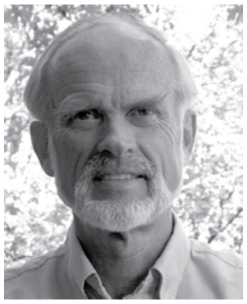
Preston G. Smith

Katherine Radeka

Flexible product development for a turbulent world—Is “Agile” NPD the answer?

Preston G. Smith, Principal, New Product Dynamics (preston@NewProductDynamics.com) and Katherine Radeka, President, Whittier Consulting Group Inc. (katherine.radeka@whittierconsulting.com)

As the debate continues about how to make the product management process more efficient, a number of new techniques have evolved. Within the “flexible” school of product development and management is “agile” product development—which came out of the software development field. In this article, the authors describe this technique and its virtues in new product development (NPD).

In today’s uncertain environment, when economic recovery is either years away or just around the corner, product developers need specific tools to help them maintain flexibility for as long as possible, so that they can make key product decisions closer to product launch.

Software development has faced this challenge for most of its history. Even today, new technology arrives in a continuing torrent, providing both unexpected benefits and challenges. Customers have become more informed and more vocal in expressing their desires. Global business creates new opportunities and new competitors. Any of these situations can provoke a change in the middle of any development project. But some of the lessons software developers have learned in dealing with this chaos apply directly to other types of products—products that are as far away from software as playground equipment and airplanes.

“Many software developers recognize that adding rigidity into a process…only leads to products that don’t satisfy customers.”

At first, software developers responded by tightening up the development process, using a rigorous “waterfall” lifecycle where work cascaded from one step to the next, that looks a lot like the phased development processes that Bob Cooper, of Stage-Gate ™ fame and others have advocated.

The reasoning behind the development of “agile”

Today, many software developers recognize that adding rigidity into a process that is buffeted by change only leads to products that don’t satisfy customers. In 2001, 17 software leaders expressed their frustration with the waterfall lifecycle’s lack of responsiveness and proposed a liberating alternative: “agile” software development.

The Agile Manifesto (agilemanifesto.org)[1] comprises four comparisons, three of which apply equally to other types of development:

- Individuals and interactions *over* processes and tools
- Working software *over* comprehensive documentation
- Responding to change *over* following a plan

The authors clearly stated that while the items on the right were important, the ones on the left mattered more. This is a revolutionary statement, because most of us revere processes, tools, complete documentation, and following the plan. But these are the very items that break down when conditions are highly volatile. The agile software community acknowledges the realities of operating in an environment where change comes unexpectedly and flexibility must be preserved as long as possible.

Consequently, in order to maintain flexibility—of software or anything else—one must think and operate quite differently from today’s norm. This article describes three such areas that developers must handle in new ways and closes by suggesting how you can implement this approach today.

The pros and cons of delaying decisions

Most managers are paid to make decisions—the sooner the better usually. Indecision is often seen as a sign of weakness, and if we knew everything we needed to know up front, that might be right. But during turbulence, early and rigid decisions are a straightjacket.

The problem with early decisions in a changing environment is that once a decision is made, changing it often has an associated cost, what we call the cost of change. The agile developer instead keeps options open as long as possible, until the “last responsible moment”—that last point in time when the team can make the decision without significantly impacting the rest of the project.

Software development expert Mary Poppendieck illustrates this “last responsible moment” with a story: “They teach airline pilots in Switzerland to make the decision about whether or not to proceed with a landing attempt at 1,000 meters (approximately 3,000 feet) above landing altitude—no earlier, and no later. Why? The weather is unstable in the Alps, and pilots need to make landing decisions as late as possible to take advantage of the best available information. Why not later? The clouds have rocks in them.” [2]

The idea is first to establish the last responsible moment, schedule it, and then start collecting information to help make a better decision when its last responsible moment arrives. Teams can gain a tremendous amount of flexibility by thinking through the key decisions where information is likely to change and then creatively scheduling their work to delay those decisions.

Note that delaying decisions in this way is not procrastination. Procrastination is being lazy about decision-making (a sign of a poor manager). In contrast, scheduling a decision and collecting information before you must make it is an anticipative, active process.

Managing the project

There are too many product development organizations that run their projects like construction projects, where change is limited by the need to create detailed architectural drawings to coordinate work across multiple subcontractors. Even in that world, classic project management requires a lot of effort to keep the plans current in the face of weather delays, customer change orders and subcontractor communication issues.

In the best of times, product development programs must accommodate much more change. Normal project planning practice is to plan the program based on detailed specifications, then follow the plan. In fact, project managers are often rewarded for following the plan. But when change is rampant, building a detailed plan at the outset is wasteful, because it will just have to be changed.

One way around this—described in both PDMA's *PDMA ToolBook 3*[3] and an article by Greg Githens in *Visions*[4]—is rolling-wave planning, which is a method of planning one segment at a time in detail, leaving later segments to be detailed later as the team encounters them. For example, a team may only create detailed plans four weeks out, with the entire program guided by high-level plans, which are easy to change because they contain only the level of detail necessary to coordinate between groups.

The other approach, which is a mainstay in agile software development and was used to develop the Boeing 777 airliner, is "loose-tight" planning, in which the team alternates between periods of tight planning and open periods where the plan can be changed easily. The Boeing 777 team alternated between periods of design, where change and creativity reigned, and periods of stabilization, where sub-teams coordinated their work.[5]

"Agile software developers apply loose-tight planning by organizing their work into a series of short development cycles."

Agile software developers apply loose-tight planning by organizing their work into a series of short development cycles. Only the current cycle has a detailed plan, and it does not change during the cycle. An open planning period happens at the end of each cycle to create the detailed plan for the next cycle.

Defining the product

One thing you can expect to change is the product requirements or specifications. Normal "best practice" is to conduct Voice-of-the-Customer (VOC) research, distill this into a list of requirements, freeze them, and develop the product to these requirements. But this often degrades into finger-pointing between engineering and marketing about the quality of the requirements, when the true cause is simply changing conditions.

This ideal state never happens: In a survey conducted with more than 1,000 development managers, there was *always* a change in requirements during design. Moreover, only five percent of designers even had complete requirements when they started designing, and for each developer who waited for at least 80 percent of the requirements, five others, under time pressure, had already started designing.[6]

Agile developers balance the need for stability in requirements against the reality of changing requirements and the need to give developers rich information about customer perceptions.

One way to overcome the dilemma of changing requirements is to define the customer or user, rather than the product, by using personas crafted carefully from your market research. Or write use cases or user stories, which describe how the user uses the product, again carefully built from solid customer research. Finally, get feedback from customers around important details that affect them and might therefore change: user interfaces, industrial designs and new features. There are excellent books in the software development literature describing these techniques.[7]

What do I do next?

Chances are, your organization has new projects about to start, projects under way and projects that have already experienced the conflict of rigidity under constant change.

If you have projects about to start, you can maximize your flexibility from the beginning stages of the project. As you plan your concept and feasibility phases, identify the key design decisions and think of ways to delay those decisions as long as possible. Begin short development cycles now to gain experience with them so that when you need them, the team is ready. Find ways to get better customer information now, before the design has frozen. Finally, consider ditching the specifications document in favor of customer personas, use cases and stories that use narrative to convey richer information about the customer.

If you have projects in the middle of development with high risk of late changes, take steps now to mitigate the effects. Catalog the major design decisions that have already been made – and that are planned to close soon – and identify the ones with high risk of change later in the project. Decide whether or not you have truly reached the "last responsible moment" for major decisions, or if some creative scheduling could provide more time and clarity. Then delay those decisions you can, and add the rest to the list of risks to monitor over the course of the program; assign last responsible moments to all of them. As work proceeds, switch to short planning cycles – one to two months works best for many teams – to incorporate more flexibility into your plans.

If you have projects that are already suffering from high degrees of late changes and late discoveries, immediately switch to short planning cycles—two weeks or less if things are truly critical—to eliminate the overhead of maintaining unrealistic schedule detail and to help make the team's issues visible immediately. Then identify the major areas of risk that have yet to be resolved and create a proactive management plan. Some major risks now probably didn't show up on the list created in initial planning. Finally, catalog the major decisions that still have high degrees of risk, and seek ways to delay those decisions.

By using these techniques to add flexibility to your product development programs, you can create the conditions that enable successful product development even in the most uncertain of times. V

Endnotes

1. See http://www.agilemanifesto.org
2. Private interview between Katherine Radeka and Mary Poppendieck, April 7, 2009, conducted for this article
3. Gregory D. Githens, "Using a Rolling Wave for Fast and Flexible Development" in *PDMA ToolBook 3*, New York: John Wiley & Sons, 2007: 397-415
4. Gregory D. Githens, "How to Use the Work Breakdown Structure to Define and Manage the Project's Work Scope," *Visions*, XXII no. 3. (July 1998): 20-23
5. Preston G. Smith, *Flexible Product Development*, San Francisco: Jossey-Bass, 2007: 188–189
6. Ibid, 32
7. See http://flexibledevelopment.com/resources.htm#Customers