

The advantages of documenting test cases include the following [Yamaura98]:

- cases allow analysis of specifications from a different angle,
- test cases are repeatable and their quality can be validated, and
- another person or team can perform the tests.

Yamaura estimates that 10% of a system's bugs can be identified during test case design, where they are much less costly to fix, both in terms of money and time. If the tests are repeatable, bugs can be reproduced so that it can be assured when they are fixed properly. Specifying test conditions, inputs, and expected outputs allows another person or persons to execute the tests (which is valuable if a project is running behind schedule or if independent testing is required.) Test suites can be validated for quality of individual tests and as well as overall sufficiency.

For a large scale project, 2 weeks devoted to test case design and 2 weeks devoted to test case execution and documentation can save 3-6 months worth of effort to fix bugs in a deployed system. Yamaura's experience at Hitachi Software Engineering suggests that if there are 1,000 bugs in a system, without careful test case design and execution, 32% of the initial bugs remain in the developed product, and each takes 2-3 days of effort to fix, so that 64-96 days of effort are required after the product is completed to remove all bugs.

Your test plan should include a paragraph outlining the overall test plan, a section that describes each test case, and a section giving an overall test case matrix.

[5 pts] I. *Overall Test Plan*: one paragraph describing general testing strategies employed by your project. This is the testing philosophy; it may also include a description of tests (such as incremental testing during code development) that are not detailed in the other sections below.

[15 pts] II. *Test Case Description*: This will vary based on the project; try to give the students advice based on the adequacy of their tests for their project.

1. test case identifier (a number or unique name)
2. purpose of test
3. description of test
4. inputs
5. expected outputs/results
6. normal/abnormal/boundary case indication
7. blackbox/whitebox test indication
8. functional/performance test indication
9. unit/integration test indication

Note that some of these categories may be inappropriate for your project and may be omitted if you can justify doing so. For items 6-9, only one term should apply for each number.

[5 pts] III. *Test Case Matrix*: summarizes the test case coverage (items 1, 6-9 in a tabular format) This is just a summary of the previous section to give an idea of the test case coverage; the coverage should be appropriate to the project and does not need to include all categories above.

[5pts] for including requirements specification; this is informational and you can comment on it if you wish, but it is not intended to be graded at this time. It should reflect the current definition of

the project, however; outdated requirements specs are not helpful for providing context for the test plan.

TOTAL: 30 points

Note that your project timeline should be updated to include both test case design and test case execution.

Definitions

The following descriptions are taken from Sommerville [Sommerville01] or Yamaura [Yamaura98].

- *normal* testing: testing with expected inputs in normal operating conditions.
- *abnormal* testing: testing with exceptional (not intended for normal use) inputs or conditions.
- *boundary* testing: testing the subdivisions of an input domain or conditions upon which decisions are made.

- *blackbox* testing: testing based on the requirements specification.
- *whitebox* testing: testing based on knowledge of the implementation.

- *functional* testing: testing based on expected features.
- *performance* testing: testing based on expected performance criteria such as speed, power consumption, accuracy within tolerances, etc.

- *unit* testing: testing of individual components or modules.
- *integration* testing: testing of interfaces between components and entire systems.

References

Yamaura, T., "How to Design Practical Test Cases," *IEEE Software*, November/December 1998.

Sommerville, I., *Software Engineering*, Sixth Edition, Addison-Wesley, 2001.