An Untested SCM Technology Solution

By Richard "Dick" Carlson

Introduction

This paper is really a sad story about how a company was bullied into believing that a technology that was not proven not to be. The drivers included money and threats from the top-tiered company. I originally published this in a 2003 issue of CM Crossroads, where articles were published specifically for software configuration management (SCM) practitioners covering Agile lifecycle management (ALM) tools, change management, build management, and other topics.

While working at a company as the Director of Process Management in 2001, several issues evolved that affected not only the software configuration management (SCM) group, but the entire software development group. I made it my primary focus to develop a method that would reduce software build time and make the overall software development lifecycle more cost-effective.

The Problem

The challenge was to automate the company's many manual business processes using a new application development system for automating e-business change. The claim was that the system could have been developed in one year using Visual Basic and SQL server. However, company executives had made some extremely bold promises to several large and highly reputable companies who funded the development effort with the proviso that the company had to use their business applications and tools. This resulted in an agreement to incorporate a range of technologies, some of which were rather complex and untested, to build an application for the company that would automate real estate escrow and title accounting, and then market portions of the application separately. There was also a commitment to use the Unified Modeling Language (UML), the Rational Unified Process (RUP), and support a Java 2 Platform, Enterprise Edition (J2EE) architecture that would cure all woes and make things easier for all. I began to sense some serious diametric opposition from the executive level down through the team level. From that point onward I knew the first thing that had to be done was to become the solution champion.

The State of the Business

Chaos and an ad hoc atmosphere prevailed throughout the organization. Architects did their own thing, which mostly entailed dreaming up ridiculous (not very practical and expensive) system concepts and peddling snake oil to the CIO, while business analysts tried unsuccessfully at first to document business requirements and business rules. Developers were expected to interpret the verbose and very complex requirements from the unfinished and dynamic (constantly changing) requirements specification. The results were very predictable. That is, the developers believed that the users were witless, and the users refused to collaborate with "arrogant" software developers. So, the software never met user expectations. Things went from bad to worse when senior management determined that the developers they hired were slow learners, so they hired contractors. At the worst point, a total of 50 contractors who were getting $125 - $250 an hour, with no hourly limitations, were hired to "fix" the problems.

The SCM problem domain was recognized when more than 150 people working on the project were given the authority to do whatever necessary to make things right, which by the way, went on for at least 18 months. We started out with a small, inexperienced SCM group who worked their tails off in an attempt to make things work. There was no change control process in place, so new requirements

were inserted at will whenever an analyst agreed to add it into the release from an irate user. It took the SCM group a significant amount of time to understand the problem domain (which no one ever did in my estimation), and then manage object configurations within that domain. The budget was adequate, and the SCM group was not ready in a technical capacity to invoke or even establish basic and essential patterns needed to implement effective SCM. Designers were not really designated, so when someone thought up a new or seemingly revolutionary concept, the design was altered to reflect the new feature set or function. Software developers wrote their code without validating any functionality (no test cases were written), and then they threw their code "over the wall" (to the SCM group) for integration builds. At that point, things got real ugly when it was determined that the applications the development group was forced to use (and mandated by senior management) had interfaces that caused more problems than coding defects. Builds broke constantly. More contractors were hired. Still there was no change control.

The Solution

As the Director of Process Management, I developed an iterative development method that would save similar projects in the future, but gave little hope to this project when I briefed the executive staff on how it worked. It was based on Scrum, was immediately embraced, and was rolled out to the project with the understanding that the transition could not hamper or otherwise impede ongoing project progress. (Yeah, right! Like any progress had been achieved so far.) I trained everyone on Scrum to the entire project group and senior managers at an off-site location so their full attention was maintained. Extensive studies were conducted within the SCM group to determine why builds took so long and why so many of those builds broke. The results were interesting to say the least. Developers were checking in untested code, which meant that there were no peer reviews being held, little-to-no supervision, and a significant amount of gold-bricking done, because user expectations were not very clear. Requirements changed faster than developers could consume pizza, and those changes were not communicated throughout the development group. Developers did not always check in their code before leaving for home each night. Parallel development was not allowed. Branching was done late in the development game and just before a required build. The Product Manager wanted change control only after a successful build had been performed, which meant that baselines were nebulous at best.

A streamlined change control process was developed in flow chart format first, so I could walk everyone through the process and get their approval. Once the new change control process was in effect, things got a little slower, but builds improved. That is, they became more successful–not faster. I suppose the reason why it was slow was that people became more conscious of their work products before promoting code to SCM.

In spite of all this, the problem's roots were a combination of workflow processes, where significant improvements were needed, and in the technology being forced by executive management. I conducted intensive Scrum training to the entire IT staff, management, and even to the directors and C-level executives. The SCM solution and the J2EE application server were at odds. Mostly, there were serious formatting issues and the SCM group had to reconcile by reformatting) the code in order to complete the builds. This went on for several months before someone finally wrote some scripts that automated the reformatting process. There are a lot of things that could have or even should have been done before things got real bad and frustrating for the development group–but they didn't and work products suffered as a result. Millions of dollars were wasted on unnecessary contractual support that

could have been solved by better internal and external communications, and a better understanding and appreciation of the values that can SCM bring to any software project.

Conclusion

The major lesson learned in this case study was that all the money in the world cannot fix an ineffective process if people don't work together as a real team and collaborate closely on a daily basis. Looking at things in retrospect, SCM planning was never completed. Come to think of it, project planning was non-existent. Someone at the executive level should have done a little more homework and perhaps listened a little more carefully to what their staff recommended. Clearly, this scenario did not have a very good set of circumstances for any SCM or release manager, but opportunities to excel abound!

About the Author

Dick Carlson has been an active Agile transformational leader for many small and large projects, and has frequently shared his experiences of successful Agile, Lean, and Scrum implementations at conferences, workshops, and symposia, and regularly advises executives and organizational leaders on the cost, quality, and schedule benefits of using those initiatives and techniques. He has actively coached teams for more than 20 years on Agile and Lean Project Management fundamentals, and follows up with mentoring activities to ensure successful project execution. Dick has also provided concentrated Agile coaching support and led many organizations, programs, and projects that varied in size from six to more than 2,000 engineers, and that ranged in costs from under $50,000 to more than $1.4B.

Dick used Scrum practices and principles to manage and form the start-up of the Agile & Lean Education Associates (ALEA) Company that began July 2013. He continues to share his practical knowledge of Agile, Lean, and Scrum through more than two decades of experience by means of Agile and Lean training and the right amount of coaching to companies and other activities that want to increase their competitive advantage. The ALEA Company website (http://www.a2zalea.com) provides information about who we are and what we do.

