Adobe Premiere Pro Scrum Adoption

How an agile approach enabled success in a hyper-competitive landscape

Peter Green
Agile Adoption Leader & Certified Scrum Trainer
Adobe Systems
Temecula, CA
tptmanpeter@gmail.com

Abstract – *Adobe Premiere Pro began an agile adoption in 2008, and has had gains in product quality, team work-life balance, and market fitness and responsiveness. This paper describes how an agile mindset has helped Premiere Pro make significant gains in market share and perception, improved quality, and created a sustainable pace of development for team members.*

Keywords – agile adoption; product ownership; quality improvements

I. INTRODUCTION

Adobe Premiere Pro is a non-linear video editor that competes against Apple's Final Cut Pro and Avid's MediaComposer for market share in key markets such as broadcasting, film / video editing, and corporate videography. Premiere Pro is a redesigned successor to Adobe Premiere, which was among the earliest non-linear editors (NLEs) when it was released on the Macintosh platform in 1991. Premiere Pro's first release in 2003 launched a fresh code base on Windows. Premiere Pro was successful, but lagged in market share and mind share behind its primary competitors after its CS4 release.

In an attempt to improve product quality, speed to market, and team engagement, Premiere Pro began to adopt an agile mindset and approach starting in 2008. Premiere Pro CS5, the team's first release using Scrum, represented a large improvement in product quality, market perception, and in the team's work-life balance.

II. IMPETUS FOR CHANGE

In 2005 and 2006, the Premiere Pro team began an effort to make the current Windows only product compatible on the Macintosh platform. This "Back to the Mac" goal was made feasible by Apple's switch to an Intel based architecture, but was still a colossal undertaking. The team used a more traditional, phased development approach in porting the code base. They spent more than 25% of the two year cycle fixing defects and trying to shore up the quality of the newly ported product in preparation for the CS4 release. This bug fix period, known internally as the "end game", put the team's capabilities to the test. In the final months of development prior to the scheduled ship date, some team members ended up in the hospital suffering from exhaustion and other effects of over-work. As is common in such a stressed system, the release quality was not as high as the team would have desired upon release, despite their heroic efforts.

During the same time period, the Soundbooth team, another product team in the same group as Premiere Pro, had adopted the scrum agile framework. Their experience of that release was vastly different. Their defect count was low, they had a very stress free end game, and they released a cross-platform 1.0 audio product with high stability and quality. The leadership of the video group investigated the results and, after taking a training course by Scrum's co-founder Ken Schwaber, held a retrospective of the CS4 cycle. Following the retrospective, Steve Warner, vice-president of engineering, Justin Cole, group program manager, and Simon Hayhurst, director of Product management, decided to move to an agile approach across the group. As Adobe's newly minted Scrum trainer, I was called upon to train the team, along with the other teams in the group, including the After Effects, Encore, and shared MediaCore teams.

III. INITIAL SCRUM ADOPTION

In the fall of 2008, the Premiere Pro team held a week-long CS5 release launch. The team began the week with two days of training in the basics of Scrum. Following this, the product management of the group helped communicate the two goals of the release, which were stability and performance. The team spent time building the initial product backlog out of the mostly technical goals for the release. The team agreed upon basic working agreements and how they would use Scrum. They chose a four week sprint cadence, built three cross-functional Scrum teams, and asked their Program Manager, Sheri Codiana, to serve as Scrum Master for all three. Their Senior Product Manager, Giles Baker, was chosen as the Product Owner, leading a group of managers that became a "Product Owner Council", which collaborated on product owner responsibilities. These responsibilities included reviewing feature requests, breaking large requests into smaller slices, and prioritizing items in the backlog. Giles, as the Product Owner, had the final say on any of these activities, but having the input and

collaboration of the Engineering Manager Paul Young, the Scrum Master Sheri, the Quality Engineering manager Laura Williams Argilla, and the User Experience designer Dave Kuspa proved key to keeping the backlog refined and ready for coming sprints.

As the teams began using Scrum, impediments began to surface immediately. Some of the major impediments included communication with remote team members, lack of skills in breaking down product backlog items into smaller items that still added value, and impedance mismatches with teams that had dependencies with Premiere Pro but that were not using an agile approach.

A. Communication with Remote Team Members

While many of the team members of the Premiere Pro team worked from Adobe's headquarters in San Jose, there were several members that worked in different locations primarily in the United States, including a group working from Adobe's Arden Hills Minnesota office and several that worked remotely from their homes in various locations. Such a setup can prove challenging when attempting to coordinate effective collaboration at meetings such as Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective.

One possible approach to this issue would be to construct the three Scrum teams as to largely collocated teams, such as a San Jose team, an Arden Hills team, and a Remote Worker team. While this may have aided in overcoming the communication challenges, the skill sets of the team members did not lend themselves readily, and so this approach was not chosen.

Instead, Scrum teams were created based on the best mix of skill sets and personalities, with only minor consideration of location. With this setup, there was concern that having some team members in an office while others were remote would result in a "home base" and "remote" separation of communication in the various meetings and conversations. Even when utilizing technologies such as video conferencing and despite everyone's awareness of the problem and agreement to try to avoid it, remote employees commonly had a difficult time breaking into conversations, sensing body language, and staying engaged in the meeting. Local attendees often forget to actively include remote attendees by soliciting their opinion or sensing when they are in agreement or disagreement. In essence, those in the room enjoyed the highest bandwidth form of communication possible: physical presence. Meanwhile those attending via technology were second class citizens from a communication bandwidth point of view.

To overcome this, the team decided to level the communication playing field. All team members were asked to use Adobe Connect, Adobe's desktop sharing and collaboration tool. The tool allows participants to share their web cameras in one pod, have a chat session in another, share a desktop with the feature tracking tool loaded in another, etc. Such sessions were sometimes referred to as the "Brady Bunch Daily Scrums", since as individuals logged into the meeting session, their faces would pop up in the video pod like the opening credits to the famous 70's television sitcom. This leveling of the playing field put everyone on equal ground. Attendees could see each other, hear each other, and no one benefitted from a higher bandwidth communication type than anyone else. This pattern is now used on other teams, and has been modified to take advantage of higher definition video and audio as those tools have improved.

B. Breaking Down Product Backlog Items into Valuable slices

One of the primary principles of agile development is to focus on delivery of small increments of value. This particular practice has proved one of the more challenging for mature teams adopting an agile approach, since it is very different from most of the developer's training and experience. Historically, developers were trained to decompose large problems into horizontal architectural layers and then to work on building out each layer, often with different groups specializing in one architectural area or another. To shift to a vertical slice approach, where the team takes the user problem and breaks out a very thin slice of value that traces through architectural layers, proved challenging for the Premiere Pro team as well. This problem initially seemed even more challenging than a typical release, since one of the primary release goals was to create a 64 bit version to utilize more memory space, and to transition from a Carbon based architecture to a Cocoa based one on the Mac, due to Apple's decision to deprecate the former technology.

To begin attacking this problem, a Q&A session was held between developers from the Soundbooth team, who had been using scrum for nearly two years, and the Premiere Pro team. This provided an opportunity to brainstorm some solutions to the problems, as well as to provide some reassurance from peers that yet, it is possible. With follow up coaching, the team began looking at ways to slice their backlog vertically. The team decided to incrementally build out the 64 bit Cocoa version in vertical slices. For example, at the end of the first sprint, they had a goal to have a release quality version of the application frame with just the Project Window up and running. This approach allowed full testing of features very early on, whereas if they had taken a more traditional approach to the port, testing would have been hampered until all of the backend and UI had been ported and joined together. The Premiere Pro team's experience with this port proved to be a model for future teams, and a proof point that nearly any engineering problem can be solved with a value focused, vertical slice approach.

C. *Working with non-agile teams*

The Premiere Pro product is part of Adobe's Creative Suite, which includes several other products such as Photoshop, Illustrator, Acrobat, and many others. The suite is not simply a collection of products – there are several features that integrate the products when used together. This provides increased benefit to users, but does come at the cost of having to coordinate plans, technology, and schedules across the many products in the suite. In addition to product integrations, Adobe has developed several core technology components that are utilized across products in the suite. For example, all of the video products utilize the same media playback engine, and the graphics products utilize a shared type engine. Tight coordination of the work of the dozens of teams that build the point products and shared components of the Creative Suite is a challenging management task. Finally, there are several core services teams which provide centralized legal, globalization, documentation, and other such services to all of the the product teams, both agile and traditionally managed.

The coordination challenges of such a complex system are made more difficult when the teams use different development approaches with different methods of planning, tracking, measuring, and change processes. The teams attempting to move to an agile approach within the Creative Suite caused such a mismatch in approaches.

The Creative Suite used a traditional Product Life Cycle (PLC). Each team building parts of the Creative Suite were required to coordinate large presentations of plans and status updates at major milestones. These milestone review meetings were attended by management of the component teams and senior executives. Planning milestones included Concept Accept, where teams presented high level plans for target markets, major features for the release, Product Requirements Document, where teams provided more detailed plans about the features for the release, and a Project Plan Commit milestone, where teams presented their plans for how they would implement the features described in the CA and PRD milestones, along with scheduling, resourcing risk mitigation, and process plans. During the development phase, two major status check milestones were utilized (Progress To Plan), wherein teams presented to senior executives current status and any major changes to the original plan. As the pending release approached, a final major milestone called Release Readiness brought the product development teams, marketing groups, and several others together to discuss fairly final plans for release – any last minute changes to planned feature sets, marketing plans and timing, and other coordination required to release such a massive product. Individual teams often utilized other milestones internally, such as Feature Complete, a milestone when they stopped developing functionality and switched to regression testing and bug fixing, Release Candidate, when all regression testing and bug fixing was complete and all integration of shared technologies was done, and Golden Master, when media and deployment testing was complete.

The group responsible for managing Adobe's PLC processes made several changes in an effort to accommodate teams wishing to use an agile approach. Several milestone definitions were made much more broadly, and milestone names were changed to reflect an incremental approach. For example, Progress To Plan milestones were a replacement for the Alpha and Beta milestones. Despite the changes, years of habit meant that agile teams had to over communicate to the dozens of teams that they coordinated with.

An example of these challenges can be found in the amount of time scheduled between the Feature Complete and Release Candidate milestones, the time period described above as the "end game" by many Adobe teams. In a pure agile approach, teams achieve both feature complete and Release Candidate at the end of iteration. No additional time period should be required for more regression testing and bug fixing. Taken in isolation, the Premiere Pro team may have had a very good chance of, if not eliminating its end game, at least significantly reducing it to one or two "hardening sprints", as they are called, wherein finalization tasks that have a high cost of iteration such as performance and media testing are performed. Since the team integrated many shared components that did not use an agile approach, the team had to plan to do several integrations during its end game. They chose to utilize the same time period for their end game as they would have prior to using scrum, both to take into account the shared component integration testing, as well as a fall back plan in case they were unable to deliver release quality code at the end of each iteration.

This plan resulted in a bit of a self-fulfilling prophecy – since they had time in the end game planned into the schedule, it became easier to choose to defer quality decisions until later. Additionally, due to the dependencies inherent in the system, the team was sometimes unable to change directions from sprint to sprint. If a change in plan required a change in any shared component, and that shared component was not using an agile process, it was difficult to accommodate, having to go through a traditional Engineering Change Order (ECO) process with the dependent teams and causing disruption to those teams' plans. In essence, the flexibility of the entire system of teams needed to be throttled to the flexibility of the least flexible team within the system.

Improvements in this situation have happened for a large part of the system, namely the shared components that were built within the same group such as the shared media playback engine and shared file exporters, since those teams have also adopted an agile approach. However, during that first cycle, they were not able to overcome all of the challenges of coordinating with non-

agile teams. In recent releases, and as more teams within the company have adopted an agile approach, this situation has slowly improved. There are plans in place for future releases to require all shared components in the suite to adopt a release train approach, wherein they all reach release level quality at least monthly. This move has been accelerated by Adobe's business strategy changes moving away from perpetually licensed desktop software and towards subscription based pricing and a mix of desktop and "Software as a Service" offerings.

IV. Results of Scrum Adoption

Scrum adoption led to improvements to product quality, work/life balance of the team members, and fitness for customers. The number of variables that differ between two large releases make it difficult to measure and attribute causality to any single change in process. Changes to market conditions, technology, and personnel can have a large effect. However, we have attempted to measure certain aspects of the pre-Scrum and post-Scrum releases for several Adobe teams, and have seen trends in a few areas [1]. We will describe results in the key areas of quality, team perception of improvement, and market perception of the release.

A. Quality Improvements

The Premiere Pro team did not adopt any new engineering techniques coincident with their Scrum adoption; therefore, we have some confidence that the quality improvements made between the CS4 and CS5 releases are largely attributable to the team's adoption of Scrum, specifically the following four practices.

1. Slicing work items into small, value focused pieces,
2. Agreeing to a Definition of Done, a quality bar that all items must meet in an iteration in order for them to be considered complete and demonstrated,
3. Building cross-functional scrum teams with both engineering and testing expertise, and
4. Scrum team discipline in actually reaching the Definition of Done for each item in each sprint. We don't wish to imply that the teams were perfect in each of these practices, only that they consistently strove to meet these goals, and to improve their ability to reach them in each iteration.

One challenge with measuring the impact of a process is that many teams don't use the same types of metrics for different process types. One exception to this rule is the tracking of defects. The Premiere Pro team kept detailed metrics on open defects at any given point in the cycle. An examination of this data shows us one way that the Premiere Pro team's quality improved after implementing Scrum.

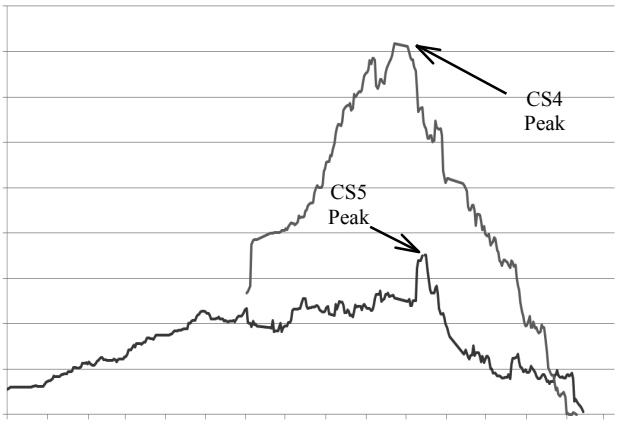

Figure 1: Open defects over time for CS4 and CS5.

The illustration *Figure 1* shows the total open defects graphed over time for two cycles, CS4 and CS5. The X axis represents time in months, backdated from the Release Candidate Declare milestone. The Y axis represents total open defects. The graph illustrates a total peak defect count for CS5 that is only 43% of the total number of defects during the same time in CS4. The Premiere Pro team has three ways to close a defect: fix it, defer it, or close it. Deferral is used by some teams to indicate that they believe it is worth the time spent to fix the defect, but that for some reason they do not have time to fix it in the current cycle. Closing a defect indicates that the team has decided that the impact of the defect to users is so minimal that it is not worth the effort to fix it, as may be the case for some edge case defects that are very difficult to reproduce.

When viewing the graph, it is important to consider whether the lower defect counts were achieved by a) fixing more defects, b) deferring or closing more defects, or c) having fewer defects injected into the system. In CS5, the team fixed 20 fewer defects per month than in CS4, so a) is not the cause. In CS5, the team deferred 5% fewer defects than in CS4, so b) is not the cause. This leaves us with the conclusion that fewer defects were introduced into the system. We attribute this to the better decomposition of work items, leading to better understanding by the team and the ability to test much earlier, the cross-functional team approach, wherein plans are made with developers and testers together, helping to prevent defects before the code is written, and the team working at a sustainable pace, where developers are not under undo pressure to meet unrealistic goals. When we also consider that the team agreed to meet quality criteria for each item, and that they did not receive "credit" (story points towards their velocity and the opportunity to demonstrate the functionality), we see an added incentive for the team to build in better quality.

B. Team Perception of Improvement

The first value of the Agile Manifesto is "Individuals and Interactions over Processes and Tools" [2]. A related principle from the Toyota Production System is "Respect for People" [3]. Accordingly, one important way to understand if a system has improved is to ask the people that use the system. We surveyed the Premiere Pro teams to gauge their impressions of how things had changed since moving to Scrum at two points: 12 months after initial adoption, and six months later at 18 months after adoption. The answers and the change between the two surveys show the team's growing appreciation for the benefits of an agile approach.

In the survey, we asked them to rate how strongly they agreed with various aspirational statements on a scale of 0 (Completely Disagree) to 10 (Completely Agree). The results to some of these survey questions are provided in Figure 2.

Statement	12 mos.	18 mos.
The Quality of our product has improved since moving to scrum.	6.5	8.2
The Communication on our team has improved since moving to scrum.	7.2	7.83
We deliver a better product to customers since moving to scrum.	6.6	7.75

Statement	12 mos.	18 mos.
If the decision were completely up to me, we would continue to use scrum for Premiere Pro (%Yes/%No)	77/23	80/20

Figure 2: Premiere Pro responses to Adobe scrum survey

It is interesting to note the difference between the results at 12 months compared to those at 18 months. At 12 months, the team was at the most challenging part of their schedule. They were completing their final sprint prior to moving into the aforementioned "end game" period. They knew that this sprint was their last opportunity to add value (new feature improvements) and that they would then shift into regression testing and bug fixing mode. This is often a challenging time for teams working on a long release as reality sinks in that we won't get everything in that we had hoped for. Additionally, though the team had been reaching an agreed upon "definition of done" for all completed items in each sprint, keeping their defect count low, there was concern that once they moved into more overall workflow testing, additional defects would be discovered. The 12 month results show that the team felt that quality had improved slightly (6.5 out of 10), communication had improved nicely (7.2 out of 10), and the overall product was somewhat better (6.6 out of 10). At 18 months, all scores had improved to a very strong 8.2 out 10 for the improvement in quality, a modest increase to 7.8 for communication, and another strong score of 7.75 for the improvement in the overall fitness of the product. Additionally, the percentage of respondents that said that they would continue to use scrum if it was completely up to them had increased from 77% to 80%. We attribute these improvements to the team's experience of a complete 18 month release cycle using scrum.

In the pre-scrum release of CS4, Figure 1 illustrates the extreme peak of defects that the team was required to address prior to shipping the product. This tremendous burden had two negative results. The first result was that in the drive to resolve the defects: accumulation of technical debt, and impact to team members' work/life balance.

In order to address the sheer number of defects present in the CS4 cycle, the team often chose quick resolution with long-term costs over slower short-term resolution with longer-term benefit. Examples of such decisions included a) deferring defects to be fixed in a later release despite the uncomfortable knowledge that the defects would likely impact customers, b) choosing to fix the symptom of a defect rather than its root cause, and c) when fixing a defect, choosing not to refactor that code area for cleanliness in order to complete the fix more quickly. These types of decisions result in a type of technical debt in the code base – we get to market more quickly, but at the long-term cost of having to pay off the debt. Due to the nearly logarithmic increase in the cost of fixing defects the later one waits[1], the results of technical debt are similar to those of financial debt with compounding interest – the longer one waits to pay the debt, the higher the cost [4].

The second result of this large queue of defects was that in order to ship, the team needed to make heroic personal efforts, often working tremendous amounts of overtime in order to meat their goals. Adobe does not track overtime, so we don't have any specific data regarding how many extra hours were worked. We do have the most extreme results of that effort, which is that several members of the team ended up in the hospital suffering the results of exhaustion by the end of the release.

In contrast, CS5 had no such mountain of defects, or to use an analogy, there was a hill of defects only 43% as high as the mountain climbed in CS4. As previously mentioned, the team had cautiously chosen to keep the end game length at six months, a caution well warranted given the technical risks of the release, including porting from Mac's Carbon to Cocoa technology, porting from 32 bit to 64 bit, and attempting to address major performance improvements with underlying engine refactoring, all without significant automated test coverage in place. Additionally, the nature of Premiere Pro makes it inherently difficult to build automated regression and unit tests for major pieces of the product, since simple business logic checks don't suffice when the output of a user's actions may be a slightly different hue in a background of

a multi-gigabyte video file. As the agile coach for the Premiere Pro team during that cycle, I attended a meeting with the product's management team just prior to the final sprint review of the last feature development sprint of CS5 before moving into the end game. I had the defect data with me, and in preparation for the end game, was considering the possibility of encouraging the team to continue developing new functionality, given the marked improvement in defect counts compared to previous cycles. As an experienced coach, I was well aware that simply recommending such a course of action is rarely effective, and instead, I began the meeting by showing the defect comparison graph to the team, and asking Paul Young, the engineering manager, what he was going to do with the defect rate being so low. I knew that Paul was aware that previous teams that had adopted Scrum and had experienced similar improvements had chosen to add feature development sprints to their schedule prior to moving into the end game, effectively increasing the ratio of feature development to defect reduction for their cycles, and I expected that given the similar situation, Paul might suggest the team take a similar approach. Instead, Paul said, with a bit of emotion "We are going to have the best release of Premiere Pro ever". After further discussion, it became clear that Paul's plan was to spend the six months doing what they had never had time to do previously, which was to refactor the code, make it clean, and reduce defects instead of defer them. For a team that had struggled for years with quality due to extreme schedule pressure, this was a major change, and one obviously welcomed by the team.

C. Market Preception

As mentioned previously, Premiere Pro's major competitors include Apple's Final Cut Pro and Avid's Media Composer products. Avid is the traditional market leader in up-scale markets, with a strong historical presence in feature film editing and broadcasting. Apple had begun to chip into Avid's lead in these markets over the course of several releases. Premiere Pro had slowly begun to move into these markets, but was still strongest in other lower profile markets such as corporate video production.

The release of CS5 coincided with a rare competitive stumble by Apple. Their own re-write of Final Cut Pro, released as Final Cut X, resulted in major features being cut, features which professional broadcasters and editors required, resulting in a shift in market perception [6, 7]. With Premiere Pro's marked improvements in performance and stability, they were poised to take advantage of this opportunity by moving up-market with these customers.

Adobe's internal Marketing Insights & Operations Team has tracked various aspects of market perception over the course of a few years. A study conducted after the release of the new versions of Premiere Pro and Final Cut Pro included a survey of customers in North America that used both products (what the study calls "Dual Users"). They were able to compare this new data with data from the previous release to assess the change in market perception. In the previous release, Final Cut Pro held an 18 point lead over Premiere Pro in overall opinion. After the new release, the positions were swapped: Final Cut Pro dropped by 13 points in overall opinion, and Premiere Pro increased by 14 points in theirs, resulting in Premiere Pro holding a nine point lead.

In this same survey, dual users of Premiere Pro and Avid's Media Composer, which was still the industry leader at this time in the feature film editing market, were surveyed for the first time. Premiere Pro lagged Media Composer by only six points in overall opinion, establishing an initial benchmark but showing only a narrow gap.

This survey also used the industry standard Net Promoter Score question Net Promoter Score[8].[6] In this measurement, Premiere Pro lagged Final Cut Pro by 21 points prior to the CS5 release. After CS5, Premiere Pro gained 11 points in their NPS among dual users, and Final Cut lost 11 in theirs, resulting in a Premiere Pro score one point higher than Final Cut Pro. This illustrates that Premiere Pro's gain was partially due to their improvement and partially due to Apple's regression. For dual users of Premiere Pro and Avid Media Composer, the initial NPS results have Premiere Pro leading Avid by six points.

V. SUMMARY

Scrum adoption helped the Premiere Pro team improve in several areas. The team was enabled to improve code quality, deliver better features for their customers, and work at a sustainable pace, greatly improving their engagement at work. Customers of the new releases of Premiere Pro have shown a marked improvement in perception and likelihood to recommend, coinciding with a decrease in the same scores for Premiere Pros' primary competitors.

The approaches observed in the Premiere Pro team may be interesting patterns for other companies and teams in similar situations to try out, and are shared below.

- Agile adoption at Adobe has been almost entirely a grass roots movement – the Premiere Pro team chose to move to Scrum based on the experience of another early adopter team (Soundbooth CS3) in their same group. Other teams at Adobe have chosen to adopt Scrum after seeing Premiere Pro succeed. This pattern has proven successful thus far, and awareness of what agile is and what needs to change have slowly moved up from team level to director level over the course of a few years, just now beginning to reach the Vice President level.

- For large teams, a Product Owner Council may help match the teams' demand for high quality, refinement, and thinly sliced product backlog items. When creating such a group, we found it was important that one member of the Product Owner Council was the official Product Owner for purposes of accountability and decision making authority.
- Level the communication playing field for distributed or non-collocated teams. If one person on a team has a low communication bandwidth limit such as phone or video conferencing, the whole team throttles their communication paths to match that bottleneck, avoiding a mismatch that can hinder collaboration.
- Nearly any software development problem can be decomposed into "vertical slices," small, user-facing increments, which will fit into a sprint of four weeks or less. However, this is a paradigm shift for experienced developers who have been trained to decompose large problems into "horizontal" or architecturally based slices. Help in this paradigm shift from experienced coaches is very important to the success of any agile adoption.
- For large programs or suites of products with shared component architectures, the ability to reach release quality for the entire system in a given time period will be throttled to the ability of the slowest component to reach release quality in that period. For Premiere Pro, this meant that they couldn't abandon their "end game" completely as various shared components that their code base relied upon still required this activity on the old schedule. The team still chose to reach release quality for all items under their direct control, and to begin to influence shared component teams to take a similar approach. Over the course of four years, this has created a ripple effect, where the components most required by the system moved to a Scrum approach for one large release, and other more extended sets of components are just now moving to the same approach. Without executive direction, expect this to take some time, and improve the parts of the system over which you do have control.
- While additional engineering practices such as those from eXtreme Programming [9] increase a team's ability to make marked improvements in code quality, the Premiere Pro team saw impressive improvements using only the Scrum framework and a vertical slice approach to product backlog items. We wholly endorse the adoption of agile engineering techniques. We also observe that even without these techniques, teams can make significant improvements by simply starting with Scrum.

ACKNOWLEDGMENT

To Justin Cole, who encouraged me to share this experience report from Adobe's own perspective after a high level version of the story that left out some important details was published in Ken Schwaber and Jeff Sutherland's book titled "Software in 30 Days."

To the Audition and Premiere Pro teams and Adobe's Digital Audio and Video organization management, for helping me to learn how to be a better trainer and coach and for having put up with my crazy ideas for many years now.

To Erica Schisler, who taught me how to be a good servant leader as a new program manager, and encouraged me to pursue better ways of delighting our customers in the face of sometimes frustrating corporate inertia.

To Marian Willeke, for her keen insights and valuable help in refining this paper as a shepherd.

REFERENCES

[1] Green, P., "Measuring the Impact of Scrum on Product Development at Adobe Systems," IEEE Xplore System Sciences (HICSS), 2011 44th Hawaii International Conference on , Jan 2011, pp. 1-10, doi: 10.1109/HICSS.2011.306

[2] Beck, K. et. Al., "Agile Manifesto", retreived from http://www.agilemanifesto.org

[3] Toyota internal document, "The Toyota Way 2001," April 2001

[4] Jones, C., "Software Assessments, Benchmarks, and Best Practices," Boston, Addison-Wesley Professional, 2000.

[5] Fowler, M., "Technical Debt", retrieved from http://www.martinfowler.com/bliki/TechnicalDebt.html.

[6] Biscardi, W., "FCPX: What Pros Find Missing from Final Cut Pro X", retrieved from http://magazine.creativecow.net/article/final-cut-pro-x-whats-missing-for-some-pros

[7] Radovsky, D, "Last Call for Final Cut?", retrieved from http://www.fool.com/investing/general/2012/01/25/last-call-for-final-cut-part-1.aspx

[8] Reichheld, F., "One Number You Need to Grow", Harvard Business Review, Dec 01, 2003.

[9] Jeffries, R. et. Al., "Extreme Programming Installed", Boston, Addison-Wesley Professional, 2000.