Flexible Project Management: Extending Agile Techniques beyond Software Projects

Preston G. Smith, New Product Dynamics

Jeff Oltmann, Synergy Professional Services

Abstract

For the development of software products, agile techniques have revolutionized the field over the past decade, especially in uncertain or changing environments, which are just where innovation is most likely to occur. How can similar approaches be applied to other kinds of projects that face uncertainty or turbulence? This presentation will answer this question, concentrating on project management aspects of the project. Initially, we describe the environment that supports flexible development, including people and team factors, keeping options open, making decisions at the last responsible moment, and creating flexible processes. Then we show how flexible project management contrasts with mainstream project management, specifically how you manage changing requirements, plan a project that is certain to change, and how you manage risk in such a project.

Introduction

Over the past decade, agile methods have revolutionized IT and software development projects, resulting in impressive successes, especially in volatile or uncertain environments. However, software projects are only a small part of the project management world. Can the success of agile be translated into these other types of projects?

Non-software projects share many of the same challenges that agile has resolved for software projects:

- turbulent environments in which changes inevitably happen at the most unwelcome time
- unstable requirements that are never complete
- customers who don't know what they want until they see it
- technology that moves faster than the project can react
- nimble competitors who put the project manager in a continual catch-up mode

Despite the common challenges, agile techniques can't be translated directly into non-software projects. Agile depends on several unique characteristics of software, such as object technologies, automated testing, and the ability to make incremental changes quickly and at low cost. Non-software projects don't have the benefits of these characteristics; however, agile provides wonderful insight into understanding how to deal with chaos. This paper covers a set of tools inspired by agile but built from the ground up and specifically designed for non-software projects.

Based on the research into flexible product development techniques and inspired by a decade of agile experience, these practical tools will help you make your non-software projects more flexible so that you can:

- lead volatile projects in the chaotic world of true innovation in which change is inevitable
- accommodate emerging and changing requirements
- take advantage of changes and reduce the disruptiveness of changes
- be more responsive, roll with the punches, or actually lead the change!

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC

- create an environment for innovation and breakthroughs

Due to time constraints, we concentrate on the flexibility techniques most closely associated with project management. For a more complete set of tools for flexible product development, see the related book (Smith, 2007).

The Environment for Flexibility

Put the People and Team First

As usual, the people on your team are the most important factors in success. In a flexible project, this is doubly true, because team coordination and communication are what stimulate progress and, in a turbulent environment where nothing stays the same, coordinating those doing the work is doubly difficult. We know that you have already devoted great effort to building a strong team, but the team factor is so important to success in the midst of chaos that we cover some of the foundational principles right at the outset.

The Right People

Some people are comfortable walking on unstable ground and some aren't, which is partly due to personal style. Some people thrive under the excitement of constant change and some are uncomfortable with it, seeking the refuge of a plan and structure. Clearly, those who need stability in order to proceed will be ill suited to a changing environment and this should be recognized upfront by both them and the project manager.

But there is an experience factor involved, too. Alistair Cockburn, a leader in the agile software world, has identified three categories, which he calls "mastery levels" (Cockburn, 2002, pp 14–18):

- **Level 1: Following.** These people are able to and are comfortable with following a single specified method. They do not have the confidence or inclination to vary from this method or to choose between methods.

- **Level 2: Detaching.** They have seen the approach of Level 1 fail enough times to know that it is not always the best way to go; thus, they are capable of pulling away to some degree and considering multiple specified methods, but they still need a framework to follow.

- **Level 3: Fluent.** These folks have been around enough changing environments that they are able and willing to improvise and adjust to building what is needed without reference to a provided structure. In fact, they may become bored and do poorly if required to follow a specified plan.

Level 3 people are a scarce resource and should be placed carefully in a project that is likely to face change. Seed them in parts of the project in which you expect or need change in order to achieve your business goals and use them to bring Level 1 and Level 2 people up through the ranks.

Commitment and Dedication

These are two distinct but related qualities that are especially critical to success in the fog of change. We like to think of commitment as "skin in the game," that is, the participant has something significant to loose if the project fails. In today's "team" environment, especially in large teams in large companies, many specialists play minor roles, and these members' contribution is unclear because it is so small and because there are always other projects. This commitment factor is what distinguishes a high-performance team from an ordinary workgroup (Katzenbach and Smith, 2001).

Dedication, as we use the term, means full-time involvement on one project—a rarity and maybe even a luxury in many companies. Clearly, it is related to commitment because a dedicated team member is more likely to be committed to that project: more skin in that game. But there is a more important reason for dedication in a changing environment. When change is continual, a dedicated team member can keep up with the "latest news" as change occurs. One who isn't involved continually will fall behind, and others will waste valuable project time bringing this part-time member up to date. If the team has many part-time members, it also has a big updating burden.

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC

Adequate Authority

Countless decisions must be made in a project, and each of these requires a certain amount of authority to make such a decision. Exhibit 1 illustrates decision types connected with a product development project. This is a long list, and you could probably expand it even further. Clearly, the team needs a certain amount of authority to make decisions, or progress will be slow as it obtains management approval for each decision. This is especially critical in a shifting, foggy environment, in which decisions arise frequently, need resolution quickly, and require information that the team is more aware of than management, when circumstances are changing rapidly.

Financial Control	Select vendors and suppliers
Prepare project expense budget	Manage vendors and suppliers
Modify project expense budget	Operational Control
Prepare project capital budget	Select product features
Modify project capital budget	Modify product features
Use project capital budget	Determine product architecture
Authorize travel	Set reuse objectives
Pay for manufacturing variances	Make reuse decisions
Establish delegation limits	Make design outsourcing decisions
Cancel project	Prepare project schedule
Management of People	Modify project schedule
Prepare staffing plan	Select development location
Modify staffing plan	Determine layout of team work area
Select team members	Determine agenda of team meetings
Hire team members	Select development methods
Remove team members	Modify development methods
Evaluate team member performance	Select engineering tools
Determine team member compensation	Select test procedures
Determine team member bonuses	Modify test procedures
Provide recognition to team members	Determine test criteria
Management of External Relationships	Set documentation standards
Select key business partners	Select manufacturing site
Manage key business partners	Select manufacturing processes
Select key technology partners	Set quality standards
Manage key technology partners	Set manufacturing yield targets
Select outside contractors	Set management reporting requirements
Manage outside contractors	

Exhibit 1 – Types of Project Authority

There are two ways to use this chart. One use stems from our observation that often a decision is delayed because the organization is unsure about who should make it. Management assumes the team will handle it, and the team is waiting for management to make the decision and give the permission to proceed. To avoid such situations, look at this list—or, even better, create a similar list for your organization—and decide in advance with management whether the team or upper management has the authority to make each decision type.

The second application is for the team to consider each item on the list and pick out a few areas where it does not believe it now has authority for certain decisions but could proceed much more effectively if it did have such authority. Then it can discuss these areas with management in hopes of enlarging the team's authority in a few critical areas. A important factor to remember here is that not only can the team make faster decisions if they are made internally, but they will be better decisions because only the team has the freshest information in a turbulent environment. Also, team members will be more highly motivated to make its decision work if it is their decision rather than management's.

In the Same Space

Several factors are leading contemporary product development teams to become more geographically dispersed:

- Corporate operations and markets are becoming more global.
- Companies acquire new units in new regions.
- The competitive environment pushes companies to obtain the best talent from wherever it is located in the world.
- Similarly, economics suggests acquiring talent from wherever you can obtain it most economically.
- Communication technologies now allow better communication at a distance.

Much has been written about modern "virtual" teams, some of it emphasizing the opportunities for dispersion that technology is opening for us and some more realistically addressing the difficulties encountered in this new mode of operation (Duarte, & Snyder, 2006).

We work with development teams facing dynamic environments and repeatedly encounter the weakened and delayed communication that occurs as teams operate at a distance. This is a difficult issue, because there is some very good evidence for the value of co-locating teams (Smith, 2007, pp 141–146), but in contemporary product development teams it is increasingly difficult to do. For the same reasons, it is perhaps the most fertile area in which you, the project manager, can improve your team's performance. Let's cover some of these opportunities.

First, if your team is divided between metropolitan areas, co-locate members in the same metropolitan area, which means, that all cross-functional (engineering, marketing, manufacturing, supply chain, and other) functions on the team are within conversational distance (30 feet or 10 meters). Because product development decisions usually involve input or concurrence from different functions, having them all in one place speeds up and improves decisions greatly. If you are not able to do this for the entire project, try to do it for the critical initial phases of the project or for subsets of your team.

Analyze the communication patterns of your team by using directed graphs to understand where the heaviest communication links are, or should be, and then take the steps to ensure that these communication partners are co-located.

Finally, arrange your product's architecture to match your geographical dispersion so that the heaviest communication occurs within product modules being developed by a co-located team, and the interfaces between these modules simplify communication between teams at a distance. Detailed guidance on these co-location opportunities and on product architecture are available (Smith, 2007).

Apply Flexibility Selectively

Flexibility is not a universal blessing: It is a set of tools and techniques that can be applied to projects *selectively* to deal with uncertainty or the anticipated changes in a certain part of the project. The reason for this is that flexibility has its price, as we will discuss below. Consequently, development processes must be adapted to the kind, location, and timing of anticipated changes.

Flexibility Pays Off When Change is Frequent

Many managers are anxious about flexibility because it leaves loose ends, which seem to be open invitations for budget overruns and slipping schedules. We believe that flexibility, when properly applied, actually reduces the range of likely outcomes in a project when uncertainty is involved.

Here is an example. Suppose your firm markets bicycle components and currently you are working on a new wheel hub. There are two styles of hubs for spoked wheels: the so-called "narrow flange," where the hub's overall diameter is about 45 mm (1.8 inches), and the "wide flange" style, which is about 75 mm (3 inches). Popular belief is that wide flanges improve torsional stiffness (beneficial) but, in fact, engineering calculations demonstrate that

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC

narrow flanges provide plenty of torsional stiffness, so the extra flange simply adds weight and manufacturing cost (Brandt, 2002, pp 61–62). After some initial discussion, the team decides to proceed on this controversial point by applying good engineering judgment and developing a narrow-flange version. This is what the project looks like at this point and this is how it is budgeted and planned:

	Cost	**Time**
Develop and test narrow flange	$100,000	3 months
Total	$100,000	3 months

About two months into the project, after visiting some bike distributors, marketing decides that wide-flange hubs will sell better so they redirect the project from the narrow-flange plans (which are now sunk costs) so that it now looks like this:

	Cost	**Time**
Develop and test narrow flange	$70,000	2 months
Develop and test wide flange	$100,000	3 months
Total	$170,000	5 months

Now, the project is two months late and $70,000 over budget. However, when the team encountered this uncertainty, they could have operated more flexibly to avoid most of the project disruption.

In the planning stage, when the uncertainty about the flange arose, the team instead could have flagged it as an uncertainty and kept it open until it was resolved. In this case, the team could have built prototypes of the two configurations, or simply bought competitive samples of the two configurations and showed them to an assortment of customers, including the distributors. Then, when the market preference became clear, they could have proceeded with one design that would have been final. By delaying the decision on this uncertainty, the project picture now looks like this:

	Cost	**Time**
Prototype and show both options	$20,000	0.5 month
Develop and test the preferred one	$100,000	3 months
Total	$120,000	3.5 months

This approach costs a little more than the first one if the team happens to pick the correct option, but is much cheaper than picking the wrong option. This approach also has the advantage of greatly reducing the $70,000 and 2-month variance in outcomes between the first two pictures. In addition, the project would finish on budget and on schedule, because the prototyping would have been planned into it.

In summary, a relatively small upfront investment in prototyping and market research resolved an uncertainty that could have been very expensive later on. The value of this extra upfront insurance premium depends on how likely the uncertainty is. This is how the likelihood of uncertainty in your project should influence how and where you apply flexibility tools.

Plan to Keep Looping Back

A fundamental difference between agile software development and its traditional equivalent is that agile operates in tight loops (iterations), whereas more traditional methods make one long pass through a sequence of steps, which is called a waterfall process. Such iteration is not just an accident; it is an essential means of dealing effectively with

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC

change. The idea is to do a small but complete bit of work, demonstrate it to customers or customer surrogates, and then repeat this loop for the next bit of work. The iterations allow you to check regularly that what you are doing is heading toward your desired goal, even as the terrain under you shifts.

Iteration is an obvious feature of agile software development. Agile teams choose an iteration length, which can be from one to six weeks (typically, two weeks) and they use this pace like a metronome for the entire project.

For non-software projects, this is not so easy, because it is usually far more difficult to divide non-software development into small pieces that can be completed for demonstration in a few weeks. Just building a working model of the latest features repetitively on a cycle of a few weeks can be either impossible or prohibitively expensive. For non-software products, the cycles will be longer; nevertheless, it is essential to find a way to work iteratively and be able to demonstrate a working model of the product on a regular basis. Without this iteration, you will become stuck in a rut and unable to make changes easily.

Keep Critical Options Open

Sometimes making decisions early in a project is a good thing, because it increases the number of stable anchor points that the project team can use to make sense out of chaos. Clearly, having many loose ends leads to blown budgets and schedule slippage; however, making decisions early in a rapidly changing environment has an insidious consequence because it may unnecessarily position the project in a corner when things inevitably change. An important part of building and maintaining flexibility is to keep options that might change open, which tends to run counter to the way project managers think and are expected to act. Project managers are usually paid to make decisions and to prune unnecessary paths, leading to what seems like greater certainty in their projects. Fortunately, there is a middle ground, called the "last responsible moment," which allows project managers to establish sufficient stable anchor points by making early decisions, while deferring other decisions to retain maximum flexibility.

The Last Responsible Moment

This is a technique for identifying and keeping options open on critical decisions that might change later, such as the decision on hub flange-width, as discussed earlier, and this process is to

- identify a decision that is uncertain at the moment and that might change later as new information arises
- determine when this decision will have to be made to avoid incurring great consequences
- schedule this point as the *last responsible moment* for this decision
- start collecting information to help make a better decision by the time its last responsible moment arrives

Several conditions can determine when the last responsible moment occurs, such as an important option expiring or project cost rising abruptly at a certain point if the decision is not made (Smith, 2007, p 155); usually, the last responsible moment is the earliest time out of all of these conditions.

The last item above is critical for distinguishing the last responsible moment from procrastination. Procrastination is simply being lazy about making a decision—putting it off because this is the easiest thing to do. In contrast, the last responsible moment is an active process in which you are busy collecting information so that you will be as ready as you can be when decision time arrives.

Making decisions this way has two benefits. The first is flexibility. By definition, carrying a decision until its last responsible moment is not expensive, and it provides you with opportunities to change direction as late as possible without incurring unreasonable costs. Second, delaying a decision this way allows you to make a better decision when the time comes, because when you make the decision, you will be working with the freshest, most complete information available to making it.

Flexible Project Techniques for a Chaotic World

Managing a project in a flexible way violates many assumptions and behaviors that project managers usually bring to a project. The normal assumption, which is actually quite effective in a stable predictable environment, is that change is costly and leads to variations from the plan in project outcomes, so the wisest approach is to nail down everything connected with the project as early as possible and follow that route to project completion. The implied objective is to finish the project as close as possible to the original plan.

However, product and service development are about innovation, and innovation is about changes from the existing order of things. In short, if you aren't changing, you aren't innovating. Unfortunately for the project manager, much of the new information leading to changes arrives after the project starts:

- Designers find better ways to configure the product.
- Engineers discover that a new technology's claims are overstated.
- Marketers observe that a competitor is offering a feature they hadn't considered before.
- Customers, when trying a prototype, find it hard to use and thus suggest improvements.

Flexible project managers, in contrast, assume that change will happen and organize their projects accordingly.

Accommodate Unstable Product Requirements

Mainstream product development theory tells us that product requirements (often called specifications) should be derived from careful market research done early in the project and then fixed so that developers are not chasing a moving target. In a stable world, if the market research is done carefully, this approach should hold, and it will certainly be less disruptive than shifting requirements. Most product development textbooks advocate thorough upfront market research and "frozen" product requirements. However, in the real world of innovation, this static view of creation seldom holds, and, in fact, research shows that it *never* does (Smith, 2007, p 32). So, rather than assuming frozen requirements and being caught off guard when they inevitably change, why not plan for change?

In general, there are two ways for the project manager to deal with changing requirements. One is to specify the product at a higher level that is less subject to change, and the other is to keep in touch with customers and users throughout development in order to provide an early warning system.

Defining the product at a higher level means not defining detailed features of the product itself—details that are likely to change—but instead defining how the product will be used or the kinds of people who will use it. Such views of the product are less likely to change and they allow more flexible interpretations at the design level as designers move forward. Here, we can draw from the software development literature. For years, software developers have used a technique called "use cases," in which they define how the user will interact with the product in a step-by-step fashion, much like a recipe. Agile software development, which has focused on changing requirements, has broadened use cases to a technique called "user stories" (Cohn, 2004). Cohn observes that the critical part of a user story isn't the story itself. The story is really just a reminder to the developer to have conversations with real users when starting to develop that part of the product. Thus, the user story is a perfect example of making decisions on design details at the last responsible moment. An alternative to user stories for defining the product at a higher level is "personas," in which developers create detailed archetypes of customers, carefully culled from thorough market research, and then they design the product to suit these personas (Cooper, 1999).

There are many effective ways to keep in touch with customers during development, and we find that companies doing this well have found customer connections that are unique to the firm's business, market, and culture. For example, program managers at a manufacturer of complex naval electronic equipment send their development engineers to do "ride-alongs" with users of the equipment. In this way, the developers learn about what users really need but may not be able to articulate. In another approach, 3M applies a technique developed in academia called the "lead user method," in which you can predict tomorrow's changes by talking to advanced users today (von Hippel, et al., 1999).

Plan the Project Expecting Change

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC

Mainstream project management generally follows the premise that thorough planning upfront is the best way to run a project. In fact, in many organizations a complete, detailed project plan is required even to obtain approval to start the project. There seems to be comfort in having a detailed plan, even if it is fictional; from this, it should come as no surprise that many project managers spend the majority of their time at their desks, updating the plan as events change (we will have more to say about this use of time in the next section). When change is the norm in a project, there are more effective ways to plan and we cover two of them.

The first method is "rolling wave planning," in which only immediate activities are planned in detail, and the rest of the project is planned in much less detail (Githens, 2007, pp 397–415). As the project rolls forward, the detailed planning also rolls forward just ahead of it. Although this seems simple, there are two important things to keep in mind. One is that when you plan work grossly, you tend to underestimate because you miss time-consuming details. You can compensate for this bias by reviewing some past projects that were planned grossly to compare the estimated project duration with the project's actual duration to compute a compression factor that you can use to correct time estimates for future projects. The other thing to keep in mind is that gross planning and planning "on the fly" run counter to the culture in many organizations that take comfort in detailed plans. You may encounter resistance to this logical technique at many levels.

The second way of planning under heavy change is "loose-tight planning," which is the approach taken in agile software development. Agile projects are typically conducted in short iterations (often called sprints) of one to six weeks, typically two weeks. An iteration is planned just when it starts, and future iterations are left unplanned. During the iteration, the team follows their plan tightly, and the loose period between iterations allows all future work to be completely replanned by working from a list of desired product features that is re-prioritized between iterations. This is a radical application of the idea; Boeing, when developing their 777 airliner, used a more moderate approach, wherein they alternated between loose periods of design and tight periods of stabilization and integration.

Manage Project Risk Continually

Nearly all the abundant sources on project risk management suggest a procedural approach to managing a project's risks: first identify the risks a project faces, analyze and compare, then prioritize them, and, finally, take action against the most serious ones and monitor your progress against your risk resolution plans. This approach is most effective when there is a relatively stable project plan in which you can identify the project's risks. If the project plan is in flux, the procedural risk management approach will miss important risks that emerge during the project. Even a regular rescan for new risks, as good practice encourages, is unlikely to keep up with a fast-changing project.

In a turbulent environment, a procedural approach to risk management must largely shift to an intrinsic one. By *intrinsic*, we mean that everything you do to manage the project is done to manage its risk. You keep in touch with customers to manage the risk of requirement changes; you create your product's architecture to fence in areas of design changes (Smith, 2007, pp 57–84); you do lots of experimenting, testing, and prototyping throughout to understand what might change and by how much (Smith, 2007, pp 85–106); you staff your team with an eye toward resource shifts; you keep in touch with suppliers to foresee changes in your supply of components; and, you create team communication tools, such as daily stand-up meetings, as an early-warning system for unforeseen problems.

In a turbulent environment, the project manager's whole job is risk management and it helps to be a bit paranoid. Weick and Sutcliffe explain how this is done by observing people who constantly face unexpected situations (Weick and Sutcliffe, 2001). At this point, it should be clear that spending time at the computer continually updating the project schedule is the antithesis of this style. On flexible projects, the project manager should be out on the floor continually "taking the pulse" of the project while watching for tomorrow's changes.

This is not to say that there is no place for a procedural approach to risk management. Turbulent projects still have some risks that can be identified well in advance and that will probably persist for much of the project. A procedural approach should be applied to these projects, but the balance must shift mostly toward the intrinsic approach.

Summary

Although some of these techniques may not seem very radical, they amount to quite a different style than normal in the industry and will probably run into resistance from the organization's cultural norms. Such cultural changes are likely to be the most difficult aspect of shifting to a flexible style.

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC

References

Brandt, J. (2002). *The bicycle wheel* (3rd ed). Palo Alto: Avocet.

Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley.

Cohn, M. (2004). *User stories applied*. Boston: Addison-Wesley.

Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis: SAMS.

Duarte , D. L., & Snyder, N. T. (2006). *Mastering virtual teams* (3rd ed). San Francisco: Jossey-Bass.

Githens, G. D. (2007). Using a rolling wave for fast and flexible development. In A. Griffin & S. Somermeyer (Eds), *The PDMA toolbook 3 for new product development*, Hoboken, NJ: Wiley.

Katzenbach, J. R., & Smith, D. K. (2001). *The discipline of teams*. New York: Wiley.

Smith, P. G. (2007). *Flexible product development*. San Francisco: Jossey-Bass.

von Hippel, E., Thomke, S., & Sonnack, M. (1999, September–October). Creating breakthroughs at 3M *Harvard Business Review 77(5)* 47–57.

Weick, K. E., & Sutcliffe K.M. (2001). *Managing the unexpected*. San Francisco: Jossey-Bass.

Originally published as a part of 2010 PMI Global Congress Proceedings – Washington, DC