



# Bridging *the* Gap

*Agile Projects in the Waterfall Enterprise*

by MICHELE SLIGER



Although agile software development methodologies have been around for almost a decade, interest in agile approaches has exploded recently as companies seek better ways to compete with their global counterparts by getting their software products to market more quickly. The economy's downturn and subsequent budget cuts have led to greater interest in how to do things faster and how to do more with less. As a result, agile methodologies—Scrum, Extreme Programming, Lean Software Development, Crystal, Dynamic Systems Development Method, Adaptive Software Development, and others—have attracted many CIOs who are looking for approaches to make product development faster, more reliable, and more satisfying to the end-user.

While these promises are enticing, most agile methodologies are designed for small, collocated, collaborative teams. Knowing these constraints—yet still feeling a need to respond rapidly to market pressures—companies with large and geographically dispersed IT divisions have chosen to forge ahead, making changes when necessary, to adopt agile methodologies within their large, historically waterfall-oriented organizations. Because it's simply impractical to “flip a switch” and have a 1,000-person IT department start doing agile all at once, these organizations must wade through a sometimes murky transition period when agile and waterfall are forced to coexist.

During such a transition, what's an IT enterprise to do? Agile and waterfall are so utterly different—from the way projects start (diving right into coding vs. spending long weeks in analysis and design) to the types of meetings held (quick, daily planning meetings vs. long, weekly status meetings), to what we expect of the team (self-organized vs. directed), and even the

expected deliverables (chunks of working code early and often vs. documentation early with code at the end). These two practices have different ways of measuring progress, determining success, managing teams, organizing, and communicating. How can they be managed as part of a cohesive project portfolio? Can agile and waterfall methodologies coexist and still make the company successful?

The answer is yes. Not only can they coexist for the interim but they can coexist for the long term, since not all companies will choose to move every software development project to an agile paradigm. The trick is in how they can coexist peacefully and not detract from operational stability and continued project success. Every transitional environment, agile or otherwise, has to deal with duality until the transition is complete. Doing this with the least amount of pain and disruption means tightly embracing one of the key agile tenets: “inspect and adapt.” Process reviews and the progress the teams have made at the end of every iteration (typically one to four weeks in length) guide decisions on how best to proceed in the next iteration. This allows teams to adjust some of the agile practices to work better in the current environment, knowing that these agile practices may change and become more “pure” as the environment and culture change over time.

Some agile purists will say that by stretching the process it's no longer truly “agile,” and in many cases they may be right. Semantics aside, it's still about improving the overall software delivery system, and the label we apply to the methodology is secondary. However, teams must be careful not to stretch agile practices so much that the teams revert to their more comfortable, waterfall habits. When agile is viewed as an à la carte menu of practices that can be adopted as the teams see fit, it's critical to adhere to a diet of key principles—continuous improvement through time-boxed iterative deliveries and reviews, implementation of the most important items first, and constant collaborative communication. As they begin their transition, teams following these principles will find ways to integrate agile and waterfall.

## The Agile-Waterfall Cooperative


Large companies have clear, valid reasons for requiring certain waterfall activities on otherwise agile projects. Project approval processes designed to weigh benefits, costs, and strategic alignment with corporate objectives are one example of a standard *waterfall-up-front* requirement. Independent Validation and Verification (IV&V) is a *waterfall-at-end* requirement for those companies in industries that must comply with external regulations. And *waterfall-in-tandem* occurs when the system under development is so large and complex that multiple teams, often working on different platforms, must work together to prepare a release.

None of these requirements will cease to exist simply because the teams are employing agile practices. Instead the teams must learn to factor these corporate needs into their agile procedures, and management must begin the investigative work of determining how to streamline these requirements and activities so they don't hamper the project.

### Waterfall-up-front

I counseled one team to use Alistair Cockburn's “barely sufficient” philosophy (see the Sticky Notes for more information), in order to avoid doing more than was absolutely necessary when faced with waterfall-associated deliverables. The team members had been newly christened as an agile pilot team when they found that they still had to go through the standard waterfall project approval process to obtain the necessary resources and funding.

Using the “barely sufficient” guideline, the team asked, “Is this something that we really must do? And if it is, then what is the simplest thing we can do to satisfy the approval process?” Project approval was a must for the team, because without approval the project team would be disbanded and the monies and staff would go to other, previously approved projects. But conversations with the financial controllers revealed that the documentation required for the approval process could be streamlined and simplified, thus saving the team



**Using the “barely sufficient” guideline, the team asked, “Is this something that we really must do? And if it is, then what is the simplest thing we can do to satisfy the approval process?”**

from having to do the big up-front design that goes against the agile philosophy.

Based on the agreement reached with the financial managers, team members did the “barely sufficient” analysis and design work required to produce a technical specification document, which they submitted with their business case for approval. The design specification was a high-level overview of the system with high-level functional specifications, but it avoided the detailed requirements that would have taken the team weeks to complete. The documents were considered just good enough for the review process,

approved the project.

Other teams with which I’ve worked received provisional funding prior to official approval, so they included the project approval requirements in their first two iterations as deliverables along with requested features. You can see an example of both approaches in Figure 1.

Don’t be afraid to put non-product items in the backlog, especially if you want to take advantage of the high visibility it affords. Ken Schwaber, author of *Agile Project Management with Scrum*, calls the list of project initiation work items the “Scrum Startup Backlog.”

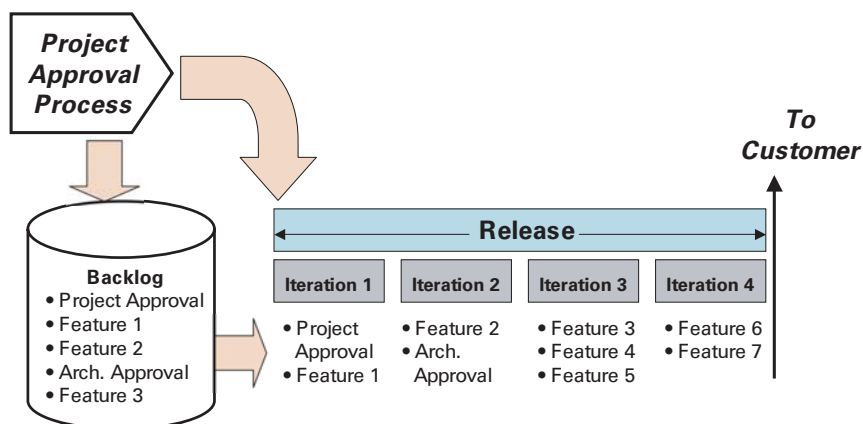


Figure 1: Waterfall-up-front

and the project was approved.

Once they had approval, team members held their first iteration planning meeting and brought along the specification that they had created. From the thirteen-page document they extracted a total of twelve sentences that represented the first few features they wanted to develop and then used this information as their initial product backlog.

They set aside the specification and rarely referenced it during the course of the release. The creation of this specification was not, however, viewed as a waste of time. Rather, team members felt they had benefited from the shared product vision created from the exercise of compiling the specification. And of course, the financial managers on the project approval board got the information they needed to determine capitalization (some companies require a technical specification before deciding whether a project’s budget can be labeled as capital or expense) and thus

### Waterfall-at-end

Another agile team I know of in a large telecommunications firm discovered that having to deal with waterfall-at-end requirements meant adding an extra iteration to prepare the required deliverables. This team produces application software for internal end-users, and it must switch from agile to

waterfall at the end of each project in order to prepare the software for production.

Passing through the required phase-gate of the team’s production department means that all documentation, meetings, end-to-end system testing and compatibility testing, and sign-offs must be planned and carried out. The team decided to use story cards to represent each item on the production checklist and allocated an entire iteration to production readiness. Team members did not implement any new features during this iteration; instead they produced documentation for production, customer support, the architecture review committee, and systems test. (Note: This company’s system test was concerned primarily with the submitted application’s compatibility with other existing applications on the network—the agile team still tested its code in each iteration.)

Figure 2 illustrates this example. If team members had any time left in this iteration, they spent it refactoring code, installing development and test system upgrades, investigating new technologies, and training staff.

An iteration dedicated to preparing the passage of potentially shippable code base to another entity is useful in a wide variety of situations: Sarbanes-Oxley auditing, FDA approvals, and IV&V. Even if your product does not need outside approval, you can use this hardening iteration to capture final screen shots for marketing and training materials, finalize the release notes, conduct performance and load testing, and other organizational activities. Just be sure that your team isn’t using this last iteration solely as a

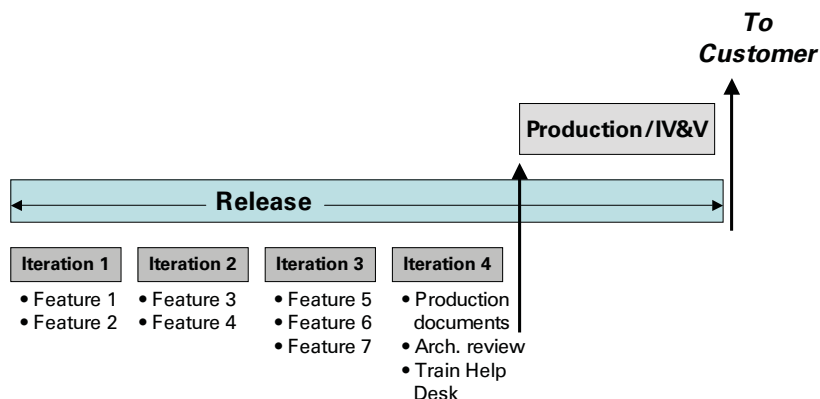


Figure 2: Waterfall-at-end

bug-fixing opportunity; that's a signal that the team is committing to more features in each iteration than it has time to test.

### Waterfall-in-tandem

A healthcare management company with which I'm currently working has multiple complex systems—systems so large they must be broken down into components, with each component having its own project teams. Data flows from legacy minis and mainframes to newer, browser-based applications and then back again—and sometimes even on to something else—resulting in huge coordination efforts to ensure successful system integration.

One of the component teams was piloting agile and realized that in order to coordinate milestone deliverables with the other teams they would have to

successful than email, overseeing multiple teams was still difficult for the waterfall managers. So, after another iteration review, the process was modified again to include other key members of the waterfall teams in addition to the waterfall project managers. All the teams eventually came together for release planning meetings, with a smaller set of waterfall team representatives attending iteration planning meetings when needed (see Figure 3). A second tier of daily stand-ups was added for the team leaders, as in the Ken Schwaber "Scrum-of-Scrums" model. For more information on how these meetings work, see the StickyNotes for a link to a nicely articulated definition and graphic by Mike Cohn. These teams, waterfall and agile, used the iteration reviews to inspect and adapt and thus created their own natural transition plan.

company-sanctioned style sheets. Instead the agile team members created a simple UI that was functional and scalable, albeit not particularly attractive. At the end of the first release, the product was deployed internally with the agile team's UI. By the next release the waterfall team had delivered, and the official screens became part of that second release. In the meantime, the end-users had access to the needed functionality, and the agile team was praised for its efforts.

### The Anomaly—Two Separate Entities

I'm aware of one organization that has decided to keep the agile and waterfall groups completely separate, with no overlap at all. This particular company set up its agile teams as a separate organizational unit, and its internal clients pay the department for the use of its agile teams for a fixed period of time (not for a set project).

These agile teams function as a type of "skunk works" operation (see the StickyNotes for more information). Copyrighted by Lockheed Martin, the "skunk works" phrase was created by a 1943 Burbank, California, team that had been tasked with creating the P-80 jet fighter from scratch. Clarence "Kelly" Johnson, who headed the team, described a skunk works operation as "a small group of experts who drop out of the mainstream company operations in order to develop some experimental technology or new application in secrecy or at speed, unhampered by bureaucracy or the strict application of regulations." These stealth agile teams evade waterfall controls thanks to the power of their executive sponsors and their paying clients, who work together to protect and incubate the teams so they can focus on their assignments. And by keeping the two separate, management avoids dealing with the issues surrounding changes needed in organizational structure, project

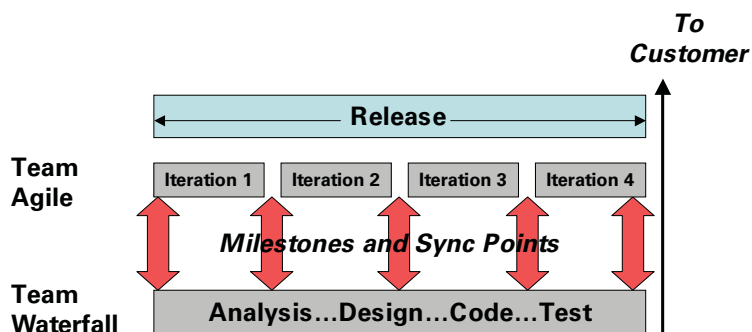


Figure 3: Waterfall-in-tandem

include the waterfall project managers in the planning sessions. This realization came after attempts to stay synced up via emails failed horribly. The agile team members realized they had left the "collaborative" out of the "constant collaborative communication" principle (sorry, constant emailing doesn't count!), and so began inviting the waterfall project managers to all their planning meetings—release planning, iteration planning, and daily stand-ups. Initially the waterfall project managers grumbled that all these planning sessions were wreaking havoc on their calendars. However, once they had attended a few sessions, the managers began to realize the value of the shared information and the improved ease and coordination of the work.

Although this approach was more

Don't be surprised, however, to find your agile teams moving ahead of those waterfall teams that aren't inclined to cooperate or are unable to deliver. Agile teams usually figure out a way to keep making progress, even without the expected deliverables on which they rely. The agile team will stub out that missing subsystem and create work-arounds, or they will simply build the module themselves.

In one instance, agile team members decided not to wait for the waterfall user interface (UI) design team to provide the

**Don't be surprised, however, to find your agile teams moving ahead of those waterfall teams who aren't inclined to cooperate or are unable to deliver.**

approval procedures, ongoing portfolio metrics and management, and overall culture that must be addressed in an agile-waterfall cooperative.

Making changes—especially of this magnitude—is never easy. But most teams have told me that the hardest part is just getting started. Once teams begin adopting some of the practices, they find the rewards surprising and well worth the continued effort. Organizations find that being able to better respond to the marketplace isn't the only reason to switch to agile. The cooperative and collaborative nature of the approach provides a more supportive, meaningful, and exciting work environment for the members of the team. Agile is a commonsense approach to software development, and the success of a waterfall-agile enterprise lies in creating a transition plan to excellence through the cycle of inspection, adaptation, and execution. **{end}**

---

*Michele Sliger has worked in software development for more than fifteen years. She currently works as an agile consultant for Rally Software Development, where she trains software development teams in agile methodologies. Previously a project manager at Qwest Communications and a consultant for Fortune 500 companies, Michele was a founding member of the engineering teams at two biotech start-ups. She is a certified Project Management Professional and a Certified Scrum Master. Michele is also an adjunct faculty member of the University of Colorado, where she teaches software project management to graduate engineering students. Email Michele at [msliger@rallydev.com](mailto:msliger@rallydev.com).*

### Sticky Notes

For more on the following topics, go to [www.StickyMinds.com/bettersoftware](http://www.StickyMinds.com/bettersoftware)

- Alistair Cockburn's Barely Sufficient methodology
- Mike Cohn on daily stand-up meetings
- More on "skunk works"
- Further reading

## The **KEY TO SUCCESS** in each of these examples is the teams' approach to transition problems—solving problems collectively, collaboratively, and with the freedom to make decisions and changes within the discipline of regular reviews. Here are several recommendations to help your organization develop ways for agile and waterfall to effectively coexist:

- Talk with waterfall stakeholders about how you'd like to work together. Answer their questions about agile, and ask for their help in making the relationship as pain-free as possible. The executive sponsor should be part of the initial discussion, to share her commitment to the process, her dedication to removing roadblocks, and the importance of each team's involvement.
- The waterfall requirements of the project should become stories in the agile team's backlog. Include waterfall stakeholders in the agile planning meetings, so that everyone understands what qualifies as barely sufficient deliverables, what assumptions the teams are making, and what dependencies exist.
- In the review and retrospective held at the end of each iteration, analyze the benefits and challenges you've just experienced and make recommendations on how to improve the experience in the next iteration. Again, be sure to include the waterfall stakeholders.
- Executive and middle management should create their own prioritized backlog of transition issues they need to focus on, implementing corporate process and organizational change iteratively and incrementally.
- Don't try to fix everything before you start agile adoption. Just dive right in—you'll find a way to work within the current constraints. Use the iteration reviews and retrospectives to make change recommendations and implement incremental improvements as you go. Be reasonable about this by focusing on only the top two or three things for the next iteration. It's a backlog of recommendations and, like the backlog of product features, they all can't be implemented at once.
- Pay attention to behaviors and avoid reverting to old habits. For example, if you notice that you're depending on the specification that you had to create in the waterfall-up-front iteration instead of having discussions with the product owner, set the document aside.
- Invite all stakeholders to participate in a project retrospective at the end. These meetings are crucial to help identify and implement broader transition changes that affect the entire enterprise.