**Documentation on Agile Model,Scrum framework and Story Pointer**

### Why Agile?

Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a potentially shippable product increment. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology is described as “iterative” and “incremental.” In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development — requirements, design, etc. — is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there’s always time to steer it in another direction.

The results of this “inspect-and-adapt” approach to development greatly reduce both development costs and time to market. Because teams can develop software at the same time they’re gathering requirements, the phenomenon known as “analysis paralysis” is less likely to impede a team from making progress. And because a team’s work cycle is limited to two weeks, it gives stakeholders recurring opportunities to calibrate releases for success in the real world. Agile development methodology helps companies build the right product. Instead of committing to market a piece of software that hasn’t even been written yet, agile empowers teams to continuously replan their release to optimize its value throughout development, allowing them to be as competitive as possible in the marketplace. Development using an agile methodology preserves a product’s critical market relevance and ensures a team’s work doesn’t wind up on a shelf, never released.

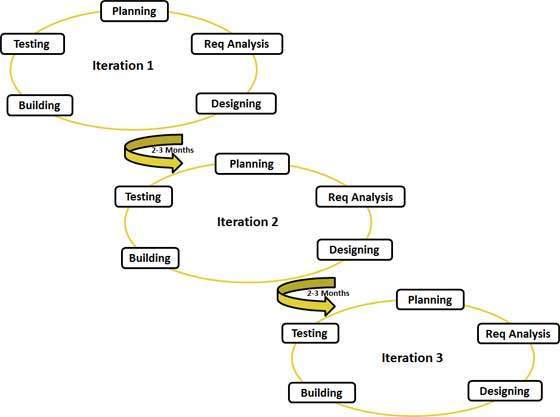
**What is Agile Model ?**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product

Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

At the end of the iteration a working product is displayed to the customer and important stakeholders.

Graphical representation of Agile model



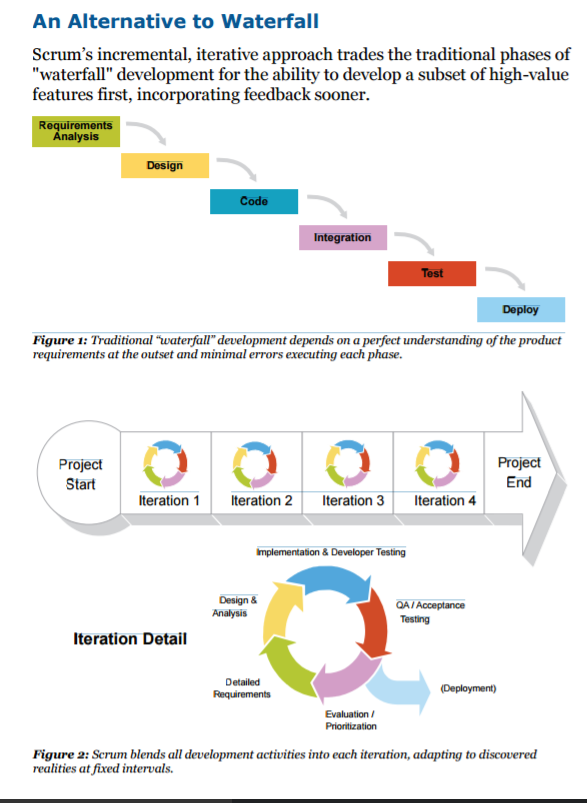
Agile Principles:

* **Individuals and interactions** - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
* **Working software** - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.
* **Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
* **Responding to change** - agile development is focused on quick responses to change and continuous development.

**What is Scrum?**

Scrum is the most popular way of introducing Agility due to its simplicity and flexibility. Because of this popularity, many organizations claim to be “doing Scrum” but aren’t doing anything close to [Scrum’s actual definition](http://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf). Scrum emphasizes empirical feedback, team self management, and striving to build properly tested product increments within short iterations. Doing Scrum as it’s actually defined usually comes into conflict with existing habits at established non-Agile organizations.

Scrum is a management framework for incremental product development using one or more cross-functional, self-organizing teams of about seven people each. It provides a structure of roles, meetings, rules, and artifacts. Teams are responsible for creating and adapting their processes within this framework. Scrum uses fixed-length iterations, called Sprints, which are typically two weeks or 30 days long. Scrum teams attempt to build a potentially shippable (properly tested) product increment every iteration.



The Scrum approach to agile software development marks a dramatic departure from waterfall management. Scrum and other agile methods were inspired by its shortcomings. Scrum emphasizes collaboration, functioning software, team self management, and the flexibility to adapt to emerging business realities.

Just to get more focus on scrum meetings and scrum actors in detail please refer the following link

<http://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf>

## [Scrum Effort Estimation and Story Points](http://scrummethodology.com/scrum-effort-estimation-and-story-points/)

Story point is a arbitrary measure used by Scrum teams. This is used to measure the effort required to implement a story.

In waterfall, managers determine a team member’s workload capacity in terms of time. Managers ask selected developers to estimate how long they anticipate certain tasks will take and then assign work based on that team member’s total available time. In waterfall, tests are done after coding by specific job titles rather than written in conjunction with the code. The downsides of waterfall are well known: work is always late, there are always quality problems, some people are always waiting for other people, and there’s always a last minute crunch to meet the deadline.

crum teams take a radically different approach. First of all, entire Scrum teams, rather than individuals, take on the work. The whole team is responsible for each [Product Backlog Item](http://scrumreferencecard.com/scrum-reference-card/#Scrum-Artifacts). The whole team is responsible for a tested product. There’s no “my work” vs. “your work.” So we focus on collective effort per Product Backlog Item rather than individual effort per task. Second, Scrum teams prefer to compare items to each other, or estimate them in relative units rather than absolute time units. (Ultimately this produces [better forecasts](http://scrumreferencecard.com/MacroMeasurementWhitepaper.pdf).) Thirdly, Scrum teams break customer-visible requirements into the [smallest possible stories](http://www.open.collab.net/nonav/community/swp/training/BacklogRefinementMeeting/BacklogRefinementMeeting.htm), reducing risk dramatically. When there’s too much work for 7 people, we organize into [feature teams](http://less.works/less/structure/feature_teams.html) to eliminate dependencies.

What does the process of estimation look like? Scrum does not prescribe a single way for teams to estimate their work. The only estimate that’s defined by Scrum is whether a team will attempt a PBI this Sprint or not, as decided in the [Sprint Planning Meeting](http://www.open.collab.net/nonav/community/swp/training/SprintPlanningMeeting/SprintPlanningMeeting.htm).

In the [Backlog Refinement Meeting](http://www.open.collab.net/nonav/community/swp/training/BacklogRefinementMeeting/BacklogRefinementMeeting.htm), the [team](http://scrumreferencecard.com/scrum-reference-card/#Scrum-Roles) sits down with the [Product Owner](http://scrumreferencecard.com/scrum-reference-card/#Scrum-Roles) to discuss the stories toward the top of the [Product Backlog](http://scrumreferencecard.com/scrum-reference-card/#Scrum-Artifacts). The Product Owner often wants effort estimates to help evaluate ROI, effectively prioritize items in the Product Backlog, and [predict which items will be ready by a given release date](http://scrumreferencecard.com/MacroMeasurementWhitepaper.pdf). So the Product Owner wants an honest appraisal of how difficult work will be.

To gather a cross section of viewpoints despite the different personalities on a team, it is often useful for all team members to disclose their estimates simultaneously. Individuals show their cards at once, inspiring the term “planning poker.” Individuals will usually choose different cards. This should trigger discussion of the implementation approach, clarification of the requirement with the Product Owner, and splitting the requirement into smaller stories (some of which will be lower priority). Often several rounds are necessary to arrive at a single effort estimation that reflects the entire team’s sense of a story’s difficulty. The refinement and clarification are more important than the estimates themselves. According to the Agile Manifesto, business people and developers must work together daily throughout the project.

**#References**

http://www.collab.net/services/training/agile\_e-learning

http://agilemethodology.org/

http://www.collab.net/sites/default/files/uploads/CollabNet\_scrumreferencecard.pdf

http://scrummethodology.com/scrum-effort-estimation-and-story-points/