When the Business Wants Waterfall: Adaption Strategies

MARJORIE FARMER, Halliburton

This work discusses how the software team at Halliburton has adopted Agile methodologies while still complying with the corporate waterfall mandate. The following are addressed: Changes encountered, both successful (e.g., engaging users) and not as successful (e.g., attempting the transition without coaching); how compliance was handled with the waterfall process, particularly with respect to extensive changes of scope during development; communication with management; Agile coaching and its success; and challenges and success experienced restructuring the roles of an Agile team.

1. SITUATION

In January 2014, I accepted the position of leader of the wireline software team at Halliburton. I joined this team after spending 15 years with the Halliburton software division as a developer, project manager, and development team manager. Most recently, I had worked with all of the business units to implement a software specific adaption of Halliburton's heavy waterfall process, referred to as LIFECYCLE.

The wireline software team consisted of 16 members, 14 in Houston, two in Singapore, and 16 offshore contractors in New Delhi. Of these, the junior most had 15 years of experience with Halliburton and the most seasoned had at least 25 years of experience. These were highly experienced, very capable software developers and testers with an intimate knowledge of the business and its needs. Several had worked in other roles in the business unit, many as engineers. Upon taking my new position, they had an established way of doing things and consistently delivered a quality product. However, they weren't viewed as flexible, and management didn't have good visibility of what they were doing or why. Management had made numerous changes to the entire wireline technology department to improve efficiency and technical capacity, and the software team was not part of this.

My mandate when I began this position was to shake things up. Management wanted to see increased openness, flexibility, and efficiency, which needed to be achieved without interfering with the team's existing ability to deliver functional software. The management perception was that the team delivered critical value, but they wanted more. One of the Singapore team members resigned before I arrived, and I inherited two open positions. At the time, the team was talking about becoming Agile, but had only begun tracking work progress in two-week sprints.

1.1 The Corporate Environment

When I joined the wireline team, Halliburton as a whole had adopted a heavy waterfall process, called LIFECYCLE, which all technology projects were expected to comply with, including software projects. This process was intended to increase the success of all technology projects, and was primarily focused on the delivery of expensive, physical tools, where prototypes might cost USD millions and minor changes could be very expensive. This process theoretically permitted software projects to make use of an Agile framework while still complying with company policy. In theory, the software team would deliver a couple of light-touch gate reviews to upper management, and then use Agile as an execution methodology. However, this approach had not yet been widely tested, and management buy-in outside the software teams was weak.

The following process components were mandatory for all Halliburton projects, including software projects:

- Gate reviews: using a defined set of template slides.
- Risk management: using a defined risk spreadsheet.
- Business case: using a defined business case spreadsheet.

Marjorie Farmer, Halliburton, 3000 N. Sam Houston Pkwy. E, Houston, TX, 77032; Marjorie.Farmer@halliburton.com

Other aspects of the process were expected, but not explicitly mandatory.

1.2 The Product

The team was responsible for one product, called Wireline InSite® services (WLI), which handled the real-time collection of wireline logging data from logging tools deep in oil wells. The product was business critical, with constantly changing requirements, and contained approximately ten million lines of code in multiple languages. A logging tool is a complex and highly scientific instrument, collecting data miles under the earth at megabytes per second by bouncing various signals off the overheated rocks below. Scientists spend years designing these tools, and other scientists spend years developing algorithms to translate pulses of sound, nuclear, or magnetic energy, into useful information regarding the properties of the surrounding environment. These algorithms, once developed, must be implemented as functional software and made to work with the actual, physical tools. This was the job of the wireline software team.

The WLI project had special challenges because requirements were driven by multiple tools projects. These projects could insert extreme changes in the software project scope and schedule on short notice as the tools projects changed requirements and slipped schedules. This also added constraints because effective testing of the software required access to the physical tools, and tool availability was limited. When a tool became available, all other priorities would need to be rearranged immediate so testing could be performed before other priorities took the tool elsewhere again.

To handle the product, the team organized itself into two groups. The tools team handled the coding that was specific to each tool and the implementation of algorithms as functional software. The software architecture tended to be consistent from one tool to another, and the major software challenges involved implementing the complex scientific algorithms that collected and processed the incoming data. The systems team handled the infrastructure and architecture of the software itself, as well as all functionality that did not directly involve communication with a physical wireline tool. When I joined the team, the WLI product was generally released every six to nine months with whatever functionality was working when the release date approached.

2. IMPLEMENTATION

2.1 Getting Started

When I joined the team in the beginning of 2014, I was already a strong supporter of an Agile software development approach because of my previous experience with other projects. I was determined that my team would convert to Agile. The team was not convinced, but I was their new manager, and they were willing to give it a try.

The conversion did not go smoothly.

When I worked out the adoption plan with my team leads, everyone agreed that the first step in our adoption was to adopt the latest version of Microsoft Team Foundation Server (TFS), which housed both the code and the work items. We were getting some pressure from information technology (IT) to migrate anyway, and the latest version of TFS included functionality for supporting Agile. That turned out to take far longer and involve far more work than anyone expected; but, a couple of months later, we finished our upgrade.

While the team was working on upgrading TFS, I was working on staffing. I was able to hire an experienced user of the product, an engineer fresh from the field, to be product owner in Houston, as well as a less experienced product owner in Singapore, and a developer in Singapore. I also began the process to move one of my contract testers from New Delhi to Singapore. This would provide two developers, a tester, and a product owner in Singapore, which was a manufacturing center where the team had access to the physical wireline tools. My hope was to have the Singapore team do much of the management of the contractors in New Delhi from a more compatible time zone than Houston.

Once the TFS upgrade was finished, we began our wholesale adoption of Agile. We started conducting standup meetings, backlog grooming, story sizing, sprint planning, demos, and retrospectives. The result of this was that, because the team spent so much time in meetings, they barely had time to get any work done. There were several issues that the team struggled with, such as:

- Backlog grooming: the team didn't know how much detail to provide in a story and therefore invested a large amount of effort in providing far more detail than was actually necessary. The team spent hours grooming, and often stayed late into the evening. Then, in the morning, someone always made sure I knew how late they had stayed, and appreciated how hard they were working to adopt Agile.

- Process understanding: everyone on the team had their own idea of how Agile was supposed to work and much of the time in the meetings was spent arguing about process questions. What exactly is a product owner and what is he doing here? What does a scrum master do and who is going to do these things? What kind of release planning document do we need? How do we do acceptance criteria? Do we really have to stand up in a stand up meeting?
- Story sizing: no one really understood how story sizing was supposed to work and so the team just used estimated hours of effort. Also, initially, the stories were too large (sometimes far too big) and tended to run over into multiple sprints.
- Definition of done: this generated a lot of vigorous discussion. The team decided that a definition of done is useful; but, don't we also need a definition of when a story is ready to go into a sprint in the first place? And, what do we do with these stories while they are in various states; and how many backlogs should we be keeping? What are the rules for moving a story from one backlog to another? And, how do we make the work management tool enable all this? The team actually sorted this one out themselves, after multiple lengthy discussions, but they probably would have been a lot faster if they had had someone to talk to about what other teams have done.
- Meeting attendance: several of the team members, mostly developers, ended up spending a lot of time in meetings they didn't consider relevant to them; this generated a considerable amount of unhappiness. Why does Bob need to be in a meeting with Michael when Bob doesn't care at all what Michael is doing and couldn't help anyway, because Michael deals with nuclear physics and Bob is an expert in sound? And, besides this, Bob had much more important things to be doing than listen to Michael figure out story sizing in yet another meeting. And, why does anyone need to know every day what he's doing anyway; Bob wants to know. Hasn't he been doing his job just fine for the last 30 years?

I initially expected to deal with these sorts of issues by supplying training and by providing coaching myself. I was able to arrange the two-day scrum master training class for most of the team; however, that turned out to be insufficient. My own coaching didn't turn out to be sufficient either, as I wasn't that great of a scrum coach, and I also struggled with time availability because of my other responsibilities. As the struggles continued, the team worked extra hours and grew increasingly unhappy. There began to be occasional shouting matches between team members and the complaints became louder. I knew I needed to do something more.

Fortunately, this was in early 2014 when the price of oil was hovering at approximately USD 100/bbl and I was still in my honeymoon period with my own management. I had some money to work with. And, I knew a good scrum coach. I called up Simon Orrell, from Snowdolphin, and he came to Houston and assisted the team. He was able to get the team aligned with respect to how the process was supposed to work. After just one week of coaching, the team was delighted with him and much happier with Agile. Half the team came by at some point to compliment or appreciate Simon. With several more sessions from him, I expected the team to become comfortable with Agile in short order.During all of this, the team acquired a new project, the Halliburton Perforating Toolkit (HPTK). This was the first additional product the team had taken on, ever, and was an example of the team's new openness and flexibility.

2.2 Observations

The initial adoption of Agile by the team turned out to be significantly harder than I expected. I had acquired my own Agile skills gradually, over a period of years, and I underestimated how hard it would be to pick it all up at once. I knew Agile worked because I had experienced it, but I didn't know how to get there. The guidance from a capable Agile coach made the critical difference and, without that, I think we would likely have given up within another couple of months.

I never really managed to acquire a good scrum master. I assigned various team members to the position, but the role never aligned well with their skill sets and the gap was a constant pain point.

I did not have strong buy in from the team when I started forcing the transition. They complied because I was their manager, but I doubt they would have continued to stay with it during the hard parts if I hadn't taken other steps to earn credibility with them. While I set the overall direction of adopting Agile, I did a lot of listening about how we did the actual implementation. Additionally, I took a number of steps as a manager to look out for my team, and they recognized that. For instance, I acquired several overdue promotions of team members, arranged a company-funded snack drawer for the team, and ensured they earned a fair share of awards when awards were being handed out. We held team parties for releases and service anniversaries and so forth. They ensured I knew all about the pain they were suffering, but they kept going. Also, I think I had an advantage because of the age and experience level of the team. They were a level headed bunch and weren't thrown by the initial rough spots. They had experienced a lot of "management initiatives" over time and were willing to patiently maintain cooperation (mostly) until it either succeeded or failed.

Once the Agile coach provided his help, a number of pains became sorted. The team was able to do their planning at the right level of detail, and he managed to align everyone on how the process was supposed to work. The meeting load became manageable. He sat with them and worked out a list of example stories of different sizes to be used for reference.

The earliest signs of success came from the testers, who liked to hear at the standup meetings what the developers were working on. Shortly after that, the tools team development lead, who had initially been quite skeptical, also began expressing his appreciation for the standup meetings, as a way to get a regular update from his own people.

One thing we never had serious trouble with was standup meetings. I think this was because, from the beginning, we actually made everyone stand up. Even when the teams were large, the meetings never went long. For a little while, we charged a quarter to anyone who showed up late, but that faded as people got used to the meeting. The standup meetings were the first Agile practice that the team saw value in because of the enhanced communication.

3. ADAPTION STRATEGIES

3.1 Checking the Boxes

While the team was busy figuring out Agile, we still had to live in a waterfall world. We managed to make that work, and even raised the team's reputation as a highly performant team through a number of strategies.

3.1.1 What we Did

A certain number of practices were explicitly mandatory according to the corporate LIFECYCLE process. All projects were required to do gate reviews, risk analysis, and business cases. Other components of the LIFECYCLE process were left to the discretion of the manager to determine applicability to any given particular project. My approach to these was to comply, well, with all the mandatory process, while shielding the team from them as much as possible. I took advantage of the discretion allowed to the manager for the remainder, and evaluated the applicability of each of these to my team's project. Only those parts of the process that were both applicable and added value were adopted.

For software projects, only two gates were required. The first was the gate between planning and execution when the project was funded. The second was the release gate, which provided approval to complete and release a product. I was fortunate in that I had a lot of experience performing gate reviews and had a good idea of the sort of information our stakeholders would be searching for. Our challenges with the gate process was that our boundary between planning and development wasn't as crisp as waterfall expected, and our scope changed wildly between planning and release. Our advantage was that, because the scope was business critical to deliver the physical tools, everyone knew we were going to fund the product, so the gate was more of a formality. It was a good communication vehicle and it forced some valuable project hygiene; however, no one was going to stop funding WLI releases. Instead of focusing on project funding issues, I focused on presenting the information that interested the stakeholders, that is, did we know what we were doing and did it align with business needs? I used the standard slide templates and then used Standard English instead of computer jargon. In general, the gate requirements were far higher level than the day to day work of Agile, and I was able to avoid distracting everyone on the team but the project manager, the architect, and the product owner. I also held pre-reviews with my most critical stakeholders before the gate reviews to get their feedback and ensure issues were addressed before the gate review itself. Under no circumstances did I want surprises in those gate reviews.

Compliance with the risk analysis was easier and I satisfied this by performing a risk review before each gate. The business case requirement was even easier because that was the responsibility of the business side, not my team.

At this time, my boss moved on to another position, and a new director moved into his position. He made the request to my team and me that we avoid the use of computer jargon, including Agile jargon, in our communications outside our team. This was a useful and sometimes challenging discipline.

Through all this, we applied our Agile principles to delivering the product. We time boxed the release and delivered the most important scope first. We tried to be always done-done with our stories, but that was harder because done-done required physical tools to test on, and those weren't always available.

3.1.2 What Happened

The software gates from my team were some of the first gate reviews within the wireline business unit because the business unit was fairly early in adopting the LIFECYCLE process. Given that we scheduled the gates, the gates were informative, our stakeholders were on board, and we seemed to know what we were doing, the business was pleased with us. After a year or two, the process people began noticing that what we planned in the planning gate and what we delivered in the release gate didn't have much to do with each other. We explained how our requirements changed because the requirements of the tools projects we supported changed, and the process people didn't like that answer. But by this time, we had established credibility as a functioning team through a regular heartbeat of delivering value. No one made any serious effort to force us to change, so we just moved on. We continued doing gates. We continued changing the scope according to business need, and the process people continued to occasionally complain.

Monthly project reports and other management communications became more interesting because we had to pull out all Agile terminology. After a few early slipups, and glares from my boss, we learned to translate Agile terminology into the corresponding English terminology. Sprints became two week work intervals. We stopped talking about story points at all. We talked about functionality rather than stories. This actually helped our credibility also because we were able to communicate effectively with our leadership in terminology they understood.

We managed to hold rigorously to our defined time box and became even more consistent about delivering on time, every time. Also, as we prioritized the most important scope first according to business need, we never failed to deliver what our stakeholders wanted. We did have to go to our primary stakeholder a couple of times to sort out conflicting priorities, but our other stakeholders accepted these decisions.

3.1.3 Reflections

There were a lot of things we did that helped us continue to be Agile in a waterfall world. The biggest of these was to establish credibility. The most important way we established and maintained credibility was by delivering useful software. We leveraged the Agile principles of always prioritizing the most important things first, emphasizing business need rather than compliance to a plan. We also delivered reliably on time every time. Then, we make a point of exhibiting competence in other areas, such as dealing with an IT audit. I also made a point of highlighting all successes to our management structure as well as any special actions or evidence of special competence by team members. This resulted in an (accurate) perception of a highly functional team, and few managers or executives wanted to mess with a highly functional team to make them comply with process.

Of all the waterfall things requested of us, the most impactful were the gate reviews, and we complied with those. I actually believe those helped our ability to deliver rather than hurt us, despite their waterfall origins. My experience from a number of software projects is that projects that involve gate reviews tend to be more successful than those that don't. The reviews force a level of discipline that can be very good for a project. Dangling problem areas are cleaned up for the review, risks are reviewed and mitigated, and pending questions are sorted out. In a perfect Agile world, this all shouldn't be necessary, but most projects fall short of that level of agility. The gate reviews also provide executives and stakeholders a better level of visibility into the project and team, and enhance the quality of project sponsorship. Lastly, the rigorous format forces communication into a language that reviewers are familiar with, discouraging the use of computer jargon.

Just as the team needed to trust each other to be effective, our management needed to be able to trust the team to know that the team was being effective stewards of the company's money. Compliance with the mandatory aspects of the waterfall process was a major factor in earning that trust.

The restriction about not using Agile terminology was actually a useful discipline. It would have provided different benefits to report to a management structure that understood software and could provide software management expertise and sponsorship; but, this wasn't how the Halliburton business units were set up. As it was, we learned critical skills about speaking to our leadership in their terminology and avoiding jargon. It made them better able understand our needs and to support us.

3.2 The Sky Falls: Oil Crash of 2014

In June of 2014, the price of a barrel of West Texas Intermediate (WTI) crude oil was USD 106. In January 2015, it was USD 45/bbl. In February 2016, it had fallen below USD 30/bbl. This is known in the oil industry as the oil crash of 2014, and it devastated the industry. The oil business is a cyclic industry and established companies in the industry, including Halliburton, have survived many of these. They knew how to survive another one. The cuts began almost immediately when the price of oil began to decline. Agile, which we had adopted when times were good, now needed to work for us when times were hard.

3.2.1 What we Did

The first cut to hit was the mandate to cut all consultants, and I had to release our Agile coach. He was about half way through our planned adoption path. The next mandate was to cut all unnecessary expenses, thus the end of the snack drawer. Shortly after that came the first round of layoffs, and I let my test lead in Houston go. Then, we had a round of cutting capital, and I found some slack in my budget and cut a few people from the offshore team. Raises were reduced and delayed, bonuses stopped, and promotions became rare.

Next was another round of layoffs, deeper this time, and I had to cut two people. I was educated by my boss that my smart decision would be to cut two people from my Singapore team and close my Singapore office. I cut a Singapore developer and the product owner there, stopped the incipient move of an offshore tester to Singapore, and moved the remaining Singapore developer to Houston. Several months after that, I was directed to cut my budget substantially but not told how. By this time, I had trimmed all the slack out of my budget, and my only option was to fire people. On this occasion, I reduced my offshore team by half. There were more cuts after that. I lost my configuration management person in another round of layoffs.

It wasn't all bad news. There were several upturns in the price of oil on the way down, and so cost pressure sometimes eased up. I had several occasions when I was able to extract unexpected money from my budget, and I mostly used that to recover headcount in my offshore team. The best upside for my team was the sudden availability of experienced wireline engineers on the job market. I was able to hire an experienced wireline field engineer as a contract tester in Houston, and also two engineers to help my team in India.

3.2.2 What Happened

The loss of our Agile coach hurt, but it turned out that he'd done enough. By this time, the team pretty much knew what they were doing with Agile and were able to continue on without him. The cutting of half of the offshore team turned out to hurt far less than I thought it would. We kept the best and most experienced, of course; and, with the smaller team, these folks were now able to spend their time working instead of supporting their less experienced team members. Productivity was not appreciably impacted.

During this time period, the experience level of my team mattered significantly. They had all been through downturns before and knew how these things worked. They also, individually, had a pretty good feel for whether they personally were at risk of losing their jobs. Because the people that I released didn't surprise anyone, they didn't worry too much. They just buckled down, with the only occasional complaint about the loss of the snack drawer, and toughed it out. I didn't have any voluntary attrition during this period.

The chance to acquire experienced wireline engineers was invaluable and substantially increased the capacity of the team. I only wish I had been able to locate the money to hire more. All three of the contractors I picked up during this period are still with the team today.

The team maintained their Agile practice, which was now becoming ingrained. With fewer people on the team, we started encountering more priority conflicts and depended on Agile to keep us focused on value. We didn't worry about the original plan, but delivered the most important functionality first. When our requirements became too much for the team to handle, we went to our key stakeholder and asked him to guide us regarding priorities. This resulted in the regular occurrence that our planned scope and our actual scope when we released had very little to do with each other, which was exposed clearly at the gate reviews. Occasionally, as noted previously, one of the process people would notice and call us on it. We would attempt to explain why software was different than tool projects, and nobody ever wanted to hear how "software is different." In the end, we just kept delivering through the downturn, and no one wanted to mess us up.

I spent a lot of my time interacting with my management and with finance people during this period as they sought to cut waste in every corner of the business. I faced two major challenges. The first was justifying my budget as a whole, as software is expensive. The second was justifying my product owner. Because tools projects didn't have product owners, management didn't understand why software teams needed them. And, the explanation that "software is different" doesn't work well in the best of times (after all, everyone always attempts to make the case "but my circumstance is different"), and it really didn't work well when every spare penny was being squeezed out of the system. I put together presentations explaining the role of the product owner, presentations justifying the value of the product owner, examples of where he had added value that we wouldn't have realized otherwise, and compliments from our users in the field who said things like "now we have someone we can talk to on the software team." I would think I had put the question to rest, and then a few months later, it would come around again, and I would do the whole exercise again.

I was able to keep my product owner, and I credit that to a couple of factors. First, there were other software teams in Halliburton, and they had product owners also. This made for a much more credible case that software really was different. Second, though, was just sheer persistence in defending him. That, too, was another case of credibility. Because the team had established that we knew what we were doing, and we said we needed a product owner, management eventually accepted that and moved on to look for cuts in other areas. That said, I'm just as glad I didn't also need to defend a scrum master position at the same time. I didn't have a dedicated scrum master the same way I had a dedicated product owner, and all the various people I tried in the position were doing other things as well. By the end of 2016, the team's budget had been cut by 25%, employee headcount had been cut by 25%, and the team was delivering their original WLI project and also the additional project of HPTK. This was with a substantially reduced budget and a substantially increased workload, and they did it leveraging Agile.

3.2.3 Reflections

Credibility was priceless in this tough circumstance. One of the things we did as part of our Agile adoption was to strictly time-box our releases. Because we were able to be flexible about our scope, this meant we released on time, every time. This consistency was useful to our users, who could count on a regular release schedule, and it also made for excellent performance metrics. There was the whole question of scope, which the process people tried to bring up occasionally; but, because we had focused on maximum value, our users and stakeholders were happy with us. This gave us some room to insist on special considerations, like flexible scope and a dedicated product owner. In general, I don't recommend drastic industry downturns. They are unpleasant in all sorts of ways. That said, there's opportunity in everything, and we did cut some waste and pick up some really good people on the team. The Agile practices that added the greatest value included:

- First and foremost, prioritizing the most important scope first and adjusting easily to significant and constant changes of scope. This allowed us to maintain user satisfaction as staffing was reduced.
- Time boxing our release schedule. We delivered on time every time, and that allowed us to build up a stock of credibility we were able to leverage.
- Standup meetings. These significantly improved intra-team communication. This contributed to faster resolution of issues, and better coordination between development and testing.
- Retrospectives. These gave team members a chance to vent and also helped ensure that issues got bubbled up and addressed.
- The product owner. He was able to significantly improve our link to our users, which also included more and better field tests of our software prior to release. He also helped to coordinate with stakeholders, especially around scope prioritization. He also acted as a touchpoint of our department with various other digital initiatives in Halliburton to ensure our business unit's needs were addressed.

4. WHAT HAS HAPPENED SINCE

One of the best measures of the value of Agile is what the team chose to do once I was no longer around to push it. I have been gone from this team for two and a half years now, and the team has settled in using their own best practices. They continue to make use of many Agile practices, but with their own team variations.

The thing they most retained are the Agile ceremonies. The senior most development lead, who was initially quite suspicious of Agile, says now that it's natural and he "can't imagine doing software any other way." They have standup meetings every day. Backlog grooming is still performed before the sprint, and the team conducts sprint planning, sprint reviews, demos, and retrospectives. Story sizing is still performed by comparing new stories to those existing, and the list of reference stories is still posted on the wall of every conference room the team uses.

The various implementation teams attempted to share their workload so, as a team, they could succeed or fail together. This worked for the systems team and the HPTK team, but failed for the tools team. The tools team was responsible for integrating scientific algorithms and physical tools into the software, and each kind of tool was very different from the others. The developer and tester who understood nuclear physics had too different of a skill set from the people who understood acoustic imaging or the people who worked with magnetics. Those people couldn't help other team members, and actually weren't even interested in wasting their time listening to some other team member's progress and implementation details. Functionally, the tools team broke down into tiny subteams of maybe one developer, one tester, and a few ancillary folks, such as the dev lead.

The product owner's workload was too high, so various team members who were expert on the business need also shouldered some of the load. The product owner now manages the initial release scope, working with the stakeholders to manage priorities. He defines features, but developers and testers define the individual stories. He manages user acceptance testing and acts as the contact point for the business.

Development leads are now acting as scrum masters. This would normally be considered a bad practice for a manager to act as scrum master for his own people, but it seems to be working here. The individual team members are sufficiently experienced and do not require nor tolerate extensive management. Therefore, the dev lead is still acting more as the classic scrum master servant leader role.

5. OVERALL REFLECTIONS

The Agile adoption turned out to be far harder than I thought it would be. I grew into my own Agile expertise over time, at least somewhat synchronously with the industry as a whole, and the transition was relatively painless. I didn't realize how much of a change this would be to tackle all at once. I had thought that some third-party training and some coaching from me would be sufficient, and it wasn't. I don't know what we would have done without the help of our Agile coach.

We succeeded reasonably well in operating in a waterfall environment. The gates provided a certain useful discipline, a time to clean up lingering issues and a framework to communicate with management, sponsors, and stakeholders in their language, which led to improved sponsorship and general management support. I can't complain about the requirement to perform risk mitigation occasionally, either. That's a good discipline and also provided us a generally accepted framework to escalate our needs. In general, we used Agile to deliver effectively, and the waterfall process to communicate effectively. The combination worked well. The regular compliance with the corporate process definitely raised management trust in the team.

The discipline of communicating in layman's terms instead of software or scrum jargon was useful. Our key stakeholder was happy to hear questions, such as "we have more work on our plate that we can do right now. What's most important?" We got the information we needed with no software jargon involved. Every so often, we would need to explain to various managers and stakeholders that we performed our work in two-week chunks, but that wasn't hard to explain either. The leads of the wireline software team have a few of their own recommendations to pass on to other teams considering adopting Agile, in a waterfall environment or not:

- Watch the human side. People are different and need to be treated differently to be most effective.
- It's good to rotate people through different roles, if only briefly, to gain experience and perspective.
- Stories need meaningful titles. Sometimes if acceptance criteria are not defined, titles are all you have.
- Keep the backlog in order so you know what's going on and have a leg up on the next release.

6. ACKNOWLEDGMENTS

Gratitude is extended to the team who stuck with it all through our successes and failures in adopting Agile. Special thanks go to the team leads, including Fabian Rojas, James Wang, Horacio Zea, and Javier Gonzales. They showed, most importantly, patience, dedication, commitment, and common sense. Also, many thanks to Simon Orrell, for his valuable coaching, without which we would not have been nearly as successful, and to Avraham Poupko, whose help and guidance in the writing of this paper was invaluable.