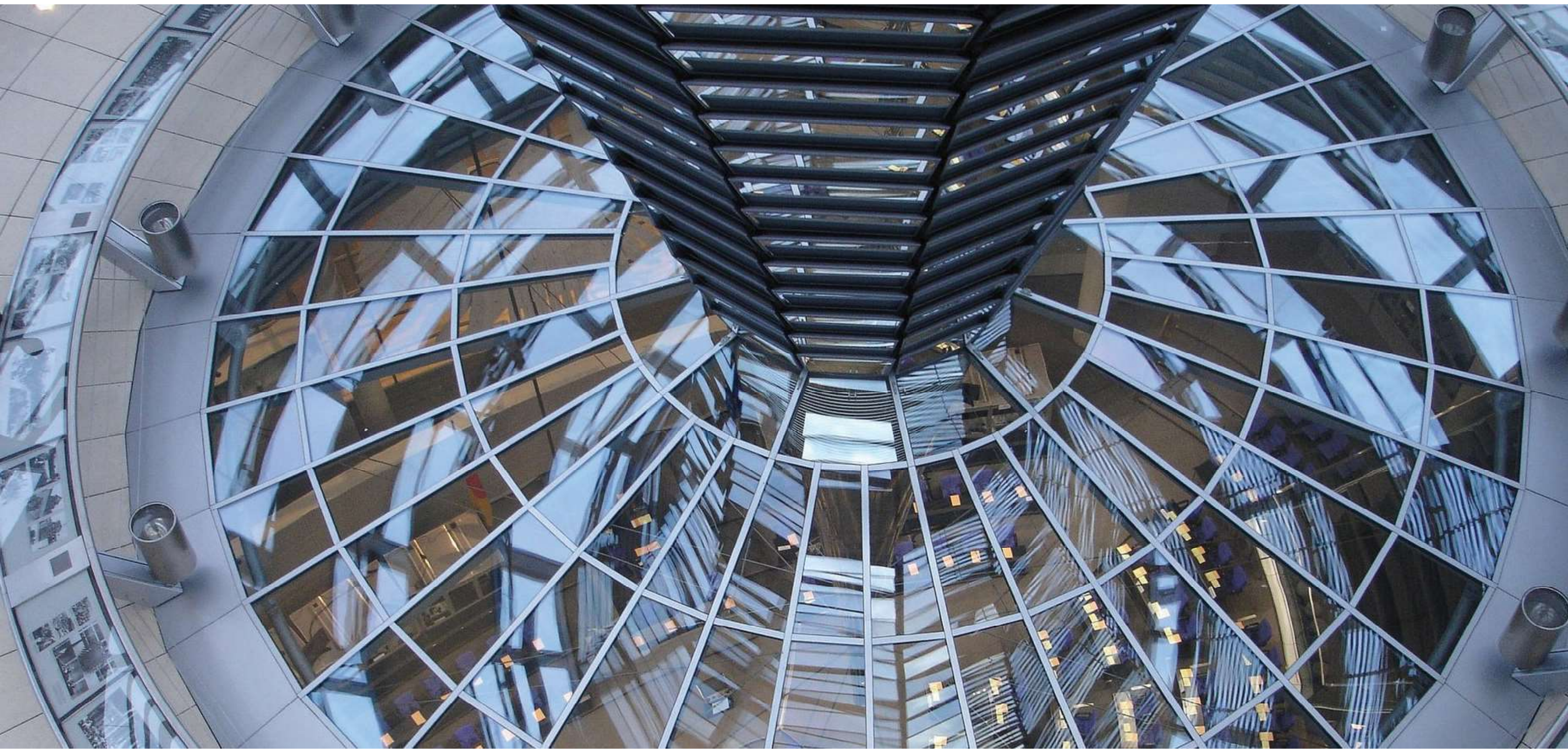
Considerations for Establishing Agile Quality Metrics

By Joe Schofield



The quest for quality continues. Manufacturers promote its importance. Consumers tend to benefit from it, and may pay more to ensure they get it. Speakers talk about it.[1] Authors write about it.[2] International standards are established for it.[3] Certifications are issued for it.[4] Organizations include it in their names.[5] A simple online search returns more than 7.2 billion hits on the word quality.[6]

Agile hardly stirs less interest, attention and scrutiny. Organizations claim their dominance in the community.[7, 8] Nearly everyone claims to be "doing agile."[9] Even "the fed" claims agile as its way of working.[10] Agile usage has spread well beyond IT as evidenced with 61% of marketing organizations using or planning to use it in their work in 2019.[11]

Despite all of the interest, no standard exists for agile today. Rather, we have a dozen or so approaches that claim a position in the agile market. Some of these frameworks capitalize on notions like iterative and incremental (concepts introduced in 1957 at IBM)[12], some on the popularity of products in the past.[13] A clear understanding of agile is further complicated by introducing quality measures across such a broad set of frameworks and techniques roughly bound together by a set of principles and a manifesto. Nonetheless, the following thoughts may advance the thinking of organizations exploring the use of agile measurements and metrics.

Most often quality is described as the product's "conformance to requirements." We could expand this definition to also include requirements for services, which are often codified in Service Level Agreements (SLAs). The quality measurement challenge begins here, since the first of 12 agile principles declares we "welcome changing requirements even late in development." An agile mindset accepts the fact that traditional requirements churn is actually acceptable (embraced?) in agile frameworks and approaches (from here forward, agile will be used to include the set of frameworks like Scrum, Crystal Clear and DAD, and more targeted approaches like Test-Driven Development (TDD)). With frequently and constantly emerging and evolving requirements, determining which set of requirements to verify could be anything but straight forward. As iterative development and incremental delivery occurs, assessing conformance to requirements needs to recognize refinement and grooming as suitable change management activities. Comparing committed features to released features is a reasonable bound for assessing "conformance to requirements"—at least until the next release.

"Establishing defect injection and detection meEsures should consider the iterative nature, the intentionally vague-to-better-understood nature of the agile work definition."

Using "conformance to requirements" as a potential definition for defects may require reconsideration as iterations and releases occur (while defects come in varying levels of types and severity, they are not the subject of this article; however, detailed analyses are available[14]). As a product owner shifts content priority in the product backlog and unceasingly grooms (changes, adds, merges, splits, deletes) specific requirements captured as stories, some which have been released earlier, defect clarity may become obscured. Acceptance criteria associated with new stories that introduce incremental change with the same feature may render previous non-conformances obsolete. As an example:

Release	Story ID	Acceptance Criteria	Comment
1	1	a – Attribute G can contain only red, or green	The chili[15] selection on a breakfast burrito
2	1a	a.a – Attributed G can contain red, green, or *none*	The restaurant owners never imagined a consumer not wanting red or green chili
3	1b	a.b – Attribute G can contain red, green, *both,* or none	The restaurant owner forgot about the New Mexico Christmas tradition of both red and green in the season though both are valid any day

While "both" and "none" are defects in Release 1, neither is in Release 3. The thinking that led to the acceptance criteria in Release 1 was incomplete (though close enough at the time), but acceptable with iterative development. What appeared to be a defect (entering "both") in Release 1 was not a defect in Release 3. Should we still count this as a defect or merely a benefit of iterative development? Do we expect (or demand) that regression tests are consistent with changes to the code? Does iterative develop tax traceability or merely justify its need?

This ongoing upheaval suggests a very different "requirements churn" in agile. The product owner has full ownership of the product backlog, changing it seemingly whimsically, arbitrarily, capriciously and ephemerally (WAC-E) (pronounced "whacky")). So then, establishing defect injection and detection measures should consider the iterative nature, the intentionally vague-to-better-understood nature of the agile work definition. And a final twist. What if in the example above, those three iterative cycles resulted in one release? Would we consider Story ID 1 and Story ID 1a to be defective? Purposely, the answer is left unanswered since the intent is to enhance our thinking about what quality means in agile before we begin to measure it.

While many organizations rely on testing, including full, daily regression testing for newly-integrated code, even world-class testing, will only remove as many as 50% of the defects in a product.[16] The other 50% or so were injected during requirements and design work; that is, story development and sprint task execution in an agile environment. Considering a sprint (or iteration) timebox for defect removal efficiency may hold promise[17] since defects can be tied directly to acceptance criteria and their associated story. This seems to be a simpler answer and is only useful once the considerations related to quality and defects during iterative development are better established.

Thus far, this article seems mostly to have offered cautions regarding quality measurements related to defects and requirements in agile efforts. Exactly right!

From Cautions to Suggestions

Improving quality, which should be the primary motive behind quality measurements, can still be driven with practices like peer reviews used in conjunction with Capture/Recapture Methods (C/RM). C/RM allow teams to predict, with statistical confidence, the remaining number of defects in a product in a peer review setting.[18] Refer to the IFPUG-cited source for step-by-step guidance of this quality enhancing technique. On agile teams, use peer reviews with C/RM on selected and critical, not all, product components. Consider swapping team members from other teams occasionally for peer reviews to cross-pollinate best-quality practices and to add an element of objectivity (account for this during sprint planning to avoid over-committing sprint commitments). Employing both design and code reviews might be worthy for inclusion in an organization-level definition of "done." Requirement reviews are a natural part of product backlog grooming when performed by the product owner and the developers. Together, these reviews have a direct impact on the quality of work of products before testing is initiated, thereby helping to address the 50% of defects not subject to testing coverage.

A related suggestion is apropos. Resist any temptation to reward teams (certainly not individuals when working with self-organized teams in agile) for either defect detection or correction. Rewarding this behavior will inspire teams to create more defects in order that they may be discovered, and potential subsequent recognition for either total number of defects found or corrected. This caution is an example of a much greater warning: beware of unintended consequences associated with the introduction of any measurement system or resultant metrics. Instead, hold teams accountable for the product they produce. Since value delivery is a major thrust of agile in general, value lost or delayed as a result of defects might be a useful quality metric; that value delivery less value lost per release.

Beware of unintended consequences associated with the introduction of any measurement system or resultant metrics

Early in my career, I heard about an organization that was going to measure the number of calls received by its service center. Customers began complaining that their calls were seemingly dropped after a couple of seconds. The "manager" of the service center went to check on the team. He heard a phone ring, saw a staff member pick-up the phone and then return it to its base. The same staff person then tallied a mark on a sheet of paper. The manager said, "What is happening here. Callers aren't getting answers to their questions." The staff person looked at the manager and said, "We are being measured by the number of calls we receive, not the number resolved." The dumbfounded manager could only blame himself for the new dilemma and the metric "calls received." He quickly replaced it with "percentage of calls resolved on first contact." In turn, this change had a negative impact on the duration of calls (they took longer, not usually seen as positive) and the wait time for calls to be answered (as call center folks were taking the time to resolve issues). Over time, applying the queueing theory and optimization stabilized expected response times.

Rethinking the Importance of Quality Expectations in Agile Efforts

Quality is no stranger to agile development. Quality is minimally implied in the ninth agile principle, which states "continuous attention to technical excellence . . ."[19] Teams that use retrospectives are constantly addressing improvements and at least some of those will be related to quality. Grooming continually improves the quality of the product backlog content. I offer these as a few examples of quality inherent in agile work.

"Teams that use retrospectives are constantly addressing improvements and at least some of those will be related to quality."

In an article released in April 2019, scruminc describes Schlumberger's use of Scrum resulting in defect reduction by about half, while also increasing productivity 25%, reducing headcount by 40% and reducing costs by 25%.[20] Reducing rework (defects) had a positive effect on productivity, which enabled Schlumberger to reduce headcount simultaneously. Historically, rework has been estimated to consume between 30% and 80% of software development cost.[21, 22] "Nailing" the requirements at the start of the sprint was a noted contributor to the Schlumberger turnaround.

In a study released in May 2019, speeding delivery, managing priorities, increasing productivity and aligning with the business all took precedence over enhancing software quality as motives for adopting agile. Improving quality was ranked even lower, ninth, as the perceived benefit of adoption. However, improved quality was listed second as a success objective with DevOps transformations. The same report found defect reduction eighth behind other agile success measures like C-Sat, value delivery, velocity, burndown and story completion.[23]

The State of Scrum survey reported a similar theme. Value delivery (71%) and responsiveness (56%) were selected over quality (44%) as most valued by executives. "Quality of life" received high scores (more than 80%) by Scrum practitioners. Seventy-seven percent of respondents expect to continue using Scrum in the future.[24]

Oddly, quality was not even mentioned in one recent project management annual survey.[25]

Don't Forget the Function Point Angle

IFPUG practitioners and researchers have often calculated defects per function point as a quality metric.[26] Function points are not a natural by-product of agile story-based requirements. Comparisons between well-defined function points and inconsistent (as intended) relative measurements using story points

aren't usually productive. However, in 2013, a case study proposed the use of elementary processes as a "common denominator" since they are found naturally in function point analysis, and are conceptually a worthwhile parallel in agile story decomposition.[27] To the extent that organizations find value in measuring defects per function point, that metric might bridge more traditional requirements definition and agile stories.

Steps Forward

One can reasonably expect the topic of quality, as it relates to our product deliveries, to continue for some time into the future. With most organizations using agile today and agile's expanding acceptance in numerous other industries, it may be time to reset the discussion on why software and other products are developed and released. Most of us willingly acknowledge a preference for quality over costlier, less useful, life-limited and defect-ridden products. Quality must therefore be an aspect of the value delivery, and is so highly-evidenced as crucial in this article. Simply stated, delivery without quality is of little value.

Delivery Without Quality is of Little Value, Agile or Not!

As you participate in assessing quality in your organizations, the closing list seems self-evident in agile organizations:

- Set expectations with all stakeholders as part of the product visioning that encompass quality attributes. Incorporate quality expectations in an organizational definition of done.
- Reduce variation with minimally-documented practices to promote consistency within and across teams. This approach doesn't keep teams from being agile; it does minimize re-learning and re-discovery. Of the 91% of organizations that offer training, 81% report improvement in practice.[28]

"Quality must therefore be an aspect of the value delivery."

- Employ techniques to detect defects early, well before testing. Less rework will lead to high productivity and lower total cost of ownership.
- Measure definitively, consistently and purposefully. Carefully consider unintended consequences and behaviors that may result from measurement activities.
- Shift the dialogue around agile measurements in general. Strengthen the focus on value delivery, priorities and releases versus cost, scope and schedule.
- Use existing reports and research to guide measurement efforts. You may learn some desirable practices or some you wish to avoid. Don't imitate other cultures that don't reflect your own. Best practices in the wrong context often make for bad practices.
- Continuous improvement doesn't happen by chance. Continuous learning feeds continuous improvement.
- Get the right people, trust them, keep them.
- Stop talking and thinking; start doing! ■

Special Thanks

I want to thank the following colleagues, active agile practitioners, for their review and insightful comments that enhanced this article and sharpened my thoughts:

- *Jennifer Turgeon, Systems Research and Engineering, Sandia National Laboratories*
- *Karen Grigg, Agile Advocate, Sr. Director of IT Development, Caesars Entertainment*
- *Brandon Hart, Technical Product Manager, A Place for Mom*

References:

[1] https://www.joejr.com/present.htm; most of these 50+ presentations deal with quality, defects and improvement frameworks

[2] https://www.joejr.com/pub.htm; most of these 35 books and articles deal with quality

[3] ISO 9000, Quality Management Systems; ASQ; 2015, 2011, 2009

[4] CSQ Certified Software Quality Analyst, as an example

[5] Quality Assurance Institute, as an example

[6] retrieved using Google on 5/27/2019

[7] 2017 State of Scrum Report identified more than 500,000 members

[8] SCRUMstudy reports more than 2,000 course registrations worldwide per week

[9] 85.9% of more than 101,000 internationally surveyed software developers use agile; Developer Survey Results; Stack Overflow; 2018

[10] 80% of federal IT projects describe themselves as agile or iterative; Agile by the Numbers, Deloitte Insights; 2017

[11] State of Agile Marketing 2018, Agile Sherpas; 2018

[12] Iterative and Incremental Development: A Brief History; IEEE; 2003; The recollections of Gerald M. Weinberg, who worked on the project, provide a window into some practices during this period. In a personal communication, he wrote: We were doing incremental development as early as 1957, in Los Angeles, under the direction of Bernie Dimsdale [at IBM's Service Bureau Corporation]

[13] Rational Unified Process with ancestral relationships with Agile Unified Process, Agile Modeling, DAD, Enterprise Unified Process, as examples; http://www.agilemodeling.com/essays/agileModelingRUP.htm

[14] The Economics of Software Quality; Jones & Bonsignour; 2011

[15] While there are a number of "correct" ways to spell the word chili (chile, chilli), I used mine—chili, green please; https://hotsaucefever.com/chile/chile-chili-chille-correct-spelling-of-the-word/

[16] https://www.ifpug.org/Documents/Jones-SoftwareDefectOriginsAndRemovalMethodsDraft5.pdf; page 15

[17] See also: http://static1.1.sqspcdn.com/static/f/702523/9242274/1288742153797/200806-Jones.pdf

[18] The IFPUG Guide to IT and Software Measurement; IFPUG; 2012; Chapter 36

[19] http://agilemanifesto.org/principles.html ; retrieved 6/2/2019

[20] Doing What A Billion Dollars Couldn't: How Scrum Inc. Partnered With Schlumberger To Solve Their ERP Implementation; scruminc; April, 2019

[21] Dr. Dobb's Report; informationweek; July 12, 2010

[22] The Economic Impacts of Inadequate Infrastructure for Software Testing; National Institute of Standards & Technology; US Dept of Commerce; May, 2002

[23] 13th Annual State of Agile SurveyTM Report; COLLABNET VERSIONONE; May, 2019; pages 7, 8, 16, 11 (as referenced)

[24] The State of Scrum 2017 – 2018; Agile Alliance;

[25] The State of Project Management Annual Survey; Wellingtone; 2018; page 14

[26] The Mess of Software Metrics; Capers Jones; March, 2017

[27] Function Points, Use Case Points, Story Points: Observations from a Case Study; CrossTalk; May / June, 2013

[28] 2017 State of Scrum Report; Scrum Alliance; page 18

About the Author:

***Joe Schofield** is a recent Past President of the International Function Point Users Group. Today, he is an enterprise agile transformation coach, an Authorized Training Partner and a Scrum-Certified Trainer with SCRUMstudy™. He has certified hundreds of Scrum Masters, Developers and Product Owners in the last few years. He retired from Sandia National Laboratories as a Distinguished Member of the technical staff after a 31-year career. Joe taught more than 100 college courses, 75 of those at graduate level. He has more than 80 published books, papers, conference presentations and keynotes—including contributions to the books: The IFPUG Guide to IT and Software Measurement (2012), IT Measurement, Certified Function Point Specialist Exam Guide, and The Economics of Software Quality. Joe has presented several worldwide webinars for the Software Best Practices Webinar Series sponsored by Computer Aid Inc.*

Joe holds six agile-related certifications: SA, SCT™, SMC™, SDC™, SPOC™ and SAMC™. He is also a Certified Software Quality Analyst and a Certified Software Measurement Specialist. Joe was a CMMI Institute-certified instructor for the Introduction to the CMMI®, a Certified Function Point Counting Specialist and a Lockheed Martin-certified Lean Six Sigma Black Belt. He completed his master's degree in MIS at the University of Arizona in 1980.