Agile Glossary

Acceptance Testing
Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.

Anti-Pattern
Repeated practices that appear initially to be beneficial, but ultimately result in bad consequences that outweigh the hoped-for advantages.

Adaptive software development
A software development process created by Jim Highsmith and Sam Bayer. Continuous adaptation of the iteration software delivery process is done using speculate, collaborate, and learn phases.

Affinity Estimating
Affinity Estimating is a technique many teams use to quickly and easily estimate (in Story Points) a large number of user stories. This is a great technique if you're just starting a project and have a backlog that hasn't been estimated yet.

Agile
An iterative and incremental approach to developing software that adheres to the Agile Manifesto and its associated principles. Ideally this is done using small, dedicated, co-located, self-organizing teams who work in close collaboration with a business customer. Agile is value-driven both in the focus on delivering the most important features first and in the ways the teams choose to work together to develop the software.

Agile manifesto
A statement of principles and values that define agile software development.

Agile Modeling
Agile modeling is a collection of values, principles and practices for modeling software that can be applied on a software development project in an effective and light-weight manner.

Backlog
A prioritized list of work to be completed. Backlogs are often divided from largest to smallest: product backlog (items for the entire project/product), release backlog (just the items for a particular release), and iteration backlog (items for the current iteration).

Brainstorming techniques
A technique teams use to generate ideas on a particular subject. Each person on the team is asked to think creatively and write down as many ideas as possible. The ideas are reviewed after the brainstorming session. Commonly used in kaizens. Some techniques are Free-from-all, Round-robin, Quiet writing.

Bug
Anything that is a potential source of dissatisfaction with the product. Bugs may be discovered during testing or even in conversation with the Business (as a defect in requirements). Depending on the size of

the bug, a team may choose to generate a story or generate a task within a "bucket" story to be able to prioritize and manage work to resolve the bug.

Build
The process of taking one or more input items (requirements, configurations) and creating one or more output items. Software compile and load is an example of a build. Lean-Agile urges the use of continuous builds, during which tests are automatically performed, as we aim for perfection in the code.

Build Verification Test
A group of tests to determine the health of a build at a high level. Typically, these tests exercise the core functionality of code and help determine whether further testing is warranted. These are also known as "smoke tests."

Burndown chart
A chart showing the work remaining over an iteration or release. Time is indicated on the x-axis, work is indicated on the y-axis (hours for iteration burndowns, points for release burndowns).

Burn up
The rate at which stories or tasks are growing, measured in hours per day.

Business Owner
The Business Owner is the person who has the ultimate responsibility for funding and using the products that are being created to deliver value to the customer.

Business Value
Business Value is what management is willing to pay for. It is a way to identify the value of “work” or a story.

Big Design Up Front (BDUF) or Big Upfront Design (BUFD)
The traditional method of trying to define all the possible requirements of a product at the beginning of the project, prior to finalizing a plan and beginning coding.

Big Visible Charts (Information Radiator)
Big visible charts are exactly what you would think they would be: Big charts posted near the agile team that describe in different ways the team's progress. Big visible charts not only can be useful tools for the team but also make it easier for any stakeholder to learn how the team is progressing. Big visible charts are an important tool for implementing the essential agile vaules of transparency and communication.

Chartering
A document that aggregates project highlights for presentation to a product manager, sponsor, or other interested party. Chartering synchronizes the customer and the project team on the project’s value and goals.

Code Coverage
A metric used to describe the degree to which source code has been tested. Code coverage is expressed as a percentage of lines of code tested over the total lines of code.

Colocation
Refers to development teams located and working in the same location. When possible colocation is desirable since it facilitates face-to-face collaboration, an important features of Agile Software Development.

Continuous Integration
"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly." (Martin Fowler)

Cost-Benefit Analysis
An examination of the relationship between the monetary cost of implementing an improvement and the monetary value of the benefits achieved by the improvement, both within the same time period.

Cross-Functional Team
Team comprised of members with all functional skills and specialties necessary to complete a project from start to finish.

Cumulative Flow Diagram
Cumulative Flow Diagrams (CFDs) are valuable tools for tracking and forecasting agile projects.

Customer
The recipient of the output (product, service, information) of a process. Customers may be internal or external to the organization. The customer may be one person, a department, or a large group.

Cycle Time
The total elapsed time to move a unit of work from the beginning to the end of a process. Cycle time includes process time, during which a unit is acted upon to bring it closer to an output, and delay time, during which a unit of work is spent waiting to take the next action.

Crystal
A family of agile methodologies created by Alistair Cockburn that focuses on people rather than process as a way to deliver software more effectively and efficiently.

Daily Stand-up
A standup meeting of the Sprint team where status is exchanged, progress is observed, and impediments are noted and removed. Typically, these meetings last 15 minutes. Each member answers three questions: • What did you do yesterday • What do you plan to do today • What is getting in your way In the literature, often called the “Daily Scrum.”

Dashboard
A type of information radiator that provides management and teams with graphs and reports indicating progress and trends and to identify potential problems.

Developer
Those members of the Sprint team who apply technical knowledge and skills to create the product. Testers may also be developers.

Development Story
When a requirement story is too large for an iteration and yet cannot be decomposed into iteration requirement stories, the developers create implementation stories to describe the requirement story will be implemented. Implementation stories: • Do not have direct business value, but support the business value of the associated requirement story • Has acceptance tests created by the implementers • Are estimable • Fit within an iteration Example: Create interface to a credit card processing system.

Development Task
Work that generates code from analysis tasks.

Distributed Development Team
Refers to development teams that work on the same project but are located across multiple geographic locations or work sites.

Done / Definition of Done
The criteria for accepting a feature as finished. Specifying these criteria is the responsibility of the entire team, including the business. There are three levels of "Done" (also known as Done-Done-Done): • Done: Developed, runs on developer's box • Done: Verified by running unit tests, code review, etc • Done: Validated as being of deliverable quality with functional tests, reviews, etc

DSDM
Dynamic Systems Development Method is a software development framework based on rapid application development and iterative and incremental delivery of code. It was developed in the 1990s in the UK by a consortium of vendors and experts in the field of software development.

Emergent Design
Allowing a design to emerge over time, as part of the natural evolution of a system. Requires good practices and testing to ensure that the system is not inadvertently allowed to decay or become overly complex over time.

Epic
A very large user story that is eventually broken down into smaller stories. Epics are often used as placeholders for new ideas that have not been thought out fully or whose full elaboration has been deferred until actually needed. Epic stories help agile development teams effectively manage and groom their product backlog,

Estimation
The process of agreeing on a size measurement for the stories or tasks in a product backlog. On agile projects, estimation is done by the team responsible for delivering the work, usually using a planning game.

Extreme programming (also referred to as eXtreme Programming and abbreviated as XP)
A software engineering methodology that focuses on a set of specific engineering practices that lead to a higher quality of software and the ability to be more responsive to the customer's needs.

Facilitator
An individual specifically trained to enable groups and organizations to work more effectively, to collaborate and achieve synergy. The facilitator is a 'content neutral' party who by not taking sides or expressing or advocating a point of view during the meeting, can advocate for fair, open, and inclusive procedures to accomplish the group's work.

Feature
A feature is a business function that the product carries out. Features are large and chunky and are implemented by using many stories. Features may be functional or non-functional. Features provide the basis for organizing stories.

FIT
Framework for Integrated Test.

Functional Test
The activity that validates a feature against customer requirements. Functional tests are usually done by a tester as part of the customer team.

Gemba
The "Gemba" is the place where value-added work is actually being done: a work cell, the developer team room, the help desk, the customer's office. Management must go to these locations to observe, evaluate, coach, and engage with the team. This is in contrast to management practices that rely on management hierarchies, team leads or formal status meetings in conference rooms or management offices.

Green Belt
(Related to Six Sigma) A person who has been trained in Six Sigma methods, has completed a process improvement project satisfactorily (possibly reviewed by a panel) and who will be involved in process improvement as part of their regular job.

Happy Path
The basic course of action through a single use case.

Harness Tests
A group of test scenarios with expected outcomes related to a specific system module to confirm that defects have not been introduced as a result of current sprint programming activity. A pricing test harness would confirm no defects from current pricing module programming.

Ideal time
Ideal time is the amount of time that something takes when stripped of all peripheral activities. Ex. American Football game is takes 60 Minutes (4 Quarters of 15 Minutes each). Ideal time is 60 Minutes. Elapsed time is three hours + time for travel

Impediment
A technical, personal, or organizational issue that is preventing progress on delivering product.

Information Radiator
A type of visual control that displays information in a place where passersby can see it and get information about the project without having to ask questions. To be effective, the information must be current and must be easy to see and understand, with sufficient detail to explain status.

Integration Story
A story that describes testing the product itself.

Integration System Test
The activity that verifies that software code does not harm other parts of the software product. Integration system test is usually done by a tester with automated tools.

Inventory
In Lean, the money invested to purchase things and organization intends to sell.

Iteration
A time-boxed period during which the team is focused on producing a demonstrable product, some amount of functionality that is ready to be shown to the customer and potentially ready to be delivered.

Iteration plan
The breakdown of tasks that the team commits to completing in an iteration. Includes all the results of the iteration planning meeting.

Iteration planning meeting
Held at the beginning of each iteration and attended by all members of the team, this meeting determines what items the team will work through to completion in the iteration and their associated tasks and task estimates.

Iterative and incremental development
It is a way of developing a product in a series of “Plan-Do-Check-Act” cycles, delivering a portion of the product (an increment) each iteration (cycle).

Inspect and Adapt (Retrospective)
"Inspect and Adapt" is a slogan used by the Scrum community to capture the idea of discovering over the course of a project emergent software requirements and ways to improve the overall performance of the team. It neatly captures the both the concept of empirical knowledge acquisition and feedback-loop-driven learning.

Just-In-Time Design
The process of waiting until you know what the design needs to be and then refactoring code to meet these new needs before adding the functionality that is forcing the design change.

Kaizen / Continuous improvement
A Japanese term that means gradual unending improvement by doing little things better and setting and achieving increasingly higher standards. Masaaki Imai made the term famous in his book, Kaizen: The Key to Japan’s Competitive Success.

Kanban
Kanban is a tool derived from lean manufacturing and is associated with the branch of agile practices loosely referred to as Lean software development. Like a task board, Kanban visually represents the state of work in process. Unlike a task board, the Kanban constrains how much work in process is permitted to occur at the same time.

Lean
Producing the maximum sellable products or services at the lowest operational cost while minimizing waste.

Lean-Agile
An approach to software development that incorporates principles, practices, and methods from lean product development, agile software development, design patterns, test-driven development, and agile analysis.

Manual Test
A document that lists the steps that a person follows to complete a test pass. Not automated.

Minimum Marketable Feature
The smallest set of functionality that must be realized in order for the customer to perceive value. A "MMF" is characterized by the three attributes: minimum, marketable, and feature. A feature is something that is perceived, of itself, as value by the user. "Marketable" means that it provides significant value to the customer; value may include revenue generation, cost savings, competitive differentiation, brand-name projection, or enhanced customer loyalty. A release is a collection of MMFs that can be delivered together within the time frame.

Non-specific Implementation Story
Stories that are created by developers that are not tied to a particular requirement story. These stories express work that developers need to do but do not have business value on their own and do not have acceptance tests. Example: Learn how to process payments.

Osmotic Communication
Is a way of information sharing by overhearing each other in background by sitting in close proximity

Pair Programming
"An Agile software development technique in which two programmers work together at one workstation. One types in code while the other reviews each line of code as it is typed in. The person typing is called the driver. and the person reviewing the code is called the observer or navigator. The two programmers switch roles frequently." (Wikipedia)
Pair programming is one of the original 12 extreme programming practices. As counter-intuitive as it may seem to the uninitiated, pair programming is more productive than two individuals working independently on separate tasks.

Planning Game
The main planning process within extreme programming is called the Planning Game. The game is a meeting that occurs once per iteration, typically once a week. The planning game includes iteration (or sprint) planning and release planning.

PDCA / Deming Cycle / Shewhart Cycle / Deming Wheel
Plan-Do-Check-Act (PDCA) cycle is an iterative four-step problem-solving process for quality improvement. Plan: development of plan to effect improvement. Do: plan is carried out. Check: effects of plan are observed. Act: results studied and learning identified for future usage. Also known as the Deming Cycle, Shewhart Cycle and Deming Wheel.

Performance Test
A test that ensures that the required level of performance of a product is met. This test checks both that the functionality works and that the time required to do the work is acceptable.

Persona
A description of the typical skills, abilities, needs, tasks, behaviors, and backgrounds of a particular set of users. As an aggregation, the persona is a fiction but is used to ensure groups of users are accounted for.

Peer Review
Peer review is a type of software review in which a work product (document, code, or other) is examined by its author and one or more colleagues, in order to evaluate its technical content and quality.

Pig
Scrum slang. Someone who is responsible for doing a task on an active iteration. It comes from the joke, "a chicken and a pig talk about breakfast. The chicken says, 'Let's have bacon and eggs.' The pig replies, 'That's fine for you. You are just making a contribution, but I have to be fully committed." Pigs are actively involved in the project.

Planning Poker / Wide band delphi
Planning Poker is an Estimation tool for user story. The idea behind Planning Poker is simple. Individual stories are presented for estimation. After a period of discussion, each participant chooses from his own deck the numbered card that represents his estimate of how much work is involved in the story under discussion. All estimates are kept private until each participant has chosen a card. At that time, all estimates are revealed and discussion can begin again.

Planning onion
Coined by Mike Cohn, this refers to the way planning is done in layers in agile, with detail appropriate to the time horizon (the closer the release, the more detail that's provided in planning; the further away the release, the higher the level of detail provided).

Process
A series of actions, changes, or functions performed in the making or treatment of a product.

Product
A collection of tangible and intangible features that are integrated and packaged into releases that offer value to a customer or to a market.

Product Backlog
The set of stories that are not yet done.

Product champion / Product Owner
The primary Business representative who represents the “voice of the customer” and the “voice of the business” to the Sprint team. The responsibilities of the Product Champion include the following: • Manage the Vision. The Product Champion establishes, nurtures, and communicates the product vision. Achieves initial and on-going funding for the project by creating initial release plans and the initial Product Backlog. • Manage the ROI. The Product Champion monitors the project against its ROI goals and an investment vision. Updates and prioritizes the Product Backlog to ensure that the most valuable functionality is produced first and built upon. Prioritizes and refines the Product Backlog and measures success against expenses. • Manage the Release. The Product Champion makes decisions about when to create an official release. For a variety of reasons it may not be desirable to release at the conclusion of every increment. Similarly, if an official release is planned for after the fifth increment it may be released (with fewer features) after the fourth increment in order to respond to competitive moves or capture early market share. The Product Champion makes these decisions in a manner consistent with the investment vision that has been established for the project. We prefer the term Product Champion to Product Owner for several reasons. • Product Champion comes from the Lean world and encompasses the notion that their role is to lead (champion) the team into discovering the best product available for the customer(s), be they internal or external. • The champion creates a team of developers to best provide value to the customer. But it is still the team that is responsible. • The champion leads, the team works with them. • The phrase “Product Owner” has the connotation that it is the Product Owner’s responsibility to create the value for the customer. It is not. It is only their responsibility to prioritize the backlog. This requires a team effort. The person prioritizing the backlog must understand value as well as cost.

Product Vision
A short statement of the vision that is driving the project, expressed in business and customer value terms. The Product Owner provides the Product Vision. Expresses who it is for, what is the opportunity, the name of the product, the key benefit, key differentiators. Sometimes called the One Big Thing or Project Charter. Tools such as the Pro Forma Press Release are helpful in creating the Product Vision.

Project
A collection of releases, iterations, team members, and stories that creates a product. May have defined end dates or be on-going. May be: • All of the software a team or related teams are working on • A specific product or product group • A version of a company's product suite • A specific client implementation

Progressive Elaboration
A technique in which the plan for the particular and designated project is being continuously and constantly modified, detailed, and improved as newer and more improved, sets of information becomes available to the project team.

Quality Assurance
A role and activity that assures integrity, does release testing. The job of QA is to prevent defects from happening in the first place... it is NOT to find bugs. See also: Release Testing

Refactoring
The disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Usually referred to when fixing bad code.

Regression Testing
A comprehensive set of tests with expected outcomes that are run prior to release to confirm that general system functionality is defect-free as a result of development in the Sprint.

Release
A deployable software package that is usually the culmination of several iteration cycles of incremental deliveries, and/or those iterations taken as a group. A version of a product that is promoted for use or deployment. Releases represent the rhythm of the business and should align with defined business cycles. A release contains a combination of Minimum Marketable Features that form a releasable product. A release may be internal and may be used for further testing.

Requirement
This term is not used in agile because it has connotations of a level of detail that is often too granular for the time horizon. Also, requirements usually begin with the phrase “The system shall...,” which tends to place the emphasis solely on the technology. Instead, agile teams focus on functionality from the users’ point of view, to ensure that they truly understand the vision and problem to be solved. The use of user stories is common, as is the use of the term “feature” instead of “requirement.”

Risk Burn Down
The Risk burndown helps measure how much risk has been mitigated.

Retrospective
A timeboxed meeting held at the end of an iteration, or at the end of a release, in which the team examines its processes to determine what succeeded and what could be improved. The retrospective is key to an Agile team's ability to "inspect and adapt" in the pursuit of "continuous improvement." The Agile retrospective differs from other methodologies' "Lessons Learned" exercises, in that the goal is not to generate a comprehensive list of what went wrong. A positive outcome for a retrospective is to identify one or two high-priority action items the team wants to work on in the next iteration or release. The emphasis is on actionable items, not comprehensive analysis.

Root Cause
A factor that caused nonconformance and should be permanently eliminated through process improvement.

Scrum
Scrum is an Agile process or framework for managing Agile projects. It is a project management process more than a methodology (the latter is rather too heavy). The Agile Alliance is a group of analysts that originally developed the Scrum processes.

Scrum Master
Responsible for the process and the health of the team. Ensures that the team is fully functional and productive. Enables close cooperation across all roles and functions and removes barriers. Ensures that the process is followed. Facilitates the daily scrum, iteration reviews, and planning meetings. Also spelled “ScrumMaster.

Scrum Team
The scrum team is a cross-functional group that is responsible for delivering the software or product. In Agile development, the team usually includes people skilled to understand customer requirements and

conduct software design, coding and testing. The scrum team is encouraged to be self-organizing and to take collective responsibility for all work commitments and outcomes. Scrum teams respond to requirements (often presented as user stories) by collectively defining their tasks, task assignments, and level of effort estimates.
The ideal size for a scrum team adheres to the magic number seven plus or minus two rule.
In scrum, the team is one of three roles; the other two being ScrumMaster and product owner.

Servant Leadership
A leadership style that lead by example and role model situations and behaviors for their teams. They challenge command and control behaviors. They listen more and speak less. They accept and reflect on feedback. They value the ideas and opinions of others. They foster a learning and adaptive environment, participative management, mutual influence, and empowered cross-functional and self-organizing teams.

SIPOC
A diagramming tool used by Six Sigma process improvement teams to identify all relevant elements (suppliers, inputs, process, outputs, customers) of a process improvement project before work begins.

Six Sigma
A method that focuses on increasing process performance and decreasing process variation through a variety of tools. This leads to defect reduction, improved profits, and higher quality. A process is considered well-controlled when its variation is within six sigmas from the centerline in a statistical control chart.

Spike
A thin piece of code that tests some portion of the system. It slices narrowly across the entire system to help developers learn about the risks, issues, and limitations of the environment. Spikes help the team discover engineering, risk, or business issues that must be addressed. Spikes are usually used in the architectural phase of the system, but are not always required. The outcome of a spike is a decision, not code.

Sprint
The Scrum term for an “iteration.” Usually, sprints are 2-4 weeks long. A time-boxed period during which the team is focused on producing a demonstrable product – functionality that is ready to show to a customer and potentially ready to be delivered.

Sprint Backlog
A list of features, user stories or tasks that are pulled from the product backlog for consideration for completion during the upcoming sprint. Product backlog features and user stories are broken down into tasks to form the sprint backlog during the sprint planning meeting.

Sprint Planning Meeting
Each sprint begins with a two-part sprint planning meeting, the activity that prioritizes and identifies stories and concrete tasks for the next sprint.

Sprint Review
A meeting held at the end of each sprint in which the Scrum team shows what they accomplished during the sprint; typically this takes the form of a demo of the new features. The sprint review meeting is intentionally kept very informal. With limited time allocated for Sprint review prep.

Sprint Team
A cross-functional group comprised of persons with skills who can perform three roles – customer (requirements & validation), developers, and test. The team selects the iteration goal and specifies work results, has the right to do everything within the boundaries of the project guidelines to reach the iteration goal, and organizes itself and its work.

Stakeholder
Anyone external to the team with a vested interest in the outcome of the team's work.

Story
A requirement, feature, and/or unit of business value that can be estimated and tested. Stories describe work that must be done to create and deliver a feature for a product. Stories are the basic unit of communication, planning, and negotiation between the Scrum Team, Business Owners, and the Product Owner. Stories consist of the following elements: • A description, usually in business terms • A size, for rough estimation purposes, generally expressed in story points (such as 1,2,3,5,8) • An acceptance test, giving a short description of how the story will be validated Facets of a story include: • Business value, direct or indirect. If this cannot be estimated, then a study story is required • Instigator of the story, a customer or developer

Story Checklist
Standard tasks that must be part of every story to be considered complete.

Story Map
Story Maps organize system features by user activity. Story maps communicate the “big picture” to teams building features.

Story Point / Relative Sizing
Story points are used in the planning game. A rating given to stories to express its relative complexity for purposes of estimating how much to load in an iteration. One approach is to use Fibonacci numbers (1,2,3,5,8 where 1 is low complexity, 8 is very complex) and then use the Planning Poker exercise to get the team to agree on estimates. The team will gain experience in how much can go into each iteration.

Story Point Burn up
The rate of progress the team is achieving in completing the project. It is measured as number of stories completed per sprint and tracked toward the full project scope. Along with velocity, it visibly shows the duration and length of the current project.

Subject Matter Expert
A person who can speak with authority on an aspect of a project or who knows to whom to talk in order to get answers. Subject Matter Experts may represent a technology, the business, the customer, process, or any other topic of importance to the project.

Task
Tasks are descriptions of the actual work that an individual (or sub-team) does in order to complete a story. They are manage-able, do-able, and track-able units of work. Typically, there are several tasks per story. Tasks have the following attributes: A description of the work to be performed, in either technical or business terms; an estimate of how much time the work will take (hours, days); an owner, who may or may not be pre-assigned; an Exit criteria and verification method (test or inspection), and an indication of who will be responsible for the verification. All tasks must be verified, not just "done"

Task board / Scrum Board (Information Radiator)
A physical board or virtual table showing the status of task items in an iteration. Items move from left to right across the task board from phase to phase (Not Started, In Process, Ready to Be Tested, Testing, Ready for Acceptance, and Done).

Team Space
A common work area that enables team to collaborate and share information. *"War room"* or *"Bullpen"* works well for Co-located teams. For distributed teams use proper tooling like Wikis, Instant Messaging, Skype, Web Cameras,

Technical Debt
The obligation that a software organization incurs when it chooses a design or construction approach that's expedient in the short term but that increases complexity and is more costly in the long term.
Test
Automatic and manual inspections of code and process to ensure correctness and completeness. Types of tests include Unit Test, Integration Test, System Test, Regression Test, Performance Test, and User Acceptance (Customer Acceptance) Test. Tests may be automated or manual. See

Test Driven Development / TDD
Test-driven development, or TDD, is a rapid cycle of testing, coding, and refactoring. TDD follows "Red, Green, Refactor" cycle.

Test Case
A set of conditions or variables under which a tester determines if a story or task is partially or fully satisfied. Each requirement must have at least one test case. Sometimes, both a positive test case and a negative test case should be used.

Test Result
The verdict from running a test. Common attributes are Pass, Fail, Uncertain.

Test-Driven Development
An evolutionary approach to development. In TDD, each test is written before the functional code that makes the test pass. The goal of TDD is specification and not validation, to think through a design before code is written, to create clean code that always works.

Tester
Those members of the Sprint team who apply testing knowledge and skills to validate and verify the product. Testers may be developers or customers and will use a combination of manual and automated methods. Testers may also act as consultants to developers to develop testing strategies.

Testing Task
Work that defines validation and verification tests for stories and code.

Timebox
A segment of time with a defined start and end and often refers to an “iteration.”

Toyota production system
The foundation of Lean Manufacturing, the TPS is the process that organizes manufacturing, logistics, and relationships with suppliers and customers at Toyota. Its emphasis is on quality, improving the flow of value through the product delivery process, and the elimination of waste.

User story
A system requirement written from the point of view of the user, also known as a user story. A story typically follows the format “As a <user role>, I want to be able to <function>.”

Unit Test
The activity that verifies that software code matches the design specifications. Typically, unit testing is performed by a developer – by the code creator or by a “buddy” – however, testers often advise developers in testing approaches. Each unit test confirms that the code accurately reflects one intention of the system.

Use Case
In Lean-Agile, use cases express the details for a requirement story. If the use case becomes so large that it cannot be implemented in a single iteration, then the requirement story associated with the use case can be broken down into iteration requirement stories first. Alternatively, if a use case is more abstract, it can represent several requirement stories. Use cases should be expressed in the style of Cockburn detailed use cases. They may be white box or black box and include a main scenario, exceptions, and alternatives. Use cases can refer to business rules and data definitions.

User Acceptance Test
The activity that verifies that software code matches the business intent. UAT belongs to the customer. They decide what constitutes an acceptable product. Unit tests help ensure that the acceptance tests are not about functional failures, but about the actual acceptability of the approach. Thus, UATs should not result in “it crashed” but "I would like the menus to be more descriptive.”

Validation
The testing activity that confirms, “you built what I asked for.”

Value Stream
The set of actions that take place to add value to a customer from the initial request to delivery. The value stream begins with the initial concept, moves through various stages for one or more development teams (where Agile methods begin), and on through final delivery and support.

Value Stream Manager
A person responsible for systematic management approach with immediate impact on the critical elements of a company’s value streams. Makes change happen across departmental and functional boundaries.

Value-Added
A term used to describe activities that transform input into a customer (internal or external) usable output. An activity on a project is value-added if it transforms the deliverables of the project in such a way that the customer recognizes the transformation and is willing to pay for it.

Velocity
Velocity measures how many “engineering hours” the team spends during an iteration. The following velocities are used in the planning game to determine how many stories the customer should give to the team to estimate for the iteration: • Story velocity. How many story points the team completes in an iteration. • Task velocity. Task velocity measures how many “engineering hours” the team actually can perform in an iteration.

Verification
The testing activity that confirms, “I built what I intended to.”

Vision statement
Your on-site customers should create and update a vision statement that describes what you’re doing, why you’re doing it, and how you’ll know if you’re successful. This provides important context for other reports. Post it prominently and reference it in conversation.

Voice of the Customer
The "voice of the customer" (VOC) is the term used to describe the stated and unstated needs or requirements of the customer . The voice of the customer can be captured in a variety of ways: Direct discussion or interviews, surveys, focus groups, customer specifications, observation, warranty data, field reports, complaint logs, etc.

Waste
Any activity that consumes resources and produces no added value to the product or service a customer receives. Tom and Mary Poppendieck identify waste is 1) anything that does not add customer value 2) anything that has been started but is not in production 3) anything that delays development 4) any extra features 5) making the wrong thing. Also known as muda.

Waterfall
A sequential phase-gated method for developing software (analysis, design, code, test, deploy).

Wiki
An editable intranet site where details of stories and tracking information may be recorded during development.

Work Breakdown Structure
The Work Breakdown Structure is “a deliverable-oriented grouping of project elements which organizes and defines the total scope of the project.” The WBS organizes work according to three primary groups: • Product work or activities • Environment and Team work or activities • Business work or activities

Work Item
In some Agile development tools, a record used to track an assignment and state of work. User Stories, Features, Bugs, Tasks are all examples of work items that may be tracked.

Working software
Software that has been coded and tested and then reviewed by the customer or product owner; software that's ready for implementation.

Work in Progress (WIP) AKA Work in Process
Any work that has not been completed but that has already incurred a capital cost to the organization. Any software that has been developed but not deployed to production can be considered a work in progress.

XP
A software development methodology adhering to a very iterative and incremental approach. XP consists of a number of integrated practices for developers and management; the original twelve practices of XP include Small Releases, Acceptance Tests, On-site Customer, Sustainable Pace, Simple Design, Continuous Integration, Unit Tests, Coding Conventions, Refactoring Mercilessly, Test-Driven Development, System Metaphor, Collective Code Ownership, and Pair Programming.

Yesterday's Weather
The expectation that a team will complete as many story points worth of work in the next iteration as they did in the last. Of course, this will only be true after they have done a few iterations and have hit a somewhat steady level. Typically this is after 3-4 iterations. The term comes from the fact that before weather satellites was the most accurate way to predict weather was to say it would be the same as the day before. Hence, "yesterday's weather" means we expect what happened before.