How Much Is Too Much? (HMTM)

(Disclaimer: The content in the paper has contextual importance. Reader or user's due care of context before using it is necessary)

"How much is too much?" a routinely bothering question for business and IT as far as software testing is concern. Theoretical possibilities of exhaustive testing are impractical under real life scenario. It is very usual to interface a situation where management is not ready to move go live deadline and testing time has been eaten up by development team or software vendor. Its therefore very important to know how much testing one should do on software so as to go live with minimum risk to business. How to save oneself from those too much testing resulting into no more value additions? Let me highlight few techniques which we have used to decide "how much is sufficient or adequate testing?"

(1) Risk based approach

Instead of looking at software defects from technical perspective, if one look at the same from business risks perspective, it would give better focus to energy going in for testing. Risk parameters can be decided as per organization's priority. Organization can also decide about standardization of coverage percentage based on risk. Following table gives very generic example of risk based test coverage.

Module / Features	Risk	Minimum Test Coverage
Financial calculations / information	High	100% of test cases
Operational efficiency	Moderate	60% of test cases
User interface	Low	40% of test cases

The risk ranking and parameter may differ from situation to situation. However, one can be fairly confident about detecting high risk items before going live if risk based approach is adopted.

(2) Confidence measurement model

The second method that we use in certain circumstances is Confidence Level measurement. The concept is picked up from Financial Audit industry where Confidence level is used for calculating sample size. Confidence level and sample size has inverse relationship. Hence while testing we keep feeding data into confidence level model which in turn throws up confidence in the system. One can decide bench mark confidence level to decide about when to stop testing. Objective Parameters that goes into confidence level calculation may includes number of test cases, tested test cases, risk assignment, failure percentage, Type of failures, success ratio of fixes, and repetitive nature of failures. After arriving at the objective number, the project manager applies subjective discount on the number and can arrives at confidence level.

(3) Defect detection trend

Observing defect detection trend over time scale can reveal when to stop testing. Reducing trend and severity of errors can provide sufficient hints at when to stop testing. However, trend should be used in conjunction with risk to arrive at more accurate predictions.

(4) Creation of minimum test sets (MTS)

MTS method generally works in regression testing. MTS helps in ensuring that minimum business functions which are needed for running regular business has not been affected due to changes in the system. MTS plus Test cases sorted based on risk based model for new modules or changes should give a fairly adequate sample size for regression testing

(5) Checklists instead of test cases

This technique is more to do with efficiency improvement of testing rather than adequacy. However, since adequacy and efficiency both are linked to same problem “No time for left for testing”, I thought of touching it here. The testing related to GUI, formats, standards etc… can be converted into checklist instead of writing test cases for the same.

(6) Automation of routine testing

Automation is also efficiency improvement related technique and is covered here for the same reason as described for checklist. Automating risk based test cases and MTS would give fairly good time saving and high confidence level for going live with the software.

Adoption of above technique may give answer to the question “How much is too much?” and “How much is adequate?”

If any of the reader has adopted any other model and has been benefited, please share the same on my regular email id madhav.bhadra@aqmtechnologies.com.

Author is director at AUDITime group and Chartered Accountant and Certified Information System Auditor with 20 years of experience in testing / auditing financial software applications. AQM Technologies is software quality management specialist with 17 years of experience in capital market, banking, insurance and logistic industry. Any further interest or inquiry for service can be routed to madhav.bhadra@aqmtechnologies.com or sales@aqmtechnologies.com.