

# Agile Glossary

---

<b>Agile Application Lifecycle Management</b>	Also called Agile ALM, Agile Application Lifecycle Management is the integrated management platform of the entire software application lifecycle, from planning to the final release. Key components of the platform include the ability to handle change management, workflow, source code management, task management, testing and bug tracking, reporting and analytics.
<b>Agile Practices</b>	Agile practices are procedures that are defined as being highly efficient to productivity, and include the following practices: user stories, cross-functional teams, unit testing, refactoring, continuous integration, multi-stage continuous integration, planning poker, burnup charts, burndown charts.
<b>Agile Development</b>	Agile development is a way of thinking about software development as expressed in the Agile Manifesto, and acts as an “umbrella” for a group of methodologies. The methodologies are based on process-centric and iterative development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. Agile development is a conceptual framework that promotes evolutionary change throughout the entire life cycle of the project and represents a new, more flexible approach to development than the traditional methods that have previously been the norm for software development.
<b>Agile Development Life Cycle</b>	The complete software development process including Agile practices such as user stories, cross-functional teams, unit testing, refactoring, continuous integration, multi-stage continuous integration, planning poker, burnup charts, burndown charts.
<b>Agile Manifesto</b>	<p>Principles of Agile software development: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:</p> <ul style="list-style-type: none"><li><b>Individuals and interactions</b> over processes and tools</li><li><b>Working software</b> over comprehensive documentation</li><li><b>Customer collaboration</b> over contract negotiation</li><li><b>Responding to change</b> over following a plan</li></ul> <p>That is, while there is value in the items on the right, we value the items on the left more."</p>

<b>Agile Processes</b>	A software development methodology based on process-centric and iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams and is collectively regarded as highly efficient to productivity. Specific processes include user stories, cross functional teams, unit testing, refactoring, continuous integration, multi-stage continuous integration, planning poker, burnup charts and burndown charts.
<b>Agile Project Management</b>	The process of planning, organizing, and managing the necessary resources in order to complete project goals while adhering to Agile practices.
<b>Agile SCM Tool</b>	Software Configuration Management tool that supports Agile Software Development Lifecycle requirements differently than requirements involved with traditional software development. These supported features and requirements of Agile SCM include feature-oriented development, sandboxing with private build before check-in, ability to revert to last good working version when integration testing fails, staging hierarchy, ability to revert and retarget changes, refactoring support and support for geographically distributed development.
<b>Agile Software Development</b>	Agile software development is a way of thinking about software development, as expressed in the Agile Manifesto, and acts as an “umbrella” for a group of methodologies. The methodologies are based on process-centric and iterative development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. Agile software development is a conceptual framework that promotes evolutionary change throughout the entire life cycle of the project and represents a new, more flexible approach to development than the traditional methods that have been the norm for software development.
<b>Application Development Process Tools</b>	Tools necessary to complete the application development process, such as Application Lifecycle Management tools, Software Configuration Management tools, Build and Release tools, security and defect tracking tools, etc.
<b>Application Lifecycle Management (ALM)</b>	Also called ALM, Application Lifecycle Management is the management platform of the entire software application lifecycle, from planning to the final release. Key components of the platform include the ability to handle change management, workflow, source code management, task management, testing and bug tracking, reporting and analytics.
<b>Backlog</b>	Also known as "product backlog," the backlog is a prioritized list of user stories and defects in order from most valuable to least valuable for a system. Backlogs include

	both functional and non-functional user stories as well as technical team-generated stories.
<b>Branching</b>	Branching is the duplication of objects under revision control (such as a source code file, or a directory tree) in such a way that the newly created objects initially have the same content as the original, but can evolve independently of the original. Branching can take two forms, static or dynamic. In static branches, copy and label operations are used to duplicate a given branch. The duplicate then can evolve independently. With dynamic branches, usually implemented in streams, only the label operation is used, to flag the point in time that a stream diverged from its parent stream. Both branching forms support some form of merging, so that code changes made on a branch can be re-integrated into another branch, as is typical in parallel development processes.
<b>Burndown Chart</b>	Representation of the number of hours remaining for completion of a project; usually represented in chart form with points plotted on an x and y axis that map a downward trend of work left to do until burning down to zero.
<b>Burnup Chart</b>	Representation of the number of stories completed; usually represented in chart form with points plotted on an x and y axis that map an upward trend of work completed until reaching 100%.
<b>Change and Configuration Management</b>	Change and Configuration Management (also known as Software Change and Configuration Management or SCCM) combines aspects of both change management and configuration management to control a software development project as it evolves through the software development process. SCCM typically includes all technical aspects of the development process, such as version control, branching and merging. Additionally, SCCM includes change related activities such as issue tracking, document tracking, and process workflows that enable development teams to control the overall process.
<b>Change Control</b>	Process in which changes to a product or system are introduced in a controlled manner with minimal disruptions to services and cost effective solutions involved in implementing the changes.
<b>Change Management</b>	Change Management enables development organizations to control, communicate and respond more effectively to rapidly changing business demands.

<b>Change Packages</b>	Change Packages enable developers and managers to group file changes together into a logical whole and enable release managers to work at the issue or task level, while still providing developers with full access to the underlying file contents of the Change Package. Once created, a Change Package allows users to move, copy, modify, merge or revert the change package.
<b>Collocation</b>	Collocation refers to development teams located and working in the same location. Collocation is usually applied at the cross-functional team level.
<b>Configuration Management</b>	Configuration Management refers to a set of practices around storing, tracking and releasing versions of a software product. Software products that enable development organizations to perform these practices efficiently are also referred to as Configuration Management systems or Configuration Management tools. Configuration Management systems will typically provide users with a variety of features, including but not limited to source code control, issue tracking, and change set management.
<b>Configuration Management Tools</b>	Configuration Management tools are the tools that make possible the practices around storing, tracking and releasing versions of software.
<b>Continuous Integration</b>	Continuous integration, one of the foundational aspects of Agile software development methodologies, is defined by Martin Fowler to be "a fully automated and reproducible build, including testing, that runs many times a day. This allows each developer to integrate daily, thus reducing integration problems." By getting changes into the main line as frequently as possible, preferably daily, and by extending the idea of a nightly build, continuous integration helps reduce integrations problems and identify and resolve problems more quickly.
<b>Cross-Functional Team</b>	Team comprised of members with all functional skills and specialties necessary to complete a project from start to finish.
<b>Distributed Development</b>	Development teams that work on the same project but are located across multiple locations or worksites.
<b>Enterprise Agile</b>	The adoption of specific Agile practices in an organization that works in conjunction with other non-Agile practices. Enterprise Agile is a highly efficient and customized practice for large organizations that have difficulty making a complete transition to Agile, as well as for organizations that already practice efficient development

	processes.
<b>Epic</b>	A user story which describes a large amount of customer value and needs to be broken down into many smaller user stories.
<b>Feature Driven Development</b>	Feature Driven Development (FDD) is an Agile method for developing software based on an iterative and incremental software development process. The main purpose of FDD is to deliver tangible, working software repeatedly in a timely manner.
<b>Hybrid Processes</b>	Development process that uses both Agile and non-Agile practices in conjunction with each other and is proven highly effective for development teams
<b>Inspecting and Adapting</b>	Agile process where teams evaluate a project by looking, listening to each other's feedback and ultimately improving the process or changing course.
<b>Iteration</b>	Microcosm of a traditional Systems Development Life Cycle (SDLC,) each of which produces working software. Iterations can be as large as 3 months but are more typically between 1 to 4 weeks. See sprint.
<b>Kanban</b>	Methodology that comes from Lean software development and has three main components: visual system for managing work, limits work in progress, and work is pulled rather than pushed through the system.
<b>Key Agile Principles</b>	See Agile Manifesto.
<b>Lean Software Development</b>	A programming concept that focuses on optimizing efficiencies for development and minimizing waste. According to Mary Poppendieck, 10 rules of Lean programming include: eliminate waste, minimize artifacts, satisfy all stakeholders, deliver as fast as possible, decide as late as possible, decide as low as possible, deploy comprehensive testing, learn by experimentation, measure business impact and optimize across organizations.
<b>Merging</b>	The process of incorporating branches back into the mainline.
<b>Multi-stage Continuous</b>	Agile method allowing for a high degree of integration to occur in parallel while vastly reducing the scope of integration problems. Multi-stage Continuous

<b>Integration</b>	Integration (CI) is an expansion upon Continuous Integration, where each developer works on his or her own task. As changes are made, CI is done against that team's branch. If CI does not succeed, then that developer (possibly with help from her teammates) fixes the branch. This way when there is a problem, only that team, not the whole development effort is affected.
<b>One Piece Flow</b>	Process in which each developer or development process works on only one piece at a time before pulling code downstream, one piece at a time, to the next process.
<b>Pair Programming</b>	Process in which two developers work together at a single workstation, where one is responsible for typing code and the other for reviewing each line of code as it is typed in.
<b>Parallel Development</b>	Parallel development occurs whenever a software development project requires separate development efforts on related code bases. For example, when a software product is shipped to customers, a product development team may begin working on a new major feature release of the product, while a product maintenance team may work on defect corrections and customer patch releases of the shipped product. Both teams begin work from the same code base, but the code necessarily diverges. Frequently the code bases used in parallel development efforts must be merged at some future date, for example, to ensure that the defect corrections provided by the product maintenance team are integrated into the major release that the product development team is working on.
<b>Planning Poker</b>	A consensus-based technique for estimating; mostly used to estimate effort or relative size of tasks in software development. Planning Poker is useful for building team cohesion and for fostering self-organizing teams.
<b>Product Backlog</b>	The backlog owned by the Product Owner.
<b>Product Owner</b>	A role originating from Scrum, but has now been widely adopted independently of Scrum. A product owner manages the product backlog, addresses questions that arise during development and signs off on work results. The product owner guides the team with what should be done and when the final product should be shipped. The Scrum team then balances out the product owner's decisions by deciding how much work should be involved in an individual sprint and estimating the amount of time necessary to complete the task.
<b>Real World Agile</b>	The adoption of specific Agile practices in an organization that works in conjunction

	with other non-Agile practices. Real World Agile is a highly efficient and customized practice for large organizations that have difficulty make a complete transition to Agile as well as for organizations that already practice efficient development processes.
<b>Refactoring</b>	The practice of continuously improving the usability, maintainability, and adaptability of code without changing its behavior. Refactoring makes it much easier to add new and unanticipated functionality. Refactoring has the disadvantage that it takes extra effort and requires changing the code.
<b>Release Management</b>	Release management comprises a broad set of activities in software development organizations that center on ensuring that software is ready to be released to customers.
<b>Release Plan</b>	A document describing scheduling, activities, resources and responsibilities related to a particular release.
<b>Release Process</b>	The software release process is the final stage in a typical software development effort, where the software product is made available for use. To ready a software product for release, the release process must ensure that all product requirements have been met, usually by executing test suites designed to exercise product functionality and correcting any defects found.
<b>SCM Software</b>	Software Configuration Management software is a software tool that enable organizations to perform the SCM practices of storing, tracking and releasing a product, and typically provide users with a variety of features including source code control, issue tracking and change set management, advanced configuration management, change packages, process management and integrated issue tracking.
<b>SCM Tools</b>	Software Configuration Management tools are tools that enable organizations to perform SCM practices and typically provide users with a variety of features, including source code control, issue tracking and change set management, advanced configuration management, change packages, process management and integrated issue tracking.
<b>Scrum</b>	Agile development project management framework based around sprints and is generally comprised of a Scrum Team, Product Owner and Scrum Master. The framework of Scrum leaves most development decisions up to the self-organizing Scrum team, where decisions are reached as a whole team.

<b>Scrum Master</b>	Person trained to facilitate daily Scrum meetings, remove impediments, oversee the team's progress through the process and track Scrum team updates.
<b>Self-Organizing</b>	A team, usually found in Scrum, that manages itself through various means of communication and reoccurring structured meetings. Self organizing teams solve development issues together as a whole and decide the best solution depending on the various team members.
<b>Software Change and Configuration Management (SCCM)</b>	Software change and configuration management (SCCM) combines aspects of both change management and configuration management to control a software development project as it evolves through the software development process. SCCM typically includes all technical aspects of the development process, such as version control, branching and merging. Additionally, SCCM includes change related activities such as issue tracking, document tracking, and process workflows that enable development teams to control the overall process.
<b>Software Configuration Management (SCM)</b>	Software Configuration Management (SCM) refers to a set of practices around storing, tracking and releasing versions of a software product. Software products that enable development organizations to perform these practices efficiently are also referred to as Software Configuration Management systems or Software Configuration Management tools. Software Configuration Management systems will typically provide users with a variety of features, including but not limited to: source code control, issue tracking and change set management.
<b>Software Development</b>	Development of software in a planned and structured process. See software development process.
<b>Software Development Process</b>	The software development process is the set of coordinated activities performed by engineers, managers and technical writers resulting in the creation of a software product. Various named software development processes are in use today, including Agile, XP, Scrum, Waterfall and Lean.
<b>Source Code Control</b>	Source code control is a common requirement in all modern software development projects that provides mechanisms for checking source code in and out of a central repository. This allows different developers to work on the same project, with reduced fears of lost code or overwritten changes. Source code control also implies a version control system that can manage files through the development lifecycle, keeping track of which changes were made, who made them, when they were made,



	and why. Finally, source code control also frequently involves the ability to group versioned files as a single release, maintain multiple active releases concurrently (branching), and join different releases (merging).
<b>Source Code Management</b>	Source code management refers broadly to the set of operations required to store, retrieve and version the files used to construct software applications. Development teams rely on source code management to organize the source code files for different releases of software, so that releases can be uniquely identified for testing, packaging and delivery to customers. Failure to do this properly results in poor quality releases and inefficient use of development resources.
<b>Spike</b>	Timeboxed investigation of feasibility via a bare bones implementation, which touches on all aspects of the full implementation.
<b>Sprint</b>	Scrum specific word describing iterations.
<b>Sprint Backlog</b>	Plan for development team to map out implementation of features for an upcoming sprint.
<b>Sprint Planning</b>	A meeting for Scrum Teams, Scrum Masters and Product Owners where the Product Owner describes priority features to the team. The Scrum Team gets enough of an understanding about the tasks discussed that they are able to choose which ones to move from the product backlog to the sprint backlog.
<b>Retrospective</b>	Meeting held at the end of every sprint review to reflect on what went well during the sprint and what can be improved upon during the next sprint. Sprint retrospectives are valued as necessary parts of inspecting and adapting, and allow development teams to plan for future output.
<b>Sprint Review</b>	In the sprint review, teams go over what stories were completed during the iteration and demonstrate those stories for stakeholders and the product owner.
<b>Stand-up</b>	Daily Meetings that are meant to quickly and efficiently resolve obstacles that any team members may be experiencing.
<b>Story Points</b>	Relative scale of effort required by a team to implement a user story.
<b>Task Board</b>	A physical or electronic board representing the state of tasks in a current sprint,

	often divided into "to do," "in progress" and "done."
<b>Timeboxing</b>	The practice of constraining the amount of time for performing any activity. Examples include iterations, spikes and stand up meetings.
<b>Unit Testing</b>	Tests that exercise small amounts of isolated functionality.
<b>User Stories</b>	Used with Agile methodologies for specifying requirements and presented as an informal statement of the requirement (usually fitting on a 3x5 index card).
<b>Velocity</b>	The velocity of a team is the number of story points associated with stories that are finished over a given period of time, often 1 to 4 weeks. For instance, if the team completed 8 stories that were each 5 points during a four week period, then their velocity is 40 story points every four weeks.
<b>Waterfall</b>	Model of a software development process in which progress flows downwards through phases of conception, initiation, analysis, design, construction, testing and maintenance.
<b>Whole Teams</b>	Team comprised of members with different functional skills and specialties that work together during all phases of development in order to complete a project from start to finish. Also known as a cross-functional team.
<b>XP</b>	"Extreme Programming," one implementation of the Agile methodology that focuses on producing the simplest coding situation for application requirements and includes practices such as pair programming, incremental design and continuous integration.