Brady Field

Software Requirements

**Important points**

“The process of eliciting, analyzing, validating, and managing requirements, often referred to as ‘requirements engineering’ plays a critical role in the success of software development projects.” [113]

“Unfortunately, there is no general consensus in either the literature or in practice for either a common requirements terminology or for a consistent requirements process. Actual requirements practices vary broadly from organization to organization, according to the culture of the organization, its maturity in implementing software engineering processes, and the domain in which the software is being developed.” [113]

“Although the related literature uses the term ‘requirements engineering’ to broadly describe the requirements process this is not a term frequently found in industry. As the requirements process, is just one aspect of software engineering, it does not in itself result in the delivery of a fully engineered product.” [114]

“Process requirements specify constraints placed upon the development process. For example ‘the system shall be developed to run on the J2EE platform,’ or ‘Model Checking shall be used to formally validate the correctness of the security protocol.’” [115]

“The results of the risk and benefits analysis provide the basis for determining whether the project should proceed or not. We should point out, however, that this decision does not necessarily need to be an ‘all or nothing’ decision.” [116]

“The role of the elicitor is to learn the needs of the users and to communicate these needs effectively to the developers. There are many different elicitation techniques, and a general consensus exists that there is no single method that is universally the best one.” [117]

“During the requirements analysis phase, the emphasis is on gaining an understanding of the product to be developed through requirements classification and conceptual modeling.” [118]

“The SRS is used to identify risks, estimate cost and schedule, drive the design and implementation of the system, and act as a contractual agreement to support eventual customer acceptance of the product.” [119]

“One additional point to keep in mind when writing requirements is to ensure that the requirements state the needs of the user and do not contain unnecessary design constraints.” [120]

“Almost every software product continues to change and evolve throughout its lifetime. If change is not managed well, the quality of the product will deteriorate and future changes will become increasingly difficult to accommodate.” [121]

“When a change is proposed, the traceability infrastructure provides the ability to trace back to the rationale behind impacted requirements so that current decisions can be informed ones, and to trace forward to artifacts such as design documents, code, and test cases in order to more completely understand how to implement the change and to identify and mitigate its possible side effects.” [122]

**Disagreements**

“The writing style of a requirement is also important. Although there is some variation in standards between organizations, a general guideline is that all requirements describing a mandatory system property use the words ‘shall’ or ‘must.’ Words such as ‘will’ are generally reserved to depict events that will happen in the future and are not used to describe properties of the system. Certainly words such as ‘ought to,’ ‘should,’ ‘would,’ ‘might,’ and ‘may’ do not belong in a requirement because they immediately introduce the idea of an optional feature and therefore do not result in contractually binding requirements.” [120]

In this paragraph the author is pointing out the importance of using proper language to describe requirements. While I do agree that this is indeed very important, I disagree with the last statement that ‘may’ requirements essentially have no place in the document. An important function of this document is to help direct the design so that the engineers can build a system to satisfy the stakeholders via satisfying the requirements. There exist multiple potential solutions to satisfy requirements, and if a solution could be implemented to not only satisfy a ‘must’ requirement, but also to satisfy, or at least open a way to satisfy, a ‘may’ requirement as well, then this solution is definitely worth heavy consideration. However, proper consideration of such a solution can only be taken if the ‘may’ requirement is actually known.

**Questions**

I have no questions. I understood everything in the article.