Crossroads News
A Monthly Publication for
Software and CM Professionals

The Need for Agility in SCM

Brad Appleton, Steve Berczuk, Steve Konieczka – May 2003

What is Agility?

Summarizing from last month's article *[1] Agility* is "the ability to both create and respond to change in order to profit in a turbulent business environment…. What is new about agile methods is not the practices they use, but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability."*[2]* Achieving agility while avoiding unproductive thrashing of the project requires high quality communication and tight feedback loops between competent team members and with project stakeholders. This allows the project team to iteratively grow the product architecture and functionality despite ever-changing customer requirements and priorities. Projects employing agile methods share the following key characteristics:

- **Adaptive** – plans, designs, and processes are regularly tuned and adjusted to adapt to changing needs and requirements (as opposed to predictive methods that attempt to develop comprehensive and detailed plans/designs/requirements "up front").
- **Goal-driven** – focus on producing end-results (working functionality) in order of highest business value/priority *[3]*. "Agile approaches plan features, not tasks, as their first priority because features are what customers understand" *[4]*. (as opposed to being document-driven or risk-driven)
- **Iterative** – short development cycles, frequent releases, regular feedback
- **Lean** – simple design, streamlined processes, elimination of redundant information, and barely sufficient documentation and methodology
- **Emergent behavior** – quality systems (requirements, architecture, and design) emerge from highly collaborative, self-organizing teams in close interaction with stakeholders.

What is Agile SCM?

One recurring theme from last month's issue ([1][5][6]) is that the key to understanding how to make SCM "agile" is recognizing SCM processes and practices must embody the agile values and principles in order to support and enable the characteristics of agile development. SCM processes and tools must eliminate bottlenecks in feedback cycles by:

- Easing up on rigid front-end controls to focus more on tracking and accommodating change rather than striving to prevent it.
- Eliminating redundant information and artifacts to focus upon the effective flow of communication
- Coordinating and streamlining development changes and communication, and …
- Automating back-end integration/build/test activities, increasing their frequency, and incorporating them into daily development tasks

Who are these people and why are they writing about Agile SCM?

We are Brad Appleton, Steve Berczuk (Steve B.), and Steve Koniecza (Steve K.). As Steve Berczuk and Brad Appleton were initiating discussion and researching new ways to enhance the documented patterns of SCM, they ran into Steve Konieczka who was researching and initiating discussion on the characteristics of SCM for the Agile Community. Both Steve B. and Brad had a natural interest in Agile in other areas of their careers and happen to believe that SCM is a key component to effective agile development. Once we crossed paths, we saw a common ground and decided to collaborate together on the topic of SCM and how it can provide agility to development teams.

This column is the result of our collaboration and we welcome input from the community. Our intent is to further discussion on the topic of SCM and ways that SCM can help development teams become more agile. This particular article is an introduction to who we are and concludes with comments on what to expect from the column in the future.

Steve Berczuk - SCM "Sympathizer" and Agile Practitioner
Early in my career I observed the power of software configuration management (SCM) appropriately applied. I was part of a Boston area development team that was collaborating with a group at the parent company's home base of Rochester, NY. As you can imagine, the issues regarding how we would manage the source code spanning the range from the technical to the social to architectural. Some of the issues that we had to resolve were:

- How to set up a distributed source management system; Because of network issues we needed to keep parts of the repository close to the group that was primarily responsible for them. We also had to come up with a build mechanism that worked at the moment and that was easily adaptable as the project components changed.
- What sort of check in and testing policy would work for both our team -- where most people were used to rapid changes -- and the other team -- which had a more staid, corporate, approach) and …
- How could we divide the work between the teams to minimize the effects of geographic distance on collaboration?

The details of the system don't matter, and one would probably do it differently today, but the system we had worked well. It helped us to work together more effectively, and seems not to interfere with the way that we worked.

As time went on, I came across more organizations where the SCM and version management systems were either ineffective at managing versions, or they seemed to hamper the software development process. SCM policies were determined by fear of avoiding past mistakes rather than with the goal of helping the team to get work done more effectively. Where there was a release engineering group, release engineering and development seemed to be at odds. They seemed to have different goals: Development wanted to change code, release engineering wanted to minimize change. The idea that we were all on the same team never seemed to make itself manifest. SCM seemed to be an obstacle.

Of course, in the end, regardless of what the SCM infrastructure was, the teams eventually produced useful products. But it seemed as if developers spent a lot of time and energy working around SCM procedures rather than treating them as part of the process. Things would have been better if the SCM process and the development process interacted in a more effective way.

Given my early experiences in seeing how SCM can be helpful, I realized that SCM can be a tool for agility. Now is a good time for me to explain that I come at SCM from a developer's perspective. I learned what I know about how to apply SCM because it was sometimes lacking in my environment, and I knew that it would help. My initial reaction is to view SCM from the perspective of how it affects day-to-day development, and that means a 'version management' centric perspective. Even from that limited perspective, developers need to understand how SCM is an essential part of their toolkit, and SCM practitioners need to realize that they share a goal with developers: Producing quality software as effectively as possible.

Steve Konieczka - SCM Consultant and Agility "Enabler"
The beginning of my career involved work with a large Big 6 consulting firm doing software development where I was exposed to many different approaches of SCM. Our success often hinged on the SCM solution employed. So when we started SCM Labs 5 years ago, we set out to do SCM with the needs of the

developer in mind. Our methodology for SCM comes from the developers' needs first - and we found that managing the corporate assets and meeting many of the standards naturally followed.

There are several SCM practices that I have always believed provided significant team productivity and software quality improvements. Over the past 5 years, however, I've experienced resistance with many of these, until I started working with Agile development teams. Some of these practices include:

- Fully automated builds with automated periodic integration and build processes.
- Automate as much testing as appropriate allowing as much regression testing as possible - then build this testing into your build process so that you can test as part of building software.
- Don't get bogged down on documenting the Configuration Items "outside" of the SCM solution - as soon as you're done documenting, you're out of date. Your SCM solution should keep that information tied to your releases and readily available.
- The SCM solution must implement "frictionless change" for the project - not roadblocks to change. SCM is not about "Controlling" change but keeping track of change. There's a difference.
- Focus the SCM solution on the needs of the developer with the "team" goals of writing the right software, on time and on budget - if you do this well, you will improve team velocity and software quality.
- SCM is not merely an administration role - if you're managing SOFTWARE, the SCM Administrator must have a deep understanding of how software is developed as well as how to administer the SCM tools being used.

There are many other areas of SCM that deserve discussion, but these seem to receive the most resistance. Over the past couple of years, I've started working with some customers who were trying out the new "Agile" methods like Extreme Programming, Scrum, FDD, etc. To my amazement, these teams were demanding many of the SCM principles that I was pushing with other customers with only limited success. They were demanding fully automated builds, built in automated testing, documentation at a high level those things that are stable and let the build system and version management system document the details. I also notice that everyone working on an Agile project is a member of the development team focused on a common goal - to develop quality software that meets the needs of the customer on a timely basis.

I believe in the Agile movement and I'm encouraged with much of the success we're seeing as it becomes more mainstream. I believe that the SCM practices mentioned above will help ALL development teams produce higher quality software in shorter timeframes. Agile projects offer an opportunity to showcase these practices and demonstrate the values for all software projects.

Brad Appleton - SCM "Anthropologist" and Agile Advocate
I began my software development career in 1987 as a part-time software tools developer to pay for my last year of college. Somehow it “stuck” because I’ve been doing some form of tool development ever since (particularly SCM tools), even when it wasn’t my primary job. I even worked for a commercial SCM tool vendor. I had aspirations of advancing the “state of the art” in SCM environments, but soon became frustrated with the vast gap between the “state of the art” and the “state of the practice.” I concluded I could do more good by helping advance the state of the practice to better utilize available tools.

Shortly after, I discovered software patterns and the patterns community with their combination of analysis and storytelling for disseminating recurring best practices. Unlike the “best practices” I was accustomed to seeing, patterns focused not just on the solution, but on the problem, and the context in which the solution made sense. For me, the problem's context and the forces were key to understanding why one person's “best practice” was often another's “worst nightmare” when improving the comfort and quality of one's work environment and processes.

I had previously amassed a wealth of knowledge about SCM tools and their implementation in both research and industry. Now I wanted to understand more about the rhyme and reason behind how to use them effectively. I spent over five years as an avid SCM "anthropologist", excavating for patterns in the use of over a dozen different well-known SCM tools and their user communities over the course of several hundred projects. During that time I met Steve Berczuk, and collaborated with him on several papers as well as a book of SCM patterns. We examined the connection between software architecture, SCM, and communication structures and observed how patterns of interaction and coordination have a profound impact on both software architecture and SCM "architecture" (and vice-versa).

When development and SCM responsibilities are segregated such that development essentially "throws it over the wall" to SCM, the two groups often take on an adversarial relationship: Many developers perceive SCM to be overly formal, rigid, and bureaucratic. At the same time, many software configuration managers and SQA professionals often perceive developers as undisciplined or ignorant of SCM concerns, constantly compromising product quality or integrity in the name of schedule or development speed.

The truth is that both sides are both right and wrong! The real problem is that they are on two different sides instead of on the same side. The integration/build "wall" is a barrier to effective communication and collaboration between developers and SCM. Something must be done to break the cycle and bring both sides together to tear down the wall! SCM is a "whole team" responsibility that must be part of regular day-to-day activities.

Software development is not some assembly-line process but is truly a knowledge creating activity that inseparably requires exploration and discovery throughout the entire lifecycle (not just in the early phases). What this ultimately means is: *Processes don't develop software. People do!* And while SCM process needs to help product quality and integrity, it is people that are at the center of creating software. So the process and the tools must first and foremost serve the people who practice and wield them (not the other way around).

These same conclusions are shared by agile development methods, which both myself and Steve had already been following for several years. Our combined interest in SCM for agile development is what led us to cross paths with Steve Konieczka (both on the scm-patterns mailing list and on CM Crossroads).

What you can expect from us

So united in purpose forged by our different but common experiences, the three of us (a developer interested in SCM, and two SCM developers) have banded together on a common mission to initiate discussion on topics that we believe will drive to higher quality software that meets the needs of our customers on a more timely basis. This CM Crossroads community is the place on the web for all of us to come together and discuss these topics and change our industry for the better.

The basic tenets representing our collaboration include:

- Agile SCM process must serve its practitioners and not vice-versa
- Agile SCM should break down traditional walls between SCM and developers to bring the whole team working together to accomplish our goals
- Agile SCM should be about responding to change rather than preventing change
- Agile SCM should track and coordinate development rather than trying to control developers
- Agile SCM should strive to be transparent and "frictionless", automating as much as possible

In the coming months, we plan on publishing some controversial concepts that we hope will spawn productive discussion. Please participate in this discussion because we will all be better for it. Some topics we plan to introduce in the coming months include:

- SCM and Team Velocity - explaining the details of how SCM can provide enhanced team velocity
- Multi-Level Continuous Integration
- Pitfalls (anti-patterns) and how to diagnose, recover/remedy and prevent

What we hope to get from you

We want your feedback and ideas about SCM and agility and growing and fostering a community to support agile SCM. We want to hear about your experiences, your concerns, your war stories and lessons learned, what has worked for you and what hasn't, and what vexes you most about reconciling SCM and agility and getting SCM and developers working together instead of against each other.

See you all next month!

References

[1] *Agile Configuration Management Environments*, by Brad Appleton; in CM Crossroads News - April 2003.

[2] *Agility* from **Agile Software Development Ecosystems**, by Alistair Cockburn and James Highsmith; Addison-Wesley, March 2002

[3] *The Agile Manifesto*, also appearing in *Chapter 1: Agile Practices*, pp. 3-11 of **Agile Software Development: Principles, Patterns, and Practices**, by Robert C. Martin; Addison-Wesley, November 2002

[4] *Agile Software Development: The Business of Innovation*, by Jim Highsmith and Alistair Cockburn; IEEE Computer: September 2001 (Vol. 31, No. 9), pp. 120-122

[5] *Making SCM Agile*, by Dick Carlson; in CM Crossroads News - April 2003.

[6] *Right-Sizing Software Configuration Management for Agile Projects*, by Steve Konieczka; in CM Crossroads News - April 2003.

Brad Appleton is co-author of **Software Configuration Management Patterns: Effective Teamwork, Practical Integration**. He has been a software developer since 1987 and has extensive experience using, developing, and supporting SCM environments for teams of all shapes and sizes. In addition to SCM, Brad is well versed in agile development, and cofounded the Chicago Agile Development and Chicago Patterns Groups. He holds an M.S. in Software Engineering and a B.S. in Computer Science and Mathematics. You can reach Brad by email at *brad@bradapp.net*

Steve Berczuk has been developing object-oriented software applications since 1989, often as part of geographically distributed teams. In addition to developing software he helps teams use Software Configuration Management effectively in their development process. Steve is co-author of the book **Software Configuration Management Patterns: Effective Teamwork, Practical Integration**. He has an M.S. in Operations Research from Stanford University and an S.B. in Electrical Engineering from MIT. You can contact him at *steve@berczuk.com*. His web site is www.berczuk.com

Steve Konieczka is President and Chief Operating Officer of SCM Labs, a leading Software Configuration Management solutions provider. An IT consultant for 14 years, Steve understands the challenges IT organizations face in change management. He has helped shape companies' methodologies for creating and implementing effective SCM solutions for local and national clients. Steve is a member of Young Entrepreneurs Organization and serves on the board of the Association for Configuration and Data Management (ACDM). He holds a Bachelor of Science in Computer Information Systems from Colorado State University. You can reach Steve at *steve@scmlabs.com*