIONS

Key measures of Software Quality, Test Coverage and Software Stability can provide senior management with an indicator of the likelihood of problems after release, and therefore help to define their technical risks to release. Simple testing metrics can support these measures and provide an objective view of this risk.