2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)

Critical Quality Factors for Rapid, Scalable Agile Development

Barry Boehm

School of Engineering
University of Southern California Los Angeles, USA
boehm@usc.edu

Doug Rosenberg
Parallel Agile, Inc. Santa Monica, USA doug@parallelagile.com

Neil Siegel
School of Engineering
University of Southern California Los Angeles, USA siegel.neil@ gmail.com

***Abstract*—Agile methods frequently have difficulties with qualities, often specifying quality requirements as stories, e.g., "As a user, I need a safe and secure system." Such projects will generally schedule some capability releases followed by safety and security releases, only to discover user-developer misunderstandings and unsecurable agile code, leading to project failure. Very large agile projects also have further difficulties with project velocity and scalability. Examples are trying to use daily standup meetings, 2-week sprints, shared tacit knowledge vs. documents, and dealing with user-developer misunderstandings. At USC, our Parallel Agile, Executable Architecture research project shows some success at mid-scale (50 developers). We also examined several large (hundreds of developers) TRW projects that had succeeded with rapid, high-quality development. The paper elaborates on their common Critical Quality Factors: a concurrent 3-team approach, an empowered Keeper of the Project Vision, and a management approach emphasizing qualities.**

Keywords agile methods, parallel agile, critical quality factors, scalalble agile, software processes, software architectures

I. Introduction

Many classical agile projects have difficulties with scalability and dealing with quality factors such as Security and Safety. They follow the Agile Manifesto principles of "Working Software Over Comprehensive Documentation" and "Responding to Change Over Following a Plan." They would build some working software, and then try to respond to change to make it safe secure, and scalable, but would find that no simple bits of refactoring would make it safe, secure, and scalable. A good example was the Thought Works ATLAS lease management project, which started out well using Extreme Programming (EP) but found that when the project reached 50 people and 500,000 lines of code, EP practices such as daily standup meetings, shared tacit knowledge of the code, 2-week system sprints, and lack of a project architecture or Big Design Up Front, were leading to project failure.

At USC, we have been experimenting with an approach called Parallel Agile, which has been successfully used to develop widely different applications such as for Location-Based Advertising, Picture Sharing, Bad Driver Reporting, and a Pokemon-Go type of video game. The performers have been groups as large as 50 of USC MS-Computer Science students working part-time, with high personnel turnover from semester to semester.. In researching the scaling of Parallel Agile to very large systems, we have identified set of Critical Quality Factors (CQFs) from some large TRW government system projects that were able to succeed in large-scale parallel development with a set of similar CQFs. Section II describes Parallel Agile and its Critical Quality Factors (CQFs), and also identifies similarities with the Incremental Commitment Spiral Process Model (ICSM). The ICSM CQFs, presented in Section II, were strongly derived from the set of successful large interactive TRW command and control software projects, which also satisfied and extended the CQFs to address forms of Parallel Agile for very large systems. The first example was the 3-version, million-line Command Center Processing and Display System–Replacement (CCPDS-R) project, described in Section III. The second example was a series of even larger TRW command-control-type projects that extended the set of CQFs and delivered highly successful results, also described in Section III.

II. Parallel Agile (PA) and the Incremental Commitment Spiral model (ICSM)

Our Parallel Agile (PA) approach (called Resilient Agile at the time), began when Doug Rosenberg, a guest lecturer in our USC software engineering courses, wanted to develop a system for Location-Based Advertising (LBA), in which drivers crossing a geofence around a restaurant or store, would be presented on their screen with a button to push to get a bargain certificate. He issued a challenge to the students in our software engineering project course to participate in the development of an LBA system by developing some of the LBA use cases in parallel.

978-1-7281-3925-8/19/$31.00 ©2019 IEEE
DOI 10.1109/QRS-C.2019.00101

It started with two homework assignments to give students a hands-on learning exercise with UML and use case driven development, with the first homework assignment called "build the right system" and the second assignment called "build the system right". Homework 1 (Build The Right System) consisted of each student identifying requirements, storyboarding screens, writing use case narratives, and disambiguating their use cases via "robustness diagrams" (conceptual model-view-controller decomposition of a use case, including both sunny-day and rainy-day use cases), while Homework 2 (Build The System Right) consisted of detailed design using class diagrams, sequence diagrams, database schemas, and occasionally state machines.

Approximately 75 students overall worked on the LBA project, with a 90% staff turnover every 3 months. During this time we did extensive prototyping of different geofencing solutions, and introduced a test-team operating concurrently with the developers to evaluate the performance of the various prototypes and conduct in-the-field acceptance testing. LBA was adopted for a time by the Santa Monica Chamber of Commerce, but the business volume was insufficient to continue.

Over the next three years we continued the experiment with three other student projects and began gathering productivity data on the various projects. This experimentation and data gathering is presently continuing with additional student projects as described below.

Our second attempt was with a photo-sharing app called PicShare, where we originally wanted to implement the same project with two independent teams, with one using Architected Agile and one using PA. Both teams produced good results, but the PA team took considerably less effort.

The third experiment was with a Crowdsourced Bad Driver Reporting System (BDR) which initially involved 15 students working in parallel for approximately 12 weeks. The BDR team developed a proof-of-concept system consisting of "dashboard camera" mobile apps for iOS (Swift) and Android (Java), and a web app implemented in Angular JS for filing, reviewing and querying video-centric bad driver reports. After 3 semesters of development the product is functional and has acquired a name, CarmaCam. Sample videos of bad driving incidents can be seen on www.carma-cam.com.

Our fourth PA experiment is a foray into game development, a Hawaiian-themed AR/VR territory game called TikiMan Go, where players throw lava fireballs at animated 3D tiki men in both VR and AR battle environments. Hawaiian Airlines is interested in adopting it.

The critical success factors for PA modify and extend the ICSM Three Team approach..

The upper Architecture and Capabilities Evolution team has as its primary functions the determination of the win conditions of a system's primary success-critical stakeholders (clients, end users, safety/security regulators, maintainers, investors, etc.); the choices of system infrastructure, the development of prototypes; and overall responsibilities of the Keeper Of The Project Vision (KOTPV) success-critical function identified in the 1988 Curtis et al. study of 19 large project practices and outcomes.

Many KOTPV functions require more than one person, often a combination of a domain expert and a technology expert. In some domains, a single person can provide such a combination, but it is rare to find a single person who can perform the KOTPV function across several domains, as Doug Rosenberg has done at USC. For the large-scale TRW projects described in Section III, the KOTPV function is provided by a team of experts, although there will generally be a KOTHV team leader such as Walker Royce for CCPDS-R and Neil Siegel for the series of large command-control systems.

The central Parallel Capabilities Development team generally develops the desired capabilities in a series of increments determined by the stakeholders, while providing updates to earlier increments as needed. The Continuous Verification and Valuation (V&V) team does not wait for some code to be tested. It starts at the beginning of the project getting ready to support the ICSM principle and PA Critical Success Factor "Evidence and Risk-Based Decisions" by evaluating the feasibility of the initial decisions of the first two teams.

They and the other two teams also reinforce the third CQF: a management approach emphasizing system qualities. This CQF has become more important for the very large government-contract project such as CCPDS-R and the Neil Siegel projects, as contract management strongly emphasizes functional requirements over non-functional (Quality) requirements. All three teams collaborate in identifying the key quality requirements and continuously addressing their degree of satisfaction. A Springer book on Parallel Agile is nearing completion.

III. SCALABILITY: CCPDS-R AND SIEGEL PROJECTS

The Command Center Processing and Display System-Replacement (CCPDS-R)was a million-line software project that finished on-cost and ahead of schedule. Its programming language was Ada, which enabled the project to develop[and consistency-check the architecture of the system and each of its components. This enabled the 47 developers to continuously develop components with the assurance that they would be system-compatible. They were complemented by 23 continuous V&V-ers and 11 extenders and evolvers of the system infrastructure. CCPDS-R is described in detail by its KOTPV, Walker Royce, in his 1998 book, Software Project Management.

The CCPDS-R CQFs were extended in a remarkable series of successful projects headed by their KOTPV, Dr. Neil Siegel. Over 20 years as a project manager through sector lead executive, he identified the key CQF as the entanglement of software functions and software controls. All of the projects finished on time and on budget, and all are liked by their users. They are described in detail in Dr. Siegel's 2011 USC Ph.D. dissertation, and will be elaborated in his forthcoming book.