Worksheet 3: Non-Functional Requirements

Updated: 18th February, 2020

1. Verifying Non-Functional Requirements (NFRs)

Imagine you are helping to develop an image/photo manipulation program. Some of the functional requirements for this system will include saving, loading and printing images, along with manipulation functions like cropping, resizing and rotating. You could write a use case for each of these.

Determine which of the following are valid NFRs. For each valid NFR, briefly describe how you would test it. For each invalid NFR, explain why it is invalid.

Note: To distinguish between functional and non-functional, ask yourself:

- From the *user's* point of view, is this a function, or a *characteristic* of a function?
- Could I write a use case flow-of-events for it?

If non-functional, ask yourself "How could I test it?" If you *can't* test it, it's not valid.

(a) The system must support the PNG, JPG and GIF file formats when saving and loading images.

(b) The system must support acquiring images from screenshots and connected camera devices.

(c) The system must save images within 1ms.

(d) The system must support images of up to 100 megapixels.

(e) The "colour enhance" function should take at most 250 ms to enhance a 10 megapixel image on an average PC.

(f) When performing the full range of image manipulation functions on images of up to 100 megapixels, the system should crash no more than once every 1000 hours it spends running.

(g) Once restarted after a crash, the system should be able to recover unsaved images open at the time of the crash.

(h) Photo-manipulated images must be suitable for inclusion in a photo album.

(i) An average user must be able, within 20 seconds, to rotate a tilted photograph so that it is within 3 degrees of level.

(j) An average user must take over 1 minute to open an image, on top of the time taken by the system to read and display the file.

2. Performance Requirements

Propose a useful and valid *performance* requirement for each of the following:

(a) A software system that enables a car to drive itself.

(b) A text-to-speech system that can read documents out loud.

(c) An equation-proof system that tries to automatically prove that two sides of a mathematical equation are equal.

Note: Since we are actually trying to do software engineering (and not hardware engineering) "performance" here means *software* performance. We are interested in, for instance, how quickly the system responds to events, or how many decisions it can make in a given amount of time. We are *not* interested in physical quantities like velocity, acceleration, temperature, etc.

3. Usability Requirements

Propose a useful and valid *usability* requirement for each of the following:

(a) A software system that enables a car to drive itself.

(b) On online banking system.

(c) A voice-activated "smart-home" system (for remotely/automatically controlling various devices around your house).

Note: Usability requirements are often expressed in terms of what the user must be able to do within a given time, or a given amount of effort. However, be careful to distinguish them from business rules.

In a business rule, the user is required to do something. In a usability requirement, the user must (on average) *be able* to do something thanks to the way the system has been (or, rather, will be) created.

4. Reliability Requirements

Propose a useful and valid reliability requirement for each of the following:

(a) A software system that enables a car to drive itself.

(b) A face recognition system for spotting wanted suspects in a crowded public place.

(c) A software system for monitoring weather data to forecast cyclones and other storms.

Note: Reliability requirements:

- Are (for our purposes) about *software* reliability, not hardware reliability.
- Must be based on a kind of measurement that actually makes sense for the particular system. Not all methods for measuring reliability actually make sense in all situations.
- Specify *how much* failure is acceptable, given that some failure is inevitable. It is not possible to require a system to be perfect. In fact, a reliability requirement is not necessarily there to ensure "high" reliability, only a *specific level* of reliability.
- Must say what *kind* of failure they're talking about. Systems may be reliable in some ways, and unreliable in others, and some kinds of failures may be worse than others.

End of Worksheet