Requirements for Test Documentation

Cem Kaner, J.D., Ph.D., ASQ-CQE

Contact Information:

kaner@kaner.com

www.kaner.com (testing website)

www.badsoftware.com (legal website)

Cem Kaner, P.O. Box 1200, Santa Clara, CA 95052

Acknowledgment

Much of this material is from the Third Los Altos Workshop on Software Testing (LAWST), February, 1998. I founded LAWST and was co-organizer of LAWST 3. That meeting agenda involved test documentation for automated tests.

The following people attended LAWST 3 and so contributed to the question list: Chris Agruss, James Bach, Karla Fisher, David Gelperin, Kenneth Groder, Elisabeth Hendrickson, Doug Hoffman, III, Bob Johnson, Cem Kaner, Brian Lawrence, Brian Marick, Thanga Meenakshi, Noel Nyman, Jeffery E. Payne, Bret Pettichord, Johanna Rothman, Jane Stepak, Melora Svoboda, Jeremy White, and Rodney Wilson.

Requirements Analysis?

If you follow a test documentation standard, such as IEEE-829, you will spend a lot of time and money documenting your test cases.

Should you adopt such a standard?

How should you decide whether or not to adopt such a standard?

Standards: IEEE 829

Examples of items included in 829-standard documentation

- Test plan
- Test-design specification
- Test-case specification
  - Test-case specification identifier
  - Test items
  - Input specifications
  - Output specifications
  - Environmental needs
  - Special procedural requirements
  - Intercase dependencies
- Test-procedure specification
- Test-item transmittal report
- Test-log

IEEE 829

- This is a highly detailed implementation guideline, but:
  - Expensive to create.
  - None of the details are things that we could easily say are irrelevant, so we can always justify spending the money.
  - ***What is the bug count for the buck?***
  - Over the volumes and volumes of test plan documentation, what does this really communicate?
  - What are the maintenance costs?
- What is the underlying requirements analysis?
  - Is this approach appropriate for all projects or just for some?
  - If some, which ones? What about yours?

Scripting

- Many people believe that a properly documented test case should be fully scripted. That is, the documentation should:
  - Describe every step in the test
  - Describe expected results from every step

Scripting: Good Practice?

Alleged Benefits

- fully reproducible test cases
- easy delegation to less experienced staff

Costs

- Expensive to create
- Tedious and expensive to maintain
- Boring to run
- Discourages or hides any exploratory testing that is done while testing from the script

How should you decide whether to use full scripting?

Requirements Analysis

- Requirements for test documentation vary from project to project.
- The following questions might help you sort out your test documentation needs.
- You probably don't have to answer all of the questions to profile your project's needs fairly well. Different questions will be more obviously relevant to some projects and less relevant to others.

Is the documentation a product or a tool?

- Product
  - Examples
    - You are a test lab, writing test plan on contract
    - You are a software developer, writing custom software and the customer wants delivery of the test docs as part of the delivered product.
  - Implication for practice
    - Follow any standards desired by the paying customer.
- Tool
  - Test docs for in-house use only
  - Do what is cost effective under your circumstances.

Is your quality driven by legal issues or by market forces?

- Driven by legal issues
  - Examples
    - Regulators will audit your development and testing practices.
    - Your software can cause personal injury. Your development / testing practices will be the focus of negligence lawsuits.
  - Implication for practice
    - Your test documentation must communicate your practices to people who don't understand your product and don't necessarily understand engineering. You may make a huge investment in paperwork to facilitate this communication, not because it improves quality but because it helps you manage legal risk.

Is your quality driven by legal issues or by market forces?

- Driven by market forces
  - Examples
    - Customers care what the product does, not how it was made.
    - Product will cause only economic harm, making negligence suits unlikely (even if it is defective). Lawsuits (if there are any) will involve claims of breach of contract or deceptive practices based on observable misbehavior of the product.
    - There are no sales / marketing representations about the process of developing / testing the software.
  - Implications for practice
    - Less emphasis on communication with future outsiders. More emphasis on documents that help you build a more satisfactory product today.

How long will these docs be active?

- Sometimes, reasonable to rely on oral tradition to carry details from one tester to the next.
  - If docs referred to only occasionally, then discarded after a few years,
    - e.g. computer games that will never develop beyond release 1.0
  - then many details can come from tester(s) who wrote the original docs and are still be around to guide new testers. The cost of reverse engineering or recreating some tests might be less than the cost of documenting the lot.
- But you need good documentation on projects that will use a software archaeologist
  - e.g. third party engineering of a product that will be in service for a long time, maintained by other organizations in the future. How do they know, 10 years later, what to test?

How quickly does the design change?

- Design changes quickly
  - Detailed test documentation (such as scripts) will be outdated quickly.
  - Software documentation is probably often out of date, therefore much testing is probably exploratory.
  - Expensive and time consuming to document expected results consistently and correctly (b/c of rapid change).
- Design changes slowly
  - Scripts might be affordable (even if not very bug-finding-effective)

Additional Questions

- Is test documentation a product or tool?
- Is software quality driven by legal issues or by market forces?
- How quickly does your product's design changing?
- How quickly does the specification change to reflect design change?
- Is testing approach oriented toward proving conformance to specs or nonconformance with customer expectations?
- Does your testing style rely more on already-defined tests or on exploration?
- Should test docs focus on what to test (objectives) or on how to test for it (procedures)?

Additional Questions

- Who are the primary readers of these test documents and how important are they (the different readers)?
- How much traceability do you need? What docs are you tracing back to and who controls them?
- To what extent should test docs support tracking and reporting of project status and testing progress?
- How well should docs support delegation of work to new testers?
- What are your assumptions about the skills and knowledge of new testers?
- Is test documentation set primarily a process model, a product model, or a tool for finding defects?
- A test suite should provide prevention, detection, and prediction. Which is the most important for this project?

Additional Questions

- How maintainable are the test docs (and their test cases)? And, how well do they ensure that test changes will follow code changes?
- Will the test docs help us identify (and revise/restructure in face of) a permanent shift in the risk profile of the program?
- Are (should) docs (be) automatically created as a byproduct of the test automation code?
- Should control of the project by the test docs come early (full set of tests developed at start of testing, used for scheduling and tracking), late (test docs completed toward the end of testing), or never (Test docs are intended to be useful but not complete. They guide testing but don't provide a structure for the overall mgmt of the project)?

In Conclusion

- Standards like IEEE 829, ISO 9000-3, and CMM have led some companies to generate huge test documentation sets.
- Before deciding on your approach, decide:
  - Who will need these docs in the future, and why?
  - What is the minimum level of detail (expense) needed to cost-effectively serve those future readers' needs?
  - What risks are you managing by investing in these docs?
  - What development / test style you want these docs to support?
- Once you understand your needs, you can invest to meet them, rather than investing to meet a standard that may or may not serve your needs.

