SearchSoftwareQuality.com

Software requirements: Why the term 'nonfunctional requirements' misleads

Missed and mistaken nonfunctional requirements account disproportionally for project and product problems. Most requirements books and courses explicitly split requirements into functional and nonfunctional and advise separately addressing each.

Terms such as 'quality factors' and 'quality attributes' are preferable but still have issues.

While seemingly straightforward, the conventional characterization and treatment of nonfunctional requirements actually may unwittingly create critical additional and seldom-recognized issues. In this tip, I explore the problems with common nonfunctional requirements scenarios and offer alternatives that can give development teams better views into user needs.

Before delving further into the mislabeling issue, let's define functional and nonfunctional requirements and explore the distinction between them. In essence, a functional requirement explains what the application does, and a nonfunctional one describes how well it does the function. An obvious comparison comes in use cases, which are a popular format for defining functional requirements of products, systems and software. Use cases describe step-by-step how an actor interacts with the system. An actor usually is the user, but also could be another system or a piece of hardware. Nonfunctional requirements, as well as business rules, are explicitly not described in a use case and must be documented in a separate supplementary specification.

How much usability do you need?

In my work, I've found that the term "nonfunctional" is misleading. It implies that such requirements exist in the abstract and can be defined all together, usually with a single brief entry for each type. Were this true -- and unfortunately, too many requirements definers act as if it is -- one would have to ask only a question for each type like, "How much usability do you need?" When it is put that way, most people realize it is a meaningless and unanswerable question.

You don't just need a bunch of usability. Usability is not nonfunctional. Usability is only relevant with respect to functionality. Moreover, usability requirements frequently differ from one function/use to another, and the differences are defined in terms of relevant characteristics, not some sizing unit as the single-entry-per-type approach implies.

Consequently, when gathering data to discover requirements, inquiries about various functions each need also to address applicable nonfunctional characteristics, of which there may be many. In turn, rather than grouping all such requirements together, they really need to be described in conjunction with the functionality to which they pertain.

More on this topic

Learn how to use nonfunctional requirements templates

Read more about issues relating to nonfunctional requirements

Learn the difference between functional and nonfunctional requirements

See how requirement types relate to software engineering

The conventional approach tends to consider both functional and nonfunctional requirements as aspects of the product, system or software expected to be created. A typical requirements definer often just expects a stakeholder to describe their desired product, system or software.

Such an approach creates creep by failing to recognize the need for first defining business requirements: deliverable *whats* that provide value when satisfied by a product/system/software *how*. Most nonfunctional requirements are real business requirements that exist within the business situation and must be met by any product, system or software. However, particular product/system/software choices may give rise to some additional nonfunctional requirements specific to the solution choice.

For all these reasons, I encourage my students and clients not to use the misleading term 'nonfunctional requirements.' Terms such as 'quality factors' and 'quality attributes' are preferable but still have issues because they easily lead to the common term 'quality requirements,' which mistakenly implies that only these factors affect quality. In fact, relevant functionality is the overwhelmingly dominant determiner of quality.

Instead, I suggest referring to all of these types of requirements as 'ilities,' because many of them end in 'ility' -- usability, availability, reliability -- and the term 'ilities' has no extra baggage connotations.

About the author:
Robin F Goldsmith, JD is president of consulting and training firm Go Pro Management, Inc. A subject expert for IIBA's BABOK®, he is the creator of the Proactive Testing™ and REAL ROI™ methodologies and author of the Artech House book, Discovering REAL Business Requirements for Software Project Success.

27 Dec 2013