Agile Experience Report
Transition & Complexity
at Cisco Voice Technology Group

“Self-organization does not mean that workers instead of managers engineer an organization design. It does not mean letting people do whatever they want to do. It means the management commits to guiding the evolution of behaviors that emerge from the interaction of independent agents instead of specifying in advance what effective behavior is.” – Philip Anderson,
The Biology of Business

Hubert Smits
Smits Management Consultancy
Boulder, CO, USA
hubert@smitsmc.com

Kathleen Rilliet
Cisco Systems, Voice Technology Group
San Jose, CA, USA
krilliet@cisco.com

Abstract—The authors spent six month in the center of the roll-out of agile practices in the Cisco Voice Technology organization. The roll-out started earlier, and is continuing after this period, and many more people contributed to the implementation. We learned that following an organizational change pattern (we used Kotter’s eight step process) is helpful to structure the program, and made us aware of where we were, and what needed doing first. We found some issues, and while working on this paper discovered that others have found similar patterns, most notably Linda Rising in her book “Fearless Change”. Obstacles we found include the necessity to obtain and maintain support of leadership, the influence of tooling on the implementation, and the need to go step-by-step: get agile skills in teams before attempting to change the full Software Development Life Cycle. We had great results from made-to-measure training of the teams around the globe, from finding a pace-maker process in the development process, and from the support of people who lead actual development programs.

Keywords: Agile, Scrum, Implementation, Large-Scale, Complexity

I. INTRODUCTION

For a technology organization committed to transition to Scrum, there is no shortage of books, training, tools and consultants to help their teams achieve agility. Turns out, this is the easy part. It gets interesting when these teams have to work together, when the solutions they build are of a high complexity and large scale. To be clear about scaling, at Cisco Voice Technology Group, we talk about integrating some 40-60 products (hardware, software – some owned by the Group, some owned by other Groups) in every release, two releases each year, multiple offer verticals for most of them, and often more than not, one team working on a single feature. Products are highly dependent upon one another - change a phone feature and the call center solution has to work with it. Teams are all over the globe, easily adding up to 1,500 people or more. And of course there is time pressure: the customer is always waiting.

II. DESCRIPTION OF THE ORGANIZATION WHICH WANTS TO MOVE TO AGILE

Voice Technology Group (VTG) is a global organization with three Business Units, with a total of about 2,500 people within the larger Cisco Systems, Inc. This Technology Group (TG) was formed approximately a decade ago, largely by way of the acquisitions of key market players. This acquisition strategy is and will continue to be a key aspect of Cisco’s and VTG’s business strategy. Presently VTG produces up to forty different products (platforms, end-points and applications). In response to market, channel and customer needs, there has been an increased focus on delivering to market integrated solutions, rather than individual products. This has been occurring over the last four or five years. More recently, Cisco as a whole is concentrating on solutions that cross TGs and incorporate many more products and solutions from multiple TGs beyond our well established network infrastructure market. This trend has resulted in an extremely complex domain within which our product and solution lifecycle processes must operate effectively.

III. WORK OF THE CHANGE IMPLEMENTATION TEAM

A. Planning for Change

When you read about agile process implementations it is often suggested to implement Scrum by using Scrum: create a backlog of process changes, prioritize, and start implementing. We used more of a Kanban approach: we had a vision and model we wanted to implement, we had a
backlog of steps to take, and when problems occurred, we would prioritize the issue, put it on the backlog, and address it when it became the highest priority. Sometimes that meant addressing an issue immediately. A lot of the process design from the previous paragraph happened that way. We ran into the problem of finding a pacemaker process and worked on it. We needed the ‘Agile Commit’ step and we just figured it out. An important lesson learned here is that implementing Agile (and probably implementing any change) into an organization is interrupt-driven, not plan driven. We had some interesting hurdles to clear, but once taken they became a strong driver in the change process. A good way to understand the structure of the change process and the steps we took is to follow Kotter’s “Eight Step Process of Successful Change”.

Setting the stage was a long process, there was definitely a sense of urgency, everybody was aware of the need to move away from the lengthy and rigid waterfall process. As a result an initiative was started to “Go Agile”. But the next step, of creating a guiding team was more about putting together a project team, rather than a leader creating a team around him. The people who got this job played a major role throughout the whole process, but a guiding team needs more than worker bees. We spent a lot of time trying to engage executives in the initiative. Without leadership backing a change it would never get done. We feared to see our work die in its infancy, before the organization could achieve the real results. That early mistake of having a consultant giving a wrong message to the executive team created a big hurdle for us; we could not get on the agenda of the executive team to debunk this obviously silly idea of leaving everything to the teams. The initiative was not killed, but all we got was a checkbook commitment. Still, that is better than nothing.

Because we had at least this executive level support, we had traction to get the project management director, a key Agile stakeholder, to work with us. She did so, reluctantly at first, and gradually became interested in the alternative approach. The battle was won when we engaged an old hand [and very respected thought leader] in the PMO - consultants can be convincing, but they never have responsibility. This senior Program Manager was convinced of the new approach, and his language brought the director over to our side. The lesson learned here is that language is important: using new words like “Roadmap” may be syntactically correct, but it can throw people off, so we went back to old language and called it an Agile Business Commit. One has to pick his battles wisely.

Now we had a leading team: we had thinkers (people knowing the old process, knowing the new concepts, and who could combine the two and foresee obstacles). We had doers (people who could train and certify the hundreds of team members, integrating the new process into the training). We had process leaders (PMO members who lead a program, or who managed the portfolio). And finally, we had leadership support that went beyond checkbook commitment.

B. Change Vision and Strategy

While all this team forming happened, and especially during the “storming” in this guiding team” we developed our change vision and strategy. Our approach was to incrementally deliver on a changed delivery process for System Releases. Our initial idea was to fully implement our changes over the next three System Releases. For each Release we had set specific, measureable priorities for changes and then started to make it happen. Choosing multiple programs was crucial, and for large organizations it shows how long it can take to implement change. The desired duration of a development cycle for a product increment was determined to be nine months, followed by work for the systems team. Thus every nine months the teams involved in product development can absorb a major change in the development process. VTG usually has two increments running in parallel, so every four to six months we could make process changes. The strategy we designed had the following steps in it:

1) System Release One – Our goal was to create healthy Scrum teams. We decided to focus our people, dollars and resources on this for the first phase, though only about one third of the product delivery teams contributing to the system release actually planned to adopt Scrum in this first phase.

2) System Release Two – Our goals were to:
   - Adopt and adapt the product Standard Agile Commit phased approach (initially aiming for three Agile Commits for any Systems Release – and therefore at Agile Commit, provide the plan for the next ninety days)
   - Work on making the system validation and trials processes agile: iterative and incrementally perform integration, regression and acceptance testing in the involved teams. This included System Component interoperability testing (integration of Products into Solutions), Alpha testing of key products, and Early Field Trials with customers in production environments.
   - Adopt a Portfolio Council which is responsible for prioritizing investments (people and dollars). The council took the Vision & Roadmap from the executive as input and worked with the Product Managers of all the products to define and maintain the backlogs for all the products involved in a system release.
   - Move to a six month system release rhythm – every six months a release finishes development, enters the final systems testing, and is ready for the packaging
process (including pricing, documentation, manufacturing, marketing materials etc.)

3) Third (and future) “release to be named later” – we are aiming to address the following:
   • Learn and adapt from Retrospectives of two earlier releases
   • Get better at estimating, integration and release of complete code more frequently
   • Document the System Release process and lighten the Commit templates
   • Create templates and adapt timing of deliverables to optimize requirements flow through

C. Make it happen

The next step in the Kotter change model consisted of work, work and more work. The sheer number of people that needed to be educated, trained and coached in the new approaches made this step a lengthy one. Not necessarily a hard step – the often-mentioned resistance to change was not a big factor for us at this point. We interpreted this as the existence of Sense of Urgency throughout the whole organization. This is what we did during the training and coaching of the teams.

1) Creating Healthy Agile Teams – System Release One (Q1-2, 2010)

The first step in shifting a software delivery organization from a waterfall approach to an agile approach was to help everyone understand where we were going. This meant teaching scrum fundamentals and team roles, training and coaching teams in the use of agile practices. Teams can deliver software in an agile way within a waterfall program, without changing many parameters at the program level. This is the approach we used for two reasons: it would allow for an incremental approach to adoption, and we knew some hardware teams were reluctant to even consider scrum. Prior to System Release One, some thirty teams and their project managers (eight hundred employees) were trained in the principles of Agile Scrum. They learned about iterations, iteration planning, backlog management, and demo & retrospectives. The project management tool of choice was purchased and implemented.

All this was relatively straightforward, it just took time due to the number of teams and people involved. The new practices, language, and delivery of increments – these things just worked. Besides the good news, though, we also got a share of bad news. First, scrum was perceived to produce lots of overhead and due to the phase implementation, sometimes both old and new templates and processes had to be used. Second, many teams thought scrum was all about managing the tool, and information was stored in multiple locations but not matching.

There is a certain amount of truth in the first complaint. The mismatch between program management using the existing tools to manage and report on the program and the teams using the new agile toolset contributed greatly to the additional overhead. However the second issue was a symptom of a deeper problem. It turned out that most teams had made the classical mistake of using the tool implementation as the driver to become agile. However good the tool, it just doesn’t work that way: a fool with a tool is still a fool. And tool training does not make a team agile; an agile tool can be used reasonably well to manage and monitor a waterfall project as well as an agile project.

Our approach was to take the focus away from the tool (not dropping it) but focus training and coaching on the practices of Scrum. Several teams got individual coaching; Scrum Masters were coached and the change leaders in the different geographical locations were pulled together in weekly calls to receive updates on process and program changes. Coaches around the globe worked with the teams to improve the understanding of agile practices. Some aha moments for us during this release included:

• We made it through the first phase (System Release One) with some teams getting healthy at the Scrum level. Or at least they claimed to be healthy. Problem was we did not have metrics to show this change, we didn’t really have a roll up view of features delivered in an agile way because of tool and reporting problems. Each team implemented the agile tool differently (some even abandoning it because it seemed like so much overhead) and went back to using excel spreadsheets.

• It’s not hard to get teams Scrumming – clear guidelines, recommendations and consultants abound not only for Scrum practice, but also for adaptations and transitions. Much has been written about experiences at the scrum team level.

• We were told that having teams split over multiple locations were be a serious impediment. Our experience was that it did take more effort and time, but we were able to use our Cisco developed collaboration tools to have some success with globally disbursed teams. Since this is a fact of life at Cisco, we had to find a way to make it work. Using Cisco web conferencing and video conferencing, desktop collaboration tools, and our own custom applications helped us a great deal. There is an extra burden on communications, precision of language, and being extra careful on clear definitions of acceptance criteria. We have had to educate management that additional time is needed to accommodate this burden. So, there is a tax to pay, but our distributed teams are definitely functional.

• We have found, though, that the most difficult part of our transition is getting everything else that must feed and support the product development teams in place. In the past, supporting and surrounding organizations counted on having a content commitment once. From this, plans were formed and timelines aligned. Incremental planning and commitment was something new for the critical internal customer teams as well (manufacturing, customer services, marketing, sales, field and partner enablement, training, demos, documentation, finance, and senior leaders.)
2) Creating a Release Rhythm – System Release Two (Q3-4, 2010)

Where change in the System Release One phase mainly worked in the “Communicate for Understanding & Buy-in” area, in System Release Two we put our efforts behind the creation of “Short-Term Wins”. We weren’t without success stories from early adopters: numerous teams were reliably producing working code out of their Sprints. Yet we wanted to see plan, delivery and integration come together. That would prove that our strategy would stand in the real world of and complex world of VTG.

And also this release held some “aha moments”:

- We agreed to make the Systems Teams (system validation and trials) the pacemaker – the metronome - of the new agile framework, and it turned out that moving to time-boxed and phased execution of their activities was actually a welcome change for many. Instead of random delivery of products to the test and trial phases, the model these groups developed had a schedule of acceptance dates, which provided the Systems Teams much more flexibility. It also gave the teams opportunities for early exposure of defects and integration problems.

- Unfortunately, the Product Teams were not able to deliver software components that could be tested or trialed early. So in effect, Systems Teams were “all dressed up with no place to go…”. It seems that the Scrum teams didn’t know how to break down their work into consumable pieces that could logically be delivered, tested and trialed incrementally. Primarily this was because our portfolio and roadmap processes were at “too high a level”.

- Tracking progress of hundreds of teams working on pieces of major features proved to be a major problem. Tools that we were using did not have roll up capability without imposing strict rules on how data was entered, labeled and updated. We found out our tools were not really scalable for the size and complexity of the programs we were working on. This proved a serious handicap in providing executive summary information about progress.

- An executive Portfolio Investment Committee was initiated with the goal of working across all of the business units in our Technology Group to set investment priorities. Two problems emerged – we did not have the management information to support the decisions of the Portfolio Committee, and information about the strategic decisions and direction were not effectively communicated to the organization so that all participants in the system release planning process were informed. The next stage of “decomposition” of Themes, Epics and User Stories didn’t really happen, so that the chunks of work were so large that the Product Teams and Scrum teams had to spend an inordinate amount of time breaking things down, (and clarifying the requirements at the lower levels) before any sprint zero or sprint planning could occur.

3) Not letting up. Create a New Culture – future releases (2011)

While the work for the Agile Transition Team in VTG is not done, it has made major steps, and will now have to incorporate the “lessons learned” into our processes (and culture) and keep moving forward. The aha moments of the System Release Two so far show that the Release Rhythm is not yet truly implemented, so we had to go back and re-enforce that step. We needed to provide more support to the Product Teams in decomposing those Themes that are top-priority and help them understand the value in a ranked priority list of requirements. The executives needed more guidance in understanding the importance of delivering their direction in smaller chunks. We’ve been inspired by “Principles of Product Development Flow” (Donald Reinertsen) and wanted to implement these concepts further into the VTG agile development process.

We continue to believe that the predictable delivery of System Releases at a regular cadence can be established with tremendous benefits to our Technology Group’s product development organization, as well as to Cisco supporting organizations. There is also a huge upside to our sales channels, world-wide support organizations and our customers with a steady supply of new value ready for customers every three months (or any acceptable interval). We must be able to execute on the strategic focus of the executives, translate that into roadmaps of value “themes” and break this down further into work “packages” consumable by the development teams. All this we needed to do while not giving up on flexibility to make change as demanded by our business, our markets, our partners and our customers. Finding the means to do this in a systematic and repeatable way would clearly give us a competitive advantage!

Our biggest aha was that we had been way too optimistic about how much we could accomplish in a couple of release cycles. We would need at least two more system releases to make the changes needed to demonstrate our achievement of the goals of Agile, and to “anchor” our new processes as a foundation for continuous learning and improvement.

IV. Conclusion

Our task is daunting, and we are definitely moving into new territory. Before us is a great challenge, but if our experience has shown us anything, it is that the hardest thing to change is the culture and thinking which is so strong within the organization. This requires us to hold strongly to our vision of what is possible. All our progress, our victories – no matter how small – we must celebrate them and solidify them as a foundation for the next steps. In spite of the organization’s hopes to make this a quick transformation… this is the work of many releases, and a journey. We will continue trying new methods, inspecting our results, and adapting. We need to continuously develop and communicate clear prioritization for those changes that bring the biggest return on investment. Yes, we need to
apply the Agile philosophy and principles from top to bottom. Or as we say in Cisco’s Voice Technology Group – we need to ‘eat our own dog food!’ If indeed we are what we eat, that is what will need to happen before we are truly, in Kotter’s words, able to “Make it Stick – change the culture”.

---


4. Forming-Storming-Norming-Performing model by Bruce Tuckman