

## FIVE MYTHS OF AGILE DEVELOPMENT

Robert Holler

### INTRODUCTION

Agile development is an umbrella term for a number of iterative and incremental software development methodologies such as Extreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM), Lean Development and Feature Driven Development (FDD). Unlike traditional software development methods, agile development processes focuses all software stakeholders (i.e., programmers, testers, writers, customers, managers and executives) on the incremental delivery of working, tested software. With agile, all aspects of software development – planning, analysis and design, development, integration, testing, etc. – are combined in short, frequent iterations, with each iteration serving as valuable input into subsequent iterations.

Agile development has made its way into the software mainstream in the past few years. As business cycles continue to accelerate and competitive pressures increase, many software development organizations have turned to agile development methods as a way to accelerate delivery schedules, adapt to changing business demands, align business and technology goals and generate competitive advantage.

As more and more organizations turn to agile development, the reality of what agile really is often gets obscured.

As with any significant industry shift, a tremendous amount of fear, uncertainty and doubt generally accompanies the transition. Agile development is in fact a fundamentally different approach to planning and delivering software, and should by no means be taken lightly. Yet with more and more companies making the transition

and more and more project successes being communicated, the opportunity for everyone in the software community to understand and learn from agile development clearly exists. Agile is by no means a silver bullet, but it has forced many of us to rethink traditional approaches to software development and apply new ideas and techniques in an effort to perform our jobs as software professionals better.

As more and more organizations turn to agile development, the reality of what agile really is often gets obscured. Drawing from our experience with hundreds of teams around the world, including many of the Fortune 100, I want to dispel several common myths in order to help clarify what agile development is, what it is not and what to believe if you are considering the transition to agile development within your organization.

### CONTENTS

Introduction.....	1
Myth #1: Agile Development is Undisciplined.....	2
Myth #2: Agile Teams Do Not Plan.....	2
Myth #3: Agile Development is Not Predictable.....	2
Myth #4: Agile Development Does Not Scale.....	4
Myth #5: Agile Development is Just Another Fad.....	4
Summary.....	5

## MYTH #1: AGILE DEVELOPMENT IS UNDISCIPLINED

Some have referred to agile development as hacking, others as “code-and-fix”. Anyone who has seen an experienced agile development team in action would likely reconsider. Continuous integration, test-driven development, refactoring and even the controversial pair programming are not practices for the undisciplined. Even more indicative of the discipline required, the continuous delivery of running, tested software every few weeks could be characterized as the ultimate software industry discipline.

Many of the practices associated with agile development are not only demanding of the individual, but typically require a significant shared commitment to the success of the entire team. Within agile development, teams are constantly delivering benefit to the organization and demonstrating their value – not once a year, but every month, every week, and in some organizations, every day. While some software development teams may never achieve this level of delivery frequency, almost all can benefit from the practices that automate and facilitate this type of delivery discipline.

Interestingly, most of the practices associated with agile development have been around for decades. The real difference is that it is only more recently that the practices have been packaged together as collections of interdependent practices. These practices, whether incorporated within Extreme Programming, Scrum, Feature-Driven Development, or any other agile methodology, really help define the innovation which has taken place. Since their inception, a high degree of developer, team and management discipline has not only been promoted, but demanded by all of the various agile methodologies. The more agile a team, the more discipline they typically exhibit.

## MYTH #2: AGILE TEAMS DO NOT PLAN

This misconception generally relates to a lack of understanding of an agile, or incremental, planning approach. Most agile teams spend as much, if not more, time planning their projects. The difference is that this planning effort is spread throughout an entire project as opposed to being compressed into the beginning of a project. As opposed to extensive, up-front planning, agile development simply follows an incremental approach to planning, which allows for initially planning at a high-level

and iteratively planning at lower levels as more and more knowledge is gained.

If we look at the actual results of many traditional project plans, detailed, task-based project plans created at the beginning of a software project oftentimes quickly fall out of synch with the technical and business realities of a project. Over time, significant energy is expended in reconciling the original plans to these current realities. On the other hand, agile development accepts this volatility and instead replaces detailed, up-front planning with “continuous planning” – for example, quarterly at the release level, monthly or weekly at the iteration level and daily at the task and team-member level.

Given that the technology, requirements, business demands, people, issues, risks, etc. are almost always in flux. In virtually all software projects, this type of continuous planning approach provides the team with the necessary process-based ammunition to much more easily and efficiently adapt to the changes. In addition, teams are also able to incrementally optimize their plans as new information emerges.

What is just as important is that as teams begin to learn what they can deliver on a daily, weekly and monthly basis, they become more accurate at longer range planning. Many of the agile teams I’ve worked with have become much more confident in both their short- and long-term planning abilities than they ever were with traditional planning approaches.

## MYTH #3: AGILE DEVELOPMENT IS NOT PREDICTABLE

As with planning, agile development methods promote a fundamentally different approach to metrics and forecasting than traditional development. The approach of traditional development is one of creating detailed, activity-based plans upon which to evaluate progress, deviation from plan and the ability to predict precise outcomes. Plans created in this manner have unfortunately oftentimes only offered the ‘perception of predictability’. After decades of experience, the results associated with this type of planning on software development projects have proven dismal at times. Just as important, the predictions associated with these plans rarely improve over time because there is no simple mechanism in place for evolving the plan as new information becomes available.

# WHITE PAPER

Agile development instead replaces detailed, speculative plans with high-level, feature-driven plans that acknowledge the inherent complexity and uncertainty of software development projects. Ongoing reconciliation of actual effort to original plans is replaced with incremental planning and re-planning at a more and more granular level throughout the development process. Because agile development operates in a rapid, iterative fashion, valuable historical data quickly emerges for supporting both short- and long-term planning.

Using this historical information upon which to base future planning, the quality of plans quickly begins to improve. As speculation is replaced with historical fact, teams are able to predict with much greater accuracy what they will be able to deliver over time. Agile teams replace PERT and Gantt charts with simpler concepts such as Velocity and Burndown charts.

As shown in Figure 1, a Velocity chart (lower left), typically measured in terms of story points, ideal days or hours, displays the volume of features a team delivers each iteration. Over a short period of time, a team's Velocity will stabilize, resulting in a very reliable planning metric for future iterations.

The Burndown chart in Figure 1 (upper left) reflects both the amount of work remaining as of each day during an iteration as well as the overall change in scope of the work within the iteration. If the slope associated with the remaining effort, represented by the red bars, is not such that the work will be complete by the end of the iteration, this is immediately apparent to everyone on the project. Not only can the team act on this now, they can use this information to calibrate the amount of work put into their next iteration plan. Simple charts and metrics such as the Velocity and Burndown charts convey as much or more information than PERT and Gantt charts, but do so in a very visual and actionable manner.

As opposed to planning once a year and executing against that plan for the remainder of a project, agile teams are constantly planning, estimating, prioritizing and delivering against the plan. As teams continue to plan and re-plan throughout the development process, each individual's planning skills continue to improve. As a result, a team's confidence in its plans increases, providing all stakeholders with a much better sense of where the project actually is and what can be accomplished.

This type of accurate, black-and-white visibility into the development process also helps build trust throughout the organization.

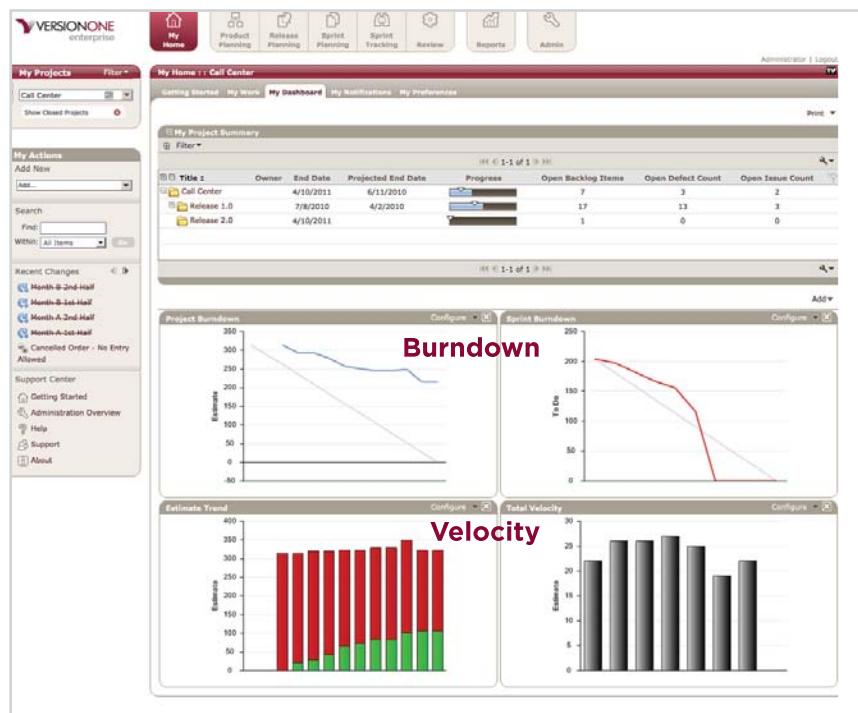


Figure 1: VersionOne Project Dashboard

## MYTH #4: AGILE DEVELOPMENT DOES NOT SCALE

Generally speaking, software development itself has scaling issues. There is plenty of evidence that would suggest this is clearly not a methodology-specific problem. The larger a project's scope, the greater the probability for failure; the greater the number of people involved in a project, the greater the communication risk and complexity. Agile development simply accepts these realities and recommends smaller projects, shorter delivery timeframes and smaller teams. Smaller teams have been proven time and time again to be much more productive than large teams.

This does not mean organizations should avoid solving large problems. Agile simply suggests that there is a different approach to solving the same problems. Agile methods promote taking large projects and breaking them down into a coordinated series of smaller projects staffed by smaller, cross-functional teams. The various teams' work is integrated at least every iteration in order to reduce risk and ensure functional and technical compatibility. There are clearly other processes that need to be instituted in order to facilitate communication, integration, architectural design and standards and decision-making amongst the teams, but these challenges have been solved before by many organizations.

Over the last decade, enough agile projects involving hundreds of people have been performed by multiple teams, in multiple locations, across multiple time zones to have a high degree of confidence in the ability of agile development to scale.

In fact, if a company has a very large, complex problem to solve, there are many reasons to prefer the use of an agile process in order to rapidly expose risks, prove business value early and to quickly institutionalize a highly disciplined approach to software development and testing.

## MYTH #5: AGILE DEVELOPMENT IS JUST ANOTHER FAD

Fads typically involve a great deal of hype with little sustained substance. In a span of less than five years, agile development has become the preferred development approach for many of the world's leading technology companies. Many of the industry's leading visionaries and practitioners have embraced and promoted agile development. And while five years ago agile development was primarily being adopted by small, collocated teams, many of today's adopters include large divisions and software organizations of enterprise IT.

A Forrester<sup>1</sup> report on agile development discusses the second wave of adoption that's now underway, with enterprise IT shops taking the lead. Results compiled within the report indicate that agile development processes are already in use in 14% of North American and European enterprises, and another 19% of enterprises are either interested in adopting agile or already planning to do so. Enterprise IT shops have found that by turning to agile processes they are better able to cut time-to-market, improve quality and strengthen their relationships with business stakeholders.

The broad-based acceptance of agile development is clearly a significant shift in the industry over the last few years. In a recent survey of VersionOne's own customers, the results corroborate those found by Forrester. The top three reasons noted for transitioning to agile development included accelerating delivery, aligning business and market needs with IT and improving visibility into the software development process. These are not the evaluation results of a fad, but instead business criteria based on real business results.

Finally, understanding agile team dynamics and collaborative decision-making techniques is important in part because agile requirements definition involves more than just the Business Analyst and the product owner. These skills enable the BA to accept input from all the team members, specify a more robust solution that meets the evolving needs of the business and help create a strong sense of confidence that the solution can be delivered to market.

## SUMMARY

Ultimately, every software organization will need to evaluate agile development methods itself in order to determine applicability and value. While some teams may determine that agile does not make sense in their environment, they should do so based on facts rather than the myths I've highlighted in this article. Regardless, every software professional owes it to themselves to understand the various benefits, risks and trade-offs associated with agile development. This type of transition can be good for an industry, and especially given the less than ideal success rates over the last decade, status quo is simply no longer acceptable by most companies.

With any new way of doing business, there is often a fear associated with the unknown. Fortunately, many software industry leaders and early adopters have been breaking down barriers and addressing the issues and challenges associated with agile development for five to ten years now. Throughout this process, and to everyone's credit within the industry, the true leaders and practitioners of agile development have continued to demand a high degree of discipline in each and every practice they promote, from test-driven development to continuous integration, to release planning and daily stand-ups. Teams that do not display this inherent planning, estimation, prioritization and delivery discipline are only putting on a façade of agility. The only ways teams can consistently deliver high-quality, working, valuable software every few weeks is through this discipline.

To many professionals who have been in the software industry for years and years, agile development is simply packaging together process, engineering and management practices that have been around for decades. The key evolutionary benefit being delivered is that the processes and practices associated with agile were designed to both adapt to and leverage the rapidly changing environment of so many of today's software development projects.

Even more indicative of the growing momentum within the software industry is that corporate IT is now driving the adoption of agile development on a much broader scale. Agile is definitely not just for small, co-located teams anymore. While accelerating time-to-market is serving as the primary catalyst for most companies transitioning to agile surveys show that many teams are

also benefiting from improved visibility, productivity, value, quality and an overall reduction of risk in the development process. The ultimate vote of confidence we hear is that a vast majority of adopters of agile development could not imagine returning to their old ways of building software, and that speaks volumes.

## AUTHORS' BIOGRAPHY

Robert Holler is president and CEO of VersionOne, recognized by agile practitioners as the leader in agile project management tools. Robert has more than twenty years of enterprise software experience, and for more than a decade, Robert has been VP of Development, CTO and CEO of several leading-edge enterprise software firms.

Robert's career in the software arena encompasses working as a consultant, developer, tester, technical writer, DBA, development manager and senior executive with organizations such as Andersen Consulting, Ockham Technologies, Tango Networks and Clarus Corporation. Through these positions and experiences, Robert became familiar with agile methodologies and iterative development processes early on, and it was his enthusiasm for agile software development that led to the founding of Atlanta-headquartered VersionOne in 2002.

*Reprinted from the May 2006 issue of  
Better Software Magazine*

<sup>1</sup> Forrester Research Report: "Corporate IT Leads the Second Wave of Agile Adoption" by Carey Schwaber with Richard Fichera

### ABOUT VERSIONONE

VersionOne is recognized by agile practitioners as the leader in agile project management tools. By simplifying the planning and tracking of agile projects, we help teams deliver better software faster. Since 2002, companies such as Adobe, Dow Chemical, Lockheed Martin, Motorola, Novell, Sony and Symantec have turned to VersionOne. Today more than 30,000 teams from over 170 countries use VersionOne.

Start small. Scale smart. See for yourself at [www.VersionOne.com](http://www.VersionOne.com).