18 Reasons Why Agile Cost of Quality (CoQ) is a Fraction of Traditional Methods

Ken Schwaber, Jeff Sutherland, and Kent Beck unequivocally knew Scrum and Extreme Programming had a far lower Cost of Quality (CoQ) than traditional methods way back in the 1990s through first-hand knowledge, application, and experience. Even those who coined the term "Agile Methods" and created the "Agile Manifesto" on that fateful day at Snowbird, Utah in 2001 seemed to inherently know Agile CoQ was unusually low. So, what exactly is CoQ? In simplest terms, it is the "total costs of achieving conformance to requirements" for a product or service across its entire life cycle (from concept to retirement). There are generally four major categories of quality engineering activity expenses that contribute to CoQ:

1. Prevention Costs. The costs of preventing defect introduction before development (i.e., training, root cause analysis, etc.).
2. Appraisal Costs. The cost of independently evaluating products and services before delivery (i.e., inspections, testing etc.).
3. Internal Failure Costs. The cost of fixing defective products and services before delivery (i.e., rework, re-testing, scrap, etc.).
4. External Failure Costs. The cost of fixing defective products and services after delivery (i.e., warranty, repair, recall, etc.).

The expenses associated with each type of cost grow by an order-of-magnitude from stage-to-stage. For instance, let's say prevention costs US $1.00 per defect. Then, appraisal costs US $10.00 per defect, internal failure costs US $100.00 per defect, and external failure costs US $1,000.00 per defect. There is some evidence to show that external failure costs may be as high as US $10,000 to $50,000 per defect (and these costs can range into the millions when one thinks of automobile recalls or aircraft disasters following frivolous or class action lawsuits). These basic economic ratios have been understood since the 1970s and were used to justify the creation, proliferation, use, and regulation of traditional, linear systems and software engineering methods.

Quality engineering economists often estimated the total number of defects created during the development of an average product or service (i.e., 10,000 defects). Then, they could estimate the cost of quality, based on the ratio of investment among these four categories. That is, emphasis upon earlier quality engineering activities reduced CoQ, while emphasis upon later quality engineering activities increased CoQ. Therefore, the emphasis was upon the earliest possible quality engineering activities (i.e., defect prevention activities), although a balanced portfolio using all four types of quality activities and cost was often sought. In spite of these models, most firms invested the lion's share of their activities in the latter two categories (i.e., failure costs).

Developing a culture of quality across organizations is quite difficult and expensive. This is often referred to as "World Class Quality." It's like winning a gold medal in the Olympics, becoming a championship body builder, or winning the World Cup. It requires a handsome investment of resources over a very long period of time, which few have achieved (i.e., less than 5%). It costs millions of dollars over decades in conventional techniques, and may even lead to bankruptcy if you bet the farm on competing on quality and lose. Today, some firms focus upon that last category (i.e., external failure costs), and economic models show it may be more cost effective to recall a product than to "do it right the first time" as W. Edwards Deming suggested.

Organizational change is required to institutionalize early quality engineering activities or form a quality culture. Organizational change is quite difficult and involves shifting deeply-held psychological beliefs and human behaviors. It requires decades, numerous initiatives, and of millions of dollars to achieve success. The level of difficulty increases exponentially with complex activities, organizations, or antithetical behaviors. The slightest change to organizations can be egregiously difficult. Have you ever heard of the "Butterfly Effect" (i.e., a butterfly flapping its wings in one part of the world can cause a hurricane in another)? I've been that butterfly and caused plenty of hurricanes (i.e., the most innocuous words and ideas may ignite a large forest fire).

Proponents of traditional methods like to point out that academic textbook approaches emphasize the use of a balanced portfolio of quality engineering activities. In fact, traditionalists are prone to point to the earliest and lowest cost defect prevention activities as the answer to the quality dilemma. Unfortunately, traditionalists fail to point out that instituting these practices is enormously difficult, expensive, and time consuming. Few so-called traditional organizations use any advanced quality engineering activities at all. Traditional methods come with portfolios of thousand page models recommending hundreds of activities, metrics, and artifacts. Agile methods aren't as well entrenched, don't come with dozens of thousand-page manuals, and have few activities and artifacts.

Welcome to the front line, trench warfare, and battle for your heart, mind, and soul. Not only are traditional quality engineering practices complex, expensive, and time-consuming, but they are manually-intensive, outdated, and come from the industrial era. Oftentimes, they don't emphasize enough defect prevention and focus upon appraisal activities like manual code inspections or late big bang integration testing. Furthermore, industrial-age philosophers mistakenly believe traditional methods scale up to complex organizations, products and services, and multi-billion dollar systems taking decades to complete. Traditional methods don't focus enough upon defect prevention, focus too much upon appraisal activities, and skip failure activities completely.

Furthermore, errantly applying manually-intensive industrial-age appraisal activities to enormously large, complex, and risky organizations, products, services, budgets, and timelines exacerbates the proliferation of defects rather than mitigates them. Mainframe operating systems from the 1960s were some of the earliest and most complex systems ever constructed by humans. They required decades and thousands of people. Productivity slowed and defects increased as people and communication paths increased. Modern queuing theory and models indicate a large scope, timeline, and budget slows productivity and increases the number of defects produced. In fact, productivity stops and defects abound on large and complex technology-intensive projects.

Innovation researchers now know that well over 80% of product and service requirements exist as tacit knowledge deeply embedded in the human psyche. Project managers, systems engineers, and software engineers who codify customer needs in the form of explicit knowledge are in for a rude awakening. Developing thousand-page request for proposals, statements of work, scope statements, charters, work breakdown structures, schedules, work packages, requirements documents, architectures, detailed designs, test plans, configuration management plans, and quality assurance plans over years and millions of dollars doesn't help. Well, it does provide a handsome paycheck, food on the table, and a few other creature comforts, but little else.

Over 80% of written requirements and other forms of documentation are defective or waste (i.e., not even needed at all). Think of Microsoft Office that took decades and billions of dollars to code tens of millions of lines of code. Well over 90% of its features have never been used at all and 99.9% of its features are never used on a daily basis. Most of its code has never been tested, would fail if encountered by users, and is full of security vulnerabilities that can be exploited to the detriment of billions of dollars. The same is true of most large-scale enterprise information systems created by Global 500 firms over the decades. Over 70% of large and complex technology-intensive projects fail, take decades to complete, are behind schedule, and way over budget.

Let's recap traditional CoQ. It's based on obsolete, manual industrial-age practices, such as code inspections, which few people use. It's used to codify thousands of errant and unneeded requirements, artifacts, and solutions. Its portfolio of techniques results in large-scale change that plunge organizations into expensive, debilitating chaos for decades. It's errantly applied to complex, large-scale projects, and technologies rapidly become obsolete. It freezes communication paths and queues, stopping productivity and increasing defect levels. It results in enormously expensive work-in-process in the form of unneeded activities and artifacts. It costs billions of taxpayer dollars in public sector projects. Finally, it's been known to bankrupt commercial organizations.

Thus, enter agile methods. Its creators understood why traditional projects failed. Dramatically reduce scope and sharply align it with business objectives. Develop a small highly-prioritized list of the top 20% of customer needs to achieve 80% of the business value. Most customer needs existed as tacit knowledge and they "don't know what they need until they see it." Implement in small batches until tacit knowledge is uncovered. Buyers and suppliers should communicate, cooperate, and collaborate in an open, empowered, and transparent manner. Use lightweight, disciplined processes and artifacts. Run thousands of automated tests every few minutes using free and open source tools. Perform frequent defect prevention on a daily, weekly, and monthly basis.

- Business Alignment. One of the highest priorities in agile methods is to align agile projects directly with business or mission objectives. Agile projects do not exist in a vacuum as autonomous, self-serving infrastructure groups, or for the greater good. Objectives are mapped directly to portfolios, epics, features, themes, user stories, and technical tasks. Business or market value is attached directly to product and service requirements. Agile teams are right-sized to satisfy specific organizational needs and priorities, and teams can track the organizational value they create on a task-by-task basis using earned business value. Agile team members know exactly how their daily tasks are aligned to organizational objectives and the value they create every day. This is an explicit form of defect prevention and waste reduction that does not exist in traditional methods and projects.
- Requirements Prioritization. A major practice in agile methods is to identify, focus upon, and deliver only the highest priority business needs, objectives, and requirements. That is, agile teams realize and recognize that only a small fraction of product and service requirements are needed and have any business value at all. Therefore, agile teams identify a small set of highly value adding requirements, quantify the value of these requirements in terms of business and mission value, and deliver these first. Over 80% of the business value is achieved by implementing less than 20% of the highest priority requirements. In doing so, agile teams can help organizations rapidly increase their mission and business effectiveness, efficiency, market share, revenues, profits, and other objectives. This is a major form of defect prevention not typically performed in traditional methods.
- Customer Collaboration. A key practice in agile methods is early, continuous, and close customer collaboration. Buyers and suppliers work together as ONE TEAM throughout the development to ensure conformance to requirements. Business and mission objectives are shared, along with responsibility for success and failure. Communication is based on rich, high-context face-to-face communications, rather than lawyers, contracts, liaisons, specifications, documents, and other inefficient, error-prone means. Customer needs exist in the form of tacit knowledge. Agile teams discover needs with rich communications, quickly develop solutions, and demonstrate physical models to create business value. This is a powerful form of low-cost defect prevention using real solutions that does not exist in traditional methods based on contracts, processes, and documentation.
- Small Scope & Batches. A major practice in agile methods is to reduce scope, requirements, team, and batch size. These are critical success factors known for decades. Over 80% of requirements are not even needed, wasteful, and result in defects. Large batches increase risks, costs, and timelines. They increase communication paths, queues, and delays, and proliferate defects. Technologies become obsolete, and the few projects that do complete are merely discarded. Agile methods reduce the size of the scope, timelines, and costs; develop value-adding solutions in smaller increments to satisfy customer needs; and produce vastly smaller numbers of defects. Reducing scope and batch size alone is a major defect prevention practice that doesn't exist in traditional methods, which aim to scale up to the largest scope, cost, timeline, risk, and complexity possible.
- Early, Fast, & Frequent Customer Feedback. Agile methods focus upon early, fast, and frequent customer feedback throughout development. This occurs from the very beginning to the end. This is the only way to solicit customer needs, test and validate your solutions, achieve business and mission value, and ultimately satisfy customers. This solves a number of problems like teasing out hidden, inexpressible customer needs that exist as tacit knowledge; quickly manifesting and validating smaller batches of inexpensive work in process as value adding solutions; and improving products and processes in hyper fast cycles to rapidly converge upon customer solutions. This uses early and continuous communications to rapidly realize high business value. This is a defect prevention practice not found in traditional methods that maximize CoQ over long periods of time.
- High-Context Communications. Agile methods rely on rich, high-context communications throughout a project. Traditional methods bar, ban, or discourage communications with contracts, lawyers, statements of work, specifications, non-value adding

processes, bureaucratic governance, voluminous documents, sporadic reviews, large batches, and extremely long cycle times. Traditional methods do so out of a misguided sense of fair market competition, division of labor, outdated manufacturing theory, and even ethics. This exacerbates poor performance, because customer needs exist as tacit knowledge that emerges through human communications. This in turn results in inordinate levels of scope, time, and cost risk, along with poor quality. Rich, high-context communications in agile methods is a key defect prevention practice often legally prohibited in traditional methods.

- Transparency & Openness. Agile methods promote transparency and openness through cooperation, collaboration, and communication. It also comes from small teams working together on a daily basis. Agile methods rely on rich, high-context face-to-face communications, as well as honesty, empowerment, and information sharing. This enables everyone to synch and tailor their behaviors, activities, and alignment to achieve business or mission value. Information is shared on walls, white boards, wikis, workflow tools, and electronic artifacts. Nothing is hidden, there is no division of labor, and little data is hidden in desk drawers, complex schedules, processes, and documentation, or withheld by cliques of power-hungry managers and developers. This is a major defect prevention activity only found in agile methods that is a major disadvantage of traditional methods.
- Small Disciplined Processes. Agile methods use small, disciplined, structured, and disciplined processes. This includes eliciting and documenting requirements and acceptance criteria, evaluating risk and business value, prioritization and scheduling, day-to-day coordination, customer demonstrations, and process improvement. This also includes configuration management, quality assurance, testing, and other support disciplines like architecture, documentation, user experience, security engineering, etc. Automated system administration, delivery, and deployment are also keys. There is a process for all activities and most of them have been automated-to-the-hilt using low-cost free and open source software. All of these practices are key defect prevention activities improving productivity and quality by 10, 100, or even 200 times over traditional methods.
- Small Cross-Functional Teams. Agile methods rely on using one or more small cross-functional teams. Agile projects focus on smaller, high-priority scope statements, requirements, architectures, designs, and solutions. They also exploit smaller batch sizes, and early and continuous validation of requirements throughout the day, week, iteration, release, and project. Therefore, the use of smaller teams goes hand-in-hand with agile methods more than any other paradigm based upon larger processes, projects, and complexity. There are more rich, high-context face-to-face communication, trust, cohesion, teamwork, cooperation, and collaboration. Silos and barriers between organizational functions are dissolved, integrated, and automated. This combines to dramatically increase productivity, quality, customer satisfaction, and business value. This is a key defect prevention activity.
- Teamwork & Cooperation. Agile methods also rely upon teamwork and cooperation to rapidly create high-quality, value-adding solutions. This is important on technology-intensive projects, which are complex by nature, and require advanced skills, education, and experience. Teamwork also applies to routine administrative or operations projects as well. Trivial problems are complex and may take months or years for individuals to solve alone. Moreover, individualized solutions may not be optimal, efficient, or effective, and may even be defect-prone or ridden. Combine experts and novices together, and complex problems become simple and efficient, and high-quality solutions can be created in a matter of minutes or hours. Even a mediocre team can do great things. This is a key defect prevention activity in a culture of traditional methods that rewards individual effort.
- Small Emergent Design. Agile methods focus on small, emergent architectures, designs, and solutions. Agile teams create just-enough design to solve a small batch of high-priority, value-adding customer needs. Agile teams develop, validate, and deliver functional and operational products and services rapidly and efficiently. Most customer requirements exist as hidden, inexpressible tacit knowledge. Small emergent designs enable agile teams to deliver solutions to customers in a matter of hours, days, or weeks. Early conceptual designs help developers rapidly converge upon valid solutions. Smaller designs enhance system performance, have fewer defects, and are more reliable. It is a key defect prevention activity not found in traditional methods aiming for expensive, wasteful, defect-ridden, low-performing, and unreliable 100 year designs with a very high CoQ.
- Flexible Technologies. Agile methods emphasize use of flexible technologies to rapidly deliver value-adding solutions. Teams identify, exploit, and use of proven higher-level abstractions instead of creating unique, one-off unprecedented designs one line of code at a time. These abstractions may be proven design patterns, frameworks, free and open source software, commercial solutions, web services, service oriented architectures, cloud services, etc. This may also include other pre-existing, dynamically reconfigurable and highly-tailorable e-commerce components. This applies to collaboration, project management, development, and solutions. This enables new and inexpensive people to achieve high levels of productivity and quality. This is a major defect prevention activity not found in traditional methods needing expensive folks with advanced skill, education, and experience.
- Frequent Multi-Tier Improvement Cycles. Agile methods are based upon multi-tier improvement cycles, which rapidly boost productivity and quality. Requirements are developed in small teams that make improvements every few hours. Daily meetings identify high-priority issues for immediate resolution, which is the second major improvement loop. Customer demonstrations and internal retrospectives are held every one or two weeks. This is the third and fourth major improvement loop. Post-mortems are held for 90 day releases and project end. This is the fifth and sixth level of improvement. These improvement cycles are multiplicative and help rapidly boost productivity and quality. Thousands or millions of automated tests are also run each day leading to more improvements. These are major defect prevention activities simply not found in longer-term traditional methods.
- Fully Automated Low-Cost Environment. Agile methods use low-cost development environments, tools, and automation. This may be free and open source or commercial tools. There are integrated environments with project management, workflow, development, and testing tools and support. Microsoft Visual Studio for .NET or Eclipse for free and open source software-based Java development are examples. This includes collaboration tools for virtual teams, wikis for information sharing, and version control tools. Build, integration, unit test, acceptance test, graphical user interface, and security testing tools are helpful. Static code analysis plug-ins can be used to run millions of low-cost automated inspections. All of this combines to help teams achieve high levels of productivity and quality. These are important defect prevention activities not found in traditional methods.
- Fully Automated CM, Build, & Testing. Agile methods use fully automated configuration management, builds, and tests. They use lightweight, structured, and disciplined processes. They are based on open and transparent communication. Artifacts belong to the entire team. They are kept in an automated environment including version control. This is egoless programming at its finest and artifacts are seen by everyone. When the code is checked in, it is automatically compiled, built, tested, and

delivered within seconds. Millions of tests are automatically executed in seconds at a fraction of traditional CoQ. This is a major defect prevention practice in agile methods, which traditionalists still have a hard time fathoming who want to perform a few manual code inspections every year and perform late manual big bang integration testing every 5 years to kill projects dead.

- Early & Often Continuous Integration. Agile methods use early and often continuous integration. Teams implement code in small batches to reduce feedback cycles and tease out tacit customer needs. This is not just a one-time automated testing event, but it is done every few minutes from the very first to last project hour. Teams pull a high-priority customer need with the highest business value right off of the requirements queue. They create a small design, implement it, and check it into version control. Automated tools take over to compile, build, and integrate the code, and run millions of tests. Developers know the results within seconds and customers can instantly validate solutions. No more late, big-bang integration to kill the project dead. This is a major defect prevention activity shortening feedback and improvement cycles to a tiny fraction of traditional CoQ.
- Fully Automated Multi-Level Testing. Agile methods use fully automated multi-level testing. They don't just use unit testing. As customer needs are identified, acceptance test criteria are also developed to satisfy customer expectations. Developers must use their experience to identify unstated performance, security, usability, quality, reliability, availability, and other non-functional requirements. Independent test, quality, reliability, user experience, certification, accreditation, and regulatory representatives should identify their criteria and tests for developers to use. As developers implement requirements and check in source code, millions of automated tests representing all of these disciplines must be executed. This is a key defect prevention activity within agile methods and failure data can be fed back into the process and classified within seconds to instantly improve the process.
- Fully Automated Deployment. Agile methods use fully automated deployment. Static and dynamic tests from multiple levels, disciplines, and regulatory bodies are run, and electronic documentation is generated. Images are created with deployment packages and manuals, in the form of source code comments and other electronic online help. Code is deployed to information technology operations groups who must run enterprises. Code may be instantly deployed to field devices, such as spacecraft, aircraft, weapons, automobiles, ships, autonomous aircraft, personal computers, laptops, networks, smart phones, wearable devices, medical devices, businesses, factories, and any imaginable equipment. This is a key defect prevention activity in agile methods, which speeds up delivery from years and decades to hours, minutes, and seconds at a fraction of traditional CoQ.

Agile methods are all up-side and very little down-side. They have the repeatability, definition, quantification, and optimization of traditional methods. However, they do not rely upon a library of thousand-page manuals, dozens of process areas, and hundreds of artifacts. They are fully automated to the greatest degree possible, eliminate most manual processes, and run millions of defect prevention, appraisal, and failure tests automatically every few seconds at a small fraction of traditional CoQ. They do not require years and millions of dollars worth of training, and done right, eliminate debilitating big bang organizational change. They are lean, have low WIP, just-in-time, focus on high-priority customer needs, and can be used to achieve scope, time, and cost constraints.

Implementing any one of these agile practices boosts performance over the use of traditional methods. Even novices using agile methods and a few of these practices for the first time experience a 10 times improvement in terms of time, cost, quality, risk reduction, and customer satisfaction. The rewards are even greater for investing a little more time implementing most of these agile practices. Projects that use more of these agile practice experience 100 to 200 times the level of productivity and quality improvement typically found by using traditional methods. Employee morale increases, expensive attrition comes to a screeching halt, and customer satisfaction, trust, loyalty, and retention exponentially increase, along with buyer and supplier business value.

So, you have a choice now. You can stick your head-in-the-sand and cling to traditional CoQ principles. You can use plan-driven, heavyweight traditional methods with thousand-page documents and hundreds of artifacts that will ruin the day. You can ignore 60 year old queuing theory showing irrefutable evidence why traditional methods freeze productivity and proliferate defects. You can insist upon manually intensive appraisal activities and nuking organizations with dirty bomb-style organizational change. You can implement large and complex projects that can't meet their scope, time, and cost objectives. And, you can throw stones from your traditional glass house at agile methods, because you don't understand its basic principles or you enjoy being a stubborn old coot.

It is possible to fail with agile methods if you pump out untested code like gangbusters. Ignoring advanced practices and principles like fully automated testing, continuous integration, continuous delivery, and devops is the fastest way to fail. Remember that agile methods are a mindset, value system, and set of behaviors not unlike W. Edward Deming's plan-do-check-act (PDCA) cycle VS. an ironclad five-step recipe. Sticking to basic backlogs, iteration plans, daily standups, demonstrations, and retrospectives instead of advanced agile principles is also a quick way to fail. You can also revert to debilitating traditionalism, because you don't know how to get coaching to achieve productivity and quality levels at 150 times the level of traditional methods at a fraction of the CoQ.

Agile methods consist of a plethora of sophisticated human behaviors VS. voluminous guidelines consisting of dozens of process areas and work product artifacts. Perhaps this is why proponents of traditional methods don't understand agile methods, or simply refuse to understand them? Maybe this is why traditionalists cling to outdated, linear, waterfall, bureaucratic, and manual practices with a very high CoQ? This also poses some difficulties for proponents of agile methods, as accepting, exhibiting, and practicing small numbers of agile behaviors may work sometimes, but not at other times. They also pose a significant dilemma as adding larger numbers of agile behaviors increases the time, cost, complexity, and risk associated with organizational change.

Are agile methods with 18 inexpensive defect prevention-focused practices a "deliver it now, fix it later" approach with high CoQ? Or, do complexity-inducing, queue-freezing manual appraisal-based traditional methods have a higher CoQ? Some early adopters used basic agile practices, saved up all testing for yearly releases, and created a ton of technical debt and defect backlogs. More and more agile groups are using advanced agile principles to ensure the lowest possible CoQ in seconds at the fraction of traditional CoQ. In essence, one can empty the defect backlogs with more sophisticated agile practices like Drano. To-be-or-not-to-be, that is the question. Should 21st century organizations trust in industrial age or agile CoQ principles? You make the call!