Agile Methodology Guidelines

Version No: 1.0

Date: May 03, 2017

Revision History

Version	Date	Prepared by / Modified by	Change Summary	Approved By	Approved On
1.0	May 03, 2017	Sridhar	First Draft Made	Anitha T G	May 18, 2017

Table of Contents

1. Introduction to Agile

During 90's some of the software programmers has decided to break away the traditional approaches of software development and move towards more flexible development style. Agile is then introduced into the industry. In 2001 few agile practitioners formed a core team and introduced the Agile Manifesto with four principles as in section 2.

Agile is a framework developed based on the iterative and incremental development approach of software engineering. It promotes evolutionary development of software system using time boxed iterative approach. Agile encourages rapid and flexible responses to changing requirements.

1.1. Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and Interactions *over* **Processes and Tools**

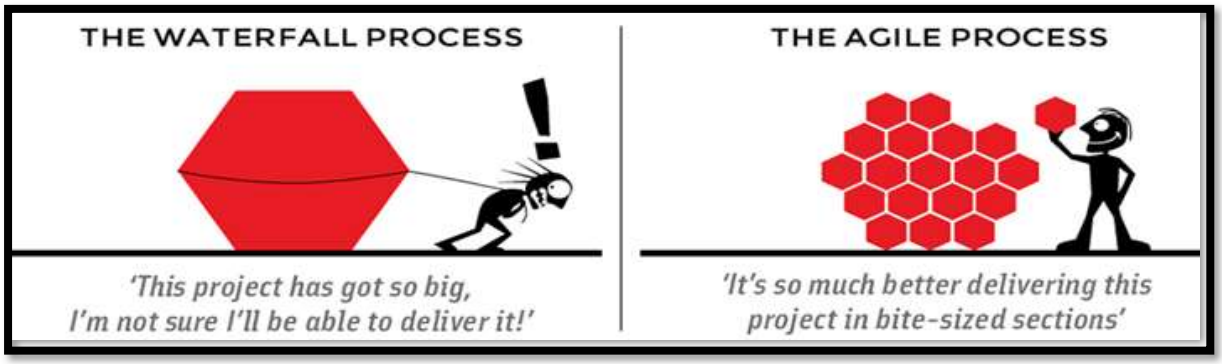
Working Software *over* **Comprehensive Documentation**

Customer Collaboration *over* **Contract Negotiation**

Responding to Change *over* **Following a Plan**

That is, while there is value in the items the right, we value the items on the left more.

1.2. Waterfall Vs Agile



2. Agile Principles

- Our higher priority is to satisfy the customers through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months of horter timescale.

- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is fact-to-face conversation.
- Working software is the primary measures of progress.
- Agile processes promote sustainable development. The sponsor, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity – the art of maximizing the amount of work not done – is essential.
- The best architectures, requirements and designs emerge from self – organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

3. Different Agile Methods

Below table provides different agile methodologies majorly heard in the industry. The table also includes the conditions that make the respective method more suitable.

Methodology	Description	Criteria for This Method

SCRUM	• SCRUM is designed for teams of 3 to 9 developers who break their work into tasks that can be completed within time-boxed iterations, called sprints (typically two-weeks) and track progress and re-plan in 15-minute stand-up meetings, called daily scrums.
	• This methodology is more aligned to project management methodology
	• Major Activities in SCRUM
	  ○ Create product backlog
	  ○ Create sprint backlog
	  ○ Sprint planning
	  ○ Daily Scrum
	  ○ Sprint development
	  ○ Customer Demo
	  ○ Sprint Retrospective
		• Customers will change their minds about what they want or need - requirements volatility
		• Unpredictable challenges—for which a planned approach is not suited

XP (Extreme Programming)	• Advocates frequent "releases" in short cycles, intended to improve productivity and new customer requirements can be adopted.
	• This methodology is more aligned with software engineering methodology
	• Engineering principles adopted includes:
	  ○ Pair Programming
	  ○ Extensive code reviewed
	  ○ Extensive Testing
	  ○ Avoid programming until needed
	  ○ Code simplicity
	  ○ Frequent communication
	  ○ Test first development
	• Life cycle of XP includes five phases
		• Emphasis on priority of customer/client stories for each system release
		• Co-located teams

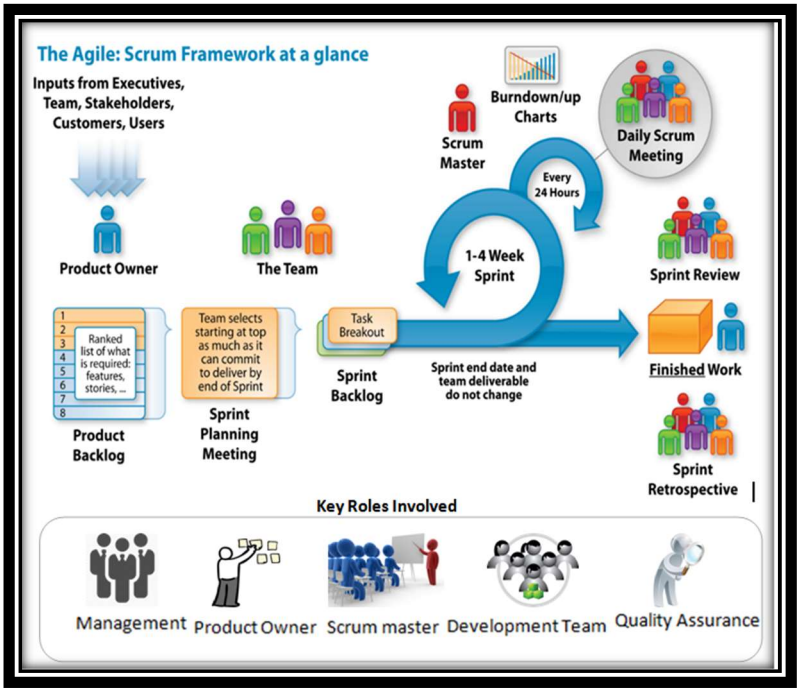
	○ Exploration Phase ○ Planning Phase ○ Release Phase ○ Production Phase ○ Maintenance Phase	
FDD (Feature Driven Development)	• The focus is more from the client value perspective i.e., features • FDD is a model driven short iteration process that consists of five activities • The first two activities establish the overall shape and the final three activities are iterated for each feature ○ Develop Overall Model – establish the scope and create the domain models by small groups. From the created models the team will select the best model for each domain. ○ Build the feature list – identify the features by functionally decomposing the domain into subject areas followed by activities / steps which are the basis for features. Features are small pieces of functions expressed as <action> <result> <object>. ○ Plan by feature – produce the development plan and assign the ownership of features as classes to programmers. ○ Design by feature – A chief programmer selects a small group of features that are to be developed within 2 weeks. Develop detailed sequence diagrams. Complete the classes and inspection. ○ Build by Feature – The class owners develop the code, conduct the unit testing, code inspection and integrate with main build.	• This model takes the advantages from XP and SCRUM and combines them with model centric techniques including domain driven design • It is scaled to large teams due to its just enough design initially (JEDI).
Kanban	• Kanban is a lean method to manage and improve work across human systems. This approach aims to manage work by balancing the demands with available capacity and improve the system capacity by addressing bottlenecks. • Work items are visualized to give participants a view of progress and process, from start to finish usually via a Kanban board. • Kanban focuses on the customer and work which meets the needs rather than individual activities. • Kanban has six general practices ○ Visualization ○ Limiting work in progress ○ flow management ○ making policies explicitly ○ using feedback loops ○ collaborative evolutionary	• This is more suitable for software maintenance environments. • Work is coming as stream of small enhancements or changes. • Value add is more important and elimination of waste is also the main focus

4. Agile Methodology

The life cycle of agile project is defined according to project nature, duration and target delivery time of the software system being developed. The project is planned into multiple releases and time boxes (are called as sprints) and the nature of sprint and the number of sprints will vary depending on the project environment and complexity. In agile each time box or a set of time boxes will ensure the working

software delivered to the customer. In each time box there will be activities involving requirements elicitation/analysis, design, coding and testing. At the end of iteration there will be customer demo to gather the feedback and a retrospective meeting to analyze the improvement opportunities.

Below diagram depicts the typical agile methodology of SCRUM approach. As mentioned in section-4 there are multiple methodologies available in Agile and we are going to discuss about SCRUM methodology in detail below.



- **Product Backlog –** Prioritized product backlog is created and maintained for upcoming releases/sprints. Stories are reviewed to ensure that they are clear and adequately defined without any conflict. Spikes and Epics are classified so that the appropriate strategy can be applied before execution. Various requirements elicitation techniques like story writing workshops, brainstorming, prototypes, wire-frames etc. can be exercised.
- **Estimation -** This process ensures sizing of user stories into story point and further estimation in efforts. Estimations can be done using Planning Poker and updated in product backlog. Efforts are updated in sprint backlog.
- **Release Planning –** Release plan is created while preparing the project management plan. The release plan contains the number and date of release, what functionality to be delivered during each release, Acceptance criteria for the release. If available, the numbers of sprints that gets into each release.
- **Sprint Planning –** At the start of the sprint the team with the product owner will identify the user stories that are part of the sprint. Identify the tasks for each user story and confirm the user

stories that can be delivered considering the velocity of the team. Velocity is termed as the actual user stories delivered within the same duration of sprint with the same resource count.

- **Daily Stand-up** – This meeting happens every day for 15 minutes at the same time where each team members will answer the following three questions
  - What is been accomplished since last meeting?
  - What is planned before next meeting?
  - Are there any impediments?
- **Sprint Management –**This Process describes set of activities carried out during sprint planning, sprint execution, sprint review and retrospection.
- **Sprint Execution –** Story requirements are converted into working code/software. Agile coding practices shall be elaborated in this process. Explicit emphasis on automation of code reviews, tests and continuous integration shall be made in this process.
- **Sprint Review / Demo –** At the end of the sprints the team will demo the application developed to the product owner to get his feedback. Product owner will provide the feedback to the team and the team will address the feedback as appropriate. Any new additions or changes will be included to the product backlog for re prioritization during the sprints planning meetings.
- **Sprint Retrospective –** After the sprint review the team will sit and analyze the opportunities for improvement. The team will answer the following questions
  - What went right?
  - What could have been improved?

Based on the answer the team will agree on the improvements/ changes expected from the next sprints and come out with the actions.

4.1. Potential Benefits

- Better and effective tracking and monitoring the project as the status is been discussed every day and the project status is being monitored every day through burn down charts.
- Early discovery of potential problems / issues if any by the team as the issues are being explicitly asked and discussed daily during stand-up meeting. This will help in reducing the unprecedented risks in the project.
- Better Quality of the System as the users/customers are early involved in providing the feedback on the software application/services. User feedback is captured in every iteration and the product/application /service will be improved in the upcoming iterations based on the feedback.
- Short delivery time leading to earlier revitalization of benefits to customers.
- Flexible to changing requirements. New or modified requirements are added into product backlog and re-prioritized by the product owner in the upcoming iterations.

5. Agile Glossary

Term	Definition
Agile	Agile is an iterative and incremental software development method, where the requirements evolve through collaboration between customers and project teams. It uses an iterative approach and flexible to changing requirements.
Agile Coach	A person who mentors or assists the project teams, users and other stakeholders in adopting the Agile practices.
Burndown Chart	Graphical representation of the quantity of work remaining in terms of effort required/ size for every sprint during the project duration. The estimated effort / planned size and the actual effort /size logged will be shown on the graph for analysis.
Daily Stand-up Meeting	A meeting not more than 15 minutes will be held at the same time every day among the team members to understand quick progress of tasks finished yesterday, working today and any blockers.
Definition of Done	Defined Criteria which must be met before a user story is considered as "done". Failure to meet the criteria at the end of a sprint implies that the work is incomplete and should not be included in the velocity.
MoSCoW	MoSCoW (Must, Should, Could and Won't) is a technique that helps the customers to prioritize the user stories based on their importance or business value.
Release Plan	A release plan represents a schedule for delivering one or more releases to customer i.e. the functionality of working software to be delivered.
Story board	It is a dashboard that shows the status of each user story in every sprint during execution. It tells the scrum master and team about how many user stories are completed and how many or still pending.
Story Points	It is a measure used to denote the size of user story and is normally expressed in numbers Fibonacci numbers. Story Point is a relative measure.
Test-Driven Development (TDD)	TDD is a programming technique in which coding and testing are very tightly coupled. In this method test cases are developed much before starting the code and while doing the coding test cases will be automatically executed.
Timebox	The development time for the project is divided into iterations of fixed time interval. Each interval is called a timebox. Each user story with its associated tasks is assigned to one of these time boxes (sprints).
User Story	User stories are the requirements for the system and written by the actual users in layman's terminology. The format of a user story is: [As a <Role>, I want to <goal> so that I can <reason>]
Velocity	The speed at which the project team are working, measured in terms of the total estimated effort/size associated with the user stories that were completed at the end of a sprint.