How do you measure success rate of large scale agile process?

Bhageerathi Bai

Bhageerathi.bai@intel.com

Abstract

To understand why there is a need for the Large scale Agile, I'm reminded of Jack Welch's words: "If the rate of change on the outside exceeds the rate of change on the inside, the end is near."

To succeed in this digital adapt-or-die environment, enterprises must be able to rapidly change the way they create and deliver value to their customers. Their ability to do that is highly dependent on their dexterity in developing software and systems. As those software and physical systems become increasingly complex, the methods used to develop those systems must allow the work culture to embrace collaboration, innovation, and speed.

Agile methods were originally designed for use in small and individual teams. Large scale Agile introduces unique challenges when different organizational units like software development teams, hardware teams, System teams, Customer Engineering teams must synchronize their activities when there is a need to interface with each other for a successful release. In this paper we present a systematic literature review on how adopting the Common Quality framework in large scale agile environment has helped us to bridge the quality gaps across these organizational units and measuring the success rate.

Why did we derive Common Quality Framework? – Motivation

- Delivering what customer **wants**
- **Open Communication** with customer
- Being **trusted** by your customer
- Synchronize release cycles to show **working software** frequently

In our attempt of combating the above challenges, we came up with Common Quality framework, consisting of a common repository and metrics that were shared across hardware, Software and System teams, raising the quality bar to be aligned with the customer expectations. This framework connects Hardware, Software and System requirements and there by prevents the individual units declaring a quality milestone in isolation. Additionally this reduces subjectivity and ensures transparency across the entire system.

Biography

Bhageerathi Bai is a Software Quality Engineer, at Intel, based in Bangalore, India. She is responsible for driving software quality gaps to closure with engineering, validation and program management office and helps teams to adhere to best software practices. She has an overall experience of 15+ years in software Quality, of which 10+ years was with McAfee (previously Intel Security) where she handled complete QA life-cycle of Enterprise products. She is a Certified Software Quality Engineer from ASQ, American society for Quality.

Bhageerathi is a Bachelor of Engineering in Electrical and Electronics from Bangalore University, India.

1 Introduction

The primary purpose of a Quality Management System is to implement an organization's chosen quality strategy by focusing on areas that are critical to successfully achieving the quality objectives, providing high quality products and services, and satisfying customers.

Large systems have more economic sensitivity to quality than do the features and subsystems that define them. A definition in **Steve McConnell's Code Complete** divides software into two pieces: internal and external quality characteristics. External quality characteristics are those parts of a product that face its users, where internal quality characteristics are those that do not.

At Intel built-in quality practices at an organization level helps every team understand and ensure that each solution element, at every increment, achieves appropriate quality standards throughout from development to release. The result is fast, continuous flow with a minimum of delays due to rework, high value delivery velocity and the highest levels of customer satisfaction.

2 Problem

Subsystem deliverables followed Individual Quality Framework and release cycles that resulted in slippage and delaying the final release.

2.1 Background

Based on our real-world learning and from applying the agile manifesto (www.agilemanifesto.org written back in 2001) and other ideas, we have derived our own top six agile principles. We successfully adapted these to meet our business objectives in a large-scale organization and put them to practice. However, we also ensured to revisit these principles and evaluate if we are still aligned with the ones that are critical to business needs at regular intervals.

- Satisfy your customer through early and continuous releases
- Welcome "Just in Time" requirements
- Don't over fill your plate
- Deliver early and often
- Cater to the Bottleneck
- Planning rhythm through collaboration

2.1.1 Large-scale Scrum is Scrum

Large-scale Scrum, is a regular Scrum, in which the concrete details need to be filled in by the teams and evolved iteration by iteration, team by team. It's about figuring out how to apply the principles, purpose, elements, and elegance of Scrum in a large-scale context, as simply as possible.

This reflects continuous improvement and a collection of suggestions enabling the team for inspecting and adapting the product and process when there are *many* teams to coordinate and collaborate with –

- **Internal -** Hardware, Software and System teams
- **Third party -** software and Hardware teams
- **Board manufacturers** and
- **Integration Vendors** (Software and Hardware Integrators)

2.1.2 Example of a subsystem which is developed following Scrum in an Ideal Case

While having Software/Hardware/System teams in different geographic locations, all teams adopted basic Scrum Principles and a scaled Agile Framework to work and deliver towards a single project or at times, multiple projects. Teams adopted Scrum which is an iterative framework for managing the projects inline to agile principles.

Agile teams - A define/build/test component team consists of following as Figure 1:

- Product Owner (Product Owner proxies)
- Feature teams mainly comprising of Architects, Developers, testers, program managers and QA Engineers
- Scrum Master

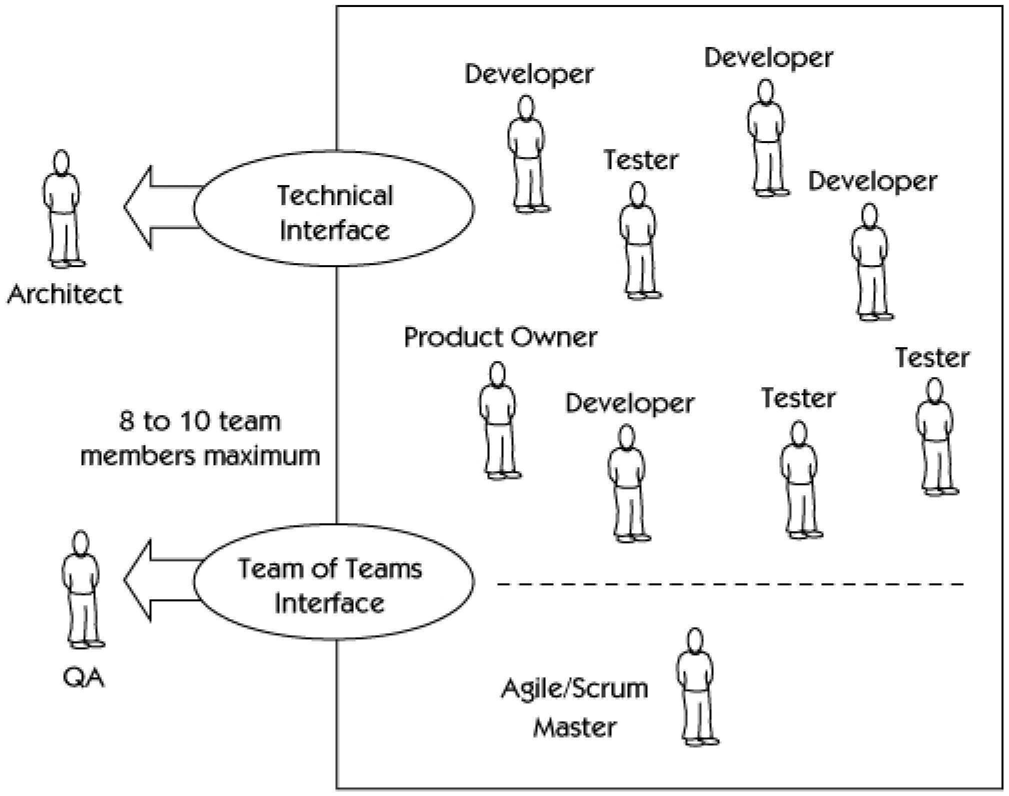

Figure 1: A define/build/test component team with interfaces to other teams

Key agile practices the agile component teams were already following:

- Agreed upon a daily 30-minute meeting - "**daily Scrum**," is a primary communication method which are coordinated to *occur at the same time every day where all the component team members were expected to attend*
- Cross-functional and collocated teams of ten or fewer team members develop deliverables in **sprints**. The teams may be distributed teams themselves or, more likely, distributed from each other. In either case, their role is the same: they define/build/test the software product, component, or subsystem of their domain
- Scrum focuses heavily on time-boxing. All the meetings starting from Planning, Sprint review meeting, standup meetings are disciplined to complete in prescribed times.
- Scrum allows requirements, architecture, and design to emerge over the course of the project.

- The **product backlog** which includes requirements to be delivered and the defects to be addressed both internal and external, is created, developed and prioritized by the **Product Owner**, an integral team member who is owning the primary responsibility of interfacing with the Integration vendors, who are customers. The Product Owner participates in the acceptance review of each story assigned in the Sprint. For certain teams, First line managers act as **Product Owner proxy** and define the requirements.
- The **Scrum master** mentors the progress and manages to clear the dependencies with the cross functional teams for enabling successful feature delivery along with clearing the Impediments the team faces.

  We have dozens of teams working together on larger systems and system of systems. Various component teams also hold a daily stand-up equivalent, typically in person or via teleconference every day so as to enable the other distributed teams to join.

  - **Synch meetings:** It is an extension of the daily stand-up and usually occurs immediately after the component team's stand-up meetings, which are coordinated to *occur at the same time every day,* where all Scrum masters of all five component teams/representatives (including subject experts at times and managers) attend. Any blocks, interdependencies, or other requirements for coordination are discussed and arrive at the common consensus to lay down the clear ownership/Plan of implementation/build drop schedule addressing the critical dependencies.
  - **The Scrum of Scrums:** This usually is on a weekly basis and members of this meeting may include some or all of the Agile/Scrum masters of all five component teams, System level QA Engineers, other representatives of the organization including Enterprise architect, Product Owners, Business Unit Owners, sales and marketing representatives, program/release managers who are point of contact to internal/external customers.
  - **Taskforce meetings:** When the Program health status moves to "delayed" from "On track" status – Taskforce meetings are scheduled *to occur as scrum at the same time every day,* to understand the risks and corresponding delay of the deliverables. This meeting will have the target audience comprising: the affected component team representative/scrum master and the other team representatives on whom the component team is blocked on. Based on the criticality, this can be tracked 24/7 where the other distributed teams would pitch in to execute in their time zones.

2.1.3 Agile Releases

Agile releases are planned at two levels: a series of fine-grained Sprint releases and coarse-grained "Potentially Shippable Increment" (PSI) releases:

- **Fine-grained Sprint releases** - Sprints are time boxed and features are delivered incrementally covering both development and validation thereby providing feature by feature release. During the Sprint planning phase, the dependencies on the other teams are clearly called out and prioritized accordingly as per the requirements and schedule. This is communicated to the relevant teams along with associated risks in case of slippage. The new increments after each Sprint may or may not be released for external use, but they are all completely developed and tested features ready for release. This is followed by the next Sprint with the same steps in order to expand the product with more features until the system level maturity is attained and reach a release milestone thus supporting market releases called – "**Potentially Shippable Increment**" (PSI)
- **Coarse-grained PSI releases** – Multiple PSI releases cater to the component deliverables and "Definition of Done" (DoD). These deliverables are tied to different milestone and evaluated in

Milestone review meetings. These product Milestones are scheduled during "**Development and Production"** Phases of the product lifecycle.

- **Milestones classification:**
  - Hardware release Milestones are defined as: Hardware Milestone (HM MS) 1, 2, 3 and Production
  - Software and System release Milestones are defined as: Pre-production (Pre-prod) 1, 2, 3 and Production.
  - Each Milestone criteria defines different entry and exit Criteria and corresponding targets.

2.1.4 Intercepting Component/System level Quality metrics during Milestone review meetings

Quality teams function independently out of agile teams - Individual Quality metrics were defined for different component/sub-system/system teams which ensured that the product meets or exceeds customers' quality expectations. It consisted of verifying the minimum qualification requirements by mapping to multiple tailored quality criteria and targets, those are used to evaluate the quality of a release candidate.

This specification applies to all Business Units across the organization and their subsidiaries and encompasses all products shipped or distributed by the organization.

The objective of the Quality specifications is to provide a framework to:

- Establish, measure, and achieve minimum quality criteria for the product development life cycle milestones
- Support software synchronization and integration (between different software and hardware components) with customized quality metrics
- Manage quality and risk of the product en route to pre-production and production releases.

Tuning to continuous changing Business requirements, the quality metrics were derived and the release candidate will be evaluated against the targets. Business Units (BU) are responsible for Qualification of their products based on these quality criteria prescribed by the quality team. A Quality Engineer is assigned to create quality plans, Monitor the progress, Measure and reporting for the Individual component/System releases.

Prior to each release, the BU must verify the minimum quality checks the framework mandates by aligning with the quality engineer who grades the quality criteria based on the maturity arrived. This is accomplished by providing the data or the evidence against each check to demonstrate that the quality has been met. If the product fails to meet any criteria, a calculated waiver or an exception is taken providing the line of sight of the fix intended. The exceptions are recorded and approved prior to externally distributing the product.

Typically Milestone release candidates go through the evaluation in the Milestone review meetings. The target audience comprises at overall system level: Quality Team, Product Development Team, Validation and Verification Teams and Program Manager.

If the product is a component in a System, the audience also includes the System Quality Manager and Quality Engineers from other business units when there are cross-team dependencies.

2.2 Challenges

A system must be managed. It will not manage itself. Left to themselves, components become selfish, competitive, independent profit centers, and thus destroy the system. The secret is cooperation between components toward the aim of the organization.

—W. Edwards Deming

As described above, the different components/System teams - software development components, hardware components, System Integration teams who function in agile mode declared their releases independently based on individual quality metrics; followed separate change control processes and repositories.

The component teams at times failed to enable a lot of features which the system teams required to deliver in their major functionality. Also at times, they independently implemented the features on requirement change requests or new requirements and declare their milestone in isolation. The system teams who would consume these components still would not be ready with required maturity to integrate these updated components.

Due to this, the system functionality and release cycles suffered featuring many compatibility issues and unsynchronized feature releases. Customers were unable to explore the final release samples with sufficient quality and maturity. This at times called for financial risks. In order to address these challenges we realized that the Business units should adhere to a common quality goals which aligns with organizational business goals and objectives, the mission of the organization and the need of its customers.

Adding to this challenge, was making this happen with teams that were geographically distributed and working on multiple projects.

- So how do we align all the agile teams to chase a common quality goal?
- What are these quality practices?
- How do these quality practices converge in large scale agile teams?
- How do we measure their success rate?

3 Common quality practices in large scale Agile

3.1 The Approach

If we don't change, we don't grow. If we don't grow, we aren't really living … Gail Sheehy

"Common Quality Framework" is designed such that it stiches different components and subsystem teams together as to follow a common quality goals against a common yardstick that is used for the evaluation. This mandates the release schedule to be synchronized based on the dependencies. System wide quality assessments are made across Hardware, software integrated into System level at each Milestone.

The main objective is to:

- Thoroughly understand the Common quality framework process, interpret the common criteria and targets defined to provide the meaningful risk assessments
- To set a common data review expectations against each Milestone
- Effectively guide the Business units through the data collection, evidence, approval and exceptions processes

Common Quality Framework acts as GO/NO-GO decision maker in the product Life cycle phases of Development and Production. Common Quality Framework ensures:

- Establish Common Quality standards across organization
- System-level assessments occur prior to component release decisions, Prevent component teams from declaring a quality level/release in isolation
- Help ask "**right**" questions to make informed ship decisions
- Transparency across entire system

- Central change control process
- Consistency of business processes
- Manage quality and risk of the product
- Improve focus on how customers test and stabilize their Systems and raise the quality bar to be aligned with customer **expectations**

  Common Quality criteria comprises of Base and Customized categories:

  - Base Criteria is a "must to have" and should be measured in all the release increments.
  - Customized Criteria – these are compliance criteria and targets specific to the third party components/Board manufacturers/Integration vendors the product supports.

With this, all the hardware, software components and Integrated System solutions are connected via their release Milestones and has to measure their working solution against the common Quality criteria and their respective targets.

3.2 Implementation

- **Scope – Quality Criteria and Targets defined:**
  As part of release readiness, Common Quality Framework touched mainly on areas like:
  - Requirements management
  - Feature Development
  - Defects and customer escalations
  - Third party Compliance
  - Manufacturing checks
  - Compatibility tests
  - Customer scenarios
  - In-house Deployment Tests
  - Documentation … etc.

  The completion targets became more stringent as the product matured through the incremental Milestones.
- **Process:** Once the common criteria and respective targets were defined in agreement with all the key team members (e.g., Engineering, Marketing, customer interfacing teams, program management office etc.) of the project, the actual results were measured and graded against the targets to understand the overall program health. This process starts early in the program and the criteria meeting trends were derived incrementally on a weekly basis. All the criteria is expected to turn **Green** which represents **"Target Met"** condition for the Milestone candidate to get promoted as a **Milestone ready** solution. Milestone review meetings are scheduled where the target audience would be members from Quality, Development, Validation and Verification teams and Program owner. Milestone candidate evaluation trends were presented here and GO/NO-GO decisions made based on actual results enabling Milestone declarations and component ship release approvals.
  However if there were any unmet criteria against defined target, then an "Exception" or Waiver would be raised. These open exceptions are expected to be "closed" before the next Milestone – example, component A had taken a waiver during pre-prod1 milestone, ensure this is addressed before their next pre-prod2 milestone review. This cannot be carried over to further Milestones unlike before.
- **Training:** These Quality definitions were analyzed in detail by the Quality experts across the organization. Then a training team evolved who was responsible for rolling out a training plan across the organization mandating all the teams attend the training/Q&A sessions and understand the framework in detail before migrating. Thus ensured all the teams and the

members had common understanding of the quality practices and targets to be achieved before releasing any working solution.

- **Adoption**: The evaluating criteria and targets are derived based on the Business objectives and customer expectations. This framework provides the Business units, the flexibility to map the quality checks to a wide range of product lifecycle models (e.g., waterfall, iterative, Agile, Scrum, etc.) thus enabling teams to effectively establish, measure and report out the quality and any associated risks as teams drive to production-level distribution.
  We adopted this in one of the large scale agile teams when the project was half way of the completion cycle - Component and System agile teams who were at pre-prod1 ready, were easily able to migrate to the **new common quality framework** during their next pre-prod2 milestone evaluation. These evaluations defined common yardstick and quality language which is generic across the organization independent of the component type (if it is hardware, software or any other subsystem)

Quality levels are mapped to plan product development lifecycle milestones in a manner that ensures appropriate visibility of a product quality and associated risks as it progressed from early definition and planning to production level distribution. The quality levels are defined with an increasing level of functionality and more stringent criteria and targets to support product progression through the life cycle.

3.3 Streamline the Release Schedule

We understand that we need to align our system level strategies to become more agile. Hence quality enterprise created a fixed set of rules that are imposed upon all the teams and leave the teams to figure out how to accomplish the mission. Following are the principles agile release trains are expected to adhere to:

- Frequent, periodic release dates for the component, System or solution are fixed and inviolate and known to all team members. Any changes to this should go through a Central change control process.
- Constraining teams to the dates means that functionality for the components must be flexible
- Certain infrastructure components, items such as common interfaces, system development kits, common installs, etc. must typically track ahead of the component teams so they are available as necessary as the components advance.
  Example, the System teams can declare Pre-prod1 milestone only when Hardware is @HW MS1 and Software @Pre-prod1 maturity so as to ensure these components do not break the baseline quality.
  Also the individual components should ensure they are validated against the latest available System configuration so that no compatibility surprises are uncovered later.
- Implement continuous integration at the System level, *so the system now has internal releases available for customer assessment just like the components had.* This gives the team the objective evidence it needs to adjust system and component scope in real time and to shake out the issues that lie around the interfaces of the system. The risk is addressed early, rather than late, in the release cycle.

By simply adhering to above principles, synchronized releases at component level and System level is achieved effectively.

3.4 Collaboration

Independent of anything else going on, how will you increase collaboration?

Having the teams largely spread across different geographical locations and relying heavily on different hardware and software Integration aspects, it was a major challenge to get all the teams' consensus on the expectations, deliverables and meet the timelines.

Due to Common Quality criteria which is set for all the team's deliverables, the teams were able to understand the deliveries and map their schedules more effectively as their evaluation metric was same to achieve. The dependencies were seen to clear considerably faster and so as to deliver as per schedule.

For Example consider Team 1 is allocated a software feature development, where the contribution is to develop for 30% of requirements and handover the solution to another Team 2, who would build upon the solution for another 50% and resume back to team 1 for rest of 20% development. In this context, the expectation is both teams 1 and 2 own the feature and have a common understanding of the architecture and logical implementations. Scrum Master of Team 1 and Team 2 were constantly in touch by attending daily Scrums and review meetings making sure the Teams are aligned by passing down the information as and when the development progressed.

The evaluation of the Quality criteria and taxonomies were common for both the teams and worked with the common understanding to meet the criteria.

3.5 Results

- Team collaboration was improvised as all chased a common quality goal
- Some of the Quality processes that were followed in an adhoc manner within the teams were well established. e.g. Requirements management traceability across the system covering all the component dependencies
- We have found the customer release samples at each Milestone met quality and good maturity
- System releases maintained zero critical exposure defects
- Relatively support calls came down due to stringent checks imposed by this framework
- Customers were clear about the deliverables and the schedule – at each Milestone they had the complete working solution be it a System or a component
- Product release - Time to Market was met as planned
- Enabled to plan the future releases with the horizon of **Predictability**
- Associated Finance Risks were in control and managed efficiently based on the compliance as the entire BU started migrating to the new Quality Framework

3.5.1 Challenges

- **Resistance to Change** - Not all the teams were ready to adopt and follow this process. Eventually adoption rate is increasing since the teams who have adopted this process are seeing tremendous value add
- **One size doesn't fit all** - There were some exceptional projects for which some of the evaluation criteria and metrics were customized to meet the quality requirements

4 Conclusion

Despite the widespread proliferation of agile practices, implementation often suffers due to lack of adequate project management support, System interdependencies, distributed teams or fear of increased interaction, to name a few.

With this small experiment of having a common quality language across organization, we could see an increase in team collaboration/transparency, asking the right questions during milestone and ship release approvals. Common Quality Framework is meant to stitch different teams in the organization who are catering to a common Business Goal. This Framework mandates to follow a common yardstick to evaluate and the release schedule to be synchronized based on the dependencies which in turn aids customers to explore the final release samples with sufficient quality and maturity.

This approach can be implemented to any large complex projects consisting of multiple engineering teams, third party dependent teams, external vendors, manufacturers, and other supporting functional teams in a geographically distributed environment. This is a tried and tested process which is proven to yield better results.

References

Book:

Scaling Software Agility: Best Practices for Large Enterprises by Dean Leffingwell

SAFe 4.0 Introduction Overview of the Scaled Agile Framework for Lean Software and Systems Engineering - Dean Leffingwell

The Certified Software Quality Engineer Handbook by Linda Westfall

Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum by Bas Vodde, Craig Larman

Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore
Product Development with Large-Scale Scrum by Bas Vodde, Craig Larman

A Practical Approach to Large-Scale Agile Development: How HP Transformed LaserJet FutureSmart Firmware by Gary Gruver, Mike Young, Pat Fulghum