

Testing maturity in an agile/CDT environment

Context

The method described in this article has been developed after a request by a manager to have insight in "testing maturity". His colleague managers had maturity scores for their teams of developers and business analysts and higher management wanted this for testing as well. We saw little value in a classic five-tiered maturity model, and were intrigued by the question: "What can you do with respect to maturity models that goes beyond scoring people into a set of pre-defined levels?" With that question as a start, we created this approach to help identify possible areas for test improvement.

What is maturity?

Let's have a look at what google says:

- The quality of behaving mentally and emotionally like an adult; a very advanced or developed form or state (Cambridge Dictionary)
- In psychology, maturity is the ability to respond to the environment in an appropriate manner (Wikipedia).

Maturity has to do with being advanced or developed, and with having the ability to respond to the environment. Both of these aspects imply change and growth – with both the person and the environment responding to each other. Mature people are able to adapt and take responsibility. That is why we like the definition by James Bach: Maturity is the degree to which a system has realized its potential and adapted to its context.¹

Michael Bolton says in his blog post²: *"if maturity means the same thing for processes as for other living things, a genuinely mature process, whether for individuals or for groups, should incorporate freedom, responsibility, diversity, adaptability, and self-sufficiency. A genuinely mature process shouldn't emphasize repeatability as a virtue unto itself, but rather as something set up to foster appropriate variation, sustainability, and growth. A mature process should encourage risk-taking and mistakes while taking steps to limit the severity and consequence of the mistakes, because without making mistakes, learning isn't possible."*

Maturity models

This is completely different from what the TMMi³ Manual (release 1.0) has to say:

- A maturity level is a well-defined evolutionary plateau towards achieving improved organizational processes.
- The evolutionary testing model of Gelperin and Hetzel has served as a foundation for historical-level differentiation in the TMMi ... Testing has, according to Gelperin and Hetzel, since [the "debugging oriented" period] progressed to a "prevention-oriented" period, which is associated with current best practices and reflects the highest maturity level of the TMMi.
- The achievement of a specific maturity level must mean the same thing for different assessed organizations.
- Degree of process improvement across a predefined set of process areas in which all goals in the set are attained.

Models like TMMi and TPI®⁴ are fixed models: everybody uses the same pre-defined model, the context is not taken into account. The authors of the model have determined

¹ Immaturity of Maturity Models: <http://www.satisfice.com/blog/archives/581>

² <http://www.developsense.com/blog/2009/10/maturity-models-have-it-backwards/>

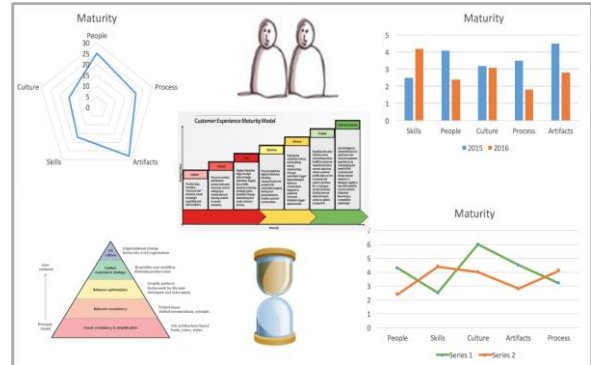
³ <http://www.tmmi.org/>

what areas are important and how to measure how you are doing in these areas. Growth is also predefined: apparently all organizations have the same growth model. Finally, the evolutionary testing model of Gelperin and Hetzel used as a basis for TMMi is from 1988. The world has changed since then... yet what they considered best practices back then still is upheld as the highest maturity level for our present day.

Designing a maturity model– mature in relation to what?

Maturity is a judgement, not a fact, but an interpretation of facts. Or at least it doesn't fit in facts.⁵

So, when designing a maturity model, a first important question is: maturity in relation to what? Do we want to know our maturity in relation to an abstract model, to other projects, to other companies, to our past selves, or to something else?



A second important question is: why are we doing this? What is the goal of gaining insight in the maturity? How would we want to proceed from that insight?

Our answers to these questions shape our answers to a number of other questions, for example: do we see maturity as something quantitative or qualitative? Do we want hard numbers, a score to judge testing by? Or do we see a maturity assessment as an expert review⁶, a means to asking questions and investigate potential problems? The expert review in this sense is a starting point for further investigation towards a solution.

A different approach to maturity

Thinking about those two questions, we came up with the following answers:

- **Testing maturity should relate to what you (as a tester, test manager, delivery unit manager, ...) think is important.** Too many different factors are involved for it to be possible to determine in general and in advance to what the testing maturity of a project/team/individual tester should relate to. This means we accept that your definition of maturity may change over time as your vision and/or your circumstances change. It may also differ per person, project and department. And it will most likely differ per company and per type of business. What is important in one context, might not be important in another. What is important for you, might not be important for me. Moreover, not only the maturity measurement cannot be done in general, but the next steps for growth can't be either. The outcome using a general model will be something "average", a "one-size-fits-all", while we want a context-specific solution...
- **The result of a maturity assessment should be valuable information on what is your vision on good testing and to what degree you live up to that vision.** The result of a maturity assessment should not only be a simple score – in the same way that a test report should not just be a Go/No-Go advice. Finding better questions to ask ourselves about our testing is more important than simple answers and score cards.

This means we see maturity as a subjective and evaluative judgement. Thus, there is no way to measure it objectively or to compare maturity levels (incommensurability). This

⁴ <http://www.sogeti.com/solutions/testing/tpi/>

⁵ Jerry Weinberg, Quality Software Management, Volume 1: Systems Thinking

⁶ An expert review in this specific context is where a test expert uses his/her knowledge and experience to evaluate the testing in an organisation, project or an individual tester. The expert will spot problems and recommend changes to improve.

view is the main thing that sets our approach apart from other testing maturity models like TPI® and TMMi. The consequences of this view can be clearly seen in the description of our approach.

Mission: better testing

What is the mission of this maturity exercise? We think the assessment should be a pathway to better testing. As a part of solving problems we think the mission should be: "An investigation of strengths and weaknesses. A starting point for a discussion about potential (testing) problems and how to solve them." Or as James Bach says: A maturity model is plan for achieving maturity¹. And this is exactly what we created. Our maturity model isn't anything like the staged, fixed models available in the market. Maybe we shouldn't call our method a maturity model, since basically it isn't. It is a tool designed to help teams assess and improve their testing. It is a method supported by a card game that helps teams retrospect and identify strengths and weaknesses in their way of working, the stuff they create, the team, their skills and context.

Finally, it's important to note that one can see testing as a performance⁷ (testing is what testers do) or as an activity (testing is testing regardless of who does it). Your paradigm of testing or how you choose to perform testing, will have consequences for the scope of your testing maturity. For example: are unit tests in scope or not?

Using the model

The model consists of a set of criteria (or heuristics) in six different areas. For details see below. To use the model you follow these steps:

1) Deciding the relevance of the criteria

The criteria are sorted into three groups per area: relevant, don't care, not applicable. Only the 'relevant' group is used in the two subsequent steps, the other two groups (and the distinction between them) are used in the analysis.

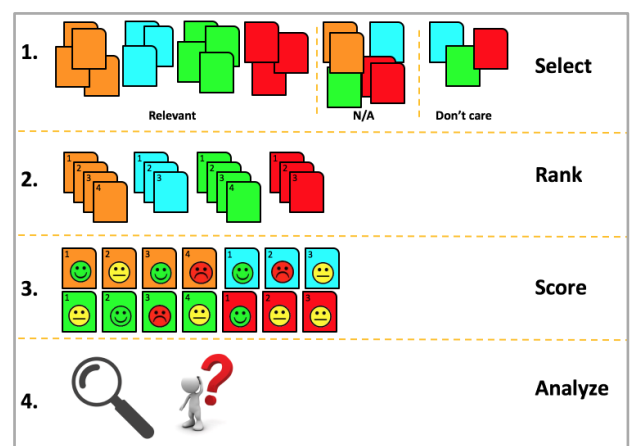
2) Stack ranking the relevant criteria

The criteria of the 'relevant' group for each area are stack ranked based on importance. Using stack ranking instead of categories (e.g. high/medium/low) forces hard choices: Yes, all these criteria are important, but is this specific criterion more important than these others or not?

3) Scoring the relevant criteria

All the relevant criteria get a 'score': green (good), yellow (to improve), red (poor). The scoring is explicitly not based on points to discourage a quantitative conclusion, i.e. the reduction of maturity

to a score. Also, points, for example on a scale from 1 to 10, suggest an unrealistic amount of precision. Besides being misleading, a high-precision scale might also lead to unproductive discussions on the difference between two scores: Should this be a 6 or a 7? Hence the simple scoring into three groups: yes (green), not there yet (yellow), no (red).



⁷ <http://www.testingcircus.com/testing-trapeze-2014-february-edition/> and <http://www.developsense.com/blog/2014/10/testing-is/>

4) Analysing the results

The analysis does not limit itself to the results of the third step. The choices made in step 1 (relevance) and step 2 (stack ranking) are also important input for the analysis. We do not want to limit the maturity analysis to how good you are in what you value: we also consider what you value and to what degree.

Improvement areas

The improvement criteria are divided in the following six areas:

1) Test Culture – where testers work

Shared patterns of behaviours and interactions, cognitive constructs and understanding⁸.

2) Context – what testers are surrounded with

Testing is not an isolated activity. (Neither is software development as a whole.) Your context, your environment may be conducive to good testing or it may not be. Criteria relating to 'outside influences' can be found in this area.

3) Trait – who testers are

Testing is executed by people. Excellent testers have the right characteristics and traits to perform well.

4) Skills – what testers do

Testing is a performance. Testing may produce artefacts through (explicit or tacit) processes, but without the proper skills from the actual people involved, the testing being done will not be very good. The skills area contains criteria to answer this very important question: do the people involved in testing have the capabilities to do what they need to do?

5) Processes – how testing is performed

The criteria in this area are mainly about interactions. Interactions between people, between people and artefacts, etc. It's about how work is getting done.

6) Artefacts – what testers create

This is the most easy and visible category, as it concerns artefacts: the things that are produced as part of the testing effort.

A note on the difference between traits and skills.

Although traits and skills cannot be completely separated from each other, we do see value on a distinction between the two. Skills relate to what you do: are you able to perform a certain activity and how well are you able to perform it? Skills can be developed through study and practice. Traits relate to how you are: do you display certain characteristics on your behaviour? Note that for you to display a trait, it needs to be part of your personality to some degree **and** your environment needs to be conducive to it. Traits can be developed through introspection and practice.

Read more:

- Maturity Models Have It Backwards⁹
- xMMwhy¹⁰
- Immaturity of Maturity Models¹¹

January 2017, updated April 2017.

Huib Schoots – huib.schoots@improveqs.nl

Joep Schuurkes – j19sch@gmail.com

⁸ <http://www.livescience.com/21478-what-is-culture-definition-of-culture.html>

⁹ <http://www.developsense.com/blog/2009/10/maturity-models-have-it-backwards/>

¹⁰ <http://www.developsense.com/blog/2011/10/xmmwhy/>

¹¹ <http://www.satisfice.com/blog/archives/581>

Test Improvement Heuristics

Testing culture – where testers work

- Image of testing
- Alignment with company vision
- Job satisfaction / motivation
 - autonomy
 - purpose
 - mastery
- Feel appreciated
- Feel responsible
- Team sport
 - shared responsibility
 - no silos
- Testing Mind-set
- Continuous learning
 - Coaching
 - Pairing
 - Training
 - Feedback

Artefacts - what testers create

- Context analysis
- Stakeholder focused communication
- Different models of product
- Risk (&value) Analysis
- Test strategy
- Test plan
- Test coverage outline
- Test Design
 - Mind maps
 - Charters
 - Testers always adding test ideas
 - Heuristics
 - Checklists
- Test results
 - Logs
 - Notes
- Test report
 - Written report
 - Dashboard
 - Testing Story
- Problem reports
 - Bugs
 - Issues
- Test infrastructure
- Test data
- Test tools
- Test automation
- Metrics
- Testware Management

Context - what testers are surrounded by

- Paradigm of testing
- Test Policy
 - Rules of engagement
 - Responsibilities
- Mission
- Stakeholder commitment
- Collaboration
 - PO
 - Stakeholders
 - In team
 - Developer relations
 - Between teams
 - Between departments
- Test organization
- Quality Assurance
- Responsibilities do not exceed authority
- Information
 - Requirements
 - Acceptance criteria
 - Manuals
 - Process descriptions
 - Product outlines
 - Architectural overviews
- Equipment & Tools
 - Hardware
 - Automation: tools
 - Probes (observation)
 - Matrices & Checklists (progress)
- Schedule

People – who testers are

- Passion
- Motivation
- Experience
- Tester Professionalism
- Tester self-defence/stand-up for testing
- Courage
- Curiosity
- Flexibility
- Collaboration
- Self-management
- Self-Aware / Asking for feedback
- Ethics
- Proactive
- Team fit
- Sceptical
- Persistent
- Diplomatic

Test Improvement Heuristics

Skills - what testers do

- Thinking
- Learning
- Context analysis
- Risk & value Analysis
- Problem Solving
- Asking questions
- Modelling & visualisation
- Estimating and planning
- Test Strategy
 - Context analysis
 - Define objectives/mission
 - Risk & Value Analysis
 - Creating product coverage outlines
 - Defining scope
 - Heuristics (HTSM)
- Testability
 - Ask for it
- Test Design
 - Test techniques
 - Chartering
 - Design Experiments
 - Heuristics
 - Oracles
 - Tours
- Generating test ideas
- Test Framing
 - Express
 - Annotate
 - Relate to mission
- Test execution
 - Exploring
- Observation
- Note taking
 - Labelling
 - Summarize
 - Listing
 - Outlining
 - Chartering
 - Mapping
- Reporting
 - Telling testing story
 - Bug/issue reporting
 - Status reporting
 - Dash boarding
 - Wrap-up & debrief
- Collaboration
- Political skills
- Negotiating
- Communication
- Technical skills
- Domain knowledge

Processes - how testing happens

- Methodology Practice
 - Agreed test procedures
- Compliance to test policy
 - Discussed with Audit
- Degree of involvement
- Model the test space and risks
 - Context Analysis
 - Product Coverage Outline
 - Test Plan
 - Test scope
 - Risk & Value Analysis
- Determine coverage
 - Test Strategy
 - Test Coverage
 - Test Conditions
 - Test Ideas
 - Design Experiments
 - Test Missions
 - Test Techniques
- Determine & apply oracles
- Configure the test system
 - Test Data
 - Test Environments
- Test Execution
 - Perform experiments
 - Run checks
 - Note taking
 - Test logs
- Evaluate the test results
- Report test results
 - Testing Story
 - Bug reports
 - Issue reports
- Defects Management
- Test Process Management