Agile Product & Project Management

Vol. 12, No. 7

Transition and Complexity at Cisco VTG

by Hubert Smits, Senior Consultant, Cutter Consortium; and Kathleen Rilliet

This *Executive Report* summarizes and explains the authors' work in implementing agile software development practices in the large and complex Cisco Voice Technology Group (VTG). The chosen solutions as well as the efforts required to implement them are discussed. The report follows John Kotter's 8 Step Process for leading change, listing results, experiences, successes, and failures.

Access to the Experts

Cutter Consortium is a unique IT advisory firm, comprising a group of more than 100 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and agile project management, enterprise architecture, business technology trends and strategies, innovation, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you *Access to the Experts*. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts — experts who are implementing these techniques at companies like yours right now.

Cutter's clients are able to tap into its expertise in a variety of formats, including print and online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products, training, and consulting services, you get the solutions you need while staying within your budget.

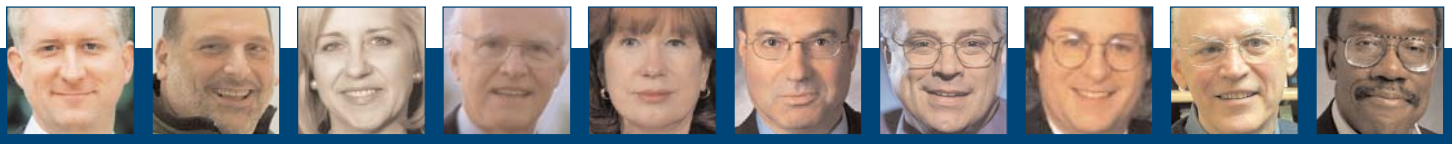
Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

Expert Consultants

Cutter Consortium products and services are provided by the top thinkers in IT today — a distinguished group of internationally recognized experts committed to providing top-level, critical, objective advice. They create all the written deliverables and perform all the consulting. That's why we say Cutter Consortium gives you *Access to the Experts*.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.

The Agile Product & Project Management Advisory Service Executive Report *is published by the Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. Client Services: Tel: +1 781 641 9876; Fax: +1 781 648 8707; Email: service@cutter.com; Website: www.cutter.com. Group Publisher: Kara Letourneau, Email: kletourneau@cutter.com. Managing Editor: Cindy Swain, Email: cswain@cutter.com. Print ISSN: 1946-7338* (Executive Report, Executive Summary, *and* Executive Update)*; online/electronic ISSN: 1554-706X.*



Rob Austin | Ron Blitstein | Christine Davis | Tom DeMarco | Lynne Ellyn | Israel Gat | Tim Lister | Lou Mazzucchelli | Ken Orr | Robert Scott

Cutter Business Technology Council

Transition and Complexity at Cisco VTG

THIS ISSUE'S AUTHORS

Hubert Smits
Senior Consultant, Cutter Consortium

Kathleen Rilliet

IN THIS ISSUE

2 Project Description

3 Analysis of the Situation at Hand

6 Implementing the Change to Create an Agile Software Development Lifecycle

7 The Act of Changing

9 Work of the Change Implementation Team

14 Conclusion and Final Thoughts

14 Endnotes

15 About the Authors

> 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*[1]

This *Executive Report* focuses on the authors' experience implementing agile software development practices in the large and complex environment of Cisco Voice Technology Group (VTG) over a nine-month period in 2010. VTG is large both in number of people (more than 2,500) and in number of products (approximately 40-50). It is complex in terms of people distribution (globally dispersed) and product dependencies (40-plus products have deep interaction with each other). We implemented agile practices and processes in this environment, effectively starting an organizational change. We used John Kotter's 8 Step Process for leading change[2] as our guide, and this report follows those steps, listing experiences, successes, and failures.

This report is aimed at the executive who is considering making a similar move in his or her software development process. Whether it is for reasons akin to those of the executive we worked with — better customer response, shorter and more certain release cycles — or for your own particular reasons, you have a long road ahead of you, and we present our experiences to offer you some useful guidance. However, keep in mind that we found the idea of "the journey is the reward" true for most of the teams we worked with, despite the fact that we uprooted them. Therefore, members of product delivery teams also constitute our audience. The hard work that is ahead of you is not without fun and not without rewards.

The lessons we learned are summarized in this report, yet before we begin we'd like to remind you of the first agile value: people and interactions over processes and tools. If one thing helped us in this change process it was our continuous work with the people: executive people, project management people, delivery people.

Everything went easier when we could interact. If we had to do it all again, we would interact even more, sometimes even forcing people to communicate with us.

Creating a vision of the new agile development process proved not too difficult and became a key communication tool. The Agile Transition Team, as we called ourselves, had a good idea based on the principles of Scrum; we stuck to the basics of Scrum and grouped the iterations into the company's stage-gate development process (see Figure 1). The effect was a process that "planned to replan" enabling product management to change product features as long as the product managers stuck to their product vision. The simple picture of "plan a release — build a release — plan the next release while building the previous one" proved simple and effective. The resulting role changes weren't necessarily easy; any time there was a role that needed to be changed, we ran into the typical resistance to change. First-line managers in particular felt there was no role for them in the "new paradigm" and completely backed out. Making their role and responsibilities clear solved this problem, but it was hard to teach these first-line managers "the art of letting go," meaning having them rely on their teams to self-organize and self-manage while the managers stayed in the position of problem solver, direction setter, and chief communicator.

PROJECT DESCRIPTION

For a technology organization committed to transitioning to agile/Scrum, there is no shortage of books, training, tools, and consultants to help its teams achieve agility. Turns out, this is the easy part. It gets interesting when these teams have to work together and when the solutions they build are of a high complexity and large scale. To be clear about what we mean by "scaling," at VTG we are talking about integrating some 40-plus products (hardware, software — some owned by the group, some owned by other technology groups in Cisco) in every release, with two releases each year, with multiple product verticals for most of them, and often with more than 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, not only representing time-zone challenges but also cultural and language barriers. And team sizes easily add up to 1,500 people or more working on each release. And, of course, there is time pressure: the customer is always waiting.

When we looked at agile at scale — at "agilifying" solution and system-level releases[3] — we found there were no guidebooks, no successful examples, and no experienced professional advice. Tools are sorely lacking the ability to scale (imagine a drop-down list with 500

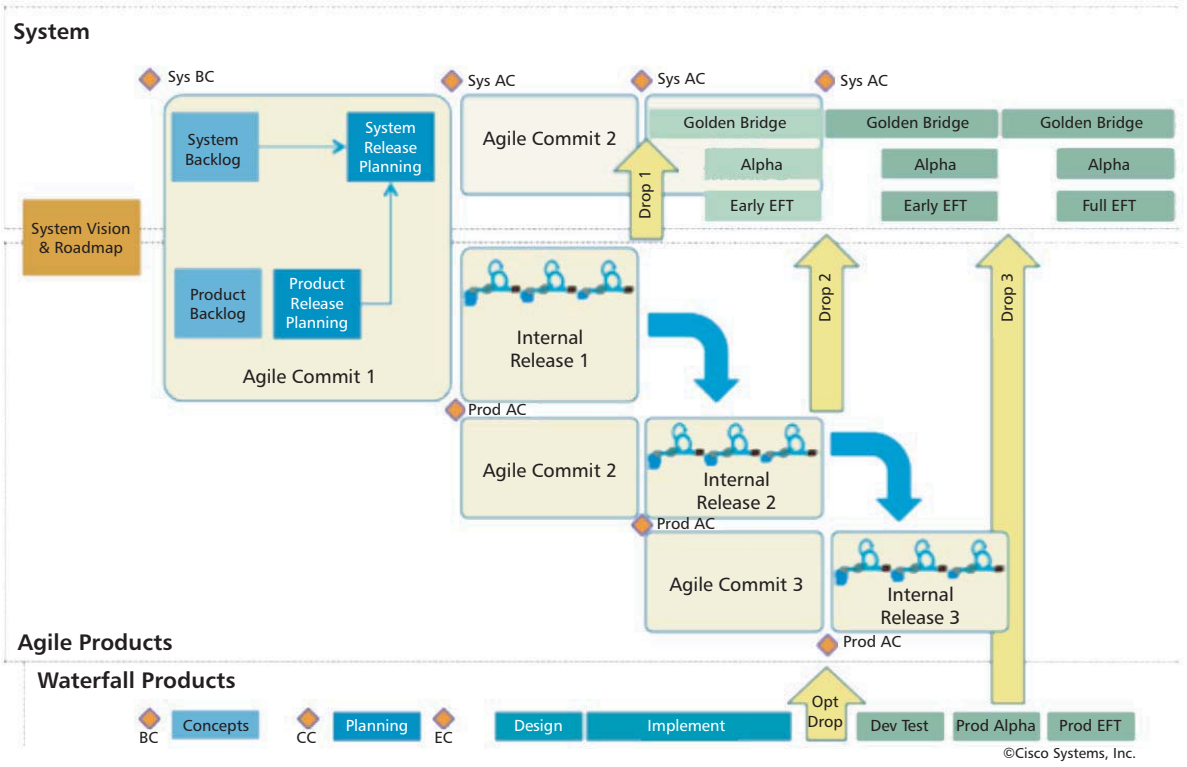


Figure 1 — Agile Unifed Communications Systems Process.

engineers to put in your sprint). So we were left to apply the agile philosophy and agile principles and invent/experiment with processes and frameworks in an organization that has highly complex technical dependencies, globally dispersed teams, and a distributed governance model. Somehow we had to make this all hang together to produce compelling solutions for our customers.

Our journey of discovery and innovation went beyond organizational change and paradigm shifts and has challenged us at all levels of the organization. And through all of this, we couldn't stray too far from a corporate agile implementation initiative that was still working at Scrum team level. And those four agile values in the Agile Manifesto[4] still stood; we had to blend them into every change, into every level of the organization. And keep it simple. Welcome to the Fun House.

In this report, we tell our story of changing the waterfall culture to an iterative and incremental delivery system. We talk about the principles of the agile software development lifecycle we created and the steps we had to take to embed this new process into the organization. We have managed portfolio and product teams, and here we tell you how we did it and how we won't do it again. We have trained teams, project managers, and product managers and have seen how training an executive team can fail and cause disastrous results. We tell you how proud we were when our system integrators started to iterate their process, picking up new functionality like clockwork. We also tell you how we moved from change management to planning for change, avoiding the wasteful drop in features that were half finished when the change occurred. Finally, we tell you how we initiated and continue to mature our process and our approach to changing an organization.

In the whole of Cisco Systems as well as in VTG, people had been experimenting with the concepts and practices of agile software development. Some teams were iterating, and some products had acquired software and coaching to support the initiative. Yet the grassroots approach did not yield results at the technology group level. Looking back at almost a year of coaching program managers and teams in their steps toward Scrum, we think that the complexity of the products in VTG was our biggest challenge. With hindsight, it is clear that in order to handle large and complex products we had to scale down. But at the time we were designing the strategy and the migration toward Scrum, that idea wasn't so obvious.

Thus, the impact of complexity on the product development process wasn't obvious to us, and it wasn't obvious or easy to explain to the people who make up this process. The most important indicator we got with regard to the complexity of the products and the impact of this complexity on the process was the number of consultants that failed to gain traction on the transition. Almost all consultants, while having a solid background in Scrum and a good list of successful implementations, approached VTG with the attitude of "let the teams figure it out and all will be good." The teams were clueless, thus management refused to take this approach, and we had to do a lot of repair work when a consultant hammered the executive team for four hours with the message "let the teams figure it out."

The impact of complexity on the product development process wasn't obvious to us, and it wasn't obvious or easy to explain.

Simply knowing what the main obstacle is doesn't present a solution. We had to figure out how to stay true to the agile principles and yet create a mechanism that controlled the product backlog from the executives downward, that allowed for major change without major waste, and that created interaction on a large and wide scale between executives and product managers. We were aware that a lot of experience had been collected around scaling agile at the delivery team level, but not a lot had been done to scale at the product owner level. The product owner role in Scrum is probably the worst-documented role in the first place, which is interesting as time and again it proves to be the role that influences the success of the project the most.

ANALYSIS OF THE SITUATION AT HAND

Before we detail the process of moving to agile in this environment, it is useful to have some more background information, including about the organization itself, the software development lifecycle in place prior to the change, the challenges the organization faced, and the reasons for it to move to agile.

Description of the Organization that Wants to Move to Agile

Cisco Voice Technology Group is a global organization with three business units, employing a total of about 3,000 people within the larger Cisco Systems, Inc. This technology group 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 business strategy. Presently, VTG produces approximately 40 different products (platforms, end points, and applications). In response to customer and market needs, there has been an increased focus on delivering integrated solutions to the market rather than individual products. This has been occurring over the last four or five years. More recently, Cisco Systems as a whole has been concentrating on solutions that cross technology groups and incorporate many more products and solutions from multiple technology groups beyond the well-established network infrastructure market. This trend has resulted in an extremely complex domain within which VTG's product and solution lifecycle processes must operate effectively and efficiently.

Recent economic conditions have required VTG to scale down expenses, which impacts such areas as travel, tools, and staffing. Additional pressures to the Cisco way of working include the globally diverse workforce, organizational silos, and collaborative/distributed decision making. While the Cisco "Culture of Working Collaboratively" can be a very powerful ally in producing innovative and market-leading solutions, it can also dramatically slow down delivery cycles. All this is coupled with a market in which change and customer demands are accelerating, creating a timing mismatch between decision-to-delivery cycle time versus market change cycle time; that is, the time needed to deliver a requested change is (much) longer than the time in which a customer updates the change request. This has led VTG leadership to look at new and — at least to VTG — innovative ways to better align with the demands of its customers and markets, as well as Cisco's desire for more productivity from its smaller workforce. Agile/Scrum seemed a good fit for us to explore, and so we started the journey.

The Software Development Lifecycle at VTG

Figure 2 shows how a classic waterfall process was used within VTG. The two layers — system and products — are the sources of requirements and places of integration. Requirements originate at one of two levels: "exec" or product. Requirements stemming from the exec level are usually solution-based (including and affecting multiple products). Meanwhile, requirements originating at the product level expand the product functionality but come from the micro level of the product as opposed to from the bigger picture of a VTG solution. However, micro does not imply small; dozens to more than a hundred people can be working on a product-level requirement. Requirements generated from the product level will still generate requirements (changes) in other products.

In Figure 2, these requirement-originating steps are seen in system concepts, system requirements definition, and product requirements. The process to define and accept the product concepts is a cycle of discussions in which

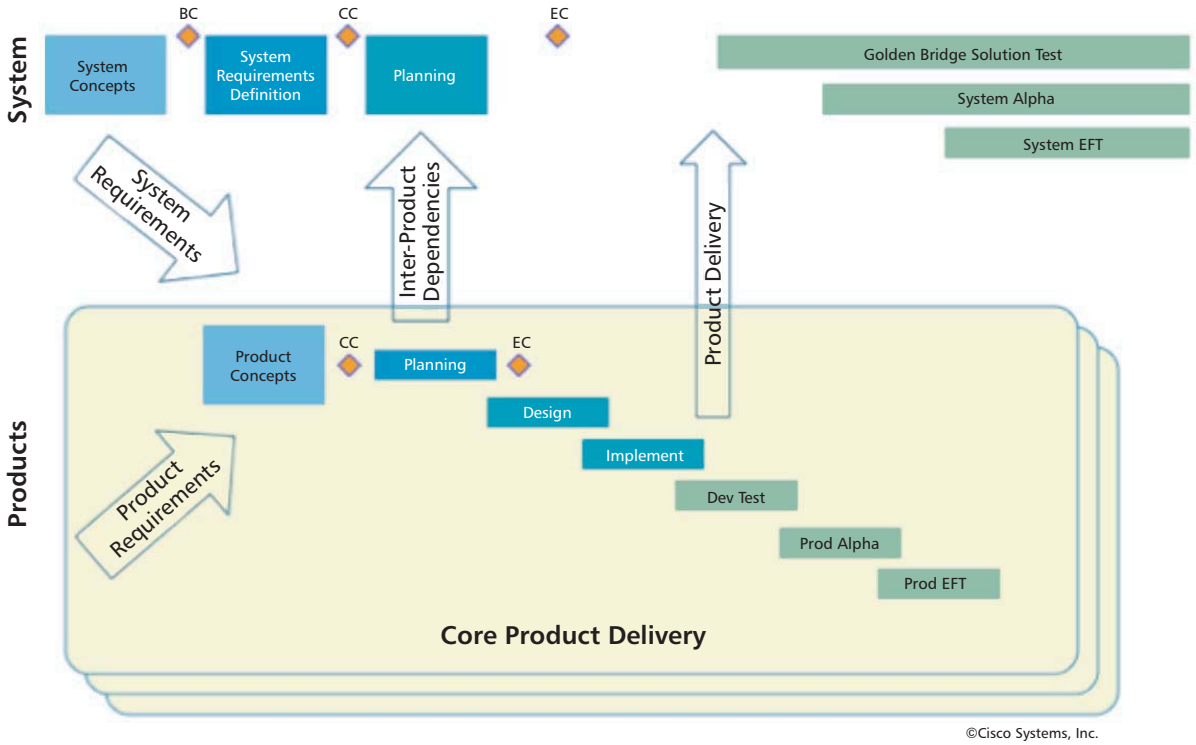


Figure 2 — Unified Communications Systems Process.

the priority of the system and product requirements is agreed. When the priorities are outlined, the process of planning starts, first at the products level, then merging into a master plan at the system level. At the products level, the balancing of requirements, estimates to deliver the requirements, and necessary people and resources are worked out within the products program management office (PMO). Once all products have worked through this exercise, the results roll up to the system level, where results and dependencies between products are verified. When mismatches are detected, the involved products are asked to solve the problem. While this time-consuming process happens, the products start to deliver the requirements and the executives start to change their minds. The delivery happens through a waterfall-style design, implement, and test process. When the product requirements are ready, they are handed over to system-level teams, which integrate and test the product changes into solutions. The changed solutions are tested, move into an alpha and beta implementation, and finally are implemented with external field trials (EFTs). Usually two of these programs run in parallel, with one in the middle of delivery while the next program is in the definition stage.

Synchronization of the players in a large program happens at decision moments, or milestones, marked as BC, CC, and EC in Figure 2. The first agreement is made at the business commit (BC) milestone: the executive team agrees on the solution requirements going into the next release. The impact of these requirements on the products and the requirements generated by the dependencies between products are agreed in the concept commit (CC) milestone. Nondelivery teams like sales and marketing now know what new features will emerge at the end of the release in both solutions and products. The impact of these features on the teams is verified during the planning stage, and the product teams commit to delivery at the execute commit (EC) moment. Like so many waterfall concepts, this implementation of a waterfall would work reasonably well if there was no change happening during all the steps.[5]

Although the waterfall process was well established, the number of exceptions at every gate in every release was evidence that it wasn't effective. Ironically, in the last major system release prior to beginning the transition to agile/Scrum, it was necessary to manage the release in three phases: three different delivery dates. Another symptom of the need for change was the large number of change proposals that had to be reviewed, analyzed, and possibly integrated into each release. This was not only time-consuming, it was disruptive and distracting.

Some Challenges of the Existing Organizations and Processes

As stated earlier, currently VTG is organized into three business units based loosely on product/technology acquisitions. This very product-focused organization has been organized as a "weak matrix" with project and program management handled within the engineering organization of each business unit. There is a solution level of activities (program management, system-level interoperability and validation, and go-to market); however, this is managed through distributed governance: agreement within product and delivery teams is established through time-consuming discussions. Linkages and alignment have been weak, with vague agreements on priorities (i.e., whether solution priorities trump product priorities). The resultant conflict of priorities, long cycle times for decisions on clear direction, and constant shuffling of the "priority du jour" (in fairness, based on real market changes and customer demands) resulted in a low tolerance for change within the workforce. Culture can have both a positive and negative effect on the ability to execute, and lack of a scaled governance process that could provide clear and concise direction quickly rendered the culture of collaboration unwieldy and, in fact, more of a burden than a help. Clearly, we needed to become more nimble.

Lack of a scaled governance process that could provide clear and concise direction quickly rendered the culture of collaboration unwieldy.

Reasons for Change

The leaders stated the initial objectives in the business transformation to Scrum as follows:

- Get closer to the customer; have true intimacy and responsiveness
- Have more flexibility to adapt to market changes without constant "replanning"
- Achieve increased productivity and accountability of the workforce (individuals and teams).
- Allow teams to be "self-managed"

IMPLEMENTING THE CHANGE TO CREATE AN AGILE SOFTWARE DEVELOPMENT LIFECYCLE

The Scrum framework is grounded in teams developing software. Application of the Scrum framework at the project, program, and portfolio levels is much more complex and requires a more adaptive approach. Since there is little in the way of research, published experience, or modeling for the "higher" levels of adoption (inside and outside software development organizations), we determined that a pilot, inspect, and adapt approach would be needed.

We wanted to incrementally move to an agile (continuously learning and improving) organization. In order to minimize the disruption to development and delivery of revenue-producing products and systems, a phased approach was chosen. We believed we would be able to take small steps, inspect the effectiveness of the changes, adapt further, and anchor our progress. We linked each change step to a system release with no change in practices and process during the time that the teams were delivering a solution and instead only from the one release to the next. As we began, we felt that it would take two to three release cycles to be at a stable and sustainable state of agile. Over time, it became obvious to us it would be a longer transition. We seriously underestimated the length of time it would take to change our thinking at all levels and how — under stress — it was very easy, and attractive, to revert to our previous waterfall methodology. Reasons for this will be discussed later.

We seriously underestimated the length of time it would take to change our thinking at all levels and how it was very easy, and attractive, to revert to our previous waterfall methodology.

Guiding Principles

We followed a few principles throughout the whole change as we felt (and got confirmation) that they would be critical to the success of the transition. These principles form the core of Scrum, and we never let go of them, even when the sheer size of the implications or the complexity of the process seemed to indicate that we should let go. These principles are:

- **Timeboxing the release activities.** Although VTG had applied some timeboxing previously in the form of a "phased approach" to various milestones/deliverables, it was primarily adopted as a survival tactic to deal with component programs that were late/unable to meet the system release timelines. Several groups that were dependent on development could not absorb or even keep track of "60 different dates for 20 different milestones," some of which seemed to be floating. Nor were those involved in managing a program able to effectively manage the expectations of the sales force and customers.
- **Deliver incremental content in line with strategies/priorities/roadmap.** This concept in particular seems in such opposition to the previous release approach (commit to all release deliverables early in the release) that product managers have not been able to embrace it in theory or practice. We have struggled with breaking down initiatives into "bite-sized chunks."[6] This becomes a major impediment to prioritizing backlogs. The number of "number one priorities" is so large that it renders a priority-ranked backlog meaningless.
- **Schedule is the least flexible (fixed); resources are somewhat flexible; content is most flexible.** Again, VTG has been a culture of pet projects and fragmented alignment on priorities (confederacy rather than union!). Agreeing to a release cycle of six months was easy, but discipline on dropping content to make a date fell away when confronted with dropping requirements "someone" felt were critical. Moving dates felt natural because that was what always happened in the past — and under stress we fall back on what is comfortable.
- **Rolling wave, incremental commitment to content delivery.** In the past, senior management, without regard to team capacity and capability, set the release goals. Aggressive plans and commitments for content were set top-down, and these were far beyond capacity and reality. Often there was no discussion on feasibility between product management and development and test prior to a business commit. It was obvious that we needed to move to a more reality-based view of capacity with a very disciplined approach to prioritizing work. To make matters worse, the planning process was frequently interrupted by priority changes from the VTG general manager and the parent group. The lack of a defined portfolio process (which addressed alignment across the contributing groups and would take care of

medium-time goals and short-term needs) also meant that each and every change resulted in a battle with all the other players on adjustments; nobody wanted to give up anything or even recognized the importance of the union over the individual units.

THE ACT OF CHANGING

Our core challenges are discussed in this section; they can be summarized as follows:

- **Steering product initiatives** — steering product initiatives at a large scale, with strong insights and strong directives from the executive level downward, within the bounds of the capacity and capability of the organization.
- **Short change cycle** — delivery taking two to three times longer than the change cycle; that is, the market requested a change in product development before the delivery cycle of the previous change was finished.
- **Moving the teams to Scrum** — a very large number of teams, distributed across the globe, needed to be moved to that agile rhythm of iterative and incremental development while staying aligned and coordinated.

Steering Product Initiatives

We had some experience from the past with bundling product features into releases, and we used these ideas to address the first two challenges. The executive team in VTG does the steering of the solution. It prioritizes the ideas and, with a frequency of three or four times a year, puts the requests in front of the product managers. That sounds like the first two levels of planning in Figure 3: vision and roadmap, and we adopted this language to reflect the responsibilities of the executive team. The vision would not change dramatically every quarter, and it was easily caught in statements like "Digital is the new analog," implying that each and every device and application in VTG needed to have digital capabilities. The roadmap would change; priorities would shift significantly — not just from one quarter to the next, but frequently and often within the quarter. This is what caused havoc in the Unified Communications Systems Process: just when the design and development were cranking out first results, the teams had to drop what they were doing and replan for slightly or significantly different work. It's no surprise that this caused long delays in delivering working software; it's a problem that all large development projects face.

Short Change Cycle

The idea to "plan to replan" started to form. Instead of raising expectations with all parties (product management, delivery teams, executives) that a plan would be solid for the duration of the release, we started to communicate that a plan would only be solid for the duration of an internal release[7] (our planning horizon). And we chose the duration of a release roughly as long as the request cycles of the executive team — a timebox of approximately three months. This was more difficult than one may think, as several initiatives requested by the executive team could not be delivered in a single internal release. Enter a new challenge: we had to motivate executives to stick to their own core priorities and resist the urge to meddle. While flexibility is important, we needed for the vision to be stable and focused on the business reasons for the change and the cost of adaptation. The concept of a vision driving the priorities of

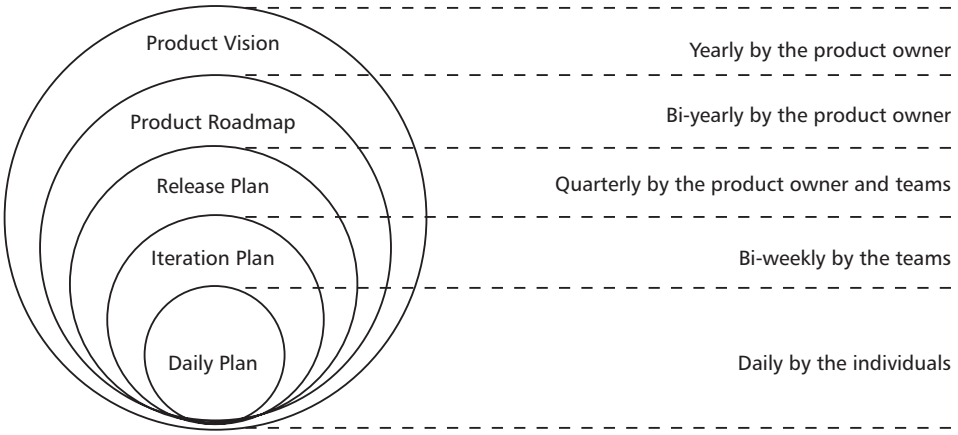


Figure 3 — Five levels of agile planning.

work in the products helped encourage clearer communication of the changes and lessen the feeling of churn among the troops. It all felt more "routine" — the benefit of planning to change! Because of this increased stability we have been able to cancel nearly every "change control board" meeting, saving everyone a lot of time and work. The executives were very accepting of this new approach to change. We were lucky here, as we had no chance to go in front of them and explain our strategy. That four-hour session mentioned earlier on how agile is about "leave it to the team" had wreaked its havoc, and the executives refused to spend their time with us again. Yet our principle to "plan to replan" worked, and they easily fell into the rhythm. The executive team became agile without knowing it.

It was all about "teams communicating to develop a plan," yet communicating with diligent orchestration by members of the systems PMO and their counterparts in the product PMO.

All this sounds easier than it actually was. Portfolio management means orchestrating the rollout of a roadmap over the large number of products in the portfolio, creating product backlogs out of it, and synchronizing dependencies between the products. It was hard work — and a lot of hard work. What helped our Scrum implementation is that we controlled this part of the program.

The agile rollout was executed from the PMO at the systems level, and both the portfolio manager and the program manager for our first agile program were systems people,[8] sitting in the same bull pen. This allowed for easy communication, creating experiments, running them, and evaluating them. This all sounds like we changed a lot in the process, and although a program looks different nowadays when compared to a program two years ago, the practices didn't change. If the product managers in VTG were able to plan a massive program out in detail, then these skills can be applied to plan a smaller plan (only a single internal release) into the necessary detail (less detail than planned out in the past). Telling participants that we would work with their known practices and skills, but on a smaller scale, helped us to convince the involved people that their life would get easier, and better yet, we wouldn't turn their lives upside down. As in every change process, a few early adopters provided the help we needed. The system product manager and his product owner team for one of the products diligently worked through the process of writing epics, prioritizing them, mapping dependencies to other teams, and creating process in and around the agile tool to communicate with other products. They designed the meeting structures to cover their own product and their own backlogs. The portfolio manager (in her role overseeing and coordinating all the products that participated in a release) carried the discussions over the products involved. It was all about "teams communicating to develop a plan," yet communicating with diligent orchestration by members of the systems PMO and their counterparts in the product PMO.

Moving the Teams to Scrum

Our third challenge centered on how to make such a large number of teams play to the tune of our release and sprint cycle. One way to do this would have been to just demand such a behavior. However, we felt that better than demanding would be to set a natural rhythm that all teams would recognize and would already use as a synchronization mechanism.

Lean software development[9] talks about a pacemaker process, and we had just such a process in our Unified Communications Systems Process. The Golden Bridge process (integration testing between products) would be key to any delivery rhythm we would be able to set. It is an essential step in the delivery, and it could not be moved into the iterations or releases for the teams since it tests the aggregation of the products. The people in the Golden Bridge process were very interested in going agile because they wanted to reduce their heavy, painful, and slow cycle time; the agile implementation team could no longer afford a three-to-six-month heavy test cycle at the end of the delivery, but a fast, one-month cycle that would test whatever functionality is available when the Golden Bridge testing starts would be much more acceptable. Read that sentence once more: fast, one-month cycle. Sounds a lot like a sprint, and that is exactly how we treated it.

So with the final step in the delivery process going into a monthly rhythm, we lined up all the other steps to this process. The dates for the start of the Golden Bridge iterations would be published, and products could choose to have functionality ready to be tested. The "release train concept" was implemented: every month the Golden Bridge train would leave the station. Products that were on time at the station would take a ride. If they missed it, they'd have to take the next ride. This may sound good, but there are two concerns with this approach. First, the Golden Bridge cannot take just any

mix of product deliveries, and, second, what happens if the major players in the release all wait for the last train to hit the station and put all their functionality on it? The first issue — finding the right mix of product features to board the Golden Bridge train — defined the next level of agile planning (the executive team owned vision and roadmap). To address the second concern, the PMO (owners of the release planning) had to create a process not only to synchronize features between products but also to order the delivery of these features in such a way that the Golden Bridge process could test an incremental product or solution chunk. This resulted in a heavy planning process; we can only admit that agility has suffered here. Where product backlog management within a product is very much people-driven, synchronizing the backlogs requires a blunt review and reordering process by the Golden Bridge (or systems) team. We are still trying to figure this one out.

The current incarnation of all this process redesign is seen in the Agile Unified Communications Systems Process in Figure 1. The vision and roadmap activities, driven by the executive team, are the start of a system release. To help communication with customers, this roadmap is captured in a systems business commit (BC). The business commit is a high-level, long-term, straw man plan for the system release. The teams in both the products and the systems groups pick up the roadmap, and product backlogs are created. These backlogs consist of "epics"[10] and a size indicator for each epic. Each of the products discovers dependencies upon other products and communicates those (either directly or during the weekly portfolio planning meeting). With the product release plan completed, the systems release management team takes on the planning at the system level, determining what the best order for the products to deliver their functionality might be with regard to being integrated and tested in the Golden Bridge process. When this order is agreed, all the teams commit to the agile commit (AC), which is a hard commit for the next three months of work, as well as a soft commit, which is for work following the three months, following the internal release, if circumstances do not change. We called the agile commit a "steel man": a detailed plan for the next 90 days. The work on the first internal release starts (teams are sprinting), and in parallel the agile commit process is repeated: changes to the system roadmap are evaluated, merged into the existing release plans, rolled up to systems level, and committed to by the product management teams. The product people are replanning as part of the delivery process; planning to replan! Some products would by choice or necessity continue to work in a waterfall fashion. The integration of those product increments would happen in a specific drop into the Golden Bridge process, depending on the content.

WORK OF THE CHANGE IMPLEMENTATION TEAM

When you read about agile process implementations, the suggestion is often made to implement Scrum by using Scrum: create a backlog of process changes, prioritize, and start implementing. We used more of a Kanban[11] 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 discussed above 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 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 take, but once taken, they became a strong driver in the change process. A good way to review the structure of the organizational change process (and the steps we took) is to follow John Kotter's 8 Step Process.[12]

Kotter's eight steps are as follows:

- Step 1 — Acting with Urgency
- Step 2 — Developing the Guiding Coalition
- Step 3 — Developing a Change Vision
- Step 4 — Communicating the Vision Buy-In
- Step 5 — Empowering Broad-Based Action
- Step 6 — Generating Short-Term Wins
- Step 7 — Don't Let Up
- Step 8 — Make Change Stick

How the structure of these steps relates to our discussion is seen in this section as we detail our change implementation.

Setting the Stage

Setting the stage was a long process. There was definitely, as follows with Kotter's first step, a sense of urgency: everyone was aware of the need to move away from the lengthy and painful waterfall process. As a result, an initiative was started to "go agile." But Kotter's next step, Developing the Guiding Coalition,

involved more of the putting together of a project team rather than a leader creating a team around himself or herself. The people[13] 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 watching our work die in its infancy, before the organization could achieve the real results. As mentioned above, 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 with this "obviously silly idea of leaving everything to the teams."

Lesson learned here is that language is important; using new words like "roadmap" may be syntactically correct, but it throws people off.

Yet the initiative was not killed. Though all we got was a checkbook commitment, that is better than nothing. Because we had that executive-level support, we could request that the product development director spend time with us. She did so and gradually became interested in our alternative approach. We won her over when we engaged an old hand in the PMO — consultants can be convincing, but they never have responsibility. The senior program manager was convinced of the new approach, and his language brought the director over to our side. Lesson learned here is that language is important; using new words like "roadmap" may be syntactically correct, but it throws people off, so we went back to old language and called it again a "business commit." One has to pick one's battles wisely. Now we had a leading team: we had thinkers (people knowing the old process, knowing the new concepts, and able to 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 led a program or who managed the portfolio); and we had leadership support that went beyond checkbook commitment.

Deciding What to Do

While all this team forming happened, and especially during the "storming" in this guiding team,[14] we developed our change vision and strategy, following Kotter's Step 3: Developing a Change Vision. We picked programs and their change goal 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 is nine months, followed by work for the systems team. Thus, every nine months the teams involved in product development could take a major change in the development process. VTG usually has two increments running in parallel, so every four-to-five months we could make process changes. The strategy we designed had the following steps in it:

- **System Release I.** Our goal was to create healthy Scrum teams. We decided to focus our people, dollars, and resources on training and coaching teams on Scrum process and practices for the first phase, though only about a third of the product delivery teams contributing to the system release actually planned to adopt Scrum in this time frame.
- **System Release II.** Our goal was to create an agile release rhythm:
  - *Adopt and adapt the product "standard agile commit" approach:* three ACs (and thus three internal releases) for any systems release. At each AC, plan the work for the teams for the next 90 days.
  - *Work on making the system validation and trials processes agile:* iteratively and incrementally do integration, regression, and acceptance testing in the involved teams. Here, this included system component interoperation testing (integration of products into solutions), alpha testing of key products, and external field trials with customers in production environments.
  - *Adopt a portfolio council that is responsible for prioritizing investments (people and dollars):* the council takes the vision and roadmap from the executive as input and works with the product managers of all the products to define and maintain the backlogs for all 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 "productization" process (including pricing, documentation, marketing materials, etc.).
- **System Release III (future release).** We are aiming to address the following:
  - Learn and adapt from retrospectives of two earlier releases.

- Get better at estimating and integration and even do external releases more frequently.
- Document the system release process and lighten the commit templates.
- Create templates and adapt the timing of deliverables to optimize requirements flow.

Making It Happen

For us, the next step, Step 4: Communicating the Vision Buy-In, 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. This was not necessarily a hard step; the often-mentioned "resistance to change" was not a big factor for us, which we interpreted as having a sense of urgency throughout the whole organization. This is what we did during the training and coaching of the teams.

Creating Healthy Agile Teams: System Release I (Q1-Q2, 2010)

The basic shift, or the first step in shifting a software delivery organization from a waterfall approach to an agile approach, consisted of teaching, training, and coaching teams in the use of agile practices. Teams can deliver software in an agile way during a waterfall program without changing many parameters for the program. This is the approach we used, and during the program prior to System Release I, some 30 teams and their project managers were trained in the principles of agile/Scrum — about 800 total people. They learned about iterations, iteration planning, backlog management, and demo and retrospectives. The agile project management tool of choice was purchased and implemented. All this was easy; it just took time due to the number of teams and people involved. But practices, language, and delivery of increments in iterations all worked.

Besides the good news, we also got our share of bad news. People in the teams mentioned experiences like "Scrum creates lots of overhead; Scrum was all about managing the tool; Information was stored in multiple locations but not matching." It turned out that most teams had made the classic mistake of using the tool implementation as the driver to become agile. It doesn't matter how good the tool is: a fool with a tool is still a fool. The mismatch between program management using the existing tools to manage and report on the program and the teams using the new agile toolset created the overhead mentioned by the delivery teams. And tool training does not make a team agile. An agile tool can be used reasonably well to manage a waterfall project: the iteration is about a year, and there are design sprints, coding sprints, and testing sprints, and the tool records it. We decided to take the focus away from the tool (not dropping it) and center on training and coaching on the practices of Scrum. Several teams got individual coaching, ScrumMasters 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 System Release I with some teams getting healthy at the Scrum level.** Or at least they claimed to be healthy. A problem here was we did not have metrics to show this change. Another problem was that 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 turned out that most teams had made the classic mistake of using the tool implementation as the driver to become agile. It doesn't matter how good the tool is: a fool with a tool is still a fool.

- **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 even able to use our Cisco collaboration tools to have some success with globally disbursed teams. Distributed teams, if they are persistent, mature, and truly self-managed, can produce good results in an agile environment. While not ideal, it is still possible. It does require a higher level of commitment, discipline, and trust especially around communication and information sharing. The leaders must take time to build that trust, overcommunicate rather than undercommunicate, and watch for assumptions that can get you in trouble. That extra effort and leadership from people is the "tax to be paid," but so far it is looking like a good investment.

- **The hard part is getting everything else that must feed and support product development 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 (manufacturing, customer service, marketing, sales, field and partner enablement training, demonstrations, documentation, finance, and senior leaders). We decided to keep the support teams outside the Scrum implementation. Not because these processes cannot iterate or deliver incrementally but to keep our scope limited, avoiding an attempt to "boil the ocean."

Creating a Release Rhythm: System Release II (Q3-Q4, 2010)

Change in the first release mainly worked in Kotter's Step 4: Communicating the Vision Buy-In. Regarding Kotter's Step 5: Empowering Broad-Based Action, we received (almost took) that approval implicitly; since we weren't stopped by the exec team, we moved on. Even better, when we built that team (see Step 2), the exec involved approved the use of agile in the next program.

In our second release (see Figure 4), we put our efforts behind the creation of Kotter's Step 6: Generating 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, which would prove that our strategy would stand in the real world of VTG.

This release also held some aha! moments:

- **We agreed to make the systems teams (system validation and trials) the pacemaker process of the new agile framework**, and it turned out that moving to timeboxed and phased execution of their activities was actually a welcome change. 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.
- **Dependency management continues to be a challenge for us.** Thus far it is still being managed manually, since our tools do not provide a practical means to manage our complex dependencies without a tremendous amount of overhead for the Scrum teams and Scrum-of-Scrum-level product teams. This they have revolted against.
- **Unfortunately, the product teams were not always able to deliver software components that could be**

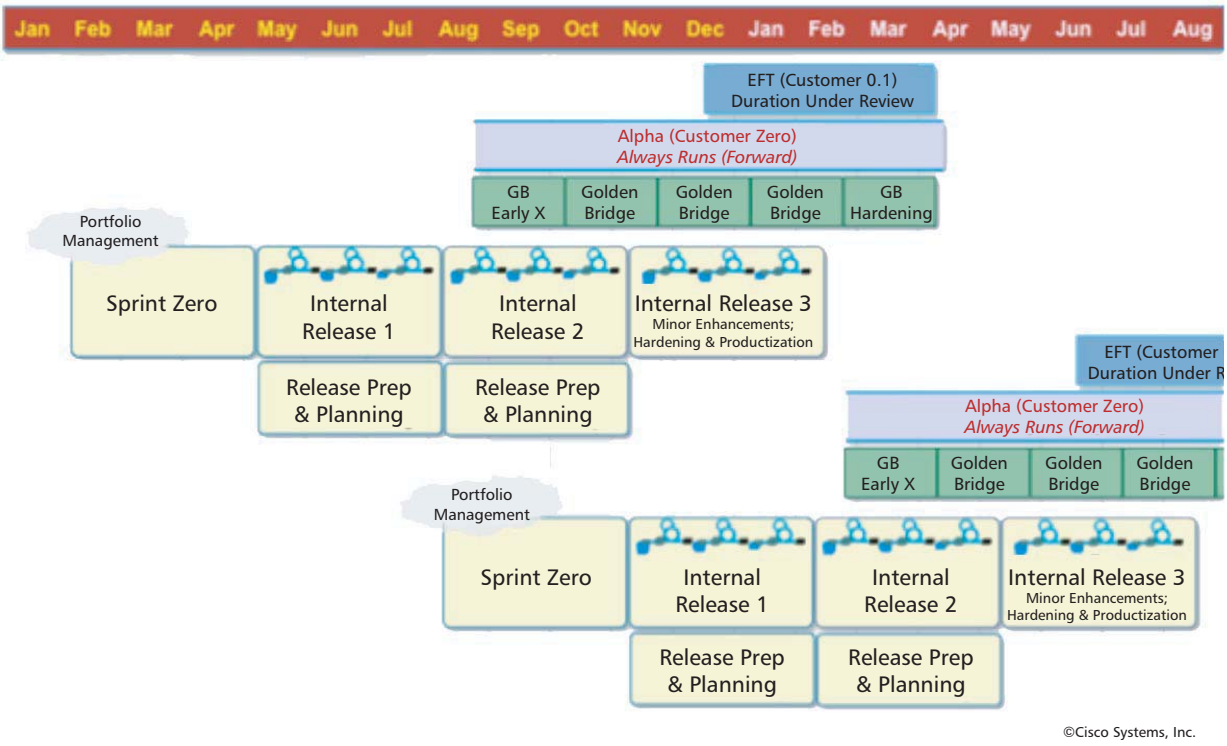


Figure 4 — Agile systems release rhythm.

tested or trialed early. So in effect, systems teams were sometimes "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 tested and trialed.

- **System release monitoring and tracking (providing a picture of our progress throughout the cadence of our lifecycle) is a challenge.** We made a first stab at naming the go-to-market priority requirements, and identifying and tracking those dependencies became the focus of our initial efforts. This was begun as part of System Release II, since we realized our mistake in not taking this into consideration only after it surfaced in System Release I. Metrics and tracking progress need considerably more work.
- **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 without extensive customization and even additional API work. This proved a serious handicap in providing executive summary information about progress.
- **Further work around the vision, roadmap, and prioritization in general continued through the initiation of an executive portfolio investment committee.** The goal of this group is to work across all of the business units in our technology group to set investment priorities. Two problems emerged: (1) we did not have the management information to support the decisions of the portfolio investment committee, and (2) information about the strategic decisions and direction were not effectively communicated to the organization so that not 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 before any sprint zero or sprint planning could occur.

Don't Let Up and Making It Stick

Kotter's Steps 7 and 8 are ongoing. The work for the Agile Transition Team in VTG is not done; it has made major steps and will now have to repeat steps and keep moving forward. The aha! moments of System Release II show that the release rhythm was not reliably implemented; we need to go back and reinforce that step. We need to provide more help to the product teams in decomposing those themes that are top priority. The executives need more guidance in creating smaller work chunks.

To date, we are only partially successful with portfolio management, but we do believe that this approach has promise. One thing that we felt would give us more leverage would be cross–business unit alignment of priorities through a more formal portfolio process. This would give us a framework within which to work and less contention on what the priorities actually are.

Systems teams were sometimes "all dressed up with no place to go."

As part of the initiative, we have also instituted a group of senior-level business unit executives who are responsible for agreeing on the investment priorities (of budget and resources) for our larger VTG organization. The intent is to provide the organization as a whole and the individual teams with the key decisions that are needed to proceed with release planning. Initially this has been at a very high level, and another discovery is that to scale up we have to go smaller. The translation of these high-level investment priorities to the level required for release planning necessitates another layer, and this too needs to be aligned. So a system release portfolio team was instituted to review and approve the roadmaps and the plans for each system release. If we run into conflicts due to budget, people, or resource constraints, this group has a fixed amount of authority to address it. However, if there is a serious conflict, the issue has to be raised to the executive portfolio committee. We are still in the beginning stages of this effort; however, this too has promise as a way forward.

We've been inspired by Donald G. Reinertsen's *The Principles of Product Development Flow*[15] and want to implement these concepts further into the VTG development process (see Figure 5). A continuous flow of system releases can be established, with a release ready for customers every three months. This process allows for a focus on the vision of the executive, while increasing flexibility and better responding to change demanded by clients.

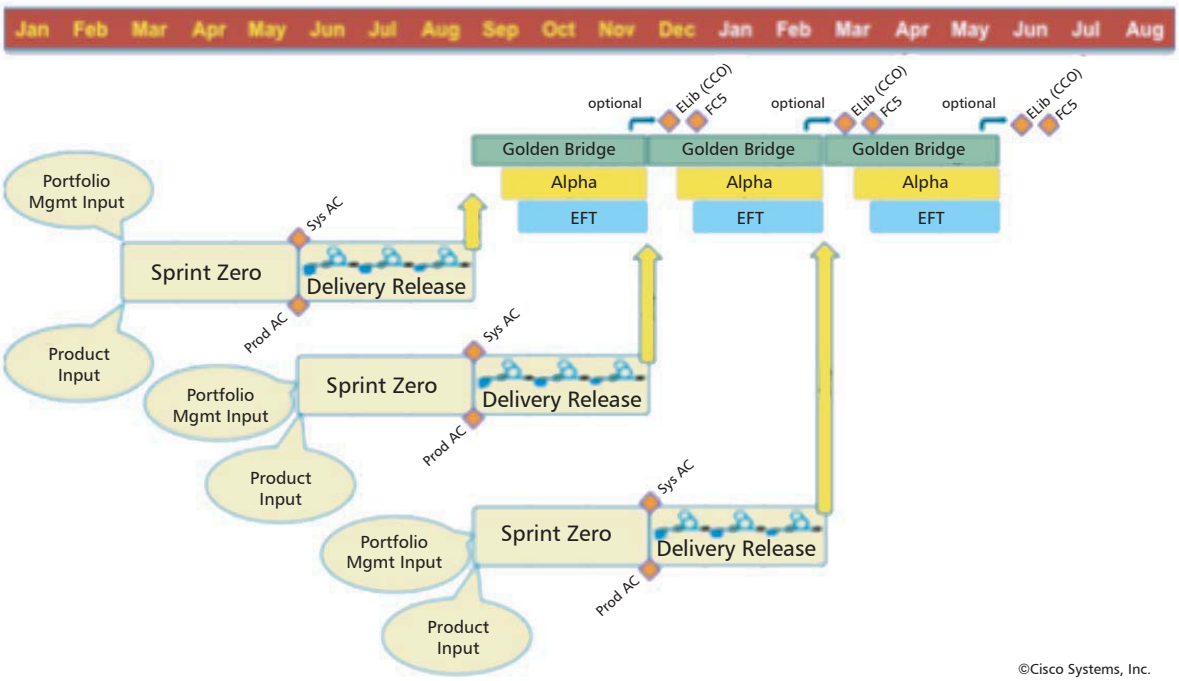


Figure 5 — Agile system release possible future.

CONCLUSION AND FINAL THOUGHTS

We have all heard it said that life is about the journey not just the destination. In the case of any major transition, there are steps forward and steps back. Persistence in the face of resistance, patience in the face of mistakes, and an open-minded, solution-oriented attitude are all key human factors in any successful change. Although it may seem easier to take a dogmatic view of agile/Scrum, the real power, in the authors' opinion, is the development of a learning organization: one that can adapt to changing market, economic, political, and cultural changes and emerge stronger and better. No matter what process or methods are used, it is still people who must do the work, make the decisions, and be accountable for their contributions (or lack thereof). If agile/Scrum is viewed as a framework, rather than a rigid list of hard and fast rules, an environment and culture can change to reward growth, creativity, innovation, and collaboration with business results that will please all stakeholders "top to bottom."

Overall, we are now in the thick of inventing, inspecting, adapting, and learning. This is an exciting time for us. One thing that has proven to be true is that whatever problems were present before the start of the agile transition are not solved automatically; no process is a silver bullet. People still have to do the work, and processes do not make up for failure to think, make the right decisions, or execute. But as each of our problems becomes clearer — better defined if you will — we have a much better chance of solving them by applying the appropriate mitigations or interventions. This entire process is like peeling an onion. Our reward is that we are slowly but surely moving into the sweeter part of that onion.

ENDNOTES

[1]Anderson, Philip. "Seven Levers for Guiding the Evolving Enterprise." In *The Biology of Business*, edited by John Henry Clippinger III. Jossey-Bass, 1999.

[2]Kotter, John. *Our Iceberg Is Melting: Changing and Succeeding Under Any Conditions*. St. Martin's Press, 2006; Kotter, John P. *Leading Change*. Harvard Business Press, 1996.

[3]A solution release includes multiple products and may include products from other technology groups. A system release includes multiple solution releases where solutions have dependencies; this is the large scale we are talking about in this report.

[4]The four agile values captured in the Agile Manifesto are: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan (www.agilemanifesto.org).

[5]Royce, Winston W. "Managing the Development of Large Software Systems." *Proceedings of IEEE WESCON 26*, 1970.

[6]This remark is based on a principle of queuing theory: the length of the queues determines the throughput of a system. We expected an audience with its roots in telecom systems, in which throughput is modeled on the very same queuing theory, to "get this." Alas, this was not the case.

[7]Internal release refers to three consecutive sprints, delivering working software into the systems (integration) process, while external release is a series of internal releases delivering software to the customers.

[8]They were coauthor Kathleen Rilliet and Michael Kalt, senior program manager.

[9]Poppendieck, Mary, and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.

[10]Cohn, Mike. *User Stories Applied*. Addison-Wesley, 2004.

[11]Kniberg, Henrik, and Mattias Skarin. *Kanban and Scrum: Making the Most of Both*. C4Media, Inc., 2010.

[12]Kotter. See 2.

[13]No man or woman is an island; the authors are part of a team, and without the colleagues in the Systems & Solutions Group, we would not have succeeded, nor had as much fun doing what we did.

[14]See Bruce Tuckman's Forming Storming Norming Performing team development model, introduced in 1965 (http://en.wikipedia.org/wiki/Tuckman's_stages_of_group_development).

[15]Reinertsen, Donald G. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

ABOUT THE AUTHORS

Hubert Smits is a Senior Consultant with Cutter Consortium's Agile Product & Project Management practice. He is an innovative, assertive, and goal-oriented agile consultant, coach, and trainer who has successfully spearheaded enterprise-level Agile-Lean enablement efforts, including those at Oracle, Avaya, BMC Software, AOL, Fender, GE Energy, Amdocs, MySpace, and Compete. Mr. Smits has worked with globally distributed teams and has coached teams in India, Argentina, Israel, and Europe. As the lead consultant at Cisco Voice Technology Group, he designed and implemented its agile SDLC, replacing multiyear waterfall processes with iterative product development cycles. More than 3,000 participants in product development at Cisco, including executives, product managers, program managers, and development teams have moved into Scrum roles and processes.

Mr. Smits is an exceptional communicator offering strong negotiation, problem resolution, and team-building skills. He enjoys leveraging agile principles and practices to mobilize teams, establish focus, promote positive organizational change, enhance performance, improve quality, and reduce team turnover. Mr. Smits specializes in working with large product development efforts in firms that require a gradual organization-wide move to agile practices. He is confident in restructuring SDLCs, interacting with IT, and managing the impact of organizational change on finance, product management, marketing, and sales. In addition to publishing with Cutter and the IEEE, Mr. Smits is a frequent speaker at agile events. He can be reached at hsmits@cutter.com.

Kathleen Rilliet, a Certified Scrum Master, is a member of both the VTG-wide Agile Transition and Systems Group Agile Transition Teams. As a Program Manager at Cisco, she specializes in product development, sales enablement, and launch processes. Ms. Rilliet has an extensive background in training and coaching in the areas of project, program, and portfolio management as well as business process development and implementation. Functioning as both an internal and external consultant, she has had an opportunity to work on business process improvement in companies large and small in technology, medical device, manufacturing, and arts organizations. Ms. Rilliet has degrees in both theater arts, dance, and organizational behavior and management. She can be reached at krilliet@cisco.com.