Software quality is perhaps one of the most sought-after but unattained goals in software development. It’s often understood, in a rather limited way, to mean the act of ensuring that a product meets its specified requirements. Unfortunately, this narrow viewpoint has led to the situation in which many quality assurance departments focus almost exclusively on testing and review and fail to address the more far-reaching questions that might differentiate between high-quality and mediocre renderings of the same product.

Just for a moment, conjure up an image of a completely bug-free system that meets all stated requirements. Now ask yourself whether it delivers quality. The answer isn’t as evident as it might first appear. The system clearly delivers all specified functionality, but the real question is, does it achieve this in a way that fully satisfies or perhaps even excites the stakeholders? What if the stakeholders failed to articulate their response-time expectations and the delivered product worked more slowly than anticipated? Or what if the system failed to appeal to the targeted user groups and therefore never made the necessary market breakthrough?

Both practitioners and researchers have widely recognized the need for a system to deliver real quality to its stakeholders. For example, Karl Wiegers identified several categories of quality requirements including attributes related to integrity, interoperability, performance, security, safety, usability, and testability. As experienced practitioners know, this list is potentially quite long.