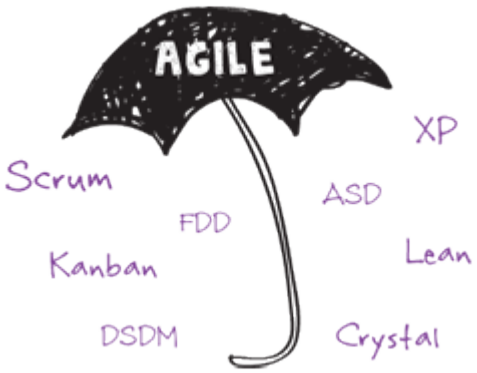
What Is Agile Methodology?

The various agile Scrum methodologies share much of the same philosophy, as well as many of the same characteristics and practices. But from an implementation standpoint, each has its own recipe of practices, terminology, and tactics. Here we have summarized a few of the main agile software development methodology contenders:



Agile Scrum Methodology

Scrum is a lightweight agile project management framework with broad applicability for managing and controlling iterative and incremental projects of all types. Ken Schwaber, Mike Beedle, Jeff Sutherland and others have contributed significantly to the evolution of Scrum over the last decade. Scrum has garnered increasing popularity in the agile software development community due to its simplicity, proven productivity, and ability to act as a wrapper for various engineering practices promoted by other agile methodologies.

With Scrum methodology, the "Product Owner" works closely with the team to identify and prioritize system functionality in form of a "Product Backlog". The Product Backlog consists of features, bug fixes, non-functional requirements, etc. – whatever needs to be done in order to successfully deliver a working software system. With priorities driven by the Product Owner, cross-functional teams estimate and sign-up to deliver "potentially shippable increments" of software during successive Sprints, typically lasting 30 days. Once a Sprint's Product Backlog is committed, no additional functionality can be added to the Sprint except by the team. Once a Sprint has been delivered, the Product Backlog is analyzed and reprioritized, if necessary, and the next set of functionality is selected for the next Sprint.

Scrum methodology has been proven to scale to multiple teams across very large organizations with 800+ people. See how VersionOne supports Scrum Sprint Planning by making it easier to manage your Product Backlog.

Lean and Kanban Software Development

Lean Software Development is an iterative agile methodology originally developed by Mary and Tom Poppendieck. Lean Software Development owes much of its principles and practices to the Lean Enterprise movement, and the practices of companies like Toyota. Lean Software Development focuses the team on delivering Value to the customer, and on the efficiency of the "Value Stream," the mechanisms that deliver that Value. The main principles of Lean methodology include:

- Eliminating Waste
- Amplifying Learning
- Deciding as Late as Possible
- Delivering as Fast as Possible
- Empowering the Team
- Building Integrity In
- Seeing the Whole

Lean methodology eliminates waste through such practices as selecting only the truly valuable features for a system, prioritizing those selected, and delivering them in small batches. It emphasizes the speed and efficiency of development workflow, and relies on rapid and reliable feedback between programmers and customers. Lean uses the idea of work product being "pulled" via customer request. It focuses decision-making authority and ability on individuals and small teams, since research shows this to be faster and more efficient than hierarchical flow of control. Lean also concentrates on the efficiency of the use of team resources, trying to ensure that everyone is productive as much of the time as possible. It concentrates on concurrent work and the fewest possible intra-team workflow dependencies. Lean also strongly recommends that automated unit tests be written at the same time the code is written.

The Kanban Method is used by organizations to manage the creation of products with an emphasis on continual delivery while not overburdening the development team. Like Scrum, Kanban is a process designed to help teams work together more effectively.

Kanban is based on 3 basic principles:

- **Visualize what you do today (workflow):** seeing all the items in context of each other can be very informative
- **Limit the amount of work in progress (WIP):** this helps balance the flow-based approach so teams don 't start and commit to too much work at once
- **Enhance flow:** when something is finished, the next highest thing from the backlog is pulled into play

Kanban promotes continuous collaboration and encourages active, ongoing learning and improving by defining the best possible team workflow. See how VersionOne supports Kanban software development.

Extreme Programming (XP)

XP, originally described by Kent Beck, has emerged as one of the most popular and controversial agile methodologies. XP is a disciplined approach to delivering high-quality software quickly and continuously. It promotes high customer involvement, rapid feedback loops, continuous testing, continuous planning, and close teamwork to deliver working software at very frequent intervals, typically every 1-3 weeks.
The original XP recipe is based on four simple values € " simplicity, communication, feedback, and courage € " and twelve supporting practices:

- Planning Game
- Small Releases
- Customer Acceptance Tests
- Simple Design
- Pair Programming
- Test-Driven Development
- Refactoring
- Continuous Integration
- Collective Code Ownership
- Coding Standards
- Metaphor
- Sustainable Pace

Don Wells has depicted the XP process in a popular diagram.

In XP, the "Customer" works very closely with the development team to define and prioritize granular units of functionality referred to as "User Stories". The development team estimates, plans, and delivers the highest priority user stories in the form of working, tested software on an iteration-by-iteration basis. In order to maximize productivity, the practices provide a supportive, lightweight framework to guide a team and ensure high-quality software.

Crystal

The Crystal methodology is one of the most lightweight, adaptable approaches to software development. Crystal is actually comprised of a family of agile methodologies such as Crystal Clear, Crystal Yellow, Crystal Orange and others, whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities. This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project 's unique characteristics.
Several of the key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process. Like other agile process methodologies, Crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions. Alistair Cockburn, the originator of Crystal, has released a book, *Crystal Clear: A Human-Powered Methodology for Small Teams*.

Dynamic Systems Development Method (DSDM)

DSDM, dating back to 1994, grew out of the need to provide an industry standard project delivery framework for what was referred to as Rapid Application Development (RAD) at the time. While RAD was extremely popular in the early 1990 's, the RAD approach to software delivery evolved in a fairly unstructured manner. As a result, the DSDM Consortium was created and convened in 1994 with the goal of devising and promoting a common industry framework for rapid software delivery. Since 1994, the DSDM methodology has evolved and matured to provide a comprehensive foundation for planning, managing, executing, and scaling agile process and iterative software development projects.

DSDM is based on nine key principles that primarily revolve around business needs/value, active user involvement, empowered teams, frequent delivery, integrated testing, and stakeholder collaboration. DSDM specifically calls out "fitness for business purpose" as the primary criteria for delivery and acceptance of a system, focusing on the useful 80% of the system that can be deployed in 20% of the time.

Requirements are baselined at a high level early in the project. Rework is built into the process, and all development changes must be reversible. Requirements are planned and delivered in short, fixed-length time-boxes, also referred to as iterations, and requirements for DSDM projects are prioritized using MoSCoW Rules:

M – Must have requirements
S – Should have if at all possible
C – Could have but not critical
W – Won 't have this time, but potentially later

All critical work must be completed in a DSDM project. It is also important that not every requirement in a project or time-box is considered critical. Within each time-box, less critical items are included so that if necessary, they can be removed to keep from impacting higher priority requirements on the schedule.
The DSDM project framework is independent of, and can be implemented in conjunction with, other iterative methodologies such as Extreme Programming and the Rational Unified Process.

Feature-Driven Development (FDD)

The FDD variant of agile methodology was originally developed and articulated by Jeff De Luca, with contributions by M.A. Rajashima, Lim Bak Wee, Paul Szego, Jon Kern and Stephen Palmer. The first incarnations of FDD occurred as a result of collaboration between De Luca and OOD thought leader Peter Coad. FDD is a model-driven, short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small, "useful in the eyes of the client" results. FDD designs the rest of the development process around feature delivery using the following eight practices:

- Domain Object Modeling
- Developing by Feature
- Component/Class Ownership
- Feature Teams
- Inspections
- Configuration Management
- Regular Builds
- Visibility of progress and results

FDD recommends specific programmer practices such as "Regular Builds" and "Component/Class Ownership". FDD's proponents claim that it scales more straightforwardly than other approaches, and is better suited to larger teams. Unlike other agile methods, FDD describes specific, very short phases of work, which are to be accomplished separately per feature. These include Domain Walkthrough, Design, Design Inspection, Code, Code Inspection, and Promote to Build.

The notion of "Domain Object Modeling" is increasingly interesting outside the FDD community, following the success of Eric Evans' book *Domain-Driven Design*.