PMWorldToday

Published in *PM World Today* – October 2010 (Vol XII, Issue X)

PM WORLD TODAY – PM TIPS & TECHNIQUES – OCTOBER 2010

Agile Isn't Just for Software

By Jeff Oltmann

Not Directly Applicable

Agile isn't just for software developers, but applying it to other types of projects requires extra creativity.

Software developers have been making revolutionary changes to the way they run their projects. A set of development techniques collectively known as Agile has swept software development companies, unleashing productivity gains, improving schedules, and leaving satisfied customers in its wake. Boehm and Turner report that the agile projects they studied improved schedule and budget performance while maintaining quality, compared to traditional software development projects. (Smith, p. 24)

Agile techniques are especially useful when projects face challenges such as these:

- Turbulent environments inevitably cause changes to happen at the most unwelcome times
- Unstable requirements are never complete
- Customers don't know what they want until they see it
- Technology moves faster than the project can react

These challenges are not limited to software development projects. Many, if not most, projects face them. Can Agile be applied to these non-software projects, too?

This enticing idea turns out to be harder than it first seems. Agile depends on several unique characteristics of software, such as object technologies that isolate change, automated testing that allows frequent and early testing, and the ability to make incremental changes quickly and at low cost. Most non-software projects don't have the benefit of these characteristics, so Agile techniques can't be translated directly to them.

Seven Principles

However, Agile provides good insights that we can use to improve non-software projects. We can creatively adapt the following seven Agile principles to non-software projects.

1. Plan to iterate
2. Put people and the team first
3. Buy options to increase flexibility
4. Delay decisions until the last responsible moment
5. Accommodate unstable product requirements

6. Plan the project to be friendly to change
7. Manage project risk continually

For example, Preston Smith describes how Johnson and Johnson has adapted many Agile practices to its business of developing lotions, creams, shower gels, and soaps! (Smith, p. 26)

In the rest of this article, I'll explore the first item on the list – *plan to iterate*. I'll dive into the other items in future articles.

Sprints Increase Flexibility

Agile's most famous characteristic is rapid iteration. Traditional methods of running a project make a single long pass through standardized steps for managing the project. First, requirements are thoroughly defined, then the entire project is planned, the plan is executed, and finally the project is closed out. This one-pass method is very efficient for projects that are well-understood from their beginning, but it stumbles when change is rampant.

In contrast, Agile software teams divide long projects into many tight loops, often called sprints. Each sprint is two to four weeks long and contains all of the definition, planning, execution and testing needed to deliver a small, useful, and complete increment of the product. In software projects, the increment is often a new set of features, such as adding a shopping cart to an existing web site. The sum total of all increments, available at the end of the project, is the final product.

This iterative, incremental approach is an essential method for dealing with rapid change. The short sprints keep software projects flexible and allow them to change direction quickly in response to new information and frequent customer feedback.

How to Plan to Iterate

Unfortunately, short iterations are harder to apply outside of software projects. It is not obvious how to divide non-software projects into small increments that can be developed, tested and demonstrated to the customer within a few short weeks. Suppose your project is to design a new automobile. You can't decide every few weeks to add or change features such as anti-lock brakes, airbags, engine design, or body styling. These features take months or years and millions of dollars to design, fabricate, and test. They are also not very incremental. A partially functional airbag system is not very useful to anyone.

Nevertheless, software projects get so much advantage from using sprints that that it is worth finding creative ways to adapt the idea. Here are some approaches that my clients or project teams have used successfully on hardware projects.

1 **Build mockups frequently.** For example, automobile designers sculpt scale models of a new body style and variations on it.

2 **Use longer cycles.** If a new iteration every four weeks is impractical, consider a three or six month cycle. An electronic design team I managed used a cycle of about six months for some critical printed circuit assemblies and six weeks for features on an integrated circuit.

3 **Make rapid prototypes**. Mechanical designers can now buy a "3-D printer" from HP for about $15,000 that will create working plastic parts overnight. Electronics engineers can update a digital system in a day rather than months using field programmable gate arrays (FPGAs).

4 **Create in cyberspace.** Automobile engine designers routinely create new designs in a software simulator. The virtual engine runs in cyberspace, where the designers can iterate on refinements quickly and cheaply before committing a lot of time and money to build a physical engine. Similarly, Sikorsky is designing a new helicopter that can fly as fast as a plane, while still hovering like a traditional helicopter. The engineering team iterated on the design, including getting feedback from pilots, by spending hundreds of flight hours in a simulator before the first prototype ever left the ground.

Endpoint

A casual peek on the internet will show fierce debates between agilists and traditionalists. Agile techniques are not a panacea, but they have proven very effective in software development. The rest of us ignore their success at our peril.

Rapid iteration is just the first Agile principle you can adapt to non-software projects. I'll discuss more in future articles. In the meantime, get creative. How can you adapt the seven Agile ideas I've listed above to benefit your non-software project? Send me your ideas and experiences to include in future articles.

Further Information

Flexible Product Development, Preston G. Smith, Jossey-Bass, 2007

About the Author:

Jeff Oltmann, PMP

Author

Jeff Oltmann is principal consultant at Synergy Professional Services, LLC in Portland, Oregon (www.spspro.com). He is also on the graduate faculty of the Division of Management at Oregon Health and Science University. His specialties include strategy deployment, operational excellence, and project portfolio management. Jeff is a seasoned leader with over 20 years of experience managing successful technology programs. He ran the Program Management Office (PMO) and a $60M project portfolio for IBM's xSeries development facility in Oregon. Jeff's hands-on program management experience includes program budgets over $100M and worldwide cross-functional teams of over 100 members. Jeff welcomes your questions and ideas. You can contact him at jeff@spspro.com or read previous articles at www.spspro.com/resources.htm.