Agile Software Development Methods

By Dick Carlson

ALEA

Agile & Lean Education Associates

"Teaching Agile & Lean from A to Z"

Introduction

Agile software development methods (ASDM) consist of a group or a set of software development methods where requirements and solutions evolve through a close collaboration between self-organizing, cross-functional teams. ASDMs promote adaptive planning, evolutionary development, early delivery, continuous improvement, and encourage rapid and flexible response to change. ASDMs are conceptual frameworks that focus on delivering working software with the minimum amount of work.

Most ASDMs provide opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular rhythms of work, known as iterations or sprints. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, ASDMs are described as “iterative” and “incremental.” Development teams using traditional methods only have one chance to get each aspect of a project right.

The methods defined below have been used by software experts for years with varying degree of success. The only methods of exception are Adaptive Software Development and Scrum.

AD - Agile Database Techniques

The goal of the Agile Data (AD) methodology is to define strategies that IT professionals can apply in a wide variety of situations to work together effectively on the data aspects of software systems. This isn’t to say that AD is a “one size fits all” methodology. Instead, consider AD as a collection of philosophies that will enable IT professionals within your organization to work together effectively when it comes to the data aspects of software-based systems.

AM - Agile Modeling

Agile Modeling (AM) is a practice-based methodology for effective modeling and documentation of software-based systems. Simply put, Agile Modeling (AM) is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and lightweight manner. Agile models are more effective than traditional models because they are just barely good enough, they don't have to be perfect. You may take an agile modeling approach to requirements, analysis, architecture, and design.

ASD - Adaptive Software Development

The practices of ASD are driven by a belief in continuous adaptation-a different philosophy and a different life cycle-geared to accepting continuous change as the norm. In ASD, the static plan-design-build life cycle is replaced by a dynamic *Speculate-Collaborate-Learn* life cycle. It is a life cycle dedicated to continuous learning and oriented to change, reevaluation, peering into an uncertain future, and intense collaboration among developers, management, and customers.

Crystal

The Crystal family was recently introduced by Alistair Cockburn [2]. It is a set of AMs covering a broad range of needs in terms of the number of developers and project criticality, which may range from C (loss of comfort), to D (loss of discretionary money) and E (loss of essential money). Life-critical projects are not yet covered in these methodologies.

Crystal methodologies are described in terms of mandatory policy standards, which are very synthetic. Implementation details are left entirely to the discretion of the team. A specific methodology for Web development, targeted at 40 to 50 developers and with criticality E, is also presented (Crystal Orange Web).

Crystal methodologies are even less prescriptive than Scrum. They explicitly take into account the project size and criticality, and this could be a plus for project management. At the time of writing this chapter, however, they are less documented than XP and Scrum.

FDD - Feature Driven Development

FDD starts with the creation of a domain object model in collaboration with domain experts. Using information from the modeling activity and from any other requirements activities that have taken place, the developers go on to create a features list. Next a rough plan is drawn up and responsibilities are assigned. Then small dynamically formed teams develop the features by repeatedly performing design and to build iteration that last no longer than two weeks and are often much shorter.

DSDM - Dynamic Systems Development Method

DSDM is more a framework than a method. The project process has seven phases, which are repeated during the lifecycle of the project, making DSDM into an iterative and incremental method. The phases are:

1. Pre-Project phase -- ensures that only the right projects are started and that they are set up correctly.
2. Feasibility Study -- assessment of whether DSDM is the right approach for the project and a definition of the problem, assessments of the costs and assessment of the technical feasibility of delivering a system.
3. Business Study -- the prime focus of attention is on the business processes affected and their information needs. Using a series of facilitated workshops to quickly gain consensus as to the priorities of the development.
4. Functional Model Iteration -- refining the business-based aspects of the system.
5. Design and Build Iteration -- is where the system is engineered.
6. Implementation -- covers the cutover from the development environment to the operational environment.
7. Post-Project -- this phase keeps the solution operating effectively.

The Underlying Principles

1. Clear definition of roles and responsibilities in the project
2. Active user involvement is imperative with four defined business roles
3. The team must be empowered to make decisions
4. The focus is on frequent delivery of products, which are defined in terms of purpose and quality and who's involved in the delivery
5. Fitness for business purpose is the essential criterion for acceptance of deliverables
6. Iterative and incremental development is necessary to converge on an accurate business solution
7. All changes during development are reversible
8. Requirements are baselined at a high level
9. Testing is integrated throughout the life-cycle

10. Collaboration and cooperation between all stakeholders is essential

Lean Software Development

In the last 25 years, lean thinking has had a tremendous global economic impact and transformed how many industries operate by changing the way they think about their work and their people. Lean works because it produces more wealth by eliminating waste, enabling any given investment of people and productive resources to produce more value while providing more meaningful, fulfilling work to the participants. *Lean Software Development* applies lean thinking to software development contexts.

Lean is not about what a team does – practices – it is about how a team decides what to do and when to do it. *Lean Software Development* starts at the level of principles derived from W. Edwards Deming who really understood the scientific approach. The application of lean thinking principles is different in each domain and it maps differently to different software development contexts as well. Practices appropriate to lean manufacturing or lean construction will not necessarily be appropriate to software development.

The lean thinking principles (adapted to the software context) are:

1. **Eliminate Waste** - If the customer does not value it or if it slows down rapid delivery to the customer, it is waste. Don't do it.
2. **Amplify Learning** - Development is about discovery and feedback. Deliver in small batches to minimize uncertainty and permit the customer to steer.
3. **Decide as Late as Possible** - Delay commitment. Keep options open as long as possible to base decisions on the best possible information.
4. **Deliver as Fast as Possible** - The best measure of organizational maturity is the speed with which it can repeatedly and consistently deliver value.
5. **Empower the Team** - Teams should design their own processes. Provide the training and leadership they need.
6. **Build Integrity In** - Concurrent development enables rich communication, the essential ingredient for system integrity.
7. **See the Whole** - Decomposition leads to sub-optimization. Focus on overall results.

Lean Software Development is not a development methodology; rather, lean is a way to think about whatever approach a team uses. Lean Software Development thinking tools support team decisions about which practices are appropriate in their unique context. Agile methods, including Extreme Programming, Scrum, DSDM, and Adaptive Software Development, are all consistent with the way *Lean Software Development* applies lean thinking to developing software.

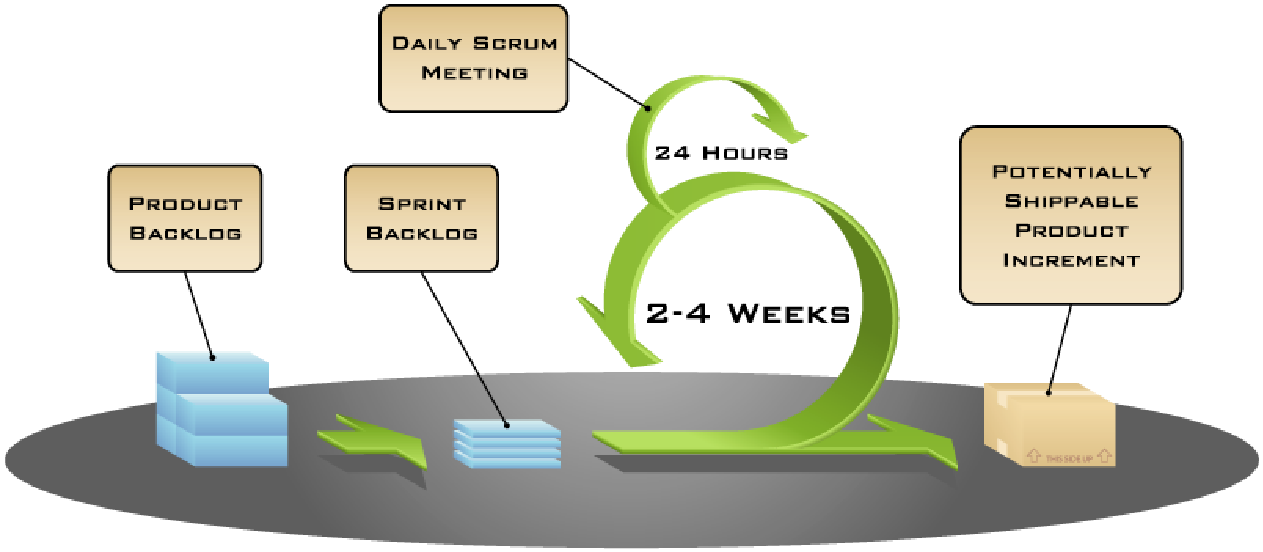
Scrum

Scrum is an Agile approach project management that includes control software and product development. Wrapping existing engineering technical practices, including those of Extreme Programming such as Test-Driven Development, Automated Testing, and Continuous Integration, Scrum generates the benefits of Agile software development with the advantages of a simple implementation. Scrum significantly increases productivity while facilitating adaptive, empirical systems development. The simplicity of Scrum includes its

The history of Scrum started in 1986, when Hirotaka Takeuchi and Ikujiro Nonaka described a new approach to commercial product development that would increase speed and flexibility, and the approach was based on case studies from manufacturing firms in the automotive, computer, photocopier, and printer industries. They called this the *holistic* or *rugby approach*, as the whole process is performed by one cross-functional team across multiple overlapping phases, where the *scrum* (or whole team) "tries to go the distance as a unit, passing the ball back and forth." This was another way of describing concurrent planning and execution.

In the early 1990s, Ken Schwaber used such an approach at his company, Advanced Development Methods, and Jeff Sutherland, with John Scumniotales and Jeff McKenna, developed a similar approach at Easel Corporation, and they were the first to refer to it using the single word "Scrum**.**" In 1991, Peter DeGrace and Leslie Hulet Stahl first referred to this as the *scrum approach*. In 2001, Ken Schwaber teamed up with Mike Beedle to describe the method in the book *Agile Software Development with Scrum*.

Scrum is an Agile approach for managing a project. Scrum was formalized originally for software development projects, but it works well for any innovative scope of work. The possibilities are endless. The Scrum model is deceptively simple as shown in the diagram below.



Copyright © 2005, Mountain Goat Software

- A Product Owner creates a prioritized list called a product backlog.
- During sprint planning, the team pulls a small chunk from the top of the product backlog; adds the selected work to a sprint backlog**,** and then decides how to implement those pieces.
- The team has a certain amount of time, a sprint, to complete its work - usually two to four weeks - but it meets each day to assess its progress (aka the Daily Scrum).
- Along the way, the Scrum Master keeps the team focused on its goal.
- At the end of the sprint, the work potentially should be shippable, as in ready to hand to a customer; put on a store shelf; deployed to a user, or shown to a stakeholder.
- The sprint ends with a sprint review and a retrospective.
- As the next sprint begins, the team chooses another chunk of the product backlog and begins working again.

The cycle repeats until enough items in the product backlog have been completed to the satisfaction of the Product Owner; the budget is depleted, or a deadline arrives. Which of these milestones marks the end of the work is entirely specific to the project. No matter which impetus stops work, Scrum ensures that the most valuable work has been completed when the project ends.

Scrum's simplicity is defined by three roles (Team, Scrum Master, and Product Owner); four time-boxed activities (Sprint Planning and Estimating, Daily Scrum or Standup, Sprint Review, and Sprint Retrospective); and four artifacts or work products (Product Backlog, Sprint Backlog, Sprint Burn-down and taskboard, and working software and associated artifacts).

The values of Scrum include Commitment, Focus, Openness, Respect, and Courage, all of which support the principles and practices of Scrum. Although Scrum has been used as an effective project management tool for software projects over the past 25 years, Scrum has been very effective in managing non software-centric projects, especially during the last 10 years where studies of successful Scrum implementations are emerging from academia, industry, and the Department of Defense.

TDD - Test-Driven Development

Test-driven Development (TDD) is the craft of producing automated tests for production code, and using that process to drive design and programming. For every tiny bit of functionality in the production code, you first develop a test that specifies and validates what the code will do. You then produce exactly as much code as will enable that test to pass. Then you refactor (simplify and clarify) both the production code and the test code.

In recipe format, the steps of TDD are:

1. The first step is to quickly add a test, basically just enough code to fail.
2. Next you run your tests, often the complete test suite although for sake of speed you may decide to run only a subset, to ensure that the new test does in fact fail.
3. You then update your functional code to make it pass the new tests.
4. The fourth step is to run your tests again. If they fail you need to update your functional code and retest.
5. Now look for and remove duplication; clean up the test code (extract common setup into the fixture/setUp(); split fixtures if they become non-cohesive; etc.)

The above sequence is repeated continuously throughout the programming process. Each cycle is framed in duration of minutes, if you're not capable to fix a failing test within minutes, and then throw away the test and the code that you wrote to satisfy the test and design a simpler test.

XBreed

XBreed is the product of mixing Scrum and Extreme Programming (XP). XBreed is the result of developing multiple applications and shared components as fast as humanly possible.

Combining Scrum and XP is very natural: Scrum provides a solid management framework, while XP provides a basic but complete set of engineering practices. The result is a lean but very mean (very effective) way to run software projects. In addition, Scrum practiced at the application team level, and provided a Shared Resources Team is involved, can lead to reusability.

XP - eXtreme Programming

Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

About the Author

Dick Carlson has been an active Agile transformational leader for many small and large projects, and has frequently shared his experiences of successful Agile, Lean, and Scrum implementations at conferences, workshops, and symposia, and regularly advises executives and organizational leaders on the cost, quality, and schedule benefits of using those initiatives and techniques. He has actively coached teams for more than 20 years on Agile and Lean Project Management fundamentals, and follows up with mentoring activities to ensure successful project execution. Dick has also provided concentrated Agile coaching support and led many organizations, programs, and projects that varied in size from six to more than 2,000 engineers, and that ranged in costs from under $50,000 to more than $1.4B.

Dick used Scrum practices and principles to manage and form the start-up of the Agile & Lean Education Associates (ALEA) Company that began July 2013. He continues to share his practical knowledge of Agile, Lean, and Scrum through more than two decades of experience by means of Agile and Lean training and the right amount of coaching to companies and other activities that want to increase their competitive advantage. The ALEA Company website (http://www.a2zalea.com) provides information about who we are and what we do.