A Pacific Edge Software

White Paper

June 2005

AGILE PHILOSOPHY APPLIED TO PROJECT MANAGEMENT

Ernest J. (Ernie) Nielsen
Managing Director, Enterprise Project Management
Brigham Young University

pacificedge® SOFTWARE

Agile Philosophy Applied to Project Management

We keep running into those “mystery words” that confuse our conversations about IT Governance. You know, terms like:

Metrics

Risk Management

Project Management

Portfolio Management

Governance

Terms for which each of us has an opinion—and probably our own definition. Depending on what we have done in the past, what we have heard others are doing, what we think the boss wants or any number of implicit criteria, we have devised our own definitions and practices.

So what makes them mystery words? The cloud that falls over our eyes—the glazed look we get when someone asks us to speak intelligently about one of these terms. The insecurity that our thoughts are not organizationally valid, and maybe we are about to get "caught" in our stupidity. The speed with which the topic is changed! Or the sometimes passionate discussions evoked as we try to persuade others to adopt our definitions and practices.

Well, a new “mystery phrase” has found its way into our hectic management lives.

Agile

While working with both large and small organizations—newly formed IT departments, newly consolidated IT departments or longstanding bureaucratic IT departments—I have more than once encountered the phrase:

“We are an agile organization.”

And the cloud forms! The executives quickly direct me to the “practitioners.” The engineers/developers explain to me—with a glazed look in their eyes—that they only design the work for the next two weeks, which makes them agile in their response to their customers/users. The customers/users—looking at their feet while they explain—suggest that agile means lots of iterative behavior, lots of changes and lots of blame that they could never get their requirements right.

Going from one “agile” organization to another, I observe that this is an atmospheric phenomenon—the cloud covers the IT universe! I have seen as many interpretations of “agile” as I have IT organizations using it.

So what was I to do when I overheard one of our own development directors telling a colleague that we are an “agile” organization! Agile to whom? By what definition? Using what processes? And so what!?

And thus began my search for the true meaning of "agile." You don't need to know the details—I'm saving that for the movie. But I would like to share the outcome—the learning and proven practices I picked up along the way.

Where did Agile begin?

In the late 1990s and early part of the new millennium, there was independent but similar work going on in the areas of customer responsiveness, change management, rapid prototyping and other development-oriented behaviors in an effort to shorten development lifecycles and get more proactive participation from the end-user community. Adaptive Software Development was introduced by Jim Highsmith in the mid-1990s, examining the language and practices of complex adaptive systems. Ken Schwaber and Jeff Sutherland were busy creating the Scrum method of development. Ward Cunningham and Kent Beck had introduced Extreme Programming (XP). Alistair Cockburn had created and practiced a development methodology that accommodated frequent change, user input and team collaboration in today's global environment. Ivan Bartolo of Malta was coaching large international teams in effecting sweeping change using the Dynamic System Development Method. These concepts and many others, including Feature-Driven Development and pragmatic programming, were closely related, but the practices and methodologies resulted in pockets of different behaviors—even warring factions.

In February 2001 a group of 17 experts, including most of those named above, met together in Snowbird, Utah, to discuss their differences, identify their similarities and see if there was an opportunity for synergy. In the words of Alistair Cockburn[1]:

"We selected the word 'agile' to describe our intent and wrote the Agile Software Development Manifesto (which reads):

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - Individuals and interactions over processes and tools
> - Working software over comprehensive documentation
> - Customer collaboration over contract negotiation
> - Responding to change over following a plan
>
> "That is, while there is value in the items on the right, we value the items on the left more."

They subsequently formed the Agile Alliance, to which "outsiders" have been invited in the format of an Agile Roundtable. It has been my opportunity over the past months to participate in these roundtable events, in which we discuss the practical application of the Agile philosophy, citing specific incidents and needs in our various organizations.

What is Agile?

Agile is a philosophy. It is not a methodology or a process. It is a way of thinking, a guide to organizational behaviors, that supports an environment in which there is, or may be, much change. In response to the frequently changing environment, Agile encourages:

- Frequent customer interaction and collaboration
- Frequent delivery of products and services
- Iterative application of supporting processes

To accomplish these values, practices are encouraged and supported by underlying process. Specifically:

- Identify requirements and prioritize them based on need versus want.
- Group requirements so that interim tangible outputs can be delivered throughout the given schedule of the project (often called increments or packages).
- Don't lose track of the overall picture while you work on the interim deliverables.
- Use your supporting development lifecycle and project management process to accomplish each interim deliverable.

Let's discuss these individually, keeping in mind that it is a discussion. This article is a continuation of the philosophical brain share that Ivan, Alistair and I hold regularly. Any one of us might say it differently, but the concepts are sound—and the practices included in this article have proven effective in silo-based as well as strategically collaborative IT organizations.

Frequent Customer Interaction and Collaboration

To avoid the "throw it over the wall" practices we have all experienced, Agile encourages an environment of frequent, yet appropriate, consultation with the customer and end users. The exercise of this philosophy needs to be tempered with the proven practices that support effective organizational behavior.

Effective practices in customer interaction and collaboration were articulated and modeled well in a study done by Dr. Ray Leavitt[2], a respected professor in the Stanford University School of Engineering. Dr. Leavitt studied the purpose and value of assembling teams during the process of developing a tangible output—which is what projects are all about.

His research shows clearly that when human beings get together as a team, they design. So during the design phase of your development lifecycle, there should be lots of collaboration and interaction—maybe even lots of meetings! But interaction should be controlled during the build phase, when developers should be building according to the design.

During the development or build phase, remember the axiom, "Meetings are to fix the problems, not find them." Collect status using automated tools, so you can keep the team focused on developing. Bring the team together only if status reports indicate that something is not working as designed. If this occurs, it is time to design again—so it is time to get human beings together again.

Interaction and collaboration kicks in again during the testing phase. However, testing should be done according to test cases or use cases that were developed during the design phase. Collaboration takes place only when bugs are discovered or functionality is not as expected, even though the design was followed. "Meetings are to fix the problems, not find them."

Frequent Delivery of Products and Services

The philosophy of frequent customer interaction leads us to shorten the amount of time it takes to get things out. If we follow the concepts mentioned above, then customer/user involvement happens at the front end for design, minimally during the build and again at the end for testing and bug fixes. In an 18-month effort, there could be a large gap between design and test. Customers feel excluded, get nervous about what the developers don't know, begin creating internal expectations of failure and set up the big "Told you so!" at the end when something doesn't work just right.

Instead, break your 18-month effort into logical groups of requirements that have a tangible output. You might do this by user groups, by functionality or by geography. However you do it, you identify shorter increments of work—typically nothing longer than 90 days. At its completion each increment delivers the working product, tested, documented and ready for the predetermined audience.

This facilitates manageable collaboration between developers, designers, customers and users at least every 90 days, and typically more frequently. Do not, however, confuse this with the Extreme Programming concept of two-week timeframes to design and build individual features or functionality. The Agile philosophy encourages the grouping of requirements, based on business value and priority to the customer, and a logical separation of deliverables. This creates manageable user expectations, manageable periods of development and manageable interaction to design and fix.

Iterative Application of Supporting Processes

Probably the most abused concept in the Agile philosophy is the notion of iteration. I recently audited the development behaviors of an "agile" IT group to help them figure out why it wasn't working. They had carried the notion of iteration to an extreme, bringing everyone on the team together every week to rethink what was done last week and "guide" the team this week. To support this thinking, they had discarded their tried and true project management process, saying that Agile does not believe in the "waterfall approach" of project management.

Hogwash! In the world of organizational behavior it has been shown over and over again that good process enables individuals to apply their expertise to a situation. The absence of process diverts the attention of the experts to figuring out how to come up with the solution rather than focusing on the solution itself. A well developed process has these characteristics:

- **Step by step** — Order of work is identified. We know what's been done, where we are and what is next.
- **Iterative** — It's okay to repeat previous steps when more information is acquired or change occurs.
- **Self correcting** — As you repeat (iterate), your result is improved.
- **Scalable** — The process can be applied to large or small efforts without eliminating steps.

There are two overarching processes that you will use in an Agile environment, and they will be repeated during an incremented effort. First, the product development lifecycle. A software development lifecycle, for example, is typically composed of investigate, design, develop, test and deploy phases for the project work. You will follow all of these steps in each increment of the Agile project. Second, the project management process, which is typically composed of initiate, define, plan, execute and closeout phases. (Gasp! This is a typical waterfall process for project management.) Again, you will follow all phases of the process for each increment.

Herein lies the beauty of the iterative nature of the Agile environment. We learn from the iteration. We apply lessons learned to the next increment. There is an aggressive design exercise at the front end of each increment in which lessons learned from the previous increment are applied, and then a proactive closeout process at the back end, in which lessons learned are identified and acknowledged to be carried over to the next increment.

In the old world of project management, we called this “phase planning” or “program management.” So the concept isn’t new, it’s only refined.

How does Agile change our project management behavior?

A definition of the word “agile” is “characterized by quickness, lightness, and ease of movement; nimble.”[3] Because the project management process is repeated for each increment of an Agile project, it becomes important to fine-tune the process steps to ensure nimbleness and ease of movement from step to step. You will find it useful to define subprocesses, including:

- **Risk Management** — Quick identification, assessment and management of what might happen during the increment
- **Issue Management** — Quick identification and management of unanticipated or unresolvable items
- **Change Management** — Quick and appropriate management of changes to scope, schedule and resource parameters, with clarity around consequences to all stakeholders

In reality, you will find that members of project teams rely more on each other, rely more on the work done to date (avoiding the not-invented-here mentality) and rely more on the project manager as a facilitator than a subject matter expert. The project manager begins to be appreciated as an expert in process and facilitation.

And, closest to my heart, project management begins to be an organizational behavior, understood by team members and managers alike—sponsors and contributors. The incremental approach to developing project deliverables discourages dependence on individual "heroics" and brings the team together to share the heroics.

Has the Agile philosophy made a positive difference?

It has! At Brigham Young University, our first effort, after demystifying the concept of Agile development, found that proactively applying lessons learned to each increment focused the skills and resources available to us, and produced the desired outcome for $100,000 less than budgeted.

In another example, we are currently building an in-house system to replace a mission-critical university function. Having just completed another such project that ran well over budget and well over schedule due to a lack of requirements definition up front, we decided to break this new project into about a dozen increments based on functionality and users. We have focused individual user departments on their specific increments, so they no longer worry about the details of entire effort. We have focused development on priority features first. And we have taken our project team through multiple iterations of design, build and test.

Project team members have become agile in moving through the process. They have become agile in identifying needs versus wants. They have become agile in interactions with other team members. They have become agile in acceptance of project deliverables. They have become agile in testing with focus, no longer extrapolating module problems to the entire project. We are on time and under budget, and the customers and users, the developers and designers, and the project manager and coordinators are satisfied and looking forward to a successful implementation of a very large application—in small parts!

We are agile, and we have successfully built upon the same development lifecycle we've been using for three years and the same project management processes we implemented in 2001. The Agile philosophy has helped us:

1. Become more appropriately collaborative with our customers/users
2. Leverage the learned and practiced processes
3. Support our customers/users by being more reliable, consistent and predictable in our delivery of their requirements in support of their business objectives and values

No clouds over this project! Our little piece of the IT universe is satisfied for now.

Footnote — Watch for the upcoming book on organizational implementation of Agile development, by Ivan Bartolo and myself. We're sure a movie will follow!

1 Agile Software Development, Alistair Cockburn, copyright 2002 Pearson Education Inc.
2 Designing Successful Matrix Organizations, Dr. Raymond I. Leavitt, Stanford Advanced Project Management Program copyright 2001
3 The American Heritage® Dictionary of the English Language, Fourth Edition Copyright © 2000 by Houghton Mifflin Company. Published by Houghton Mifflin Company. All rights reserved.

Pacific Edge Software, Inc.
11100 NE 8th Street
Suite 600
Bellevue, WA 98004

425.213.1090
www.pacificedge.com