

Issue 13, Free Digital Edition



# QUALITY MATTERS

WWW.QUALITY-MATTERS.ORG

## WHAT IS AGILE QUALITY?

by ERIK VAN VEENENDAAL

## THE VALUE DOES NOT WAIT FOR THE NUMBER OF YEARS

By OLIVIER DENOO

## QUALITY AND THE DATA STREAM

By PETER VARHOL AND GERIE OWEN

## AGILE TESTING TAKING NEXT STEPS

By KARI KAKKONEN

# WHAT IS AGILE QUALITY?

By Drs. ERIK VAN VEENENDAAL, Bonaire

Agile software development typically provides benefits such as the ability to better manage changing priorities, improved project status visibility, higher team morale, increased team productivity and better delivery predictability. However, in practice there are still many organizations that are struggling with Agile and scaling Agile. It also has become apparent that moving towards Agile does not always result in improved software quality. This is among others confirmed by both the 12th (from 2018) and 13th (from 2019) Annual State of Agile report that both (surprisingly to many) show, that a majority of the organizations using Agile do not (yet) report benefits in terms of software quality (see figure 1).

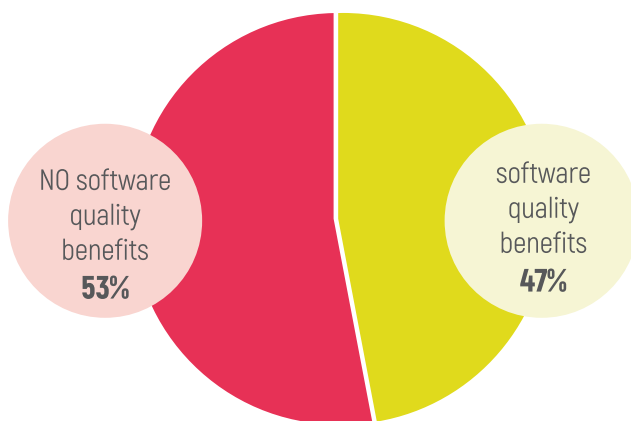


Figure 1: Agile organisations reporting benefits for software quality

## BUILT-IN QUALITY

The above triggered me, made me think and inspired me to write this paper. "Agile teams build high-quality products". "Agile team members write high-quality code". "Agile teams produce functionality quickly by not sacrificing quality". These are common statements one comes across in the context of Agile development. Quality gets mentioned a lot in discussions about Agile and in fact all twelve Agile principles promote quality either directly or indirectly. Continuous integration, test-driven development, acceptance test-driven development, test automation, definition-of-done, acceptance criteria, retrospectives, etc. are all more or less common Agile practices that should result in achieving higher levels of software quality. Quality is said to be 'built-in' when using Agile delivery methodologies as they have a focus on early and ongoing testing in various forms and continuous feedback from the customers, who (themselves or their representatives) are integrated in the team. Of course this can only work, when Agile is done well; you need the right product owner, collaboration techniques etc. to pull it off.

## EXPECTATIONS, STAKEHOLDERS AND MISUNDERSTANDINGS

The KPMG report Quality in Agile Teams from 2017 clearly states that lack of quality in Agile teams is largely due to improper testing. "Despite the teams trying to deliver in shorter iterations, the teams do not spend enough time planning for the testing types and phases." In addition, the quality problem could possibly also be traced back to the fact that there are often different expectations about Agile quality by different

stakeholders, e.g., a software developer may well focus on code of high-quality, whereas a product owner will focus on fitness-for-use. Quality is certainly an ambiguous term that has many definitions, expectations, and is often misunderstood. Have we, as an IT- industry, clearly defined what is meant by software quality in an Agile context? Do we at an organizational, project or product value stream level, discuss the ambiguous term quality upfront with stakeholders and know their expectations? One of the statements within the Agile manifesto reads "Working software over comprehensive documentation". But what does working software mean? Does it relate to the intrinsic value of quality code, software that is according to specification, or software that is fit-for-purpose?

So quality can imply different things. What do we mean when we speak of a quality product? This question gets even more interesting when we start comparing industries and products. Clearly a software product in the medical domain is expected to have a higher level of quality than any software game. But again, what exactly do we mean by a higher level of quality?

### VISION DOCUMENT

Quality is neither intangible nor immeasurable. It is a strategic imperative that can be quantified and put back to work to improve the bottom line. Within an organization, project or product value stream, a discussion should take place on what software product quality means in their context. What are we aiming for, and how will it be measured? We should by no means assume what others expect, but rather communicate, discuss and ultimately define software quality. To make it more tangible, we should define definition-of-done criteria based on and linked to the established common understanding of software quality. It's important to deploy and ensure the whole teams shares this understanding, and to document it unambiguously in a quality policy, test policy, business case, release level, mission or vision document, whatever does the job. My personal preference is to also print it, stick it somewhere on or near the task board, visible for all. This will ensure the whole team has the 'correct' software quality in focus, and moreover the quality objectives we are trying to achieve are clear, including how they contribute to business/customer value.

### QUALITY DEFINITION FRAMEWORK

I certainly do not want to provide or suggest a new definition for software product quality. Many before have thought about quality in depth. However, I will comment on how I believe these definitions can be applied to and fit in an Agile context. One way to answer the question "What is (Agile) quality?" is to start with the framework put forward by Dr. David Garvin. His framework distinguishes five approaches to define and determine software quality: manufacturing-based (also referred to as 'production-based'), product-based, user-based, value-based and transcendent-based (see figure 2). Garvin does not claim that any one of these approaches is sufficient by itself. Rather, a well-rounded view of quality requires all five. I will describe and discuss each of these approaches/definitions briefly from the perspective of Agile quality. Having used this framework many times in practice, I strongly believe it is a highly useful tool for having the much needed Agile software quality discussion in an organization, project or product value stream.

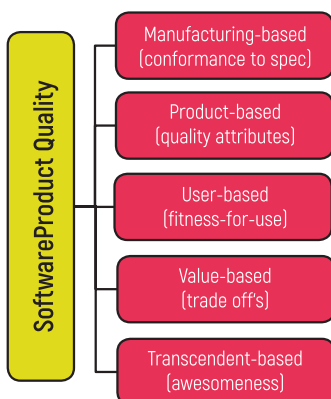


Figure 2: Five distinct approaches to software product quality [Garvin]

### THE MANUFACTURING BASED DEFINITION

This definition points to the manufacturing, i.e. the specification, design and implementation, process of software products. Quality depends on the extent to which requirements (user stories) have correctly been implemented in a software product. One of the leading advocates for this definition of quality was Philip Crosby. In the 1970s he proclaimed that "quality is free" because doing something right the first time is much cheaper than fixing it later. The price of non-conformance is the expense of doing things wrong. Quality comes from the prevention of defects, not their detection. I personally highly recommend this book to all involved, or interested, in quality and testing. To me, it's a must read! As stated, Crosby defines quality as conformance to carefully thought-out requirements. Interestingly, contrarily to what many people think, Crosby also puts the customer at the heart of quality. "The customer is the one you have to make successful. You have to understand what they need and learn how to give it to them."

The main problem with the 'conformance with requirements', there was never a way to be confident requirements were accurate and up-to-date. This because the wrong people or not all the right people were involved in their elicitation and validation, documentation was poor or simply because requirements and/or context changed in the timeframe from project start to delivery.

This is of course where agile really comes into its own and one of the biggest drivers for Agile adoption - when done well - can solve a lot of these problems through much closer collaboration and speed to market. In Agile we have put things in place with the objective to address, or at least minimize, these issues, e.g., user stories, INVEST criteria, acceptance criteria, stakeholders involvement (product owner), refinement sessions, managing the backlog, short iterations, feedback loop, etc. All of this should bring us closer to being able to state that it is a quality product when it complies with the defined requirements (user stories).

### THE PRODUCT BASED DEFINITION

According the product-based view, product quality is not just one homogenous item, rather it is determined by a number of defined characteristics or attributes of the product. Quality is based on a well-defined set of software quality attributes. The most well-known example of a set of software quality attributes is provided by ISO/IEC 25010 (see figure 3). The ISO/IEC 25010 standard provides consistent terminology for specifying, measuring and evaluating system and software product quality and is as such the successor of ISO/IEC 9126. In practice these software quality attributes are often referred to as 'non-functionals' although they also cover functionality attributes, in the case of IDO/IEC 25010 under the header of Functional Suitability.

#### ERIK VAN VEENENDAAL

CISA, Bonaire

Erik van Veenendaal ([www.erikvanveenendaal.nl](http://www.erikvanveenendaal.nl)) is a leading international consultant and trainer, and a recognized expert in the area of software testing, quality and requirement engineering. He is the author of a number of books and papers within the profession, one of the core developers of the TMap testing methodology and the TMMi test improvement model, and currently the CEO of the TMMi Foundation. Erik is a frequent keynote and tutorial speaker at international testing and quality conferences. For his major contribution to the field of testing, Erik received the European Testing Excellence Award (2007) and the ISTQB International Testing Excellence Award (2015). You can follow Erik on twitter via @ErikVeenendaal.



The various software quality attributes should ideally be measured in an objective and quantitative way. In the product-based view the level of quality of a software product is determined by its reliability, usability, scalability, etc. Differences in the quality of products of the same type can be traced back to the way specific attributes have been implemented.

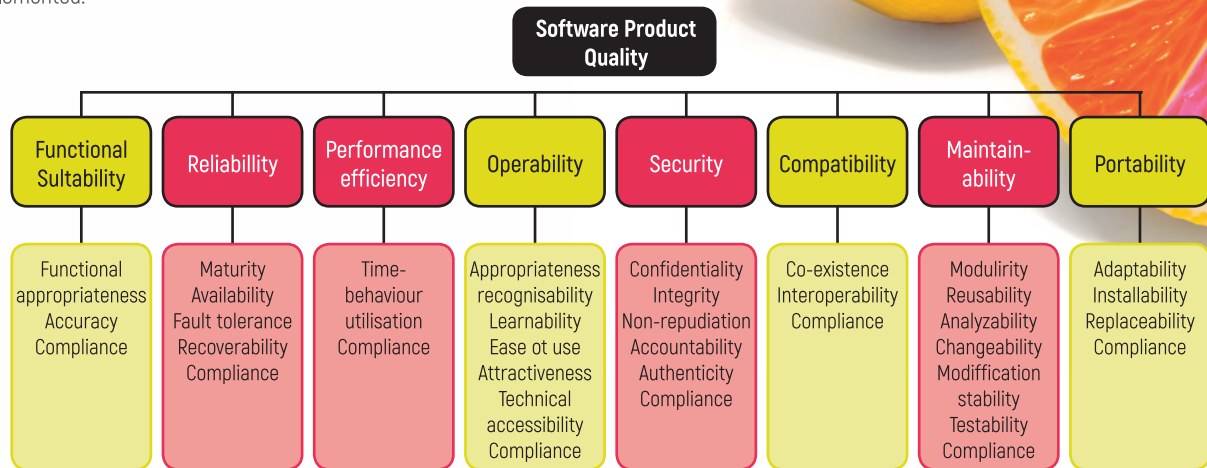


Figure 3: Software product quality attributes (ISO/IEC 251010)

An interesting factor in getting this right is how comprehensively the product owner understands the product requirements, e.g., a product owner from the business may have a very clear understanding of the business outcomes, but not care very much about a product's technical features, which will however determine the products long-term stability, performance and maintenance cost. In other words, Agile may succeed at ensuring functional quality, but may not automatically factor in non-functional software quality. So how do you ensure all aspects of a product's quality criteria are factored in equally, according to their importance for the product's benefit realisation?

Applying the product-based definition, it is important to discuss with stakeholders from a different background and with different perspective on a strategic level which quality attributes are of importance for the products being developed within the organization to consider them to be a quality product. As a personal lesson learned, it is of utmost importance to also critically discuss the rationale (business reason) for each of the identified quality attributes to prevent gold plating. Using the strategic product quality view, this exercise is repeated at lower level when starting the development of a specific new product, again with multiple stakeholders and typically not just the product owner. During the iterations the identified software quality criteria are translated into (non-functional) user stories with acceptance criteria, who subsequently are implemented, measured and tested. Often the mere existence of the quality attributes already makes the difference.

### THE USER BASED DEFINITION

Quality is fitness for use (also referred to as fitness for purpose). This approach to defining quality comes from Joseph Juran. Juran is generally considered to be one of the founding fathers for quality and quality management. He defined quality as fitness for use. "An essential requirement of products is that they meet the needs of those members of society who will actually use them. This concept of fitness for use is universal. It applies to all products and services, without exception."

The user-based definition states that software quality should be

determined by the users of a product in a specific business situation. Different business characteristics require different 'qualities' of a software product. The user-based definition starts from the assumption that individual users/customers have different needs and those products that best satisfy those needs are considered to have the highest quality. According to the user-based quality definition, quality is a subjective concept that cannot just be determined on the basis of only quantitative data and metrics.

This quality definition is related to the building the right product and the validation process. Specifically within Agile, it is related to the requirements (user story) elicitation process, refinements sessions, involvement of the product owner, sprint reviews with stakeholders and testing with domain-based testers and end-users based on user scenarios.

The user-based quality definition has of course, at least in my opinion, an almost perfect match with Agile. The first Agile principle states, "Our highest priority is to satisfy the customer ...." Quality is fitness for use. A high-quality product does what its customers want it to do, supporting the way they use the product. However, at the same time a quality product should do no more and no less than what it was intended for. The use of the term quality in real life influences our thinking, often it is seen as high or low in terms of the perceived awesomeness of a product. Beware to distinguish between fitness for use in context and general awesomeness!

### THE VALUE BASED DEFINITION

This definition states that software quality should always be determined by means of a decision process on trade-offs between time, cost and quality factors. The value-based definition emphasizes the need to make explicit trade-offs, to be done by means of discussions with stakeholders, e.g., product owner, users, developers and testers. This quality definition also relates to risk-based testing and the good enough paradigm. How much testing is enough? How much quality do we want? Which product risks shall be mitigated? etc.

In the real world, we often see quality being compromised for speed, so there is a trade-off between the two that needs to be managed. This

leads to negotiations regarding what flaws would be acceptable to meet a given deadline and how to set customer expectations accordingly. It may sound somewhat theoretical at first, but at a closer look it isn't. For instance, when you estimate a user story, what percentage of the effort is assigned to development, and what percentage is assigned to reviews, verification and validation? Spending relatively more effort on reviews, verification and validation should provide a higher level of quality, however it will also mean the velocity will go down. This is an example of a trade-off discussion that should be held when defining software product quality, the related quality objectives and definition-of-done.

Suppose that at the end of an iteration some spare time is available. The trade-off question then occurs: "Shall we do some additional testing to raise the level of quality, or will we try to deliver an additional user story?" I think I know the answer for most teams, which probably shows a thorough value-based quality discussion is needed to provide direction and manage quality expectations. Of course an additional user story is only an option, when product quality objectives are clearly defined in the definition-of-done and met for all developed user stories.

### THE TRANSCENDENT DEFINITION

According to the transcendent approach, quality is synonymous with innate excellence, absolute and universally recognizable: 'You will know it when you see it'. Transcendent quality recalls Plato's concept of beauty as an "ideal form." This 'esoteric' definition states that quality can in principle be recognized easily depending on the perceptions and the affective feelings of an individual or group of individuals towards a type of software product. In this view quality is something that is intuitively understood but nearly impossible to communicate, such as beauty and art. "I like it, because I just like it." Noted that when different people are asked what quality is, their understanding of quality is most often different. Although being the least operational one, this definition should not be neglected in practice. Often a transcendent statement about quality can be a first step towards the explicit definition and measurement of quality.

For example, a product can perform perfectly for its purpose or according to specification, but still be perceived as low in quality, because it's not pretty enough or just different. Likewise, aspects of a product can be needlessly over-engineered based on pre-conceived ideas of what constitutes (high) quality. Often this is seen in the form of insistence on best practices or state-of-the-art practices which may or may not be relevant in a given context.

In order to become mature, we need to move beyond the transcendent definition in most industries. Crosby already stated that quality should not be defined as 'general awesomeness', 'goodness' or 'elegance'. Perhaps in some industries, e.g., gaming, it may be the way to go. Why a game is appealing and successful is often hard to grasp, perhaps this somehow relates to the transcendent definition of quality.

A way to possibly make the transcendent perception of quality more tangible is using the Kano model during the definition of quality objectives, definition-of-done and requirements. The Kano Model of product development and customer satisfaction was published in 1984 by Dr. Noriaki Kano, professor of quality management at the Tokyo University of Science. Kano says that a product or service is about much more than just functionality. It is also about customers' emotions. The model assigns three types of attribute (or property) to products and services: threshold attributes (basics), performance attributes (satisfiers) and excitement attributes (delighters) (see figure 4). The excitement attributes are the surprise elements that can really boost your product's competitive edge. They are the features that customers don't even know they want, but are delighted with when they find them. Excitement attributes are the so-called "wow factor", and can often be discovered in conversations, on what product quality means, in an open-minded transcendent way.

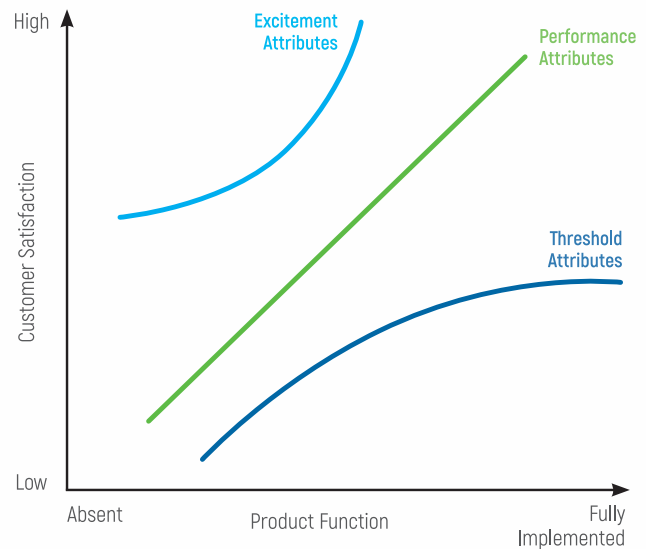


Figure 4: The Kano model

### HOW TO ACHIEVE AGILE QUALITY

Before starting Agile development and testing activities there must be consensus about what product quality really means in a specific business or product context. The objectives in terms of quality must be clear. Otherwise what are we aiming for? Only then can wrong expectations, unclear promises and misunderstandings be avoided.

The existence of the various types of quality definitions shows that it is not so easy to determine the meaning and relevance of software product quality in context, and thereby the focus of the development and testing activities. Agile practitioners have to deal with this variety of definitions, interpretations and approaches. I have learned over the years that in discussing the team's strategy and approach, it helps to also start a discussion about product quality. What does it mean to the stakeholders and what is expected? The framework as presented in this paper has proven to be highly useful and easy to apply. As already stated, in practice it is often a mix of the various definitions, a well-rounded view of quality requires all five. The discussion will make things much clearer to all, and expectations become more aligned. Such a discussion should not only take place on a project, product or value stream level, but also on an organizational level to drive improvement activities.

Understanding what quality means and what the quality objective are in context, allows the team to choose the right Agile practices and methods to achieve these objectives. Agile methodology techniques themselves have been designed to incorporate quality in a more practical and meaningful way, using elements of traditional quality management methods and testing regimes and integrating them in the design and development activities. However based on what is stated before, we need to tune the Agile 'built-in' quality approach.

Quality can be established within a project from the start through promoting an understanding of the underlying quality objectives and purpose of the project within the team. Applying the right mix of these quality features is key to fulfilling the Agile promise of 'built-in' quality, but an Agile project needs to invest some resources above and beyond the immediate delivery of the product to achieve this.

Defining what is meant by Agile quality in context, managing expectations and working towards these goals using the correct Agile practices with the right balance, will make a difference. It will not be the silver bullet, there are other (quality) challenges as well in applying and scaling Agile, but let's use this approach to bring the percentage of Agile organization that report software quality benefits to a much higher level.

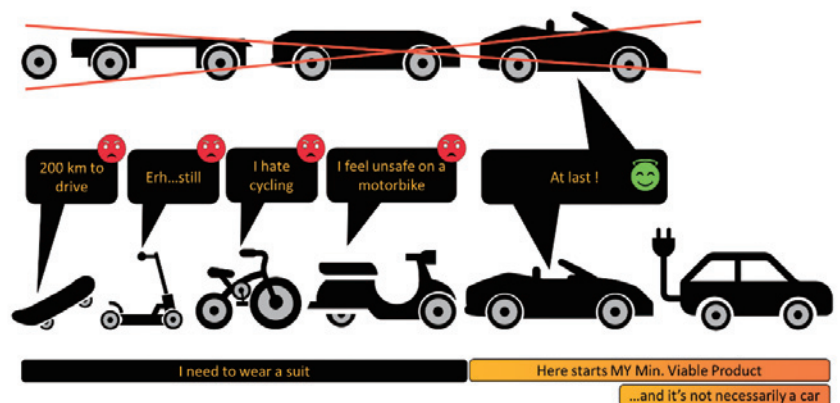
# THE VALUE DOES NOT WAIT FOR THE NUMBER OF YEARS

*« ...la valeur n'attend point le nombre des années » – Corneille – in « Le Cid »*

By OLIVIER DENOO,  
BELGIUM

While attending a conference, a colleague once tried to convince me of the benefits of an iterative approach, compared to a sequential development approach. According to him, and to the well-worn marketing slogan, Agile could only deliver value faster. So, he started to describe to me, with the energy of a convinced zealot, the example of the design of an automobile according to the Agile vs. traditional approaches.

## VALUE FASTER, HUH ! REALLY ?



Just as he had happily switched from the scooter to the motorcycle, I interrupted him with three sledgehammer arguments, which blew his false good idea to smithereens:

- I work 200 km from my home
- I have to wear a clean suit when I'm in business, rain or shine
- I don't feel safe at all when I ride in a vehicle with less than 3 wheels (or even 4 for good measure)