For example, either agile or waterfall software development approach needs to be conducted for a nonfunctional requirement deal with usability, performance, reliability nonfunctional requirements evaluation to validate the quality of software. However, not all these nonfunctional requirements are equally testable. Some are based on inherently subjective or ambiguous criteria and must be refined to be verifiable. In this analysis we look at five such nonfunctional requirements and conclude whether they are verifiable and if so, we provide acceptance tests for the ones that can be tested, for others we explain why they cannot be tested directly.

First requirement, the user interface must be user-friendly and easy to use, its Nature is subjectively hence direct testing will not be possible. User-friendly and easy to use are concepts that could be defined in a measurable way. These features are usually evaluated by user experience studies, not binary tests. This usability requirement should be made measurable, e. g., in terms of the percentage of users completing give tasks or collected survey values related to user satisfaction. Pressman and Maxim (2014) stated that Usability requirements also need to be stated in a testable form as without it, they remain near data i.e. unclear and open to interpretation. Another way could be through usability tests, asking users to achieve specific goals and measuring success with metrics like completion rate, error rates, and satisfaction scores. So, if 80% of users can complete a task without help and rate the interface as good or great, perhaps the task requirement is met.

The second requirement, "The number of mice clicks the user needs to perform when navigating to any window of the system's user interface must be less than 10," is clearly verifiable. It is a clear and viable requirement that can be measured. Acceptance Test — After analyzing each navigation path in the user interface, count the number of mouse clicks from the

main screen to each window in the system. The requirement is fulfilled if each path takes less than ten clicks. It can be done manually while performing User Interface Testing or through automation tool where simulated actions are done and replication of user action is performed. Its clarity and specificity make it an excellent example of a testable nonfunctional requirement.

Likewise, **the third requirement**, "The user interface of the new system must be simple enough so that any user can use it with a minimum training" suffers from ambiguity as well. Terms like "any user" and "minimal training" are vague and context dependent. It doesn't specify the experience level of the users, or what would qualify as "minimum" training. According to Wiegers and Beatty (2013), nonfunctional requirements should avoid the use of vague terms, and instead, they should include measurable benchmarks. To make this requirement testable, we'd have to know the user demographic as well as acceptable training duration and outcomes. It may be expressed like this: New users should be able to lead to core tasks after 30 minutes of training time, with a success rate of 90%. Such a revision would allow for usability testing and objective measurement.

The fourth requirement, "The maximum latency from the moment the user clicks a hyperlink in a web page to the moment the rendering of the new web page starts is 1 second over a broadband connection" is a clear, measurable, and therefore testable performance specification. Tools like Lighthouse or automated browser testing frameworks can help verify this, measuring the time between user interaction and the browser response. As Bass, Clements, and Kazman (2012) explain, latency and other performance attributes are usually quantitative and need to be defined with environmental assumptions, including broadband speed, to make sure the measurements are valid. In Fig. 1, an acceptance test would be to click on all hyperlinks

under test conditions, and check that, regardless of presence or absence of a URL in the cache, the system starts rendering the new page within one second in every case.

The last requirement, "The need to recover easily and minimize loss of important data in case of failure" is testable only partially. Data loss can be quantified by simulating hardware failure, measuring the loss of data integrity afterwards, but the term “easy to recover” is subjective. This allows for the requirement to be written in a way that it can improve testability, for example – recovery time and acceptable levels of data loss can all be specified. For example, one might say that the system must recover service within five minutes of a crash and retain 99% of user-entered data. In the absence of such specifications, there is no way to definitively test the requirement.

Essentially, if nonfunctional requirements are clearly defined, they can be tested as per the goal. Requirements described in unclear or subjective language must be transformed into quantifiable and observable terms to be verifiable. With the help of structured testing methods and defined metrics, nonfunctional requirements are effectively part of the software development and validation process.

Wordcount: 792

References:

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3rd ed.). Addison-Wesley. https://www.amazon.com/Software-Architecture-Practice-3rd-Engineering/dp/0321815734

Pressman, R. S., & Maxim, B. R. (2014). *Software engineering: A practitioner's approach* (8th ed.). McGraw-Hill Education. https://www.amazon.com/Software-Engineering-Practitioners-Roger-Pressman/dp/0078022126

Wiegers, K. E., & Beatty, J. (2013). *Software requirements* (3rd ed.). Microsoft Press. https://www.microsoftpressstore.com/store/software-requirements-9780735679665