Agile Product & Project Management

REPRINT

CUTTER CONSORTIUM

# How Agile Projects Measure Up, and What This Means to You

by Michael Mah, Senior Consultant, Cutter Consortium; with contribution by Mike Lunt

In this *Executive Report*, we share observations about real-world agile projects and how they measure up against waterfall and other projects in terms of productivity, schedule, and quality. This was made possible by juxtaposing them against trends from a contemporary worldwide database of more than 7,500 completed projects. Specifically, we look at more than 20 agile releases from five companies. Two of the companies are examined in specific detail, each having achieved best-in-class levels of performance by implementing agile in very different ways. The report examines what was learned from this analysis and illustrates how your organization can measure your own projects, agile or not, and how to communicate results to team members and decision makers who face decisions about shaping an IT organization to meet future challenges head-on.

Executive Report

*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; E-mail: service@cutter.com; Web site: www.cutter.com. Group Publisher: Kara Letourneau, E-mail: kletourneau@cutter.com. Managing Editor: Cindy Swain, E-mail: cswain@ cutter.com. ISSN: 1536-2981.*

# How Agile Projects Measure Up, and What This Means to You

## THIS MONTH'S AUTHOR

Michael Mah
Senior Consultant, Cutter Consortium

## IN THIS ISSUE

## THE WORLD OF TECHNOLOGY AT-LARGE

We live in a world that is driven by technology — the Information Age. This is a world that is quite different from the Agricultural Age, which produced modern farming methods and the rapid production of food, and the Industrial Age, which created factories that revolutionized rapid production and the manufacturing of goods.

In this Information Age, if you live in a house, talk on the phone, drive a car, take public transportation, work in an office, go to the bank, or eat at a restaurant, then computers, software, and information flow are all around you. Worldwide, it's been estimated that people and companies invest trillions of dollars in IT. More than 60% of that is estimated to be in software, services, and mobility. Technology is revolutionizing the rapid flow of information around the world.

For those of us who work in the field of IT, we know that the following are also true:

- Today, many companies that we work with or for are global companies.
- Much of the time, teams are spread out and are rarely in the same place: many perceive that the "world is flat," and thus, it's been an accepted norm for people on a project to be dispersed.
- Software and systems that we are asked to build today are more complex and harder to make than the ones we built yesterday.
- We have less time for projects than we used to; time is accelerating.
- Building complex software inside a tight deadline is very hard to manage, often resulting in high defect rates and reduced scope at delivery.

More than 20 years ago, at the dawn of the modern Information Age when the technology world began its dizzying acceleration, many organizations were criticized because projects took too long and cost too much. In the 1980s, studies by the US General Accounting

Office and the UK Department of Trade and Industry reported high percentages of software projects that overran their budgets, missed their schedules, or were cancelled. When projects did make their dates, it was often achieved by cutting scope.

Our industry responded by creating new processes; the intent was to build better software — faster, at lower cost, with higher quality. The process movement in the 1980s looked at Japan's manufacturing success and its adoption of quality principles by gurus such as W. Edwards Deming and decided that a similar path would create better software. We emulated total quality management (TQM) ideas employed in manufacturing, created such terms as "the software factory," and crafted software development processes such as the Software Engineering Institute's Capability Maturity Model (CMM®) and ISO 9000/9001.

In the midst of this process movement, we examined whether companies were successful or not, and one answer came in the form of an oft-quoted research study called the Standish *Chaos Report*, which effectively said that software projects still take too long and cost too much.

## AGILE CLAIMS A BETTER WAY

In February 2001, a group of 17 people convened at the Snowbird ski resort to discuss new ideas, ski, relax, and find common ground about how to tackle this problem. What emerged was the Agile Software Development movement. This group included individuals such as Cutter Fellow Jim Highsmith, Kent Beck, Ken Schwaber, Alistair Cockburn, and many others. They produced the Manifesto for Agile Software Development.[1] In part, it states:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> **Individuals and interactions** over processes and tools
>
> **Working software** over comprehensive documentation
>
> **Customer collaboration** over contract negotiation
>
> **Responding to change** over following a plan

In his book *Agile Software Development Ecosystems*, Highsmith says that although this was a group of recognized software development "gurus," they adopted use of a key word: *uncovering*.[2] It was chosen to demonstrate that they didn't claim to have all the answers. What they were committed to was *doing software development* differently (themselves) and simultaneously examining their *values* in creating computer software. Through their experience of "doing," they would helps others do things better as well.

The question was, if they did things in this new way, would it work? How would you demonstrate this fact to decision makers? How would one measure that?

What agilists believed was that changing *how companies and teams work* — effectively changing the culture — was needed to make better software. In a 2002 *Cutter IT Journal* on XP and culture change, Beck said, "This is hard, and it might be easier [in some ways] to go back to 'the old way.' So why bother?"[3]

In answering his own question, he articulated that projects "could be more successful in the sense that they would produce more functionality and fewer defects." You could extend that line of thought to say that creating software would take less time and cost less money in addition to being higher quality. Is Beck right? Could we offer answers to decision makers who might consider an agile approach? Do agile projects produce more functionality with fewer defects? And if so, by how much?

## ENTER SOFTWARE MEASUREMENT: THE QSM DATABASE

To answer these and other questions, we require data. Fortunately, such data exists: a large modern repository of more than 7,500 projects collected worldwide from many of the world's leading companies, with performance trends on schedule, effort, defects, and productivity statistics for projects small, medium, and large. This research is collected and updated each year by my company, Quantitative Software Management (QSM) Associates, and is used for benchmarking engagements on behalf of clients at Cutter Consortium.[4]

The database is located in a repository called SLIM-Metrics™ and contains project statistics collected from more than 500 organizations across 18 countries throughout North America, Europe, Asia, Australia, and Africa. The projects span all size regimes, application types, and development methodologies, including offshore/outsourced, inhouse, new development and maintenance, package implementation/ERP, waterfall, incremental development, and, increasingly, agile development.

### Data Streams in on Agile Projects

Today, more companies are incorporating agile practices in developing software. Here is a synopsis of findings from recent data collected from agile practitioners:

- **Agile teams have metrics.** Many keep reliable records of their schedules (time), the team members working on their projects (effort), requirements/features, and the "stories" (a size metric) accomplished for each work iteration. This can be easily drawn as a staffing sketch showing the team headcount by month, from the start to the end of a project or release. This profile can provide an image depicting essential metrics: time, effort, and size. They also keep track of a fourth metric: bugs (defects). We can then load this information into a project database like SLIM-Metrics.

- **Agile trends can be plotted on charts for schedule and defects, as well as staffing and effort.** Smaller releases, medium-sized releases, and large releases are charted from left to right. This enables us to compare agile projects of disparate size on one trend so agile projects can be positioned against industry projects of similar size.
- **As a group, the projects were mostly faster than average.** About 80% were below the industry average line. Some took longer for various reasons. Some companies were much faster — in some cases, developing software in half the time. These generally used larger-than-average teams. Even though many of the smaller releases used small teams, some, apparently in response to deadline pressure, used larger teams than average. One company applied seven parallel Scrum teams totaling 95 people, where the norm was about 40 people.
- **Waterfall projects showed higher defects when using large teams trying to meet fast schedules, sometimes four times higher than average.** On many agile projects, we saw a far lower number of defects in some of the companies. The best performers had high-maturity XP and Scrum teams. These project teams showed defects that were 30%-50% lower than average. Other less-mature agile teams had defect rates that were more like waterfall projects.

## WHAT AGILE PROJECTS LOOK LIKE

### Case 1: Follett Software Company (Colocated XP)

Let's examine how these observations were derived, beginning with a company that implemented agile XP with teams of paired programmers physically colocated in one large room. This company is Follett Software Company in McHenry, Illinois, USA, which agreed to be named for this study. (Follett develops commercial software products used by school districts to manage educational resources such as textbooks, library books, and other educational media.)

Follett initially chose XP for several reasons. It felt that its older waterfall process resulted in longer schedules and higher defects than desired. At the time of the XP transition, it was also transitioning to create a new product line and software architecture; thus, it seemed reasonable to move to the XP development approach simultaneously. Other reasons that XP appealed to it included:

- An ability to be more flexible to change
- Tighter feedback loops during iterations with key stakeholders, including customers, marketing, and management
- A more appealing approach to dealing with project uncertainty when things were not well-known up front
- Moving away from previous homegrown processes toward more industry-standard processes
- XP iterations aligned well with its move toward a Java/SQL architecture
- Translating desired features into stories and then weighting story complexity using story points (on a 1-7 scale)

After building several releases with the new methodology, Follett also decided that it was very important to communicate results in management terms to senior executives. This included productivity, time to market, project cost, and quality. In effect, management was an "investor" and key sponsor to the XP initiative. Shadowing this entire initiative was some speculation by senior executives who wondered whether offshoring was a more viable option. Their own historical waterfall experience led them to question the degree to which XP really improved productivity and quality. Without metrics, the debate was more charged with opinion and speculations than actual data.

Follett contacted me in the spring of 2006 to conduct a productivity benchmark study against industry data. I met with its team, which included about 15 people in one large room, where we collectively drew a sketch of their XP projects across three major releases on a whiteboard. Figure 1 shows a photograph of its XP room with paired programmer workstations. Figure 2 depicts one of the sketches for the story (requirements) activity and the build/test activity for one of its releases that spanned about six months with a peak staff of 25 people.

Let's first consider Figure 2 to summarize the numbers that emerge from this picture. We begin with the left side of the graph depicting the start of the story phase with two people working on story definition starting

Figure 1 — Photo of XP room.

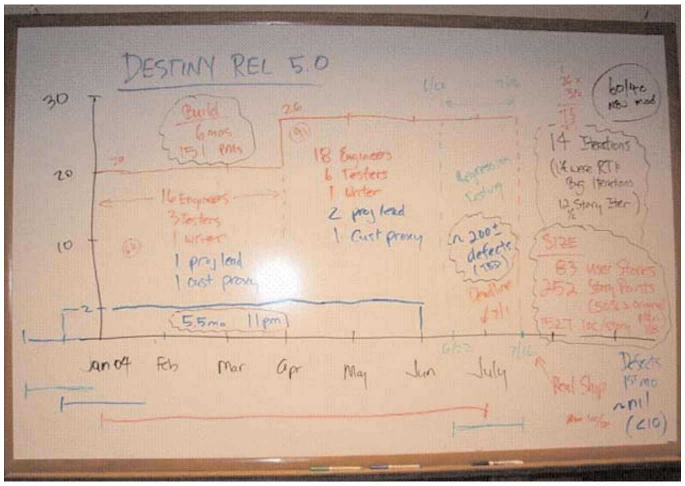


Figure 2 — Whiteboard sketch of Follett's Destiny 5.0 release.

in mid-December. This work continued to the first of June, upon which it concluded. The elapsed time for this phase was 5.5 months. For each month, two person-months work of effort were expended by the two full-time staff. Thus, the total work effort over the 5.5 months was two person-months per month x 5.5 months = 11 person-months.

Next, we turn our attention to the build phase, which includes testing. This commenced in January with 20 full-time staff and proceeded for three months. From that point forward (beginning in April), the team ramped up to 26 staff and continued through the middle of July. The total elapsed time for the build phase from January to mid-July was 6.5 months (the sketch reads six months, which is incorrect). The expended effort was 60 person-months for the first three months, plus 91 person-months for the rest of the build phase, or 151 total person-months.

Next, we consider the volume of work produced by the team. Across 14 iterations (which each lasted two weeks), the team produced software that satisfied 83 stories, comprising 252 story points. This was tallied by examining the story points assigned to each of their 83 stories using their 1-7 scale, with more complex stories receiving higher point assignments. The volume of software worked on in this release comprised 126,748 lines of new code and 91,783 lines of changed code, for a total of 218,531 lines of code.

Finally, during regression/QA testing, which occurred in the last month of the build phase, the team found and fixed about 200 defects. Upon further review, it determined that the correct count was 121 when removing duplicates.

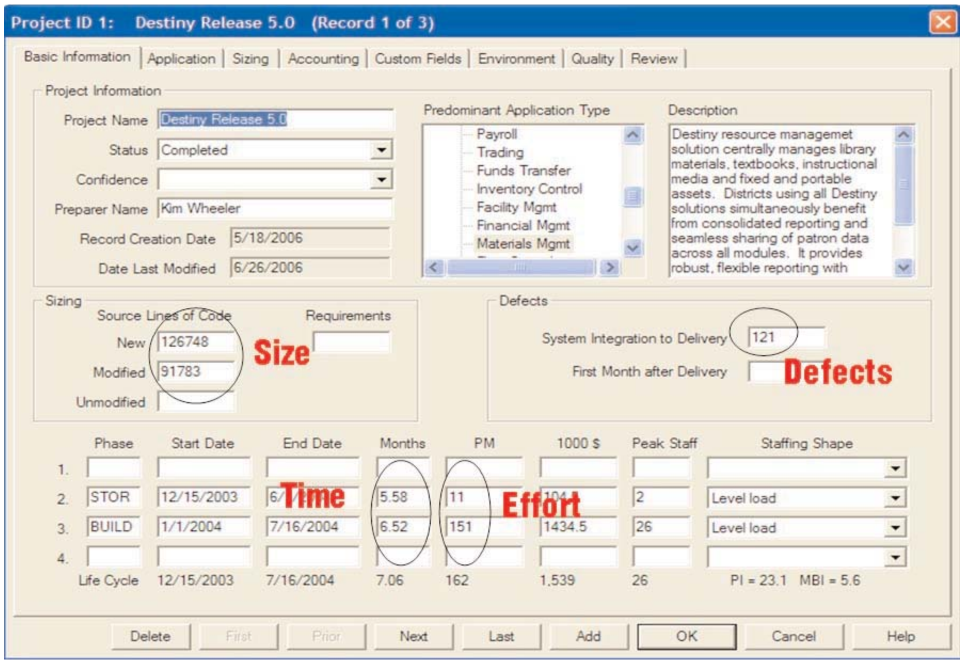
In summary, we have four core metrics for the build phase: the time being 6.5 months; the effort at 151 person-months; the size was 218,531 lines of new plus changed code for the 83 stories and 252 story points; and the system integration/QA defects were 121. For the story phase, the time and effort values were 5.5 months and 11 person-months, respectively. This is shown as a SLIM-Metrics data record in Figure 3.

For Follett, staffing profiles and data records were constructed for five additional releases over a three-year period, for a total of six XP projects.

## Case 2: BMC Software (Distributed Scrum)

The next company profiled in detail is BMC Software, based in Austin, Texas, which also agreed to be named for this study. BMC creates applications that allow clients to manage distributed systems environments, including network, applications, databases, and operating systems.

BMC's group chose Scrum for its agile implementation. What was distinctive about its environment was the fact the group was managing seven Scrum teams across four distributed locations, with time zones differing by as much as 12 hours. This was one of the largest agile teams that I encountered, with 95 full-time equivalent engineers, developers, testers, and project leads. When I asked about the primary objective for implementing Scrum, the team replied that its previous product release cycles were in the 12-to-18-month range. The goal was to dramatically shorten time to market while keeping defects as low as possible.



Project ID 1: Destiny Release 5.0 (Record 1 of 3)

Basic Information | Application | Sizing | Accounting | Custom Fields | Environment | Quality | Review

Project Information

| | |
|---|---|
| Project Name | Destiny Release 5.0 |
| Status | Completed |
| Confidence | |
| Preparer Name | Kim Wheeler |
| Record Creation Date | 5/18/2006 |
| Date Last Modified | 6/26/2006 |

Predominant Application Type: Payroll, Trading, Funds Transfer, Inventory Control, Facility Mgmt, Financial Mgmt, Materials Mgmt

Description: Destiny resource managemet solution centrally manages library materials, textbooks, instructional media and fixed and portable assets. Districts using all Destiny solutions simultaneously benefit from consolidated reporting and seamless sharing of patron data across all modules. It provides robust, flexible reporting with

Sizing

| Source Lines of Code | | Requirements |
|---|---|---|
| New | 126748 | |
| Modified | 91783 | |
| Unmodified | | |

Defects

| | |
|---|---|
| System Integration to Delivery | 121 |
| First Month after Delivery | |

| | Phase | Start Date | End Date | Months | PM | 1000 $ | Peak Staff | Staffing Shape |
|---|---|---|---|---|---|---|---|---|
| 1. | | | | | | | | |
| 2. | STOR | 12/15/2003 | 6/ | 5.58 | 11 | 104 | 2 | Level load |
| 3. | BUILD | 1/1/2004 | 7/16/2004 | 6.52 | 151 | 1434.5 | 26 | Level load |
| 4. | | | | | | | | |
| | Life Cycle | 12/15/2003 | 7/16/2004 | 7.06 | 162 | 1,539 | 26 | PI = 23.1 MBI = 5.6 |

Figure 3 — SLIM-Metrics data record for Follett's Destiny 5.0 release.

It didn't surprise me then to witness such an aggressive team size. What would be interesting to discover was whether defects would be high or low. From industry statistics, we knew that doubling staff sometimes results in 4x or higher defects than the industry average. Complicating this was a team that was physically spread across four cities and two continents, with one team in India. The team of 93 people was more than twice the industry norm of about 40 people for the amount of software contained in the release. Figure 4 shows a photo from our whiteboard session to sketch their staffing profile. Figure 5 depicts the sketch from one of their two releases that were measured.

Before we begin reviewing the BMC sketch, note the similarities in the general shape of the story and build phases compared to Follett. Both have rectangular, level-loaded staffing profiles with high degrees of parallelism between the story and build phases. For the most part, this seems to be a characteristic shape for agile projects. What differs is the magnitude of the team size; in this case, 93 people versus 26 for Follett. What also tends to vary is the duration of the release; however, in these instances, both happen to be about six months.

Let's examine the left side of the graph in Figure 5 depicting the start of the story phase for the BMC Scrum project. It begins at 10 full-time staff for the first two months beginning in June then steps down eventually to five people when the story work concludes at the end of September. The elapsed time for

Figure 4 — BMC software whiteboard session.

this phase was four months. The total work effort was 32 person-months.

Looking at the build phase, we see that it started immediately with 93 people in July and stayed at that level for just over 5.25 months when the release reached the "go live" milestone on 8 December. The total elapsed time for the build phase from July to December was 5.25 months. The expended effort was 93 person-months per month for 5.25 months, or 488 person-months.

The volume of work produced by this large team spanned 11 two-week iterations. Across these 11 iterations, the team produced software that satisfied

918 stories. (They did not break each story down with story-point values.) The volume of software comprised 834,540 lines of Java and 2,835 lines of XML code. About 25% of this total represented new code, while the remaining 75% was modified/changed. The total was 837,375 lines of code.

During regression/QA testing, which occurred in the last two months of the build phase, the team found and fixed 635 defects. At immediate glance, I intuitively knew that this was a low number for almost a million lines of code produced by a team of almost 100 people. The trendline from the QSM SLIM-Metrics database would later confirm this.

Summarizing this BMC release, the four core metrics for the build phase were: time at 5.25 months; effort at 488 person-months; size was 837,375 lines of new and modified Java/XML; with 635 defects found during QA and regression testing. This is shown as a SLIM-Metrics data record in Figure 6.

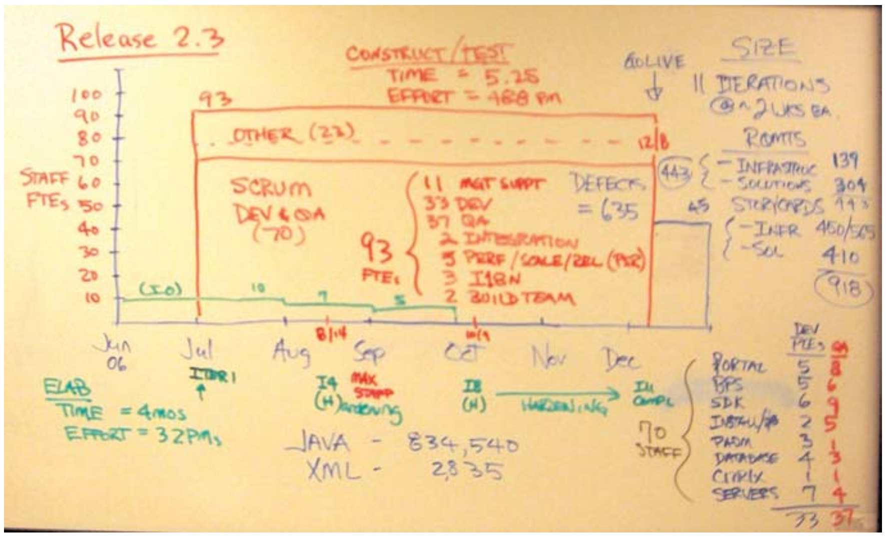
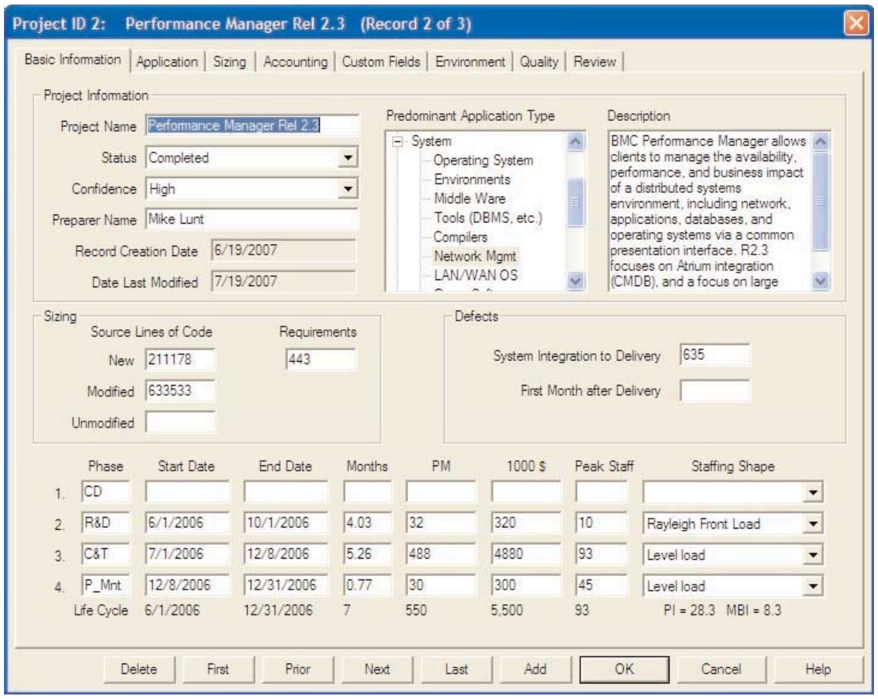


Figure 5 — Whiteboard sketch of BMC's Performance Manager 2.3 release.


Basic Information | Application | Sizing | Accounting | Custom Fields | Environment | Quality | Review
Delete | First | Prior | Next | Last | Add | OK | Cancel | Help

Project ID 2: Performance Manager Rel 2.3 (Record 2 of 3)

Project Information

| | |
|---|---|
| Project Name | Performance Manager Rel 2.3 |
| Status | Completed |
| Confidence | High |
| Preparer Name | Mike Lunt |
| Record Creation Date | 6/19/2007 |
| Date Last Modified | 7/19/2007 |

Predominant Application Type: System — Operating System, Environments, Middle Ware, Tools (DBMS, etc.), Compilers, Network Mgmt, LAN/WAN OS

Description: BMC Performance Manager allows clients to manage the availability, performance, and business impact of a distributed systems environment, including network, applications, databases, and operating systems via a common presentation interface. R2.3 focuses on Atrium integration (CMDB), and a focus on large

Sizing

| | Source Lines of Code | Requirements |
|---|---|---|
| New | 211178 | 443 |
| Modified | 633533 | |
| Unmodified | | |

Defects

| | |
|---|---|
| System Integration to Delivery | 635 |
| First Month after Delivery | |

| | Phase | Start Date | End Date | Months | PM | 1000 $ | Peak Staff | Staffing Shape |
|---|---|---|---|---|---|---|---|---|
| 1. | CD | | | | | | | |
| 2. | R&D | 6/1/2006 | 10/1/2006 | 4.03 | 32 | 320 | 10 | Rayleigh Front Load |
| 3. | C&T | 7/1/2006 | 12/8/2006 | 5.26 | 488 | 4880 | 93 | Level load |
| 4. | P_Mnt | 12/8/2006 | 12/31/2006 | 0.77 | 30 | 300 | 45 | Level load |
| | Life Cycle | 6/1/2006 | 12/31/2006 | 7 | 550 | 5,500 | 93 | PI = 28.3 MBI = 8.3 |

Figure 6 — SLIM-Metrics data record for BMC's Performance Manager 2.3 release.

## WHAT BENCHMARK TRENDS SHOW FOR AGILE PROJECTS

Having collected these measures, one might ask how to interpret their meaning. Yale University Professor Edward R. Tufte, the author of such seminal books as *Visual Explanations* and his most recent, *Beautiful Evidence*, once said, "The essence of quantitative thinking involves answering the question 'Compared to what?'"[5]

With that, we posed questions to ourselves about agile projects from five companies that were benchmarked, including Follett and BMC Software. All were ostensibly "agile," producing a range of results. In a Cutter Consortium interview on XP and culture change, we previously recalled that Kent Beck said:

> XP projects that are technically successful — in the sense of producing more functionality and fewer defects — offered greater transparency and control for the sponsors than projects done in the old ways of working.[6]

Considering Tufte's notion of creating visual comparisons and Beck's specific hypothesis about agile projects producing more functionality and fewer defects, let's consider charts that might offer visual evidence, if any, in at least these two domains to answer the following questions:

- **Do agile projects produce more functionality?** Another way of asking this is, "Do they produce functionality faster than industry (i.e., "older ways")?
- **Do agile projects exhibit fewer defects?** A useful apples-to-apples comparison is the number of defects found during QA and regression testing (agile projects versus industry trends).

With that in mind, we might recall the Follett XP project illustrated in Figure 2, where the team produced 218,531 lines of new plus changed code, satisfying 83 stories and 252 story points in 6.5 months. Two questions could be: Compared to historical projects in the SLIM-Metrics database, is 6.5 months faster than industry? What about the other five XP releases? We might ask the same questions about the BMC Scrum team and its projects.

Figure 7 illustrates the schedule performance for agile projects versus an industry trend line from the QSM SLIM-Metrics database.

This chart depicts the build phase schedules for 23 agile projects with smaller projects shown on the left, medium-sized projects in the middle, and larger projects on the right. The six Follett XP releases are shown on the right as gray circles. The two BMC Scrum releases are shown as gray squares. The smallest projects were in the range of about 10,000 lines of new and modified code, with the largest approaching nearly one million lines of code.

Vertically, the Y-axis is labeled in months from the start to the end of the build phase. Follett's XP release encompassed 6.5 months for the 14 iterations from start to finish. The industry average schedule depicted by the centerline is 10 months — 3.5 months longer! (Note that

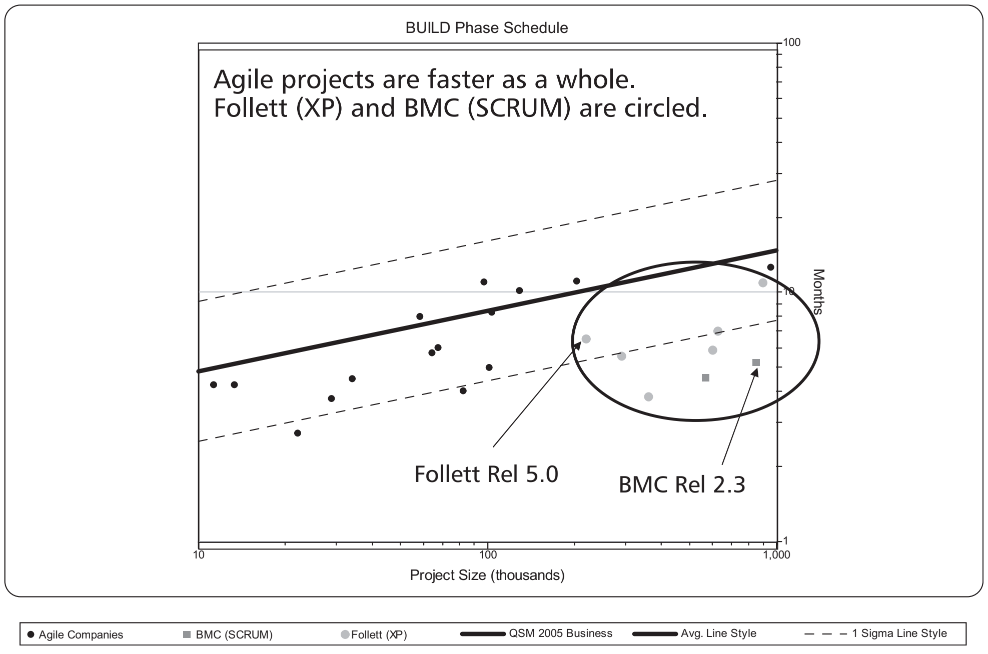


Figure 7 — Schedule trend: agile projects vs. industry.

the upper and lower dashed lines represent the range-span of the industry data from high to low, effectively a bell distribution curve.)

In the BMC case, its release spanned 5.25 months across 14 iterations. The industry average schedule for a project involving that many stories and code is more than a year!

What's especially noteworthy is how the data from both Follett and BMC (circled) are consistently faster. All of the gray data points for the most part are clustered far below the industry average (mean). Some more, some less, depending on the uniqueness of project complexity and team dynamics from project to project. Variation is normal; in other words, for the most part, one can interpret that both of these exemplary companies are about twice as fast as industry; their project schedules are about half the norm — 50% shorter times. Moreover, when you look at the black dots representing projects from three other companies, they too are generally fast. Overall, 18 of the 23 projects in this sample are below average line (faster) in terms of speed.

However, one caveat is in order — team size. One might consider the fact that when projects employ larger teams, this shortens project duration. Indeed, nine of the 15 projects from the other three companies used larger teams than average, some significantly more so. They said that their companies have "schedule-driven cultures," and this plays out in how aggressively they staff their projects. In BMC's case, it used twice the industry average team size — 90 to 95 people when the norm is about half that. Certainly, some of its schedule performance is due to "brute force staffing." The rest might be considered a result of their agile approach. A staffing trend is depicted in Figure 8. Note that all six of Follett's releases straddle almost exactly on the industry average line. Removing this variable, one might interpret that its schedule performance — about twice as fast as industry — is due to its XP approach.

Finally, let's consider defects. The Follett XP 5.0 release project exhibited 121 defects during QA and regression testing as an example. Is that fewer compared to industry for projects of this code and story size? We depicted how this, as well as the other five Follett XP releases (gray circles), compared against the defect trend. Collectively, they were 50%-66% fewer compared to our database. A defect trend line is shown in Figure 9.

With respect to the BMC Scrum projects, the two gray squares representing their releases straddled the average centerline. However, what is not apparent is this: when we examine waterfall projects that attempt to compress schedule by using an aggressively large team — in this case, 90 to 95 people instead of 40 — we often see defects rise geometrically, mostly due to communication complexity among the team. This is frequently exacerbated if they are not in one location.

It's chaotic to run a team of nearly 100 people. Defects easily rise by at least a factor of 4x as observed from the industry data. BMC's Scrum releases were at only 1x. One might conclude that had the team not had the high-bandwidth communication effectiveness from its Scrum

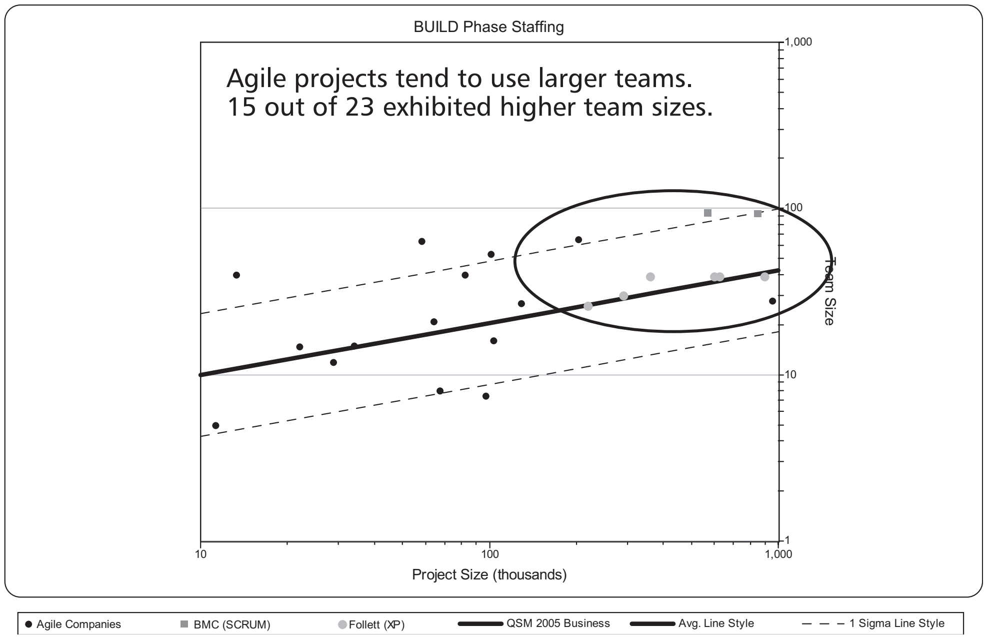


Figure 8 — Staffing trend: agile projects vs. industry.

process, that it could easily have suffered poorer quality and higher defects, which it did not.

Returning to Beck's assertion that agile projects are technically successful by delivering more functionality with fewer defects, the verdict is in for Follett and BMC Software: it is an absolute "yes" — by a very significant margin.

In summary, with regard to the other three companies — all ostensibly implementing agile but less mature — the verdict is mixed. Schedules are mostly faster, but defects are generally not faring as well.

## What This Means to the Bottom Line: Cost, Schedules, Defects

Tables 1 and 2 summarize the levels of performance achieved by these two organizations, one using a colocated XP approach and the other using a distributed Scrum strategy. Reminding ourselves of Tufte's challenge to answer the question, "Compared to what?" the tables provide a side-by-side comparison of each company's aggregate performance against industry averages for a project of the same software size. Table 1 shows the industry comparison for our first case study company, Follett Software.

We took the average size of all six of Follett's XP releases and computed the aggregate cost performance, schedule, and defects for its team. We derived an industry average productivity value from the QSM SLIM-Metrics database for this sized project and determined the cost, schedule, and quality assurance (QA) defects found and fixed during the QA phase.

Not only does this show a dramatic schedule reduction of almost five months compared to industry norms, with twice the level of quality (half the defects), but for each project, we determined that Follett saves a whopping US $1.3 million. Across six releases, you can multiply those savings, and the total cost savings is $7.8 million. That's enough to make a CFO blush.

Table 2 shows the industry comparison for our second case study company, BMC Software.

We took the average size of two very large BMC releases and also computed the aggregate cost performance, schedule, and defects for its team. Once again, we derived an industry average productivity value from the QSM SLIM-Metrics database for a reference project for comparison and determined the cost, schedule, and QA defects found and fixed during the QA phase.

Table 2 shows a very interesting, but different story — primarily because a different dynamic reveals itself due to team size.

In Follett's case, its staffing was right on the industry average. However, in BMC's case, it runs teams of about 92 people versus an industry average of 40. This aggressive team–size approach is designed to maximize speed and to shorten time to market. In this, it indeed succeeded, by completing its releases at a whopping 8.7 months faster than the industry norm.

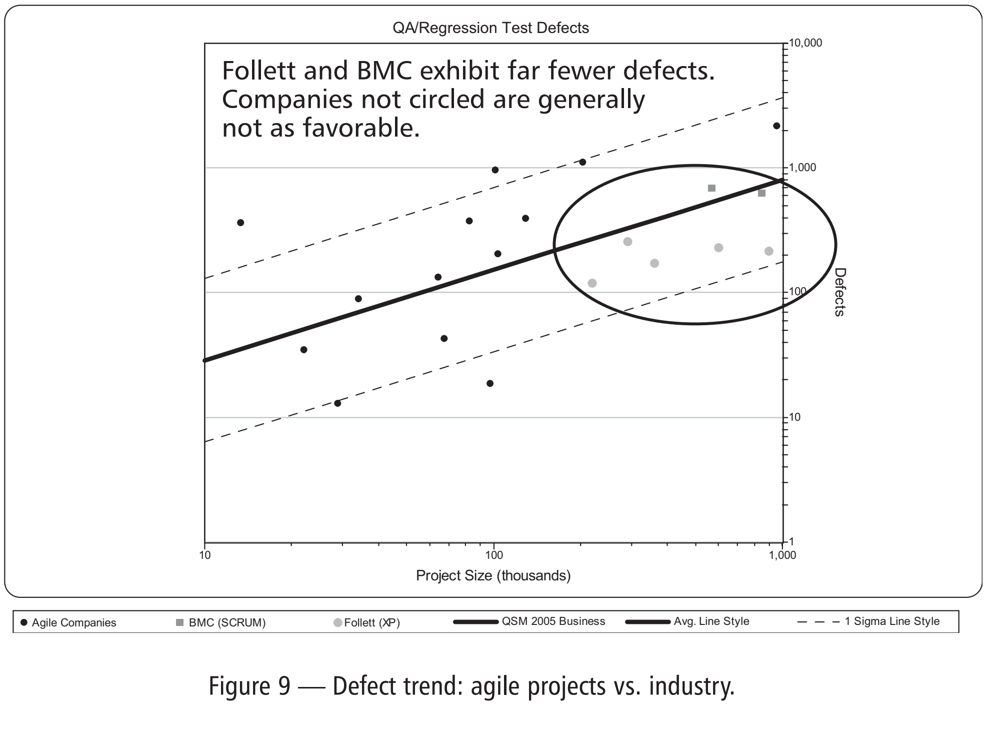


Figure 9 — Defect trend: agile projects vs. industry.

Table 1 — Follett Software Colocated XP vs. Industry Average

| | Industry Average | Current Performance | Delta |
|---|---|---|---|
| **Project Cost** | $3.5 million | $2.2 million | -$1.3 million |
| **Schedule** | 12.6 months | 7.8 months | -4.8 months |
| **Defects During QA** | 242 | 121 | -50% |
| **Staffing** | 35 | 35 | N/A |

* Using average project size of 500,000 lines of new and modified code

Table 2 — BMC Software Distributed Scrum vs. Industry Average

| | Industry Average | Current Performance | Delta |
|---|---|---|---|
| **Project Cost** | $5.5 million | $5.2 million | -$.3 million |
| **Schedule** | 15 months | 6.3 months | -8.7 months |
| **Defects During QA** | 713 | 635 | -11% |
| **Staffing** | 40 | 92 | +52 |

* Using average project size of 700,000 lines of new and modified code

Recall the fact that, typically, when waterfall projects attempt to compress time by doubling staff, industry data shows defects easily at 4x the average and sometimes 6x as team communication complexity rises. What is remarkable about BMC's Scrum approach is that it ramped up with a large team, successfully accelerated time to market but *suffered no defect and quality impact*. In fact, defects in aggregate were 11% fewer compared to the 1x industry average. Finally, despite running with a staff of 92 people versus 40, its cost on a project basis was $300,000 less because of crossing the finish line in such a short schedule (upon which the team moves to build the next release).

In summary, we have two companies employing two vastly different styles of implementing agile software development and both succeeding in highly dramatic ways. Follett used normal-sized teams and built applications fast, with half the defects, simultaneously saving millions of dollars. BMC emphasized time to market, and while still beating industry averages when it came to cost, achieved a nine-month time-to-market advantage while keeping defects low. One might infer that this schedule advantage was designed to maximize its top-line income by opening a revenue pipeline for its product almost nine months faster.

## How They Got There

Of the five companies shown on the benchmark charts, both Follett and BMC stand out in terms of scaling agile. This is exemplified by noting that the data points for both Follett and BMC are on the right side of the trend figures shown earlier. They are each building large-scale software applications in excess of a half million to a million lines of code. BMC is also managing one of the largest Scrum teams we've observed.

I asked Follett what roadmap it employed to construct its XP room and how it adapted XP to its team, and one answer given was, "We read Beck's books and implemented nearly all of the practices." This was an impressive accomplishment, but it also added that it had taken a couple of years to achieve these levels. It confided that early experiences were not at this caliber of performance.

In the case of BMC, it gave extensive commentary of having used coaching to implement its Scrum process. It should also be noted that maintaining high degrees of visibility across a geographically disbursed team isn't readily possible using 4x6 story cards or sticky notes on a wall. To scale agility at this level, while overcoming 12-hour time zone differences across four major cities, required a technology infrastructure that included Rally Software tools and audio-video communications technologies. That said, Mike Lunt from BMC Software emphasized that not all work was done remotely.

Critical colocated team meetings, especially "Iteration 0" release planning was done face-to-face.

In order to capture the "how-to's," we had the pleasure of conducting a special Cutter Webinar entitled "An Agile Metrics Chat" featuring a Charlie Rose–style town meeting conversation with Kim Wheeler of Follett Software and Mike Lunt of BMC. Some of the questions and answers from that Webinar have been excerpted and included in this report (see Appendix).

With regard to BMC, I once asked Israel Gat, Senior VP of BMC and a key executive sponsor of its Scrum initiative, to what he attributed much of BMC's success. Mr. Gat replied, "Well you see, we have this thing called a 'secret sauce.'" I asked about the ingredients in this secret sauce recipe. Thus, we're privileged to present the following section authored by Lunt titled "The Secret Sauce."

## "THE SECRET SAUCE"

At this point, you might be wondering what's behind the numbers and whether these results be replicated at other software shops. The good news is that any team with the right motivation and training can implement agile and scale it to the levels listed above. The not-so-good news is there is not a one-size-fits-all approach. While a tailored process will be required to meet each organization's unique personality, some hurdles remain the same in moving to agile within any large software business.

For the distributed systems management (DSM) business unit within BMC, we set out on a course to revamp the method for creating and delivering software to customers. Our customer base demanded a more dynamic approach than the previous 12-to-18-month releases; consequently, we needed to change the development delivery mechanism so that we could gather customer feedback and return results in a much shorter time. The team chose to use the Scrum recipe for agile, and three years later, the DSM team is one of the most productive development teams in the world in terms of size and time to market. The following sections detail what makes BMC's version of agile unique in its approach to software development and lists the special ingredients for its "secret sauce" for scaling agile to large global teams.

The five key components to BMC's version of agile are:

1. Buy-in
2. Ready to ship
3. Dusk-to-dawn teamwork
4. The backlog(s)
5. Holding back the waterfall

### Buy-in

The decision to convert to a new development methodology is one of the most disruptive events that a well-established software organization can endure. To make this happen, enough momentum must be built up to prevent a complete stall. Within our business unit, we have a VP-level proponent who is willing to foster the methodology.

With this top-down sponsorship, ScrumMaster training and consulting support were used to form a core group that were energized and passionate about making the process work. The term "passion" has often been used to coin why agile has worked at BMC, and this is a subtle but essential characteristic needed for any large team to flourish with agile.

While the team now operates in a state of continuous improvement, the sequence of events up to this point has not always been a pleasant stroll across a grassy meadow. In the beginning, some accepted and ran with the new methodology; others did not. These were the "agile casualties," as some within the company have referred. While the business side of the equation was screaming for a much-needed improvement, some in the engineering group saw no reason to change the process that had been used for years. These team members transferred to other positions or chose to pursue other paths, but over time, more people began to see the agile process help both internally within the R&D teams and externally with customers. As we have matured, less focus has been placed on converting to agile and more focus has been placed on enhancing and extending the methodology to deliver more internal efficiencies.

When asked to help other teams that are struggling with some part of their agile conversion, we often see a lack of buy-in from one or more parts of the organization. These teams lack executive support and/or support from all of the functional groups.

While some may doubt the agile advantage altogether, the reasons for this reluctance could range from disbelief in short iterations to having doubts about the lack of process documentation. Even though many of these issues are rooted in previous organization structures (see Conway's Law[7]), BMC has been able to garner enough buy-in to overcome these common internal struggles.

This ability to foster a new methodology across archaic group formations is what has allowed BMC to scale its agile implementation within one business unit, and now, this success is spreading to other engineering groups within the company.

### Ready to Ship

Staying releasable means different things to a lot of people, but as the number of agile teams increases, the criticality of staying releasable increases exponentially. While being able to literally ship new product to a customer at the end of each iteration would be the nirvana of agile development, this approach is not practical for all products. For instance, our products have a significant reliance on integration points from other teams, which may not be following an agile approach or even a similar release schedule, and in some cases, we require mean time between failure (MTBF) testing of 30 days for a generally available product.

As with many agile approaches, BMC's efforts involve a nightly build, a nightly test integration suite that runs on top of the nightly build, and an iteration demo that occurs every two weeks. While these are common to most agile software shops, BMC places a significant sense of urgency on keeping the software close to shippable at all times.

**Starting the project off with all of the stakeholders in one room results in many positive outcomes.**

Extra effort is given to make each iteration "releasable," generally meaning that all new functionality works as expected and that the product is ready to enter a final hardening period for customer availability. A more formal definition for our releasability state is as follows:

- The software builds can be installed in its current state (an hourly/daily requirement).
- New user stories cannot be accepted with any severity 1, 2, or 3 defects (an iteration requirement).
- No severity 1 or 2 defects exist. If regression defects are found, they must be prioritized over new user stories in iteration planning (an iteration requirement).
- Shipping to a customer is 1 (2 max) hardening iterations away (an iteration requirement).

Another way that we stay releasable, which is a lesson borrowed from the XP recipe book, is that all code check-ins require a peer code review, typically done by a subject matter expert or more senior developer than the person writing the original code. For each code check-in, the reviewer is noted and a checklist of reminders is added to the notes associated with the changed code. What appears to be a redundant paper trail ultimately provides a subtle reminder when time is short.

Much of our process related to staying releasable might sound like Agile 101; however, when five to 10 teams are involved, the necessity cannot be overstated. Broken functionality may cause an entire team to be delayed; consequently, there is a high sense of urgency related to solving any issue that has forced the product out of a releasable state. In many ways, the "stay releasable at all times" mindset is what has allowed BMC's multi-team approach to avoid the large integration traps seen in other large teams.

### Dusk-to-Dawn Teamwork

As Michael will attest, the metrics for teams that are globally distributed tend to trend toward lower productivity and potentially higher defects as the lines of communication increase, while the efficiency of these lines decreases due to time and distance. At BMC, we do have overlapping times with our remote locations, and we use every form of communication possible to promote information flow, such as instant messaging, Wikis, videoconferencing, and periodic travel to build relationships. Beyond the common communication techniques implemented by many global organizations, BMC takes intensive measures related to organizing all of the agile teams to ensure roadblocks and misdirection do not occur. As Mountain Goat Software founder Mike Cohn mentions in his book *Agile Estimating and Planning*, "but this is so much work …",[8] we have discovered that this exhaustive level of communication is necessary when dealing with a team structure that never sleeps (literally, in some cases).

To kick things off, we conduct a large colocated-release planning meeting for each major release, and sometimes we will have portions of the team travel to common locations to conduct iteration planning. Starting the project off with all of the stakeholders in one room results in many positive outcomes.

First, each person is educated on the themes and the top user stories in the backlog. In other words, everyone gets an idea of what's being considered and what could be possible for the next release, even though there is a clear understanding of the potential change in priority that often occurs during the release. Second, the teams benefit from face-to-face interaction and a shared sense of purpose, and excitement is created. Once the team has "left the huddle," the harder work of keeping the groups communicating and in harmony begins.

Two of our most popular methods of communication are well known in the Scrum community, even though they serve additional purposes with our multicross-continent time zone setup.

First and foremost, daily Scrum of Scrum (SOS) meetings occur with leads/leaders from every team, and these meetings occur early in morning (US time), so they are friendly for participants in India, Israel, and so on. During these meetings, all of the teams get a chance to hear what is happening across the teams and determine whether roadblocks exist within one team that may cause a problem for other teams. An important aspect to keeping these meetings short and productive is focusing on what is occurring that could affect other teams.

A second important method of communication is our iteration demos, which occur every two weeks during nonhardening iterations. Again, these are done at globally friendly times, and all of the teams get to see what the others have done. This provides a sanity check for consistency and allows each team to accept critiques on the work from the past two weeks. As the transparency increases, teams become more comfortable with being open and potentially "exposed," which leads to discussions around the validity of the use cases as well as the proposed solutions. Since these meetings are recorded, remote support teams and others can always use these meetings as ongoing training for the new release well in advance of it being made available to customers.

While release planning, SOS meetings, and iteration demos are common among large agile teams, BMC differentiates its implementation by rallying the agile teams around these well-known means of communication and maximizing the time spent to solve additional problems. In all cases, the main goal is to make small-course corrections instead of having big rework items, especially with integration of the various parts. As rigorous as these approaches may appear on the outside, the process is never placed on cruise control because the teams can quickly get off course.

## The Backlogs

Single agile teams have the luxury of a single backlog with prioritization occurring in one concentrated location, but when multiple teams are involved, the ability to maintain and distribute the most recent backlog items becomes a serious topic.

For our teams, the decision to have one all encompassing backlog versus multiple backlogs has remained a continual debate in some minds, but our current backlog model takes advantage of both approaches. Our setup for handling user stories across multiple teams allows for our teams to be geographically separated and takes into account the varying skill sets that might exist on these teams. Having this system of priorities is a unique way BMC keeps its teams all heading in the same direction while giving each team autonomy to behave as a typical agile team.

**We've found that once the teams have momentum with agile methods, they quickly become able and ready to take on the next user story, and so on.**

To accomplish this, we have created an overarching strategic backlog that is used as guidance and a way for the agile teams to peek inside the heads of those setting the direction at the strategic level. The strategic backlog items cover multiple product lines and form a set of guidelines for the individual product lines to base their backlogs. The product owners for each product line then create a separate backlog focusing on customer needs and working toward delivering various aspects of a larger suite of products.

These backlogs are reviewed on a weekly basis by an internal "steering committee." The steering committee contains stakeholders from various parts of the organization as well as from other teams where integration occurs. Reconciliation of priorities between strategic objectives and individual backlogs are discussed, and changes to the relative rankings are often made in one or both locations.

With such a large suite of products and high number of teams, one area where agile teams can quickly lose balance is with the bandwidth of the product owner role, which is filled by a group of product managers in our case. With so many teams and a regular stream of user stories arriving from our 4,000-plus customer base, there were not enough product managers to extrapolate customer ideas while simultaneously injecting this into the development engine.

To address this issue, we invented the role of "requirements architect" to bridge the ever-changing gap between product management and R&D. The requirements architects are responsible for breaking down the user stories into more granular software requirements as the code is developed. As author Dean Leffingwell points out in his book *Scaling Software Agility*,[9] this role ultimately helps improve collaboration with the product management team and allows agile teams to work in a just-in-time mode without being hindered by lack of immediate access to a product owner.

**Comparing disparate practices in software development such as waterfall versus agile is both easy and at other times daunting.**

In each of these methods, our ability to scale the backlog prioritization and communication is a key element. We've found that once teams have momentum with agile methods, they quickly become able and ready to take on the next user story, and so on. Without proper alignment and ready-to-go backlogs, teams can quickly wander into uncharted territory and lose one of the key agile tenets, which is completing the highest-ranked customer items first.

### Holding Back the Waterfall

Constantly trying to improve the agile process at every iteration and release boundary is the best way we have determined to keep the process healthy and to look for the next big advancement. We realize we need more test-driven development (TDD), more work in maintaining our backlog, and so on, and in each new release, we pick certain items to improve upon and focus on these items.

Often, a key result of these retrospective meetings is determining that we have undone some agile process we used several months ago. This might be as simple as making sure the number of "overhead" projects is not increasing to a point where prioritization is being skirted. Just as BMC has obtained services from Cutter and QSM Associates (via Michael Mah), using an outside source to assess the process can help overcome internal struggles and point out new areas of improvement. The important part is to consistently set aside time to reflect on what is (or is not) working within individual teams and across the entire team structure.

Just as prototypes in software are often discarded in favor of a redesigned substitute, software process ideas can be as easily discarded and replaced with improved methods. In some cases, a transitional process is implemented to overcome the resistance to change as the group or parts of the group gain a better understanding of the benefits. In other cases, there is almost unanimous agreement to change some part of an existing practice, and the change takes effect immediately.

Ultimately, the process is never assumed to be good enough. As BMC's customer needs have evolved, our ability to remain flexible in developing and delivering software has created a competitive advantage. In conclusion, as the pendulum swings further away from the waterfall approach, we plan to look for ways to decrease the time between customer demand and software delivery to weeks or even days.

*This concludes Lunt's section of this report.*

## OBSERVATIONS ON METRICS PRACTICES IN AGILE ENVIRONMENTS

Comparing such disparate practices in software development as waterfall versus agile is both easy and at other times daunting. The easy part — from Cutter's and QSM's perspectives — is knowing from history what has come before in software development.

Another easy part comes from measurement practices in many agile organizations. As stated in an earlier section, many in fact keep good metrics records, contrary to popular belief. How did this manifest, especially since many agile proponents originally equated software measurement with "heavy" methodologies? The answer is simple: agile projects — not just waterfall ones — require project estimates, release planning, and iteration planning. To accomplish these tasks, one of the practices that agile teams employ is estimating project size. To identify team velocity, they express work accomplished per iteration (usually two to four weeks) for a team that is staffed with a given number of people. This captures time and effort.

### Good Agile Metrics Practices

As described in his book, Cohn explains that agile teams separate estimates of size from estimates of duration.[10] For example, if one must move a large pile of dirt from one side of a yard to another, one can look at the dirt pile, assess the tools such as shovels and a wheelbarrow, and directly estimate the job at five hours, bypassing a size estimate of the dirt pile. However, if one were to estimate the dirt pile to be 300 cubic feet, and ascertain that a wheelbarrow could carry 6 cubic feet at a time, then 50 trips would be required to move the whole pile. The end result is a more accurate and reliable time estimate for a dirt-moving project. Agile teams, in a similar fashion, estimate the size of their software projects, and then determine how many iterations may be needed over a span of time for a software release. Although designing and building software is far more complex than moving dirt, many project teams, including those from our case study companies (including Follett and BMC), estimate size using metrics such as story counts ("features") and story points (complexity of the feature on a numeric scale). As a result, they translate these size estimates into more accurate schedules.

By translating desired features expressed by a client or user into stories (typically, a description that can fit onto a 4x6 index card or equivalent) and rating the complexity of this story or feature, say on a scale of 1-10, agile teams capture size metrics that even some waterfall teams do not. They do this in order to assess how much functionality can be accomplished inside a given deadline or time frame. That is the software estimation quandary that all teams face: how much can we do in this amount of time?

At the end of an iteration, or a series of say, 10 iterations in a 20-week release, agile teams have a size inventory of the stories and story points that they have accomplished by creating software. This includes writing new "added" code to an existing base and often modifying previously written code to produce the functionality required by a user story. In the Follett and BMC examples, the Follett team (26 people) completed 44 stories and 252 story points through 11 iterations in a 6.5-month schedule, while the BMC team (93 people) completed 918 stories across 14 iterations in a 5.25-month schedule. The Follett team produced software totaling more than 218,000 lines of new and modified Java and XML, while the BMC team produced about 857,000 lines of new and modified Java and XML.

Last, by tracking defects, both teams historically capture four core metrics that include time (months), effort (person-months), size (stories, points, and code), and defects (bugs).

### Agile Metrics Challenges

What is daunting about how agile teams capture metrics is that no real standard exists to define a "story." What comprises a feature or story in one company, such as a bank or insurance company, can be vastly different from a story at an Internet startup. Even weighting a story by applying a story point scale is vastly different. At a recent gathering of more than 100 developers at an agile conference, I asked the audience what scale it uses for determining story complexity. One person described a scale of 1-3. At the other end of the spectrum, several said they used a scale of 1-100. One said they used a scale of 1-200. It was all over the map.

While Cohn says his personal preference is to use a scale of 1-10, part of the problem is that agile experts historically advised that conforming to one "standard scale" is unnecessary; all that matters is to weight one story to another by relative weights (i.e., a story assigned six story points is twice as difficult as one rated a three). As a result, it's rare that any two organizations size projects in the same way. To some, it might not matter, since an objective is to quantify accomplishments and forecast future iterations *within a specific project or release*.

**Proponents of agile methods claim a better way of creating software than their predecessors.**

What becomes difficult is comparing two different projects in terms of work accomplished or comparing two different organizations. Methods of defining stories or story points may be disparate. Companies that don't comprehend or understand how much software they produce are at a disadvantage compared to those who do. The good news is that it's not that hard. One team I worked with at a client organization initially had no idea how much code was involved in its release. Another manager overheard this, provided the team with a link to an open source code-counting utility that she used, and the information was derived that same afternoon.

A strong benefit of this is an understanding of the amount of software required for stories and story points. Understanding these proportions can even allow a team to predict the code potentially required for a future iteration or release, comprising "X" number of stories. That enables the team not only to understand its past, but to use this valuable information to better anticipate its future.

## CLOSING THOUGHTS: WHY AGILE WORKS

We began this report by observing that software is all around us in this technological age, inside all of our marvelous machines and devices, including laptops, cell phones, iPods, houses, cars, airplanes, and information networks. Software plays the music, makes the phone call, moves currency through the bank and credit networks, places the online order through a Web site, and gets us through the checkout line at the grocery store.

Proponents of agile methods claim a better way of creating software than their predecessors. Some industry observers feel that it is *revolutionary*, while others claim it is more *evolutionary*. The truth probably involves an intersection of the two. But in the final analysis, there is the idea that agile methods should produce more software, with fewer defects. Is it true? The answer appears to be yes, for those practitioners who have evolved their craft thoughtfully and deliberately as exemplified by two of the organizations profiled in detail in this report.

Interestingly, each chose to implement agile in significantly different ways: one a colocated XP approach, the other a distributed Scrum method. These two different flavors of agile both exhibited success in cost, schedule, and quality to varying degrees. The outcome clearly seems oriented around how each dealt with schedule compression and staff buildup, which influenced their respective outcomes in the three dimensions of time, cost, and quality.

Both delivered more functionality with fewer defects. The other three organizations — while good performers in their own right — achieved fast schedules and productivity but not at the defect and quality performance levels of the first two. Why? This seems clear: having a head start helps. The two leaders had more mature implementations of agile and are experiencing far greater success than the more "neophyte" organizations. This leads us to an interesting observation: a good way to lead the race is to start sooner.

So why do agile methods work? The reasons are many and varied. In addition to what both Follett and BMC expressed in their words, I offer these thoughts and observations, culled from interacting with many remarkable teams and agile experts:

- **Domain knowledge.** Rapidly moving from stories to code requires smart people who innately understand the problem being solved by creating working code, quickly. Having this manifest from mind to machine, efficiently, demands software designers who truly understand the business and can virtually "mind-meld" with marketing teams, business analysts, and other subject matter experts without the burden of a long learning curve.
- **Short feedback loops.** Optimizing the transmission of knowledge between team members occurs when you have short feedback loops. This is directly achieved by paired programmers. Code reviews are instantaneous, and learning and execution are accelerated from the one-on-one direct interaction. The shared reality of looking at the same thing together, face-to-face, complete with body language, voice tone, and interpersonal nuance maximizes the bandwidth of the communication channel. This also is the concept when users or customer proxies work together with a paired team to accelerate the transmission of story concepts to the designers/builders who are tasked with creating the code.
- **Time-boxing/first things first.** Short iterations force teams to focus on the most important features first. By making clear and discernible progress every two to four weeks, teams then anticipate what the next most important features are in the queue, and this concentrates their thinking and their action, creating efficiency.
- **Avoiding burnout.** XP practices, like 40-hour work weeks, enable teams to work and maintain a sustainable pace. Short-term bursts of overtime are fine on occasion, but data shows that productivity collapses on successive iterations if teams overwork for long periods.
- **Craftmanship over craft.** As Agile Manifesto coauthor Robert C. Martin (affectionately known as "Uncle Bob") describes, creating quality at the start with clean code means taking pride in what you do, at every moment, without compromising one's professionalism. Agile practices encourage this early on with simple design, TDD, pair programming, and refactoring. Mature agile teams don't wait to test in quality in at the end. It starts up front, not building junk for a QA team to clean up later. Getting it right first prevents costly rework and reduces defects.
- **Transparency.** The best teams don't hide anything. This is especially exemplified in a large XP room where everyone lives and works side by side in a broader community. Scrum teams accomplish this with daily stand-ups, demos, and Scrum sessions. Nothing goes underground. Velocity and burn-down charts keep everything out in the open.
- **High-bandwidth communication.** Techies are often stereotyped as introverts. There are times when nothing can be further from the truth. The best teams have wide-open pipes where information is flowing rapidly and effectively. Domain knowledge is important but moving that information around is also crucial. Knowledge doesn't stay locked in one place.

- **Avoiding waste and costly rework.** All of these practices cumulatively encourage expending work effort without waste. Projects that have slipped their schedules the most and exhibited the highest defects exhibit endless do-overs where work was thrown out repeatedly. Let's not use the term "refactoring" to cover up wastefulness either. It's one thing to refactor inefficient and dead code from long ago but another thing to rework code over and over because it was done incorrectly in the first place.

We close this report with this idea: understanding productivity and agile development takes on new meaning when we look at real quantitative data through the lens of the observer. Moreover, when we realize that metrics can be made simple and easy, it clarifies our ability to appreciate that — understanding what's happening inside this movement is readily available to anyone, with real numbers. This is a solved problem, and it can be taught to anyone.

Integrating these measures, along with the stories from real-world managers and leaders from the case studies we shared, hopefully has communicated an understanding of this agile world in a new way that has not been presented until now. It's our desire that this has made a difference to you, our readership, so that you might emulate and replicate what we've illustrated here. I would welcome your comments and ideas; you can e-mail me at mmah@cutter.com.

### QSM SLIM SUITE

Technology used for this benchmark analysis was the QSM SLIM Suite (www.qsma.com) comprising:

- **SLIM-Metrics.** This is a database containing industry trend lines and benchmarks derived from more than 7,500 completed software projects collected from more than 500 organizations across 18 countries.
- **SLIM-Estimate.** This reliably estimates the time, effort, cost, and quality for software projects and releases under tight deadlines whether they are waterfall, agile, package implementation, inhouse, or outsourced. It uses historical data to calibrate its forecasts, resulting in reliable estimation accuracy. It does this in a fraction of the time compared to traditional methods and produces estimates, probability profiles, and work plans that can be exported to many popular project management tools.
- **SLIM-Control.** This uses curve-fitting techniques to assess the "in-flight" productivity of a software project and generates an accurate predictive schedule forecast to completion — as high as 90%. It assesses the impacts of requirements changes and dynamic project changes to predict where a project is headed based on its current progress.

## ENDNOTES

[1]"Manifesto for Agile Software Development," 2001 (http://agilemanifesto.org).

[2]Highsmith, Jim. *Agile Software Development Ecosystems.* Addison-Wesley Professional, 2002.

[3]Beck, Kent. "XP and Culture Change." *Cutter IT Journal*, Vol. 15, No. 9, September 2002.

[4]Quantitative Software Management, Inc.; and E.M. Bennatan, Tom DeMarco, Jim Highsmith, Tim Lister, Jim Love, and Michael C. Mah. *Cutter Benchmark Almanac: Application Development Series, 2007-2008 Edition.* Cutter Consortium, 2007.

[5]Tufte, Edward R. *Visual Explanations: Images and Quantities, Evidence and Narrative.* Graphics Press, 1997.

[6]Beck. See 3.

[7]According to Wikipedia, Conway's Law is "an adage named after computer programmer Melvin Conway, who introduced the idea in 1968. It concerns the structure of organizations and the corresponding structure of systems (particularly computer software) designed by those organizations. In various versions, Conway's Law states: Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations. If you have four groups working on a compiler, you'll get a 4-pass compiler. Or more concisely: Any piece of software reflects the organizational structure that produced it." (See http://en.wikipedia.org/wiki/Conway's_Law.)

[8]Cohn, Mike. *Agile Estimating and Planning*. Prentice Hall PTR, 2005.

[9]Leffingwell, Dean. *Scaling Software Agility: Best Practices for Large Enterprises.* Addison-Wesley Professional, 2007.

[10]Cohn. See 8.

## RECOMMENDED READING

DeMarco, Tom, and the Cutter Business Technology Council. "Reassessing XP." Cutter Consortium Business Technology Trends & Impacts *Executive Report*, Vol. 6, No. 8, August 2005.

Mah, Michael. "The Making of the Agile IT Executive." Cutter Consortium Business-IT Strategies *Executive Report,* Vol. 6, No. 10, October 2003.

Putnam, Lawrence H., and Ware Myers. *Five Core Metrics: The Intelligence Behind Successful Software Management.* Dorset House Publishing Company, Inc., 2003.

## APPENDIX

The following is an excerpt from a Cutter Webinar titled "An Agile Metrics Chat" featuring a Q&A with Kim Wheeler of Follett Software and Mike Lunt of BMC.

**Q:** *Right off the bat, some people might say, "OK, we're talking about two companies, XP, and Scrum. Follett chose XP colocated teams in a large room. Why did you choose XP?"*

**A:** (Wheeler) Let me start with some history. A long time ago, our company went through some rough times. Schedule delays were the norm. Shipping rates and software were astronomical — off the charts — and we were trying to test in quality instead of build it in. We were wasting tens of thousands of dollars burning CDs that would just have to be thrown away because we'd find a critical showstopper. I think you get the point, it was just ugly.

Our leader, George Gatsis, is a huge supporter of process improvement, and although we were making some small improvements, we needed something that was going to take us to the quantum leap if we were going to remain successful as a company and as a development organization. So George attended a presentation on agile development and XP at a Chicago [Software Process Improvement Network] SPIN meeting. He came back, brought us the presentation, and said, "I want you to consider this." I don't know about you, but when my boss says you should consider this, it means you very well better consider it.

We went up to the bookstore, and we bought Beck's books. As a leadership team, we read them and there was some initial resistance — especially against the paired programming from our lead developers — but we decided as a team that we were going to go ahead with the process by jumping in with both feet. We implemented the process by the book. Desperate times called for desperate measures. And six years later, here we are, executing some of the best projects in our history. Then we found out that we are executing some of the best projects in the industry.

**Q:** *Awhile back, you called this room* [refer to Figure 1 on page 4] *the "Destiny room," where you have the colocated programmers. Tell us a little bit about this room where you used the XP process.*

**A:** (Wheeler) When we first started XP, we had 13 people sitting in a really small, old room with folding tables that sagged in the middle because we had 50-pound monitors sitting in the middle of them. But the Destiny room is actually a bigger room that has 34 people in it and a smaller room that has 20.

Every person has exactly a 4x3-foot table on wheels. That's all the personal space you get, but we are organized into three subteams, so whenever subteams are going to switch, which is generally every two to three weeks, you just unlock your wheels and roll your station, plug in, and you're ready to go. All of our production code is paired. Our confidence and our products come from the fact that we have 42,000 unit tests, and we have 21,000 acceptance tests that run on a nightly basis to prove we didn't break anything from the day before.

One of the things that is really unique from a management prospective is that each team is policing. If someone doesn't carry their weight, everybody knows, and it's on the table for discussion at postmortem. We subscribe to a 40-hour work week, so it's pretty intense; you are on for eight hours. There's no opportunity to go on and launch your browser and search for Christmas presents for your kids.

It's lively, it's interactive, and we have integrated management. I think the bottom line is, it's just a fun place to be.

**Q:** *What are some of the benefits from that energy and having everybody working together in such close proximity, and what are some of the downsides?*

**A:** (Wheeler) The downsides are pretty obvious. We sometimes like to call it the Destiny Petrie dish, because when you have 34 people working in such close proximity and somebody gets a bug, it's shared by all of us (laughter). We're taking some measures: everyone is getting hand sanitizer on their desk, and we have multiple USB ports. People bring their own keyboards to the pairing station as opposed to sharing a keyboard.

Another negative is there isn't any privacy. Also people don't have personal office phones so if you need to make a personal phone call, you need to go somewhere else. It can get noisy, but we've worked really hard to control the elevated noise, the exuberance, the excitement.

As far as the plus-side goes, I can tell you that when we first started, people were a little uncomfortable with the loss of privacy, but I can tell you with certainty that there is not one person who works on that team of 54 people that would go back to the old environment.

**Q:** *You were under pressure to outsource and send work to India. What happened after the executive team — CEO, CFO, CIO, etc. — saw how XP was doing in your world?*

**A:** (Wheeler) You know, I call that day one of the best days of my life.

We did have pressure to outsource something, and I really believe that if we began to do that, we were going to lose confidence from our team and confidence in management. At the time that this was going on, the market for job developers was wide open, and we have really talented people who could have gone anywhere. So we got those results, and George asked you [Michael Mah] to present those to our executive committee and the noise to outsource literally evaporated.

Like I said, I call it one if the best days of my life.

**Q:** *How did you overcome resistance to agile development and paired programming from your lead developers?*

**A:** (Wheeler) It was interesting because the paired programming specifically was the challenge. Like many, our developers have egos and think they do it just as well as anybody. It was from the area that we came from, we thought, we have got to do something drastic. It was a team decision to commit to it, and it was a decision based on drastic measures. It was a bottom-up decision. Once we jumped in, it didn't take long for things like paired programming to prove themselves very valuable, and the team was bought in.

**Q:** *Tell us about your attrition when switching to XP. Did you lose people?*

**A:** (Wheeler) We did not lose people initially. Again, we started small; we only had 13 people, and we were dedicated to the cause. Over the last six years, we have had exactly three people leave on their own volition and two of those were moved. Our attrition rate is incredibly low.

**Q:** *You mentioned running automated tests every night. Are you using continuous integration during the day? How many of the tests do you run in CI* [continuous integration]*?*

**A:** (Wheeler) Yes, we are using continuous integration, and we do have a set of tests that we call the "smoke unit," because clearly you can't run all those units at one time. I would say it represents about 15%, and that smoke unit has to pass before you can promote your code.

**Q:** *For the measured projects, what other factors besides agile processes led to productivity gains on these projects? Were these teams made of experienced members in the organization? Was the team and/or organization very familiar with the type of project being developed?*

**A:** (Lunt) For us, it was a mixed bag. The offshore teams were much less experienced. Some were only two to three years out of college. The core team on average had seven to 10 years of experience. In general, we had a more experienced team.

**A:** (Wheeler) For Follett, our team definitely had experience with the organization. All of us had lived through the ugliness described before. As far as the product, we started XP with a rebuild of our core product line on a new technology front. So we were very experienced with the product concepts and what the product needed to do to please the customers who were unfamiliar with the technology that we were building on.

**Q:** *What kind of technical metrics do you gather on a regular basis (e.g., lines of code, unit tests measuring)?*

**A:** (Lunt) For us, because we are using Rally, the number of stories is easily measured. We also measure defects on a daily basis. As far as lines of code change, that's something that's easy to capture in reverse. I would say those three things are the key metrics.

**A:** (Wheeler) For Follett, on an iteration basis, we're measuring things like story points and task points and also code coverage. We have a code-coverage tool. Things like lines of code and staffing are measured on a project-by-project basis.

**Q:** *How do you define a story point?*

**A:** (Lunt) For us, we don't use a purest version of story point. Cohn talked a little bit about that in his book *Agile Estimating and Planning*. We typically use an ideal developer day [IDD], which on average in each Scrum team determines what the number of ideal developer days that each developer has based on their past experience and history. It's somewhat intuitive. You can look back and see if this person estimated that for a two-week iteration that one developer had seven IDDs and another had four, we would start to average things out. On average, our teams would have four to six development days in a two-week iteration.

**A:** (Wheeler) For Follett, we do have story points. This is how we define them. They are really a level of difficulty for us. We have a scale from 1-7, and if something is more difficult than a 7, then it gets a 7+ and that just means it needs more definition and needs to be broken down into multiple stories. So when we are planning a project and we have ballparked all the stories, we can look at the buckets — the ones, twos, threes, and so on. The velocity accounts for itself, so when we first started, we picked the velocity. We said, "We think this thing can do 15 story points," and then we used the concept of yesterday's weather; if in this iteration the team did 18 story points, then we plan 18 story points for the next iteration. If something happens in that iteration and we only hit 16, we'll take the number 16 going forward.

**Q:** *Are tech writers colocated with developers, and do they deliver documentation at the end of each iteration?*

**A:** (Wheeler) Great question. Yes, our tech writers are colocated with the development team, and we essentially have three forms of documentation that we produce for a given release. We have what I call "on-screen help," which is context-sensitive help. And then we use a tool from Adobe called RoboHelp. It's a search engine type of help. And then we actually produce some paper documentation, which goes out in the form of PDF called "quick help." The piece of documentation that is delivered on the iteration basis is the on-screen help, which helps specifically for those features. The RoboHelp and the paper documentation (quick help) essentially follow.

**Q:** *What were some of the lessons learned from your project?*

**A:** (Lunt) To switch models from a waterfall to agile approach is difficult. From what I have seen, you need to have executive-level sponsorship at some point, as well as needing to build a core group in the lower layer of the organization that is building in some type of passion momentum that helps switch over. I think it's a top-down and bottom-up approach that has to occur at the same time in order to switch over.

**A:** (Wheeler) I would certainly agree with that. One of the principles of XP says if you're doing XP today the way you were doing XP yesterday, you're not really doing XP. And I can tell you at Follett, we have not arrived yet. We do lessons learned at every postmortem every two weeks, and those get turned into action items to continue to help us to improve.

## ABOUT THE AUTHOR

Michael Mah is Director of Cutter Consortium's Measurement & Benchmarking practice and Senior Consultant with Cutter's Business Technology Trends & Impacts, Agile Product and Project Management, and Sourcing & Vendor Relationships practices. He is also Managing Partner of QSM Associates, Inc., a firm specializing in software measurement, project estimation, and "in-flight" control for both inhouse and outsourced/offshore development. QSM has developed and maintains one of the largest databases of more than 7,500 completed projects collected worldwide, with productivity statistics and trends on cost, schedule, and quality from more than 500 organizations and 18 countries.

With more than 20 years' experience, Mr. Mah has written extensively and consulted to the world's leading software organizations while collecting data on thousands of projects. His background is in physics, electrical engineering, conflict resolution, and mediation. Mr. Mah's work examines the dynamics of teams under time pressure and its role in contributing to success and failure. He can be reached at mmah@cutter.com.

## ABOUT THE CONTRIBUTING AUTHOR

Mike Lunt is a seasoned director of software engineering groups, with more than 14 years of software lifecycle experience in both online and product-based software. Mr. Lunt has implemented rapid development and agile techniques within enterprise organizations, such as BMC Software, as well as for several startups. Visit his blog at www.mikelunt.com/blog.

# Index

CUTTER CONSORTIUM

# Agile Product & Project Management Advisory Service Published Issues

## Upcoming Topics

- **Succeeding in the Contemporary World of Project Management**
  **BY BOB WYSOCKI**
- **Scrum Today**
  **BY BRIAN J. DOOLEY**

This index includes Agile Product & Project Management *Executive Reports* and *Executive Updates* that have been recently published, plus upcoming *Executive Report* topics. Reports that have already been published are available electronically in the Agile Product & Project Management Resource Center. The Resource Center includes the entire Agile Product & Project Management advisory service archives plus additional articles, Webinars, and podcasts delivered by Cutter Consortium Senior Consultants on the topic of agile.

**For information** on beginning a subscription or upgrading your current subscription to include access to Cutter's Resource Centers, contact your account representative directly or call +1 781 648 8700 or send e-mail to sales@cutter.com.

## Executive Reports

| | |
|---|---|
| Vol. 9, No. 9 | How Agile Projects Measure Up and What It Means to You BY MICHAEL MAH |
| Vol. 9, No. 8 | Agile Software Package Implementations BY SAM BAYER |
| Vol. 9, No. 7 | Refactoring in the Context of Enterprise Architecture BY SEBASTIAN KONKOL |
| Vol. 9, No. 6 | The Four Pillars of Agile Adoption BY NANCY VAN SCHOOENDERWOERT |
| Vol. 9, No. 5 | Social Project Management BY DAVID COLEMAN |
| Vol. 9, No. 4 | Agile Adoption for Organizational Change: Improving Time to Market BY AMR ELSSAMADISY |
| Vol. 9, No. 3 | Moving the Herd: Facilitating Multiparty Project Teams Toward Common Goals BY MOSHE COHEN |
| Vol. 9, No. 2 | Is Design Still Dead? BY KEN ORR |
| Vol. 9, No. 1 | Transitioning to Agile Project Management: A Roadmap for the Perplexed BY SANJIV AUGUSTINE AND ARLEN BANKSTON |
| Vol. 8, No. 12 | Negotiating Resources, Deliverables, and Deadlines Within the Global Organization BY MOSHE COHEN |
| Vol. 8, No. 11 | Emulating the "Bazaar": Open Source–Style Development Within the Firm BY JOSEPH FELLER |
| Vol. 8, No. 10 | From the Ashes: Resetting Expectations and Restoring Relationships After Project Disasters BY MOSHE COHEN |
| Vol. 8, No. 9 | Requirements for Managing Requirements BY SUZANNE ROBERTSON |
| Vol. 8, No. 8 | From Startup to Enterprise: Creating a Quality-Friendly Development Environment for All Methodologies BY MEGAN FOLSOM |
| Vol. 8, No. 7 | Project Management Cultures: The Hidden Challenge BY ROB THOMSETT |
| Vol. 8, No. 6 | A Business Value Focus for Portfolio Management BY KENT J. MCDONALD |
| Vol. 8, No. 5 | An Organizational Structure for Agile Projects BY DAVE ROONEY |

## Executive Updates

| | |
|---|---|
| Vol. 9, No. 16 | Managing Offshore Development: Establishing Software Development Processes BY STACEY BERLOW |
| Vol. 9, No. 15 | Insecure Code: An Agile Look at the Debian Debacle BY LAURENT BOSSAVIT |
| Vol. 9, No. 14 | Software Project Planning: Part III — Divide and Conquer BY E.M. BENNATAN |
| Vol. 9, No. 13 | A Practical Guide to Crafting an Agile Adoption Strategy BY AMR ELSSAMADISY |
| Vol. 9, No. 12 | Flying the Friendly Skies: Navigating Agile Projects BY DAVID ROONEY |
| Vol. 9, No. 11 | Software Project Planning: Part II — Back to 1978 BY E.M. BENNATAN |
| Vol. 9, No. 10 | Software Project Planning: Part I — What Can We Learn from *Star Trek*? BY E.M. BENNATAN |
| Vol. 9, No. 9 | Agile Software Development: A Customer-Centric Way of Doing Business BY STACEY BERLOW |
| Vol. 9, No. 8 | Qualities of a Great Project Manager BY MARTIN BAUER |
| Vol. 9, No. 7 | 'The Big Splash': Discovering How Wet Our Leadership Behaviors Can Be BY DAVID SPANN |
| Vol. 9, No. 6 | The Agile Project Management Revolution BY MARTIN BAUER |

## What Do Clients Receive?

***Executive Reports:*** Clients receive *Executive Reports* — written by the top thinkers in project management, product management, agile methods, or portfolio management — providing tools for transferring knowledge throughout your development groups. Don't miss these upcoming reports:

- Succeeding in the Contemporary World of Project Management
- Scrum Today
- Agile Service Orientation

***Executive Summaries:*** Each *Executive Report* is accompanied by a fast-reading *Executive Summary*, providing you and your project managers with a quick synopsis that highlights the key points made in the *Executive Report*.

***E-Mail Advisors:*** Receive solid advice each week on implementing best practices in project management, collaboration, agile software development, and leadership, from Jim Highsmith and other members of the agile team.

***Executive Updates:*** A steady flow of *Executive Updates* provides insight into the product and project development challenges organizations like yours are facing, as well as analysis of Cutter's exclusive survey data.

## Inquiry Privileges

Cutter's Access to the Experts inquiry program provides specialized advice directly from Cutter's renowned experts. Every inquiry is fielded by a Cutter Senior Consultant, Fellow, or Faculty Member with hands-on experience addressing agile integration challenges at organizations like yours. You get tested answers to your questions, unlike those you'd get in any other inquiry program.

## Consulting and Training

Cutter Consortium offers consulting and training customized to the unique business drivers, culture, technology, and history of your organization. From project management to collaboration, software development practices, portfolio management, agile leadership skills, project governance, and agile enablement, accessing Cutter's experts gives you the confidence that comes from relying on the best minds in the industry.

## Becoming a Client

Cutter Consortium's research and content provide you with the support you need to explore and implement today's best practices in agile product management, project management, and software development. For example, in recent months we've helped our clients:

- Leverage performance management systems to achieve organizational agility
- Embrace a collaborative leadership model to manage evolving environments
- Optimize testing and inspection practices
- Promote innovation by implementing agile project management

If you would like to evaluate Cutter's information services or discuss how Cutter's consultants can help you apply agile techniques at your organization, just call us at +1 781 648 8700 or send e-mail to sales@cutter.com. Or you can fill out the form below and send it by fax to +1 781 648 8707.

☐ ***YES!*** **I am interested in learning more about Cutter Consortium's *Agile Product & Project Management* practice.** Please contact me with more information.

CUTTER CONSORTIUM

| | |
|---|---|
| Name | Title |
| Organization | Dept. |
| Address/P.O. Box | |
| City | State/Province |
| ZIP/Postal Code | Country |
| Telephone | Fax |
| E-Mail Address | |

660*C05SUM

Call **+1 781 648 8700**; send e-mail to service@cutter.com; fax to +1 781 648 8707; mail to Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA; Web site: **www.cutter.com/project.html.**

About the Practice

# Agile Product & Project Management Practice

Cutter Consortium's Agile Product & Project Management practice provides you with information and guidance — from experienced, hands-on Senior Consultants — to help you transition (or make the decision to transition) to agile methods. Led by Practice Director Jim Highsmith, Cutter's team of experts focuses on agile principles and traits — delivering customer value, embracing change, reflection, adaptation, etc. — to help you shorten your product development schedules and increase the quality of your resultant products. Cutting-edge ideas on collaboration, governance, and measurement/metrics are united with agile practices, such as iterative development, test-first design, project chartering, team collocation, onsite customers, sustainable work schedules, and others, to help your organization innovate and ultimately deliver high return on investment.

Through the subscription-based publications and the consulting, mentoring, and training the Agile Product & Project Management Practice offers, clients get insight into Agile methodologies, including Adaptive Software Development, Extreme Programming, Dynamic Systems Development Method, and Lean Development; the peopleware issues of managing high-profile projects; advice on how to elicit adequate requirements and managing changing requirements; productivity benchmarking; the conflict that inevitably arises within high-visibility initiatives; issues associated with globally disbursed software teams; and more.

**Products and Services Available from the Agile Product & Project Management Practice**

- The Agile Product & Project Management Advisory Service
- Consulting
- Inhouse Workshops
- Mentoring
- Research Reports

**Other Cutter Consortium Practices**

Cutter Consortium aligns its products and services into the nine practice areas below. Each of these practices includes a subscription-based periodical service, plus consulting and training services.

- Agile Product & Project Management
- Business Intelligence
- Business-IT Strategies
- Business Technology Trends & Impacts
- Enterprise Architecture
- Innovation & Enterprise Agility
- IT Management
- Measurement & Benchmarking Strategies
- Enterprise Risk Management & Governance
- Social Networking
- Sourcing & Vendor Relationships

## Senior Consultant Team

The Cutter Consortium Agile Product & Project Management Senior Consultant Team includes many of the trailblazers in the project management/peopleware field, from those who've written the textbooks that continue to crystallize the issues of hiring, retaining, and motivating software professionals, to those who've developed today's hottest Agile methodologies. You'll get sound advice and cutting-edge tips, as well as case studies and data analysis from best-in-class experts. This brain trust includes:

- Jim Highsmith, Director
- Sanjiv Augustine
- Christopher M. Avery
- Paul G. Bassett
- Sam Bayer
- Kent Beck
- E.M. Bennatan
- Tom Bragg
- David R. Caruso
- Robert N. Charette
- Alistair Cockburn
- Jens Coldewey
- David Coleman
- Ken Collier
- Ward Cunningham
- Rachel Davies
- Doug DeCarlo
- Tom DeMarco
- Lance Dublin
- Khaled El Emam
- Kerry F. Gentry
- Chet Hendrickson
- Sid Henkin
- David Hussman
- Ron Jeffries
- Joshua Kerievsky
- Bartosz Kiepuszewski
- Brian Lawrence
- Mark Levison
- Tim Lister
- Michael Mah
- Lynne Nix
- Siobhán O'Mahony
- Ken Orr
- Patricia Patrick
- Roger Pressman
- James Robertson
- Suzanne Robertson
- Alexandre Rodrigues
- David Rooney
- Johanna Rothman
- David Spann
- Rob Thomsett
- John Tibbetts
- Jim Watson
- Robert K. Wysocki
- Richard Zultner