

The Need for a Test Process Maturity Model

Gerrard Consulting has been conducting test process improvement projects since 1991. To help us to improve our clients' test practices by focusing on what is most important, we continue to refine our approach to test process improvement. All process improvement methods require an initial assessment of current practices and this is used to measure the current capability, identify shortcomings and guide the improvement process. For several years we have been seeking a process model that helped us to assess an organisation's testing maturity objectively, and which could be used to identify a set of appropriate improvements.

The Capability Maturity Model (CMM) is the best known Software Engineering process model. It has as its foundation an incremental set of maturity levels. Each level consists of a set of Software Engineering practices that an organisation must use for them to reach that maturity level. To reach the next level of maturity, the organisation must implement the practices identified as part of the next level and so on. The CMM attempts to provide a sequenced set of process improvements to reach the ultimate process capability.

The CMM (and related models) have been found lacking when it comes to testing practices. The detail presented in these models is sparse, to say the least. Attempts have been made to refine the CMM as well as to come up with alternative testing-specific models. We have found that even these models did not match the way we conducted testing improvement projects. These models are based on assessments that identify whether certain good practices are used or not, and present a staged sequence of process improvements. The recommended process improvements consist of the good practices not currently being adopted. The assumption is that these practices will increase testing effectiveness and improve software quality.

Remedy-Based Models are Inadequate

Several problems with such approaches (particularly the CMM) have been documented but we would emphasise one in particular. We believe that these models are all solution or 'remedy-based' and miss the point. Consider what might happen, if a doctor adopted a remedy-based diagnosis process. If you had a headache, and the doctor asked you a series of questions relating to possible remedies, this would probably perplex you: 'are you taking aspirin?', 'are you taking penicillin?'... These questions are not related to the problem and would be very unsatisfactory, unless of course, you were a hypochondriac and wanted to take a lot of pills.

Process assessments that are remedy-based are also unsatisfactory. Most organisations wishing to improve their test practices have one or more specific problems they wish to solve. E.g. 'testing costs too much', 'our testing isn't effective enough', 'testing takes too long'. Answering NO to questions such as, 'do you conduct inspections?', 'do you use a tool?', 'are incidents logged?' should not mean that inspections, tools and incident logging are automatically the best things to do next. The remedies recommended may be based on the sequencing of practices in the model, *not because it will help the organisation solve its software development problems.*

We fear that many organisations use remedy-oriented approaches blindly. Assuming that an organisation's problems can be solved by adopting the 'next level' practices may be dangerous. The cost or difficulty in adopting new practices may outweigh the marginal benefit of using them. For example, an organisation might use 80% of CMM level 2 practices and 60% of level 3 practices, but would not be assessed at a level higher than level 1. If the organisation adopted the last 20% of level 2 practices would they automatically benefit? There might be some benefit, but it is more likely that those practices are not adopted because the benefits are marginal or negative at this time.

We believe that process improvement methods that use remedy-based approaches are inadequate because they do not take existing problems, objectives and constraints into consideration.

Process Models and Process Improvements

In our experience, the major barriers to improved practices are organisational, not technical. Most of the difficulties in the implementation of improved practices are associated with changing management perceptions, overcoming people's natural resistance to change and implementing workable processes and management controls.

For example, management may say 'testing takes too long' and believe that an automated tool can help. Buying a tool without further analysis of the problems would probably waste more time than it saves: time is spent getting the tool to work, the tool doesn't deliver the benefits promised, so the situation is made worse and the tool would end up as shelfware.

The underlying issue to be addressed is most likely due to a combination of problems. Management doesn't understand the objectives of testing; the cost of testing is high but difficult to pin down; developers, testers, users may never have been trained in testing; the quality of the product delivered into testing is poor, so takes forever to get right. To address the management problem, a mix of improvements is most likely to be required: management awareness training; testing training; improved definition of the test stages and their objectives; measurement of the quality of the product at each stage etc. etc.

We believe that the assessment model must take account of the fact that not all improvements are a good idea straight away. Some improvements are expensive; some save time, but the changes to the way people work may be dramatic; some improve the quality of the testing, but take longer to perform. Very few improvements save time, improve quality, cause minimal change and pay back after two weeks. Recommended improvements must take account of other objectives, constraints and priorities.

The Test Organisation Maturity Model (TOM™)

Gerrard Consulting has developed a Test Organisation Model, TOM™ to address the primary concern that the outcome of the assessment should address the problems being experienced. The assessment process is based on a relatively simple questionnaire that can be completed and a TOM™ score derived without the assistance of a consultant.

The questionnaire appears on the following pages and is governed by the following:

- The questions focus on organisational rather than technical issues and the answers, in most cases, can be provided by management or practitioners (try both and compare).
- The number of questions asked is small (twenty).
- The objectives of the organisation assessed should be taken into consideration and prioritised. (Do we want to get better, or do we want to save money?)
- Questions relate directly to the symptoms, not remedies. (What's going wrong, now?)
- Symptoms are prioritised. (Release decisions are made on 'gut feel' and that's bad, but we are *more* concerned that our sub-system testing is poor).
- The scoring system is simple. All scores and priorities are rated from one to five.

The Improvement Model

A potential process improvement may help to solve several problems. The improvement model is a simple scoring/weighting calculation that prioritises potential improvements, based on the assessment scores and priorities. The model has a library of 83 potential testing improvements. For each symptom, a selection of improvements has been deemed most appropriate, and weighted against the objectives and constraints.

When the questionnaire is completed, the scores are entered, and a prioritised action list of potential process improvements is generated.

How the TOM™ Questionnaire is used

When completed, the questionnaire can be used to calculate a TOM™ level. Since you must answer twenty questions with scores of 1-5, you can score a minimum of 20 and a maximum of 100. If you repeat the questionnaire after a period, you can track progress (or regress) in each area. If you send the completed questionnaire to us, we will enter the data into our TOM™ database. We are using assessment data to survey testing practices across industry. The database also has a built-in improvement model. Based on the assessment data entered, the model generates up to seventy prioritised improvement suggestions. You can use these to identify the improvements that are likely to give the most benefits to your organisation.

Completing the Questionnaire

The questionnaire has four parts. Parts one and two are for administrative and analysis purposes. Parts three and four are used to calculate a TOM™ level and, and then analysed in the testing improvement model to generate prioritised improvement suggestions.

You may complete a questionnaire for your entire organisation or for a particular project. Please copy this form and use it for multiple assessments. (You might consider giving the form to both managers and practitioners for comparison).

Part 1 - About Your Organisation

We use this information to analyse the assessment data we have collected across industry.

Part 2 - About You (the Assessor)

When we have analysed the data on your questionnaire, we can contact you easily.

Part 3 - Your Objectives and Constraints

You may have overriding objectives or constraints that will influence which improvements will help you most. For example, some improvements improve quality, but may be expensive to implement. Rate the objectives and constraints in the range one to five. You can give the different objectives the same weightings e.g. 5, 3, 5, 2, 2 or grade them 5, 1, 3, 4, 2

Part 4 - Your Symptoms

This part asks questions about twenty symptoms of poor test process maturity. The questions must be answered with a score (1-5) and a priority (1-5).

For each numbered question, there are three examples of maturity. If one of the three columns headed Low, Medium or High resembles your situation, score a 1, 3 or 5 respectively. If you are 'in between' score 2 or 4. You *must* score each question between 1 and 5. If you score less than 5 for a symptom, you should assign a priority to that symptom. The priority indicates how much it 'hurts'. A priority 1 symptom can be ignored. A priority 5 is extremely painful and must be addressed.

What Next?

Add up the individual scores for each of the symptoms. The result is a TOM™ maturity level.

Mail completed questionnaires for the attention of Paul Gerrard at the address below or enter your questionnaire results ON-LINE at our web site: www.gerrardconsulting.com.

We will email or post your completed assessment form and a prioritised list of up to seventy potential test process improvements to you FREE.

Information provided on these questionnaires is treated in the strictest confidence.

Part 1 - About Your Organisation

O1	Organisation Name															
O2	Address															
O3	What is your organisation's main line of business? (underline one)	<table> <tr> <td>Computing</td> <td>Retail</td> </tr> <tr> <td>Defence</td> <td>Transport</td> </tr> <tr> <td>Financial</td> <td>Consultancy</td> </tr> <tr> <td>Health</td> <td>Government</td> </tr> <tr> <td>Manufacturing</td> <td>Telecommunications</td> </tr> <tr> <td>Media</td> <td>Mining/Construction/Oil</td> </tr> <tr> <td colspan="2">Other (please specify)</td> </tr> </table>	Computing	Retail	Defence	Transport	Financial	Consultancy	Health	Government	Manufacturing	Telecommunications	Media	Mining/Construction/Oil	Other (please specify)	
Computing	Retail															
Defence	Transport															
Financial	Consultancy															
Health	Government															
Manufacturing	Telecommunications															
Media	Mining/Construction/Oil															
Other (please specify)																
O4	How many staff do you employ?															
O5	How many developers do you employ?															
O6	How many testers do you employ?															
O7	What is your annual turnover?	<table> <tr> <td>Less than £1m</td> <td>£11-100m</td> </tr> <tr> <td>£1-5m</td> <td>More than £100m</td> </tr> <tr> <td>£6-10m</td> <td>N/A</td> </tr> </table>	Less than £1m	£11-100m	£1-5m	More than £100m	£6-10m	N/A								
Less than £1m	£11-100m															
£1-5m	More than £100m															
£6-10m	N/A															
O8	Do you develop on mainframes?	Yes / No														
O9	Do you develop on midrange?	Yes / No														
O10	Do you develop on client/server?	Yes / No														
O11	Do you develop web applications?	Yes / No														
O12	What development languages/tools are used?															
O13	What development methodology is used? (underline one)	<table> <tr> <td>Structured</td> <td>Rapid Application Development (RAD)</td> </tr> <tr> <td>Information Engineering</td> <td>Prototyping</td> </tr> <tr> <td>Object Oriented</td> <td>Incremental</td> </tr> <tr> <td>Evolutionary</td> <td></td> </tr> <tr> <td colspan="2">Other (please specify)</td> </tr> </table>	Structured	Rapid Application Development (RAD)	Information Engineering	Prototyping	Object Oriented	Incremental	Evolutionary		Other (please specify)					
Structured	Rapid Application Development (RAD)															
Information Engineering	Prototyping															
Object Oriented	Incremental															
Evolutionary																
Other (please specify)																

Part 2 - About You (the Assessor)

A1	Your name	
A2	Project or department	
A3	Telephone	
A4	FAX	
A5	Electronic mail	

Part 3 - Your Objectives and Constraints

	<i>Objective/Constraint</i>	<i>Priority (1 low, 5 high)</i>
C1	I want to decrease time required to test	
C2	I want to decrease cost of testing	
C3	I want to increase the quality of testing (and systems)	
C4	I want to minimise change to current practice	
C5	I want to get a quick payback	

Part 4 - Your Symptoms

	<i>Symptom Description</i>	<i>Low (score 1)</i>	<i>Medium (score 3)</i>	<i>High (score 5)</i>	<i>Score (1-poor 5-good)</i>	<i>Priority (1-low 5-high)</i>
S1	Testing has no deliverables of value	Testing deliverables are not identified, not valued, not used for anything	Test deliverables are used to make product release decisions	Test deliverables are used to make product release and process improvement decisions		
S2	Testing has little purpose	Testing is essential to achieve sign-off	Testing is essential to detect faults in software	Testing is essential to both detect and prevent faults in software		
S3	Test objectives are less important than development objectives	When development slips, testing time is squeezed to achieve deadlines	When development slips, testing time is squeezed, but extra resources are applied to achieve the deadline.	When development slips, a risk assessment is conducted and a decision to squeeze, maintain or extend test time may be made.		
S4	Testing is expensive, but the costs are not visible	Test activities are not identified or distinguished separately from development activities so costs are not visible	Only system and acceptance test activities are identified and tracked and costs recorded	Static tests (reviews, inspections etc.) and all dynamic test stages (unit, integration and system and acceptance tests) are tracked and costs recorded		
S5	Of the faults that are found, there is a perception (based on evidence) that many should have been found in earlier test stage(s).	Faults are found in acceptance tests that should have been found in sub-system and system tests.	Faults are found in system tests that should have been found in sub-system tests.	Faults found would not be expected to have been detected earlier		
S6	Users perceive that they are responsible for testing	Users compensate for poor development and system testing by staging large acceptance tests; developers and system testers regard user testing as a fallback	User and system testing is not coordinated; users insist on large scale tests	System and user testing is coordinated; users contribute to system test planning and prepare their own tests to complement system tests		

Test Organisation Maturity Questionnaire

V1.0

	Symptom Description	Low (score 1)	Medium (score 3)	High (score 5)	Score (1-poor 5-good)	Priority (1-low 5-high)
S7	Significant requirements and design defects are found by system and acceptance testers	Requirements documents are used 'for guidance only'. Acceptance testers regularly find defects so serious that they cause project slippage or force de-scoping decisions.	System testers have difficulty deriving tests from specifications. Test planning and the tests themselves reveal missing and ambiguous requirements that require late reanalysis or redesign.	System and acceptance tests reveal subtle or minor requirements or design defects. Defects found usually require re-coding, rarely reanalysis or redesign.		
S8	Lack of ownership of the test process	No one owns testing, it is assumed to be done by developers (who don't want it)	Developers, system testers and users own their test stages, but work independently	Overall approach to testing developed and agreed by consensus - developers, testers and business users and project management are involved in master test planning		
S9	Users won't buy-in to the test approach	Users only become involved when the time comes to acceptance test	Users contribute to system as well as acceptance test planning	Users are fully involved in the preparation of the test approach, system testing, user testing and have visibility of test activities		
S10	It is difficult to get volunteers to test, to hire and retain testers	Testing is perceived to be easy - anyone can do it; testing is not recognized or rewarded; training is not given; management favour developers over testers	There are some career testers, some have been trained; testing is recognized, but not rewarded	Testing is seen as a viable career path; testers are promoted to management positions and are rewarded		
S11	There are gaps in the testing - features of the system may be released untested	Tests are not based on requirements or design documents, there are no test inventories or means of measuring coverage against requirements or specifications	Test inventories are used to define the scope of system and acceptance tests and cross-reference requirements; formal test techniques are sometimes used to design black-box test cases.	Test inventories are use for all testing and are reviewed against requirements and specifications, formal test techniques are used for test case design, tools are used to measure code coverage		
S12	There are gaps in the testing - testing does not address user concerns	System testing does not take account of user needs and usage patterns, user testing is ad-hoc and disorganized	Some system testing is based on the business process, user tests cover variations in data but not business process	System tests are based on business transactions and cover business process paths, user testing focuses on usability and fit with the business process		

	<i>Symptom Description</i>	<i>Low (score 1)</i>	<i>Medium (score 3)</i>	<i>High (score 5)</i>	<i>Score (1-poor 5-good)</i>	<i>Priority (1-low 5-high)</i>
S13	There are gaps in the testing - some non-functional requirements of the system may not be tested	Non-functional requirements and concerns are not documented, non-functional tests are not performed	Some informal non-functional tests (usually performance) are performed but are squeezed and often left incomplete	Non-functional requirements are documented and used to steer testing; non-functional tests are planned early and given enough time in the project schedule to be meaningful		
S14	There is no evidence available to show that testing has been thorough	Test plans do not set measurable targets, test records do not provide an insight into thoroughness	Test inventories are used to define quantifiable targets. Test design techniques are sometimes used.	Functional test techniques are consistently used to design tests. Source code analysis tools are used to measure structural coverage.		
S15	Test budgets are unreliable because there is no rational method of estimating test activities	Test budgets are calculated as a percentage of development budgets	Test budgets are estimated, based on 'received wisdom' and experience	Test budgets are estimated based on previous project metrics and an assessment of the risks to this particular project		
S16	When testing stops, no one knows whether enough or too much testing has been done.	Testing usually stops when time runs out. Some tests may never be run.	Testing is time-limited, prioritization happens after estimates and timescales are committed to	There is a defined prioritization policy. Tests are prioritised before test plans are fixed. Tests that are de-scoped and the associated risks are identified and understood.		
S17	Decisions to release into production are based on 'gut-feel' and not an objective assessment of the quality and readiness of the software	Test records provide scant details of the stability of the features of the system. Testers argue against release, but cannot provide objective evidence.	Testers log and categorize incidents. Features of the software can be assessed in terms of defects found, deferred, fixed/re-tested and outstanding. Little or no regression testing done on final version of software.	Defects are analyzed for severity, location. Assessment of the stability of all system features can be made. Regression tests on final version of the software have been run without incident.		
S18	Test environments are rarely adequate and never ready in time	Test environments are created just before testing needs to start	Test environments are requested and built in good time, but testing time is lost through lack of control, support resources and environmental instability.	Test environments are available in time for testing, are adequately supported and testing is rarely held up through environmental problems.		

	<i>Symptom Description</i>	<i>Low (score 1)</i>	<i>Medium (score 3)</i>	<i>High (score 5)</i>	<i>Score (1-poor 5-good)</i>	<i>Priority (1-low 5-high)</i>
S19	When software is first delivered into test, there is a delay of days or weeks before the first tests pass successfully	Testers complain that early builds are incomplete, unstable and not ready for testing. Testing time is lost through unreliable builds and test environment. No tests pass first time because of 'fundamental' software defects.	Testers complain that the configuration of the test environment causes major problems. Testing time is lost through the software behaving differently in test than in development. Early tests fail because of configuration discrepancies.	Testers are able to run tests on the day the software is installed in the test environment. The software is testable, insofar as it is stable enough to be usable on day 1. Incidents raised reflect functional faults in the software and not unreliable build		
S20	Testing is not focused, systematic or effective	Testers never know whether enough testing has been performed because they rely on gut feel. Testers are uncomfortable that they may be blamed for bugs in production.	Test documentation is produced in large volume; the simplest test design techniques are sometimes used. Much effort is spent testing more complex functionality but it is never tested 'thoroughly'.	Testers focus their attention on the key risks and select test strategies specific to the risk and type of software under test. Testers produce objective and quantifiable test coverage targets and tests consistently meet them.		

Suggestions for Improvements

If you have an objective that is not listed, please let us know in the space below.

If you have a problem that is not reflected in the questions above, perhaps we need to add it to the symptoms list. Please let us know.