**Agile Introduction**

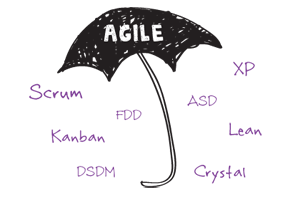
Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.

The authors of the Agile Manifesto chose “Agile” as the label for this whole idea because that word [represented the adaptiveness and response to change which was so important to their approach](https://martinfowler.com/articles/agileStory.html).

It's really about thinking through how you can understand what's going on in the environment that you're in today, identify what uncertainty you're facing, and figure out how you can adapt to that as you go along.[1]

**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](https://resources.collab.net/agile-101/what-is-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.

**Lean and Kanban Software Development**

[Lean Software Development](http://www.poppendieck.com/index.htm) 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

**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](https://resources.collab.net/lifecycle/versionone-lifecycle-for-alm-product-overview).

**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

**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.

**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](http://www.dsdm.org/) 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. 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](http://www.featuredrivendevelopment.com/) 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

References:

1. **‘Agile Alliance’, *“What is Agile Software Development?”*, ‘March 05, 2019’** [**https://www.agilealliance.org/agile101/**](https://www.agilealliance.org/agile101/)**, April 18, 2019**

**2. ‘Agile Methodologies for Software Development’, ‘Agile Methodology’, ‘January 15, 2019’,https://resources.collab.net/agile-101/agile-methodologies, ‘April 18, 2019’**