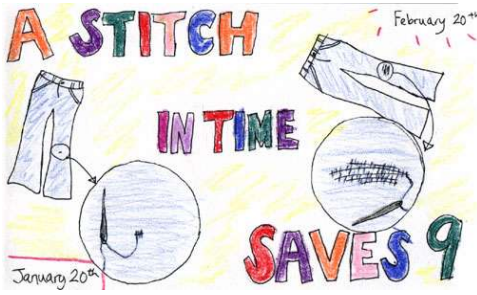


Measurable and Testable Requirements

1 Introduction

A stitch in time saves nine says a proverb. It explains a lot about the cost of fixing the bugs in software development. The cost of fixing an error is lowest in the first phase of software development i.e., requirements. This is because there are very few deliverables at the beginning of a project to correct if an error is found. As the project moves into subsequent phases of software development, the cost of fixing an error rises dramatically since there are more deliverables affected by the correction of each error. Of the bugs rooted in requirements, many are due to poorly written, ambiguous, unclear, and



incorrect requirements.

Measurable and testable requirements go a long way in reducing the cost due to errors found in the requirements phase. In this article, we look at how to make requirements measurable and, thus, testable.

2 Measurable and Testable Requirements

A requirement is measurable if there is an unambiguous way of determining whether a given solution fits that requirement. Requirements, if they are to be at all useful, must be measurable. That is, they must be given quantifiable characteristics so that testers can accurately determine whether the eventual solution satisfies a requirement, and so that the client can determine if the requirement's value is worth the cost of construction.

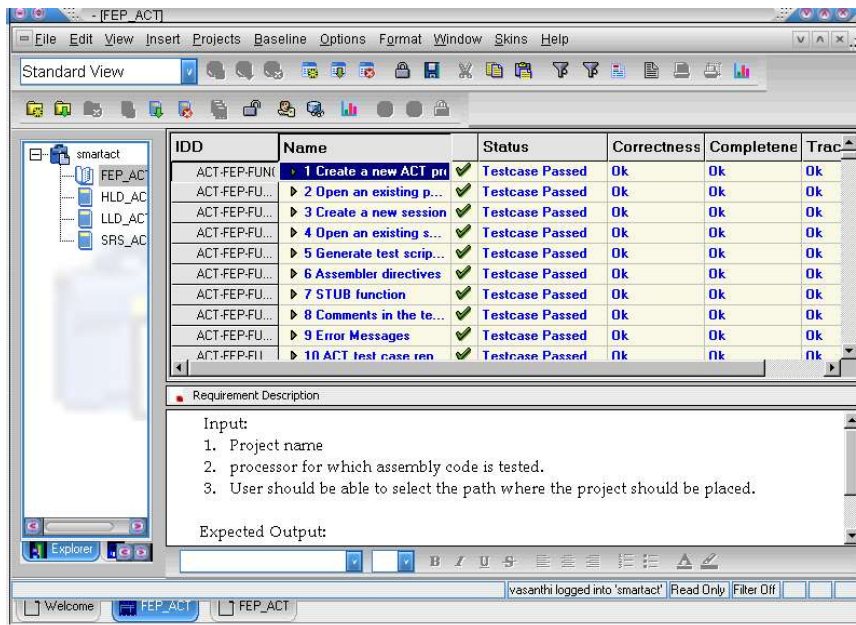
A requirement is seen to be testable because it is linked to one or more tests, which together show that it has been met. A requirement that can't be verified isn't a well-formed requirement.

Conversely, a test is seen to be worthwhile because it verifies one or more requirements. It has a trace back to each of those requirements, which it verifies. A crucial part of test design is the engineering of a complete set of verification traces, so that the system is known to conform to its requirements. A tool like ReqTrack is very helpful for constructing and checking large numbers of traces.

Following sections explain some of the guidelines to make sure that the requirements are measurable and testable while eliciting the requirements.

2.1 Assigning attributes

At first glance, a requirement is just a piece of text, possibly with a 'shall' in the middle. But this isn't enough. Requirements are shared, so each requirement must have an identifier enabling people to refer to it uniquely. Product Managers also need to know the priority and status of each item: the requirement becomes a database record, with a set of fields or slots or attributes holding different pieces of information.



The screenshot shows the smartact software interface. On the left is a tree view with folders: smartact, FEP_ACT, HLD_AC, LLD_AC, and SRS_AC. The main window displays a table with the following data:

IDD	Name	Status	Correctness	Completeness	Trac
ACT-FEP-FUN	1 Create a new ACT pri	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	2 Open an existing p...	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	3 Create a new session	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	4 Open an existing s...	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	5 Generate test scrip...	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	6 Assembler directives	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	7 STUB function	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	8 Comments in the te...	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	9 Error Messages	Testcase Passed	Ok	Ok	Ok
ACT-FEP-FU...	10 ACT test case rep	Testcase Passed	Ok	Ok	Ok

Below the table, the 'Requirement Description' section is visible, showing the following text:

Input:

1. Project name
2. processor for which assembly code is tested.
3. User should be able to select the path where the project should be placed.

Expected Output:

2.2 Deterministic requirements

The goal is to derive a set of requirements that are testable. The requirement is testable if it is written in such a way that each statement is deterministic. Deterministic means that for a given starting condition and a set of inputs, the reader can determine exactly what the expected outcomes will be. Each statement in the requirements can then be used to prove or disprove whether the behavior of the software is correct

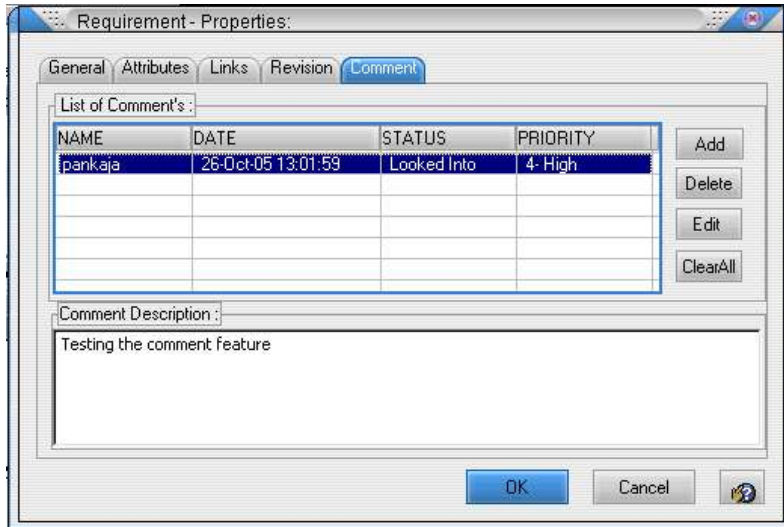
2.3 Removing ambiguity

An Ambiguity Review is a test of the requirements to insure that they are written in a clear, concise and unambiguous manner.

The Ambiguity Review occurs prior to the review of requirements for content by the domain experts. The intent of the Ambiguity Review is to provide the domain experts with a better quality set of requirements to work from, so they can better identify missing

requirements, and improve the content, completeness and accuracy of all requirements. The Ambiguity Review can be based on a checklist of common problems people have in writing requirements like dangling else, built-in assumptions, implicit cases etc.

A requirements management tool, such as ReqTrack can facilitate the review process very well. It can increase the review efficiency because of the improved logistics in making the requirement available to the reviewers and the increased visibility of the comments from the various reviewers without the need to manually merge and redistribute them. They are entered into the database with the requirement itself.



2.4 Requirements creep

Requirements creep is one of the most common risks in software projects. It happens when new requirements are added to the product after the formal requirements phase without any control. New requirements often "leak" into the product and nobody knows how they got there and where they originated. The result is that the eventual product and the original plan to build it bear very little relationship to each other.

To prevent failure, even during requirements creep, it is vital to insure that the new requirements are measurable and that the requirements process includes a measuring and testing activity to which all requirements are subject, without exception. Having a strong change control process supplemented by an automated tool like ReqTrack can control the problems of requirements creep and make sure that the requirements are measurable and testable.

In a change control process an approving authority is identified to approve all the new requirements based on a standard checklist. Whenever there is a new requirement or requirement change, a change request is generated which is approved by the approving authority before acceptance. An automated tool can assist by making the change request and its approval status visible to the whole team. The history of each requirement along with the date of change, author, comments etc. is entered in to the database.

A standard checklist of the approving authority can check whether the new requirement or requirement change is viable and relevant.

2.4.1 Relevant Requirements

Relevant requirements are those that can be shown to be part of the context. To test whether the requirement fits with the context, examine the flows of data to and from the system on the context diagram.

- Do any of these flows carry the data needed for this requirement?
- Do any of the flows bring data into the system to be stored for this requirement?
- Do any of the flows carry data from the system that is a result of this requirement?

If none of the boundary data flows contribute to, or are a result of, the requirement, then either the requirement is irrelevant or the context is incomplete.

2.4.2 Viable Requirements

Viable requirements are those that the organization is capable of both building and operating once they are built. For a requirement to be viable, answer to the following questions should be in affirmative.

- Does the organization have the technological skills necessary to build the requirement?
- Does the organization have the money and the time needed to build this requirement?
- Is this acceptable to all stakeholders?
- If we build this requirement, will everybody use it?

3 Software requirements metrics

As the requirements become more and more measurable, the metrics generated from these requirements become more and more reliable. Software requirements metrics are leading indicators of project scope, growth, stability, and progress. Software requirements metrics characterize the problem that a project is addressing, as opposed to metrics such as LOC that characterize the solution. Software requirements metrics are also available very early in a project and can form the basis for early analyses and predictions of project plans, alternatives, risks, and results. Following are the examples of metrics, which can be collected in requirements phase whose collection will be easier with an automated requirements management tool.

- Function Points - To predict size or cost and to assess project productivity
- Number of requirements errors found - To assess quality
- Change request frequency - To assess stability of requirements

4 References

<http://www.atlsysguild.com/index.html>
<http://www.stsc.hill.af.mil/index.html>
<http://easyweb.easynet.co.uk/~iany/index.htm>
<http://www.benderrbt.com/>

Pankaja P K
Manager - Software

She heads the SmartWorks and ReMa teams at Accord. She also manages DO178B verification projects.

Write to her at [feedback at ReMa](#)