

From your reading of chapter 4 and your own knowledge from other sources, discuss the following:

Is it ever possible to have the requirements definition document be the same as the requirements specification?

What are the pros and cons of having 2 documents?

For the most part, it appears that, since the requirements definition and specification are written for two different audiences, users and developers (Pfleeger & Atlee, 2006), it would not be advisable to merge these two documents; at the same time, the following passage from the IEEE's recommendations for software requirements specifications (SRS) makes this less clear:

The SRS should be unambiguous both to those who create it and to those who use it. However, these groups often do not have the same background and therefore do not tend to describe software requirements the same way. *Representations that improve the requirements specification for the developer may be counterproductive in that they diminish understanding to the user and vice versa* [emphasis mine] (IEEE, 1998).

In the IEEE's recommendations, the user still needs to be able to understand the Requirements Specification; however, it is possible that the IEEE's SRS recommendations do not distinguish between a requirements "definition" and a "specification," and are putting them both into the same document.

Such a merging often results in "confusion between concepts and details (Kim, 2009)." As the definition provides high-level concepts for the user, while the specifications get down into the details to which the developer may relate.

There is an interesting exception to this rule that I found online, and that is when the user and the developer are both working in the same environment, developing software. Open Trusted Computing merged their Requirements Definition and Specification into a single document without having to fear going over the user's head or water down technical details, because their users are software developers and the definition and specifications deal with interfacing with their application architecture's application programming interface (API) (Kuhlmann, et al. 2007)); in other words, the system interface is interfacing only with other software components.

IEEE (1998). *IEEE Recommended Practice for Software Requirements Specifications*. The Institute of Electrical and Electronics Engineers, Inc. New York, NY.

Kim, Sangwook (2009). *Requirements Definition and Specification*. Mobile Media Laboratory, Computer Science & Engineering, KNU. Retrieved from Mobile Media Laboratory on May 1, 2009 at: <http://woorisol.kyungpook.ac.kr/lab/prof/SoftEng/ch7.htm>

Kuhlmann, Dirk (2007). *DO2.2 Requirements Definition and Specification*. Open Trusted Computing. Villach Austria. Retrieved from OpenTC on May 1, 2009 at:

http://www.opentc.net/deliverables2006/OTC_D02.2_Requirements_Definition_and_Specification_update.pdf

Pfleeger, S., & Atlee, J. (2006). *Software Engineering: Theory and Practice*. New Jersey: Prentice Hall.