Chapter 4

An Agile Project Management Model

Principles and Practices

"Hi, Maya, it's Herman."

"Hi, dude, how's your project going?" asked Maya.

"Pretty well. Things are actually moving along. We've implemented a few of the agile practices. But you know, I'm just an action kind of guy. Just explain the practices to me. The principles stuff still seems like fluff."

"I thought they were fluff, too, at the beginning," Maya responded. "But once you've used APM for a while, you'll understand that it's the principles that make the practices flexible."

"OK, apply principles to iteration planning."

"Think about Deliver Customer Value. For us, this principle guides the selection of features. We're constantly asking ourselves whether one feature is more valuable than the next."

"But won't most of the features ship anyway?"

"Possibly, but we keep two things in mind. Sometimes we release incremental versions, and the goal of always having a releasable product really keeps us on our toes."

"Isn't early release unlikely?" Herman asked.

"Mostly, but last summer we had a customer with a critical problem. We were able to take a product that was three-quarters finished, do a couple of quick special features for the customer, and deliver it in three weeks. The customer was blown away, and this one spends megabucks."

"So, on to the Employ Iterative, Feature-Based Delivery principle. That seems redundant with the practice to me, but maybe that just reinforces its importance."

"That's the general idea. See, you already understand this stuff, Herman! You just don't trust it yet." Maya grinned to herself. She knew it was hard for Herman, who worked for a 75-year-old insurance firm, to break with the conservative, traditional approach that Great Mid-West had always taken to everything, but he was trying, and she gave him lots of credit.

"That's fine as far as it goes, but does this scale? You and I both know that any competent PM can complete a small, short-term project on force of will alone, but that all changes when you scale the whole thing up."

"Yeah, but that's where the Simplify principle comes in. It's how you scale everything up without tripping over yourself and your process. For example, on Jupiter, the new project I'm running, we have a big team. And to make matters worse, one feature team is distributed. Adopting a collaboration strategy was a given, but we also figured that we might need some additional documentation to keep everyone in sync."

Herman interrupted with a "See, I told you so."

"But the Simplify principle keeps it from turning into the pounds of paperwork you generate," Maya laughed. "We're bears about using the simplest documentation that accomplishes the goal. We work with just a few documents and keep them as informal as we can. Then we adapt them from time to time as we find what works and what doesn't."

"So, you use the principles to help adapt practices to specific situations," Herman said.

"Right. Without these guiding principles we could get hung up on the specific practices instead of understanding the intent of the practices. They keep us from going overboard."

"One I really get hung up on is Encourage Exploration," Herman replied. "This whole notion of responding to change over following a plan, of actually embracing change, is really foreign."

"That's a tough one," said Maya. "We can respond to change and deliver reliably. It's all in how you look at the relationship between uncertainty and time."

"My management doesn't care. They just want a commitment," said Herman.

"There is no way around the fact that the higher the uncertainty, the wider the potential schedule variation," said Maya. "It's too bad you aren't here—I need a whiteboard to draw this, but imagine that the probability curve of schedules is a skewed distribution curve, one with a long tail of possible very late delivery dates. High exploration-factor projects have a lot of possible dates based upon technology and requirements volatility."

"Everyone around here just assumes a project is a project is a project. There's no allowance for riskier projects—we just get the mantra 'on time, on budget, on scope' over and over. It's like a broken record."

"Don't remind me. Hopefully we're past that. In effect, the role of dates changes. In low-uncertainty projects, dates are predictions. For high-uncertainty projects, dates are boundaries, as in 'We will deliver as many features as possible by June.' Does that make sense?"

"I think I understand, but I'm not sure if the folks around here will get it," Herman said.

"It took us a while to work through it, too," Maya continued. "That's why the principle Encourage Exploration is critical to reducing people's anxiety. I have to keep encouraging people and reminding them that responding to change is part of our day-to-day work. We always have a few people from a conformance-to-plan type organization who get a little crazy at first. We project managers have to encourage them."

"Lots to think about," said Herman. "Bye for now."

An Agile Process Framework

Process may not be as important as people, but it's far from unimportant. Process has gotten a bad rap in agile circles (much of it deserved) as being static, prescriptive, and difficult to change. But process, per se, doesn't have to be negative, although in many companies the move to "improve process" leads down a slippery slope to standardization and certification, at which point the static, prescriptive, and difficult to change criticisms are generally accurate. Process, like anything else, must be tied to business objectives. If the business objective is repeatable manufacturing, then a prescriptive process may be completely justified. However, if the business objective is reliable innovation, then the process framework must be organic, flexible, and easy to adapt. An agile process framework needs to embody the principles described in the last two chapters. In addition to supporting business objectives, the framework needs to:

- Support an envision, explore, adapt culture
- Support self-organizing, self-disciplined teams
- Promote reliability and consistency to the extent possible given the level of project uncertainty
- Be flexible and easy to adapt
- Support visibility into the process
- Incorporate learning
- Incorporate practices that support each phase
- Provide management checkpoints for review

The APM model's structure—Envision-Speculate-Explore-Adapt-Close— focuses on delivery (execution) and adaptation (see Figure 4.1). It is based on the Speculate-Collaborate-Learn model first described in *Adaptive Software Development* (Highsmith 2000). In the APM model, the Explore phase replaces the Collaborate phase in the earlier model. Although collaboration practices dominate the phase, the "action" is better described by Explore. Similarly, while Learn reflects the monitoring portion of a feedback-gathering phase, Adapt completes the loop. Not only is the team learning, it is also taking action on that learning.

The APM phase names reflect both activities and results. For example, the Envision phase results in a project vision. Furthermore, the departure from traditional phase names—such as Initiate, Plan, Manage, Control—while subtle, is significant. First, "Envision" replaces the more traditional "Initiate" to indicate the criticality of vision. Second, a Speculate phase replaces a Plan phase. Words convey certain meanings, and those meanings arise from systematic use over time. The word "plan" has become associated with prediction and relative certainty. "Speculate" indicates that the future is uncertain. We know the future of any project, particularly high exploration-factor projects, contains uncertainty, but we still try to "plan" that uncertainty away. We have to learn to *speculate* and *adapt* rather than *plan* and *build*.

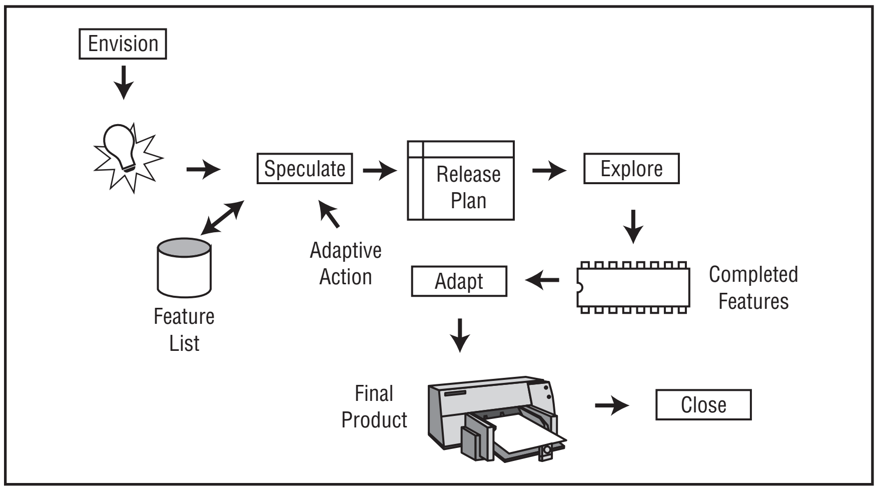

Figure 4.1 The APM Process Framework

Third, the APM model replaces the common Manage phase with Explore. Explore, with its iterative delivery style, is *explicitly* a nonlinear, concurrent, non-waterfall model. Questions developed in the Speculate phase are "explored." Speculating implies the need for flexibility based on the fact that you cannot fully predict the results. The APM model emphasizes execution and the fact that it is exploratory rather than deterministic. Fourth, a team practicing APM keeps its eyes on the vision, monitors information, and adapts to current conditions—therefore the Adapt phase. Finally, the APM model ends with a Close phase, in which the primary objectives are knowledge transfer and, of course, a celebration.

To sum up, the five phases of agile project management are:

1. **Envision**: determine the product vision and project scope, the project community, and how the team will work together
2. **Speculate**: develop a feature-based release, milestone, and iteration plan to deliver on the vision
3. **Explore**: deliver tested features in a short timeframe, constantly seeking to reduce the risk and uncertainty of the project
4. **Adapt**: review the delivered results, the current situation, and the team's performance, and adapt as necessary
5. **Close**: conclude the project, pass along key learnings, and celebrate

Phase: Envision

The Envision phase creates a vision for the customers and the project team that covers what, who, and how. Absent a vision, the remaining activities in getting a project off the ground are wasted effort. In business-speak, vision is the "critical success factor" early in a project. First, we need to envision *what* to deliver—a vision of the product and the scope of the project. Second, we need to envision *who* will be involved—the community of customers, product managers, project team members, and stakeholders. And, third, the project team members must envision *how* they intend to work together.

Phase: Speculate

The word "speculate" first calls to mind an image of reckless risk taking, but actually the dictionary definition is "to conjecture something based on incomplete facts or information," which is exactly what happens during this phase.[1] The word "plan" has come to connote certainty and prediction, while the more useful definition of plan, for exploratory projects at least, is speculating or hypothesizing based on incomplete information. Colleague Ken Delcol makes a great observation: "People believe when they plan that they introduce certainty, which is far from the truth. What they introduce is something to gauge their performance by. Then, when the gauge does not reflect reality, they fail to replan." APM consists more of envisioning and exploring than planning and doing—it forces us to confront the reality of today's precarious business environment and highly volatile product development environment.

The Speculate phase, which is actually an extension of and interactive with the Envision phase, consists of:

- Gathering the initial broad requirements for the product
- Defining the workload as a list of product features

1. Encarta® World English Dictionary, © 1999, 2000 Microsoft Corporation.

- Creating a delivery plan (release, milestones, and iterations) that includes schedule and resource allocations for those features
- Incorporating risk mitigation strategies into the plan
- Estimating project costs and generating other required administrative and financial information

Phase: Explore

The Explore phase delivers product features. From a project management perspective there are three critical activity areas during this phase. The first is delivering planned features by managing the workload and using appropriate technical practices and risk mitigation strategies. The second is creating a collaborative, self-organizing project community, which is everyone's responsibility but is facilitated by the project manager. The third activity is managing the team's interactions with customers, product management, and other stakeholders.

Phase: Adapt

Control and correction are common terms applied to this lifecycle phase. Plans are made, results are monitored, and corrections are made—implying that the plans were right and the actual results, if different from the plan, are wrong. "Adapt" implies modification or change rather than success or failure. In projects guided by the philosophy that responding to change is more important than following a plan, attributing failure to variation from the plan isn't productive. A purely ad hoc process fails to learn from its mistakes, whereas the incorporation and retention of lessons learned are key pieces of APM.

After the Envision phase, the loop will generally be Speculate-Explore-Adapt, with each iteration successively refining the product. However, periodically revisiting the Envision phase may be necessary as the team gathers new information.

In the Adapt phase the results are reviewed from customer, technical, people and process performance, and project status perspectives. The analysis looks at actual versus planned, but even more importantly, it considers actual versus a revised outlook on the project given up-to-the-minute information. The results of adaptation are fed into a replanning effort to begin the next iteration.

Phase: Close

Projects are partially defined by the presence of both a beginning and an end. Many organizations fail to identify a project's end point, often causing perception problems among customers. Projects should end—with a celebration. The key objective of the Close phase, and the "mini" close at the end of each iteration, is learning and incorporating that learning into the work of the next iteration or passing it on to the next project team.

Judgment Required

Because of product and project management's long history of favoring serial development processes, any figure like that of Figure 4.1 can take on a serial appearance. However, while a project may follow the general sequence of Envision, Speculate, Explore, Adapt, and Close, the entire model should be considered fluid. The wording of production-style models implies linearity—Initiate, Plan, Manage, Control—while the APM terms were selected to imply iterative evolution—Speculate, Explore, Adapt.

An overemphasis on linearity leads to stagnation, just as an overemphasis on evolution leads to endless, and eventually mindless, change. With either model, development team members, customer team members, and executives need to exercise keen judgment in its application.

Project Size

The core values and principles of APM are applicable to projects of any size. Similarly, the practices described in the next few chapters are applicable to projects of any size. However, for project teams that exceed 50 or so people, additional practices or extensions to the described practices may be necessary—some of which are described in Chapter 9. As project teams get larger, more documentation, additional coordination practices, increased ceremony, or other compliance activities (financial controls, for example) are usually needed. However, even these expanded practices should still be governed by APM's values and principles. For example, the principle of Simplify still applies to a large project; it just means to employ the simplest practice that works for a team of 150 rather than one of 15.

A 500-person team can't be as agile as a 10-person team, but it *can* be more agile than a competitor's 500-person team. By focusing on delivery, self-organization, and self-discipline, even larger teams burdened with complex coordination issues can readily adapt to business, technology, and organizational changes.

Agile Practices

The next four chapters describe specific practices that align with the agile values and guiding principles for each of the APM framework phases. These practices should be considered a "system of practices," because as a system, they reinforce each other as they align with values and principles. But they do more than align; they implement. Principles without practices are empty shells, while practices without principles tend to be implemented by rote, without judgment. Without principles, we don't know "how" to implement practices—for example, without a Simplify principle we tend to overdo the formality and ceremony of almost any practice. Principles guide practices. Practices instantiate principles. They go hand in hand.

Aligning principles and practices prompts the realization that the holy grail of "best practices" is a sham. A wonderful practice for one project team may be a terrible practice for another. Practices are just practices—various ways of carrying out some goal. A practice is only good or bad within some context, which might include principles, problem type (e.g., exploratory), team dynamics, and organizational culture.

The practices in the following chapters have proven useful in a variety of situations. Some could be useful in production-style projects, just as practices not included may be very useful in agile projects. In selecting and using these practices, I've used these guiding principles:

- Simple
- Generative, not prescriptive
- Aligned with agile values and principles
- Focused on delivery (value adding), not compliance
- Minimum set (just enough to get the job done)
- Mutually supportive (a system of practices)

Few, if any, of the practices described in the following chapters are new. Some of them are variations on a theme of practices described by others. Some are well known; others are not so well known. For example, risk management practices are widely described in the project management literature, while others, like participatory decision making, are not. Therefore, common practices such as risk management will be briefly described and other resources will be referenced, while less well-covered practices such as decision making will be described in more detail.