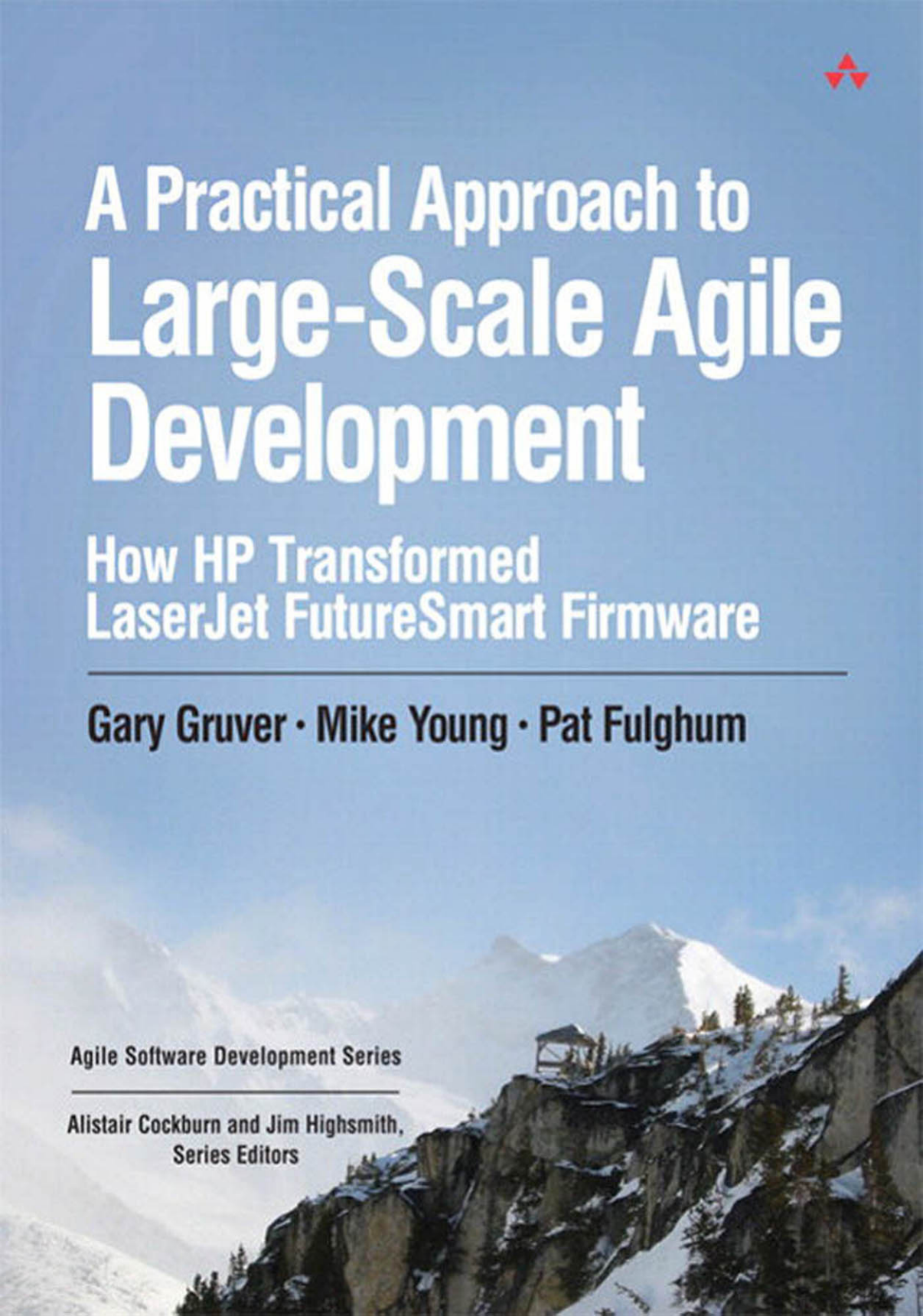
A Practical Approach to
Large-Scale Agile
Development
How HP Transformed
LaserJet FutureSmart Firmware
Gary Gruver · Mike Young · Pat Fulghum
Agile Software Development Series
Alistair Cockburn and Jim Highsmith,
Series Editors

A Practical Approach to Large-Scale Agile Development

Metrics Are a Conversation Starter

It's also all about how you respond to pressure and bad news. Historically, we had a culture where no one would speak up if they weren't going to hit a schedule or were struggling with solving a difficult technical problem. Have you ever heard of "schedule chicken"? It was alive and well with us! No one wanted to be the bad guy, which means we learned about problem areas late, and there were a lot of burned-out, lonely people trying to get off the critical path but rarely succeeding. Our managers work closely with the teams to understand how much work they can handle and when. Many organizations use consultants and agile coaches for this, but we've always relied on our own managers (we've focused on training all our managers on the agile principles we promote and let them work the details with their teams). This requires good tools for tracking real-time what is and isn't getting done. The key, though, is *not to manage by metrics but to use the metrics to understand where to have conversations about what is not getting done***.** Always trust that engineers are doing the best they know how or can in the situation. People want to do a good job; so when the work is not coming along, there is usually some key reason: a need for training or technical help, a need to improve architecture or processes, or there is just too much work in the system and you are going to have to adjust priorities to make progress on that particular vector. Having clear, integrated priorities for all work makes a big difference, especially when you give the message: "If you're working on something at the top of the priority list, you have access to anyone, anytime to get whatever help you need."

The worst thing that can happen with metrics is that people start feeling beat up, which means they will start hiding data. If the management team does not understand the constraints because of the team's concerns about getting beat up for not being on time, then you never know where to help or how to move resources around. If teams see management as helping to move resources and provide creative and helpful changes, the issues will be made visible early, and you can respond and adjust in real-time. This leads to a positive reinforcing cycle that is key to the agile management approach.

At the most stressful times of development and deadlines, we've had several friendly competitions with balloons placed with "the team with the most issues/change requests," and so on. This has helped turn stress into a more enjoyable process. Most people enjoy "delivering innovation" more than the endgame of a release, but when it is a game or a friendly competition, everything gets closed out more efficiently and people have fun along the way.

Iterative Model of Agile Management

We came up with a model we call the Iterative Model of Agile Management (see Figure 5.1). Agile at its best is not just a development model but a way to manage people and processes. It shows that management doesn't "know it all" and is willing to listen, adjust, and change for the betterment of everything, with more focus on real people and what they care about. This model is all about setting new objectives for the whole organization every four weeks, and then putting metrics in place to track how we're doing against those objectives. Those metrics are then used continuously, not to drive people harder, but to have a conversation to discover where the problem areas are and what we can do to help. Then we learn from it and adjust whatever is needed for the next cycle.

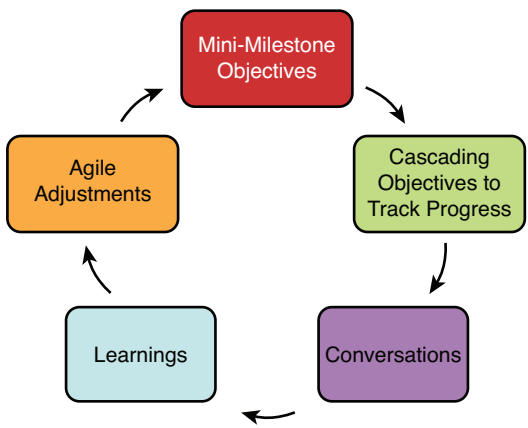

FIGURE 5.1 Iterative approach to agile management

Mini-Milestone Objectives

We start each monthly Mini-Milestone (sprint) with a clear set of objectives and high-level priorities for the organization. The idea here is to set aggressive goals the team feels are possible and important to achieve in 4 weeks. It is surprising how much you learn in a month and have to adjust based on discovery during development. We typically drive hard for these stretch goals but usually end up hitting around 80% of what we thought we could at the beginning of the month. During the Mini-Milestones we don't track team-based or program-level burndown charts, but we do track progress. We had some teams try burndown charts, and we looked at it for the overall program but felt the overhead for tracking accuracy was taking capacity away from maximizing throughput. Instead, we have an overall set of metrics we track during the MM that helps align everyone in the organization and gives us a clear view of where we are not meeting the objectives. The objectives include user stories we implement and qualify but also other initiatives including early hardware support, endgame activities, and tool/process improvements (whatever's important or urgent to the business becomes an objective). Table 5.1 is an actual list of prioritized objectives for our MM30 sprint (2½ years into our agile experience).

TABLE 5.1 Sample Mini-Milestone Objectives (MM30 Objectives)

Rank	Theme	Exit Criteria: Objective Met/*Objective not met*
0	Quality threshold	P1 issues open < 1week L2 test failure 24-hour response
1	Quarterly bit release	*A) Final P1 change requests fixed* B) Reliability error rate at release criteria
2	New platform stability and test coverage	A) Customer Acceptance Test 100% passing B) All L2 test pillars 98% passing C) L4 test pillars in place D) L4 test coverage for all Product Turn On requirements E) 100% execution of L4 tests on new products
3	Product Turn On dependencies and key features	A) Print for an hour at speed to finisher with stapling B) Copy for an hour *at speed* C) *Enable powersave mode* D) Manufacturing nightly test suite execution E) Common Test Library support for four-line control panel display

4	Build for next-gen products	*A) End-to-end system build on new processor* *B) High-level performance analysis on new processor*
5	Fleet integration plan	Align on content and schedule for "slivers" of end-to-end agile test with system test lab

Cascading Objectives to Track Progress

The fact that these metrics and prioritized MM objectives are online and available for everyone does a great job of aligning the organization. The metrics at the overall program level are a summation of the metrics for each section manager and project manager. Therefore, everyone in the organization has an aligned set of goals to strive for and track each month. These metrics range from the status of requirements development to change requests to tests passing. The tools even enable us to track individual code contributions throughout the day or summary metrics for different time periods. The ability to track this information and have it online has basically eliminated tracking and status meetings. We don't require scrum meetings or scrum of scrum meetings to know what's going on. Within an hour of reviewing the metrics in the morning, it is easy for the management team to know more about the status of the program than we ever thought possible. In fact, it got to the point where the developers nicknamed the Director of Engineering "the unblinking eye" because he never misses anything and always knows what is going on in the organization. At the same time, he spends very little time in meetings getting status or tracking progress. The nickname started one day when the Director noticed that someone broke the build 15 minutes after it happened. He stopped by the engineer's desk to see what happened and the engineer promptly responded, "What are you, the unblinking eye? Don't you miss anything?" This led to a great April Fool's joke where the entire floor was decorated in the *Lord of the Rings* theme. There were bloodshot-eye balloons, name plates we changed with the theme, and the online build environment icons were even changed. It may seem silly, but it really helped to make it a fun environment where you wanted to work when you knew it was safe to make fun of the Director of Engineering.

We send out an automated daily early morning email with a summary of the objectives and the metrics, with a link to get to an hourly version of the same data online. We've included samples of some of the key metrics we track in Figure 5.2. (Each cell in the tables is a drill-down to the details through each layer of the organization.) These all help us make sure we're both maintaining the appropriate code stability and making appropriate progress in each sprint. We go into more detail on the meaning and purpose behind these metrics in later chapters.

Open Change Requests

Section Manager	P1 Punch list	P1	P2	Newly Found	Newly Fixed
Section A	0	2	20	2	2
Section B	0	0	3	0	0
Section C	3	14	29	0	2
Section D	0	5	6	0	1
Section E	0	4	52	4	1
Section F	3	14	25	1	2
Total	6	39	135	5	8

Requirements Status (current sprint)

Section Manager	Not Started	Under Dev	Imple-mented	Under Test	Verified	Total
Section A	5	3	0	0	4	12
Section B	14	20	5	27	12	78
Section C	3	7	0	4	4	18
Section D	8	10	1	0	15	34
Section E	11	11	6	19	10	57
Section F	17	20	9	6	22	74
Total	58	71	21	56	67	273

System Test Execution [test runs are "out of date" after a week (auto) or a month (manual); drill-down shows breakdown by team]

Product	Simulator Auto Pass/Exec	Emulator Auto Pass/Exec	Product HW Auto Pass/Exec	Product HW Manual Pass/Exec	Out of Date Exec
Product A	15288/16114 (95%)	2132/2352 (91%)	10/10 (100%)	390/411 (95%)	141
Product B	8251/9262 (89%)	359/589 (61%)	2/17 (12%)	107/132 (81%)	20
Product C	5577/6113 (91%)	224/1035 (22%)	1/4 (25%)	155/167 (93%)	46

Integration Queue / Automated Builds Status

Daily Summary Metrics for 05/24/2011

Builder Type	Build Type	Builds Processed	Builds Passed	Builds Failed	Branches Processed	Branches Passed	Branches Failed
⊞ Stage 2 Builder	Stage2 Combined	10	4 (40%)	6 (60%)	133	105 (79%)	28 (21%)
⊞ Stage 1 Builder	Stage1 CEQbar	36	31 (86%)	5 (14%)	36	31 (86%)	5 (14%)
⊞ Stage 1 Builder	Stage1 Qbar	50	42 (84%)	8 (16%)	50	42 (84%)	8 (16%)
⊞ Stage 1 Builder	Stage1 XPQbar	23	21 (91%)	2 (9%)	23	21 (91%)	2 (9%)
Totals for all Stage 2 builds		10	9 (90%)	1 (10%)	133	105 (79%)	28 (21%)
Totals for all Stage 1 builds		109	94 (86%)	15 (14%)	109	94 (86%)	15 (14%)

FIGURE 5.2 Sample metrics for prompting daily conversations

Conversations

The majority of the time is spent having conversations to understand where and why we are not meeting objectives. If you are going to use the flexibility of agile principles to improve the effectiveness of your organization, you must constantly be seeking to understand what is working

and what can be improved. *You can't manage by metrics and only drive to meet the numbers. The metrics help you understand where to go to have conversations about what is and isn't working.* People want to do a good job and are doing their best, so when they are not meeting a clear objective that they felt was achievable at the beginning of the month, it's important to understand why and what needs to be adjusted so you can help. You also need to be careful with how the management team uses this new detailed view into the development process. The engineering director frequently knows within an hour when someone has made a big mistake and broken main. It is very easy with this visibility to create a destructive, blame-type culture where everyone slows down to the point of trying to make everything perfect before committing, which would kill productivity. At the same time, if there are repeat offenders, it should lead to conversations to reset expectations. A lot of this happens through peer pressure when people are bringing in code that is impacting the productivity of the engineers, but management does need to play a role (although carefully) because knowing so much can be used to drive the wrong behavior.

Learning

One example of what we learned from this iterative approach to agile management is that the management team also needs to make sure it is using the tracking data to help understand where to move and adjust resources. During the development process, we try to actively advertise the bottleneck for the program. This helps the organization understand that if there are general tasks, like test failure triage where another team can help out, that a non-bottleneck, non-critical-path team should step in to lead the initial investigation. The broader organization should also look at loaning developers to the bottleneck and adjusting other resources to help. If you can create this culture and really help out when needed, it makes everyone feel that they are on the same team. It's suddenly not such a bad thing to be the bottleneck, because the organization is there for support. When we used this process the past few years in our development, the bottleneck shifted several times, based on helping out and