Requirements Engineering: Specification & Validation

Software Requirements and Design CITS4401 Lecture 7 part 1

Key ideas (this week)

1. Testing requirements

   Convince me!

2. Prototyping requirements

   Show me!

3. Requirements Specification Document

   Tell me!

4. Managing requirements

   Maintain me!

Why do requirements need tests?

- An essential property of a software requirement is that it should be possible to **show** (validate) that the finished product satisfies it.
- Requirements that cannot be validated are really just “wishes.”
- An important task is therefore planning how to verify each requirement.
- In most cases, acceptance tests are prescribed based on how end-users typically conduct business using the system.

Source: SWEBoK Guide 6.4 Acceptance Tests

Requirement Test

- A test for a requirement is a way to demonstrate whether a system satisfies the requirement.
- Best to think about and write the tests at the same time as writing the requirement.
- The test is a type of contract for that requirement
- See also “Fit Criterion” – how will we know if this requirement has been satisfied?

Testing Requirements

- Engineers aren’t typically great at assessing the testability of requirements.
- Agile deals with this by bringing **testers** in right from the start
- (vs Waterfall when testers see everything much later in the process when it is often too late to change/fix the requirements as they are documented and agreed).

Verifiable Requirements Specs

- Definition. A requirements spec is **verifiable**

  iff (if and only if) every requirement statement is verifiable

  iff there is some finite cost-effective way in which a person or machine can check to see if the SW product meets the requirement

- We can use test cases, analysis or inspection to decide

Some Objective Metrics for Non-functional Requirements (1)

- Performance Speed
  - Number of processed transactions per second
  - User/event response time
  - Screen refresh time

Example 1: The system shall respond to user requests in < 1 second when the system is running at normal user load of <100 concurrent users.

Example: The traffic gate shall close in a most 3 seconds.

- Size
  - Kilobytes
  - Number of RAM chips

Example 2: The installed app requires less than 200 MB of memory on an Android phone.

Some Objective Metrics for Non-functional Requirements (2)

- Reliability
  - Mean time to failure
  - Probability of unavailability
  - Rate of failure occurrence

Example 3: The system shall be available to users for at least 1400 minutes in any 24 hour period.

- Robustness
  - Time to re-start after system failure
  - Percentage of events causing failure
  - Probability of data corruption on failure
- Integrity
  - Maximum permitted data loss after system failure

Example 4: No more than 1 minute of entered data can be lost if the system crashes.

Some Objective Metrics for Non-functional Requirements (3)

- Ease of Use
  - Training time taken to learn 75% of user facilities
  - Average number of errors made by users in a given time period
  - Number of help frames

Example 5: Users who have completed the system tutorial should be able to complete the PlanTrip use case within 2 minutes.

- Portability
  - Percentage of target-dependent statements
  - Number of target systems

Example 6: The app shall run on all Android phones models since 2015 for all operating system versions from 7 onwards.

Exercise

Propose an acceptance test for each of the example requirements in the previous slides

Some Objective Metrics for Non-functional Requirements (1)

Example 1: The system shall respond to user requests in < 1 second when the system is running at normal user load of <100 concurrent users.

Test: Write a script to run 100 common user requests and launch this script 99 times (worst case). Measure the total response time for each request (or all) and test 100 requests can be served in <100 seconds.

Example 2: The installed app requires less than 200 MB of memory on an Android phone.

Test: Generate an executable file for the app on (different versions) of Android phone and check that the app size is <200 MB for all versions.

Some Objective Metrics for Non-functional Requirements (2)

Example 3: The system shall be available to users for at least 1400 minutes in any 24 hour period.

Test: Write a script to run continuous "normal" interactions and run it for a long term. Record any time the system is not available. Note these types of requirements are hard to test effectively.

Example 4: No more than 1 minute of entered data can be lost if the system crashes.

Test: Write a script that continuously enters data. Write another script that crashes the system at random times. For each crash, measure the amount of data that was lost ie not saved before the crash.

Some Objective Metrics for Non-functional Requirements (3)

Example 5: Users who have completed the system tutorial should be able to complete the PlanTrip use case within 2 minutes.

Test: Run a training session (or provide online tutorial) for a group of users. Give them a test at the end of the session and measure the time it takes them to complete tasks (in this case, the PlanTrip use case).

Example 6: The app shall run on all Android phones models since 2015 for all operating system versions from 7 onwards.

Test: Collect a test set of devices (or simulations) and also a set of app tests with a pass criteria. Run each of the app tests on each of the required app/op sys combinations.

Hard-to-test requirements (1)

System requirements

Requirements which apply to the system as a whole.

In general, these are the most difficult requirements to validate irrespective of the method used as they may be influenced by any of the functional requirements.

Hard to test for non-functional system-wide characteristics such as usability.

Hard-to-test requirements (2)

Exclusive requirements

Requirements which exclude specific behaviour.

For example, a requirement may state that system failures must never corrupt the system database.

It is *not possible* to test such a requirement exhaustively.

Hard-to-test requirements (3)

Some non-functional requirements

Some non-functional requirements, such as reliability requirements, can only be tested with a large test set.

Designing this test set does not help with requirements validation.

But the tests are still necessary for system delivery and for test during development.

If you can’t make a test?

- We can use test cases,
- analysis or
- inspection to decide
- If a requirement is has been satisfied

Dilbert Testing

DID YOU ASK THE LAB IF THEY HAVE A WAY TO TEST TRAFFIC LOADS ON OUR PROTOTYPE?
I MET WITH THEM FOR AN HOUR AND EXPLAINED THAT WE NEED TRAFFIC LOAD TESTS.
BUT YOU DIDN'T ACTUALLY ASK IF THEY COULD DO THE TESTS?
WELL... NO... BUT...
IT'S THEIR JOB TO DO THE TESTS, AND THEY WOULD HAVE MENTIONED IT IF THEY DIDN'T HAVE A WAY TO DO IT.
BUT YOU DIDN'T ASK.
THAT WAS THE CONTEXT OF THE MEETING.
IF THEY COULDN'T DO THAT SORT OF TEST THEY WOULD HAVE MENTIONED IT SOMETIME DURING OUR HOUR TOGETHER.
MAYBE YOU SHOULD ASK.
GAAA!!! OKAY! I'LL ASK!
ARE YOU FREAKIN' KIDDING ME? !!!
I WONDERED WHY YOU DIDN'T ASK.
DilbertCartoonist@gmail.com
www.dilbert.com

Summary

- Each requirement should have a test to determine whether it has been satisfied (or not)
- Non-functional requirements can be hard to test objectively
- Requirements tests can be a form of contract for delivery
- Thinking of ways to test your requirements and test them is hard. You need to practice that skill
- Requirements can also be tested by analysis or inspection