Debunking Agile Myths

by Dick Carlson

It is often interesting and sometimes amusing how often people believe "facts" that are not true. Misinformation, confusion, and a lack of practical experience create challenging barriers to the acceptance and application of Agile and Lean, to software and systems engineering techniques. Since the Agile Manifesto for Software Development was developed in 2001, large and small companies alike have experienced an increase in productivity, improvements in employee satisfaction, and higher customer satisfaction when Agile practices are used. [1]

This article discusses and analyzes 10 of the most common misconceptions about Agile practices and techniques and identifies real-world examples that debunk these misconceptions. Included are actual examples of successful applications of Agile that have resulted in the development and deployment of deliverable software and other products within a wide range of businesses and industries.

The misconceptions to be addressed in this article include:

1. **Agile methods are undisciplined and not measurable.**
2. **Agile methods have no project management.**
3. **Agile methods apply only to software development.**
4. **Agile methods have no documentation.**
5. **Agile methods have no requirements.**
6. **Agile methods only work with small co-located teams.**
7. **Agile methods do not include planning.**
8. **Agile only works for small project teams.**
9. **Agile development is not predictable.**
10. **Agile development does not scale.**

AUTHOR'S NOTE: The company's name is protected by a very strict code of ethics regarding specific business practices and the identification of its customers. Therefore, references to the company, its customers, and the name of any program cannot be disclosed.

My apology to those readers who believe that a more important misconception has been omitted in order to include a lesser misconception. This list is based on my own personal experiences and opinions.

Applications of Agile

The application of Agile to projects has resulted in higher success rates over many years because of its specific practices and principles, including:

—Simplicity.
—Short iterations.
—Embracing change.
—A sustainable pace.
—Customer satisfaction.
—Verifiable collaboration.
—Daily stand-up meetings.
—Prioritizing requirements.
—Continuous improvement.
—Time-boxed work sessions.
—Close customer collaboration.
—Frequent planning and estimating.
—Frequent releases of working features.
—Product demonstrations and artifact reviews.
—Motivated, self-organized and cross-functional teams.

Agile Myths

The following are myths collected from my personal experience and from the experience of others. All are presented in a relative order of popularity.

1. Agile methods are undisciplined and not measurable.

This myth is among the most common and the most damaging when attempting to explain an Agile approach to an uninformed executive or customer. Though many people attempt to find discipline and predictability in traditional projects, these projects often involve extensive processes, excessive meetings, and extensive data management that do not achieve the desired discipline and predictability.

Agile projects use a different approach that involves a disciplined method with a diminutive focus. Some people are unaware that a "disciplined method" is involved. Agile projects are conducted through a series of short iterations (aka "sprints"), each of which requires teams to:

—Establish a goal or set of goals.
—Make a plan and determine what to build during each sprint.
—Estimate the effort required to build each item.
—Decompose the items into quantifiable tasks.
—Establish a team commitment to sprint goals.

This means that team members, sponsors and managers must commit to the sprint and that team members are expected to maintain focus throughout the duration of the sprint. Now this is discipline!

The evidence debunking this myth includes the completion of a multitude of projects that have implemented Agile rigorously with positive, measurable results. Further evidence in the VersionOne 10th Annual State of Agile Survey revealed that the top three benefits of Agile include:

1. An ability to manage changing priorities: 87 percent.
2. Increased team productivity: 85 percent.
3. Improved project visibility (transparency): 84 percent.

The survey also showed that the leading causes of failed Agile projects during 2015 were:

1. A company philosophy or culture at odds with core Agile values: 46 percent.
2. A lack of experience using Agile methods: 41 percent.
3. An absence of management support: 38 percent.

Three of the leading reasons indicated by "How Success is Measured with Agile Methods" included:

1. On-time delivery: 58 percent.
2. Product quality: 48 percent.
3. Customer/user satisfaction: 46 percent.

2. Agile methods have no project management.

This myth was likely formed by those unfamiliar with Scrum, which is an Agile project management approach that focuses on delivering the highest business value first. Scrum is implemented through a series of short sprints (two to four weeks in duration), with each sprint producing an increment of potential functionality. After each sprint, decisions are made to either

release the product as is or to continue enhancing the product until it is ready to be delivered or deployed. Scrum is simple and straightforward, and its activities, practices and rules are few and easy to learn. Scrum minimizes project planning because team members select their own work and self-organize.

While most projects have a project manager or someone responsible for project requirements and expenditures, the evidence debunking this myth is supported by the global implementation of Scrum. Scrum is an Agile project management framework that includes three roles (the development team, the product owner, and the Scrum master), four activities or working sessions (sprint planning, the daily Scrum or standup, the sprint review, and the sprint retrospective), and a few work products including the product backlog, sprint backlog, task board, burn-up chart, operational feature, and/or working software. The Scrum master facilitates all working sessions, arranges facilities, ensures all team members are fully functional and productive, and makes certain that everyone follows the agreed-upon Scrum process. The product owner collaborates closely with stakeholders for release planning and manages the product vision, project requirements, product road map, and product backlog. The team plans for, builds and manages chosen deliverables during each sprint.

Further, problematic processes do not achieve wide acceptance. Agile methods continue to receive wider acceptance.

3. Agile methods apply only to software development

This myth is most likely based on a combination of the principles expressed in th e Agile Manifesto for Software Development used in most Agile projects. [2] Only three of these principles apply specifically to software development. The remaining nine principles are generic and apply to just about any kind of project, circumstance, or situation that must be managed. In effect, the remaining principles can be used in anything we do. Whether they are applied on the job, at home, or in our communities, a majority of Agile principles can be used in almost any scenario.

One case with evidence debunking this myth is how quickly a five-day, comprehensive Agile-training course was developed in 60 days when it was originally anticipated to take six months. The project started with six people developing a 12-module course. After the first few weeks, the training deck contained more than 550 slides, many of which were redundant. As most team members were pulled back into other projects and support diminished, it became apparent that our small team of three had to rethink its approach. The solution was a no-brainer. We applied Scrum practices that reduced the six-month effort to 60 days and reduced the training deck from more than 550 slides to 350 slides. When presented, the course was very effective, and the course developers and instructors received kudos for the quality of the training.

Another example is a large project that used the Scrum project management approach. At the beginning of the project, it had failed to create several critically needed product backlogs across several engineering domains that should have contained enough user stories to define the operational capabilities and functionality of the application under development. Thus, several domain-related sub-projects were established. Each of these sub-projects planned and executed a series of short sprints, each with a goal to develop requirements in user story format and create product backlogs so software development teams could develop software and write tests based on those requirements. The results of this effort produced hundreds of user stories that were written and prioritized in each of the product backlogs so that the software development teams were able to use the backlog to build a wide array of functionality.

Another example is a project on a large defense program that was chartered to develop detailed specifications for equipment racks that was outsourced to an offshore company to build test benches. Management knew that projects of this type typically ran over budget and failed to meet schedules, so they looked to the software engineering effort that was using Scrum to better manage the project. Project stakeholders decided to use a seasoned Scrum master and identified a product owner who possessed the requisite technical domain knowledge, understood the challenges of the project, and was available to support the project's life cycle that was limited to 12 months. The result was that the project finished ahead of schedule by four months and saved 20 percent of the projected costs.

4. Agile methods have no documentation

This myth was formed from a misunderstanding of the Agile Manifesto principle that states, "Working software over comprehensive documentation." To some, this means "no documentation," but to others it means that documentation specified by contract will be developed and delivered as agreed upon, but will be developed incrementally — just like other work products specified in the contract.

Evidence that debunks this myth comes from hundreds of projects where contractual documentation was developed and delivered (or is in development and will be delivered). These documents follow the same life cycle pattern as all other project deliverables except that they are delivered incrementally to avoid the risk of developing product features that may exceed cost thresholds or never be used.

Scrum is used to manage hundreds of projects that are required to develop suites of documentation, including software and systems requirements specifications, software and system design specifications, software test descriptions, system validation descriptions, software and system user manuals, training manuals, and other documents. This is not unusual, and most of these projects are capable of preparing these documents incrementally while the products are being built.

5. Agile methods have no requirements

This myth may be a consequence of misunderstanding the Agile Manifesto, or it may have been originated by someone with limited software or system development experience.

The evidence against this myth is that all software and system development methods start with the definition and development of requirements. After all, what does one build if there are no requirements? For the most part, Agile projects — especially those implementing Scrum or Extreme Programming — convert known requirements into user stories, which are descriptions of desired functionality or an aspect of a feature told from the perspective of the user or the customer. Stories focus on the "what," not the

"how," and shift the center of attention from writing to discussion. Stories provide a quick way of handling customer requirements without having to create formalized requirements documents and without the administrative overhead required to maintain them.

User stories are meant to collect performance behavior and to be able to respond to changes in requirements faster, more accurately, and with significantly less overhead. User stories are controlled and managed in a product backlog, which is a prioritized list of requirements in a story format. During sprint planning, the team selects the customer's highest priority items from the product backlog to be completed during the sprint. Selected backlog items are then decomposed into quantifiable tasks, completed by team members, and verified as complete by the product owner. User stories also give the advantage of highlighting the value of a requirement or feature. Knowing the value of each story assists with ranking the stories.

There are many projects that have used Scrum to define and develop requirements within a short time period in order to avoid costly program delays. Many projects begin with immense challenges where customers require short product turnarounds and teams struggle with the logistics of project preparations, including determining the appropriate stakeholders (such as domain and subject matter experts), staffing, facilities, timing, requirements resources, and so on. The application of Scrum during project execution greatly increases the potential of project success by setting up and synchronizing multiple teams to define and develop requirements with software and product development teams. A viable approach to the execution of Scrum on such projects is depicted in the diagram below.

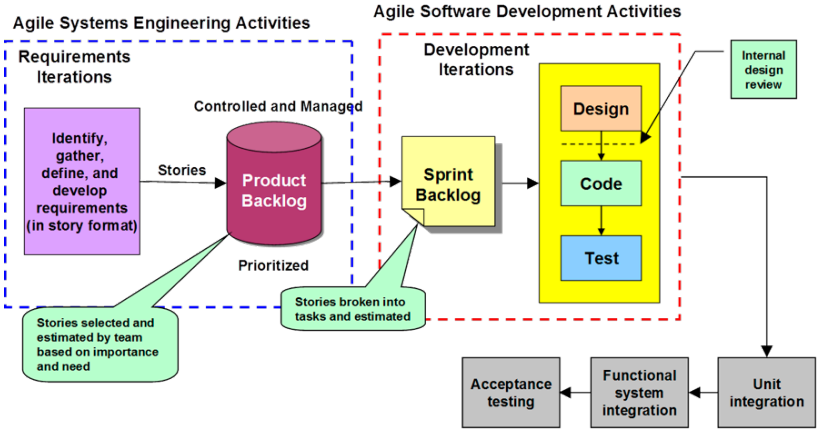

Figure 1.

6. Agile methods only work with small, co-located teams

This myth was likely started by people who cannot imagine what it would be like if project members were dispersed or distributed. Actually, the attitudes of some of the original Agile Manifesto authors believed this myth at first. Most have learned through personal experience. With the tightening of travel budgets and popularity of working with offshore or otherwise distributed teams, this is a reality being lived and experienced every day for many projects around the globe. Situations vary, but difficulties that come with distributed teams are similar regardless of the circumstances. For example, all distributed teams face unique challenges that arise from the loss of rich social and physical interactions. Another challenging aspect is the transition from a face-to-face environment to an environment where video conferencing, instant messaging, email, shared work areas (repositories), shared calendars, and workflow and collaboration tools replace the convenience of a physical working environment. In spite of these challenges, we find ways to adapt and perform.

At one time this myth was more true than false. The Agile Manifesto was established in February 2001. In the 16 years that have passed since then, our communication capabilities have increased immensely. The capabilities of WebEx, Skype, and GoToMeeting barely existed in 2001. We have learned and experienced much since 2001. While face-to-face communication remains the best method, new tools make telecommuting feasible and cost effective. A team in Burbank, California, can work with a team in Bangalore, India. It has been done effectively.

7. Agile methods do not include planning

This myth may come from a major misunderstanding of the Agile Manifesto, or perhaps from a rumor of an Agile project gone bad. Were backlogs and parking lots created? Were burn-down and burn-up charts used? Were daily stand-ups held? All of these artifacts and activities are part of planning and controlling.

Rumors result when those who do not practice Agile or do not understand it form negative opinions of activities without empirical experience. Poor Agile projects include those that failed to yield favorable results.

Evidence against this myth is in the application of Scrum on Agile projects. Scrum includes a significant planning effort that begins with the creation of a vision, a product road map, some release planning, and some sprint planning. Creation of the vision, product road map, and release planning are conducted during Sprint Zero (aka Iteration Zero, initial planning, or something similar) prior to any project activity and is not usually time-boxed like other Scrum activities. Sprint Zero is a term meaning a planning session that takes place before project execution. Sprint Zero is executed at a high level and may include the customer, end-users, and select project and support personnel.

Sprint planning is conducted on the first day of every sprint and is typically time-boxed at four to eight hours. In sprint planning, the product owner and the team determine what must be completed during the sprint. The team estimates all items targeted for a sprint's completion, selects a certain number of product backlog items, then determines how to build those items during the sprint. Clearly, this is planning!

8. Agile only works for small project teams

This myth is unfounded as projects using Agile practices are conducted every day around the world by both small teams and, many times, by much larger teams.

The evidence against this myth is the lack of knowledge and empirical experience on the part of the myth's originators. The Scrum framework recommends small teams of five to nine self-organized and cross-functional people who possess software designing, coding, and testing skills. Teams staffed with

10 to 15 systems and software engineers are quite common, although it is recommended that larger teams be divided into smaller, synchronized teams.

Further evidence of this unconfirmed myth can be explained by the Disciplined Agile Delivery (DAD) life cycle model, where the focus is on the delivery portion of the system life cycle from a project's beginning to its release. [3]

9. Agile development is not predictable

This myth is debunked by the fact that many of the teams that use Agile practices and principles every day experience a significant increase in productivity and work output due to their strong commitment and focus. Regardless of the challenge or product being developed, planned, and estimated, the end goal is always the same — a potential increment of working software at the end of each sprint.

Evidence debunking this myth includes the following:

- –Agile efficiently and effectively 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 with smaller granularity throughout development activities. Reduced granularity results in increased insight.
- –Agile development operates in a rapid, iterative fashion, so valuable historical data quickly emerges for supporting both short- and long-term planning.
- –The use of simple charts and metrics that show velocity and task burn-down that emphasize visual and actionable effects convey as much or more useful information than PERT or Gantt charts.

10. Agile development does not scale

This myth does not consider that, generally speaking, software development has scaling issues. Evidence against this myth is that this is not a method-specific problem. The larger a project's scope, the greater the probability of failure; the greater the number of people involved in a project, the greater the communication complexity and, therefore, the risk. Agile development accepts these realities and recommends smaller projects, shorter delivery times, and smaller teams. Smaller teams have proven to be much more productive than larger teams.

Agile methods promote taking large projects and decomposing them into a coordinated series of smaller projects staffed by smaller, motivated, self-organized, and cross-functional teams. Work output is integrated at least every sprint in order to reduce risk and ensure functional and technical compatibility. The recommendation is to take, where possible, a large project and decompose it into smaller subprojects. Creating these smaller subprojects increases the probability of success.

For many years, a significant number of Agile projects involving hundreds of people have been conducted by multiple teams, in multiple locations, and across multiple time zones while experiencing a high degree of confidence in the ability of the Agile development process to scale appropriately. If a com-

pany has a very large, complex problem to solve, there are many reasons to prefer the use of an Agile approach to expose risks quickly, demonstrate business value early, and institutionalize a highly disciplined approach to software development.

For more than three years, I assisted and coached a very large program within a very large, multi-national company. My assistance helped to transform the company's traditional development methods to Agile. The results were a reduction in schedule by more than 60 percent and a cost reduction of just under 40 percent. The organization consisted of 5,000 personnel that included software engineers, system engineers, and a large group of highly skilled assemblers and construction technicians.

Conclusion

As previously stated, the aforementioned myths are collections of Agile myths that have been debunked through personal experience. Further searches on this topic will reveal different myths reported by others in the field. To recap the myths reported in this article:

"Agile methods are undisciplined and not measurable" is disproved by the multitude of projects, past and present, that depend on commitment and focus on the part of everyone involved. It is also disproved by measured evidence in terms of cost and time savings.

"Agile methods have no project management" is disproved by the fact that everyone committed to an Agile project is taught to be self-managing. It is also disproved by the many projects that are using Scrum as a project management approach.

"Agile methods apply only to software development" is disproved by the many projects around the world that are using Scrum to manage the development of products unrelated to software. I have coached and facilitated numerous teams that produced nonsoftware products.

"Agile methods have no documentation" is disproved by the fact that documentation is produced on virtually all projects, regardless of whether it is contract-specified or project-generated. Essentially all of the teams that I have coached and facilitated over that last 15 years have produced value-added documentation.

"Agile methods have no requirements" is disproved by its ludicrous implications. Products are not developed through osmosis; products are developed from creative thinking that is defined and developed through extensive analysis and documented as specific requirements.

"Agile methods only work with small, co-located teams" is debunked by the fact that there are hundreds of Agile projects functioning with geographically distributed team members. I have coached and facilitated many teams that were not co-located.

"Agile methods do not include planning" is debunked through planning sessions that take place at the start of every sprint, during the daily stand-ups, and prior to project start during Sprint Zero, where all project and release planning activities are conducted. Projects that do not plan are doomed to fail.

"Agile only works for small project teams" is debunked by my own experience. I have coached and facilitated teams as small as two people and as large as 13 people. Very small teams are productive but cannot complete as much as larger teams can. However, I maintain through experience that teams of five to seven members are by far the most productive when developing software products.

"Agile development is not predictable" is debunked by evidence of progress and success throughout an Agile project's life cycle through rigorous, rapid, and incremental planning, iterative and evolutionary development, and the application of simple metrics.

"Agile development does not scale" is debunked by the evidence that smaller teams are more productive than larger teams, and through Agile's highly collaborative na-ture, identifies issues and risks early and drastically reduces their impacts through viable mitigation strategies

REFERENCES

Manifesto for Agile Software Development (http://agilemanifesto.org/).
Scaled Agile (http://www.scaledagile.com).
Scott Ambler (http://www.ambysoft.com/scottAmbler.html).
Software Engineering Institute, Carnegie Mellon University.
10th Annual State of Agile Survey, VersionOne, 2015.
Systems & Software Technology Conference (SSTC), 2011, Salt Lake City, Utah, May 2011.
National Defense Industry Association (NDIA) Systems Engineering Conference, October 2010, San Diego, California.
Systems & Software Technology Conference (SSTC), 2010, Salt Lake City, Utah, April 2010.

Recommended Reading:
"Agile Software Development with Scrum," Ken Schwaber and Mike Beedle, Prentice Hall, 2001.
"Questioning Extreme Programming," Pete McBreen, Addison-Wesley, 2003.
"Death March," Edward Yourdon, Prentice Hall, 2nd edition, 2003.
"Agile & Iterative Development," Craig Larman, Addison-Wesley, 2004.
"Agile Project Management," Jim Highsmith, Addison-Wesley, 2004.
"Implementing Lean Software Development," Mary and Tom Poppendieck, Addison-Wesley, 2007.
"Succeeding with Agile: Software Development Using Scrum," Mike Cohn, Addison-Wesley, 2010.

NOTES

1. The Agile Manifesto appears on the Agile Alliance website at http://agilemanifesto.org/
2. http://www.agilemanifesto.org/principles.html
3. The DAD model was hypothesized by Scott Ambler, who believed that Agile needed to extend beyond construction and into production. https://www.ibm.com/developerworks/community/blogs/ambler/entry/disciplined_agile_delivery_dad_lifecycle14?lang=en

ABOUT THE AUTHOR

Dick Carlson has a Bachelor of Science degree in business management. He has held certifications as a Scrum Professional, Scrum Master, and Scrum Product Owner, and in Lean-Agile Project Management. He is an accomplished software engineering process analyst, and has shared successful experiences of Agile, Lean, and Scrum implementation at conferences, workshops, and symposia. Dick's engineering career spans more than 40 years, and he has taught courses in mathematics, electronics, CMMI, configuration management, data management, Agile, Lean, and Scrum for more than 30 years.
dcarlson@iascar.us