Infosys®

Building Tomorrow's Enterprise

Infosys Global Agile Developer Certification

Agile Overview

Study Material

www.infosys.com

Revision History

Version	Date	Remarks
1.0	Dec’2014	Initial Version

Table of contents

Table of Figures

1 Introduction

Agile software development is based on the Agile Manifesto introduced in 2001, as given below:

- 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 on the right, we value the items on the left more.

There are 12 principles which are the pillars of the Agile Manifesto. They are:

- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Close, daily co-operation between business people and developers
- Face-to-face conversation is the best form of communication (co-location)
- Projects are built around motivated and trustworthy individuals
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances

Several Agile methodologies have evolved over a period of time. Agile is often misunderstood as Scrum framework which is the most popular among the Agile methodologies. There are other Agile methods such as Extreme Programming (XP), Dynamic System Development Method (DSDM), Feature Driven Development (FDD) along with Lean and Kanban which can be adopted based on the scenarios in which Agile adoption is intended.

It is always important to understand the business context in which any particular Agile methodology or a combination of Agile practices can be adopted.

Agile methodologies adopt an iterative and incremental delivery of working software. Each iteration follows a time-boxed approach where working software gets developed and delivered based on the prioritization of the requirements.

Some of the popular Agile Software Development Methodologies include:

- Scrum
- Extreme Programming (XP)
- Feature-Driven Development
- Dynamic Systems Development Method (DSDM)
- Scrumban
- Agile Unified Process (AUP)
- Agile Modeling
- Crystal
- Lean Software Development
- Kanban

In this module, Scrum, XP and Kanban methods are detailed out for the reader to understand the concepts. Scrum and XP are mostly used in Agile development scenarios whereas Kanban is used in Maintenance and production support scenarios where visibility of work in progress is key.

2 Scrum

Scrum is not an acronym, the name is taken from the sport of Rugby. It brings the analogy where Team works together to successfully develop software. Scrum is an iterative and incremental framework for application or product development. The development of the project is achieved through iterative cycles known as Sprints. At the start of each Sprint, a cross-functional team selects items from Product Backlog and commits to complete the items by the end of that particular Sprint. Every day the team gathers for a short meeting to discuss the progress and road blocks, if any. At the end of the Sprint, the team reviews the work product with stakeholders and demonstrates what has been built. The feedback is then incorporated in the subsequent Sprint. At the end of each Sprint, Scrum emphasizes that the integrated working software is fully tested, and potentially made shippable. The Sprints are strictly time-boxed and occur sequentially. The end date of a Sprint does not get extended, regardless of the completion of the work initially planned.

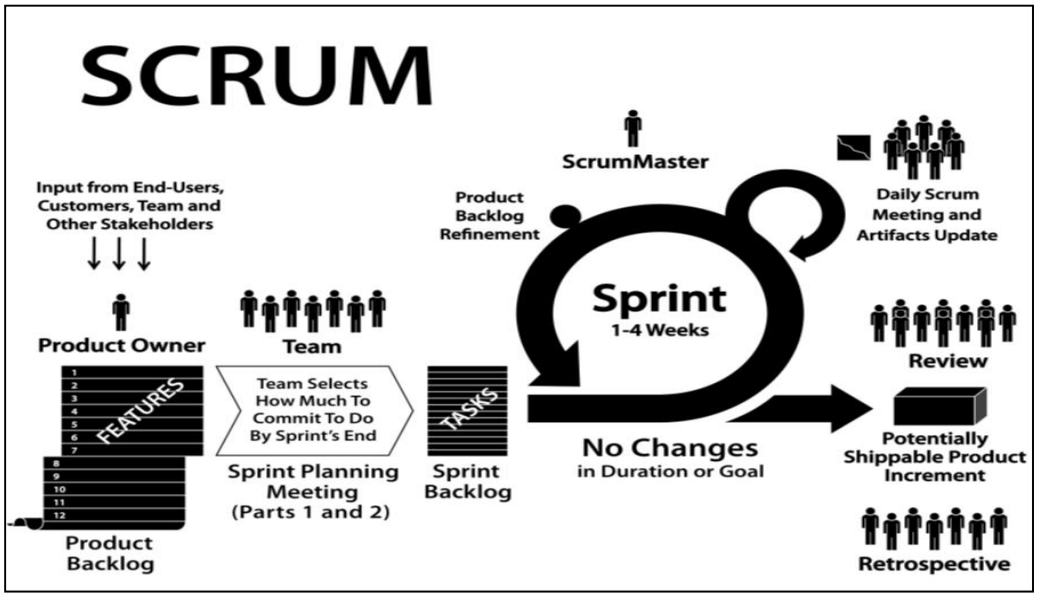


Figure 1 : Scrum

Source: Scrum Primer V1.2 by Pete Deemer

2.1 Scrum Roles

Scrum team consists of three roles: Product Owner, Team and Scrum Master. Team here refers all the members working for the project delivery. There are other contributors to the success of the project including managers, SMEs (Subject Matter Experts), architects, customers and vendors. These additional contributors facilitate the overall success of the project through their expertise and experience.

Product Owner

The Product Owner is responsible for**:**

- Providing the vision of the project to the team
- Maximizing value of the project along with the work of the Scrum Team
- Managing the Product Backlog
- Clearly articulating Product Backlog items
- Prioritizing the items in the Product Backlog so as to best achieve goals and mission of the project
- Refining the backlog whenever necessary

- Ensuring that the Product Backlog is visible, transparent and clear to all and is self-explanatory on what the team will work on next

Any team member recommending a change to a Product Backlog item's priority, has to discuss this with the Product Owner.

The Team

Team converts Product Backlog into increments of potentially shippable working software at the end of every Sprint. Team is structured, self-organized and manages its own work.

A team has the following characteristics:

- Team size in Scrum should be approx. 7 persons (five to nine is the ideal team size)
- The team is recommended to be cross-functional with skills in analysis, architecture, interface design, database design, development, testing, documentation
- Adheres to Scrum principles while working in the project
- It is self-organized and decides what to commit and how best to accomplish that commitment
- The accountability of the work product belongs to the team as a whole

Scrum Master

The Scrum Master is responsible for ensuring that the Scrum team adheres to Scrum values, practices, and rules. The Scrum Master is a Change Agent, a Coach, a Protector, a Problem Solver, a Process Owner. To put together, he can be called as "The Servant Leader".

Responsibilities of Scrum Master are as follows:

- Helps the Scrum team to adopt Scrum
- Guides the Team to success by serving the team to become knowledge-wealthy and self-organizing
- Helps the Team to learn and apply Scrum to achieve the desired objective of the project
- Facilitates Scrum events
- Coaches the Team to be cross-functional
- Removes impediments for the Team's progress

Scrum Master cannot be the Product Owner, manager of the Team or the Project Manager.

2.2 Scrum Events

Scrum uses time-boxed events to ensure that there is no wastage of time during the execution of the project. It has five events – Release Planning, Sprint Planning, Daily Standup meeting, Sprint Review and Sprint Retrospective.

2.2.1 Release Planning Meeting

At the project initiation, the Release Planning meeting focuses on the strategy to turn the vision into a successful project in best possible way. This meeting establishes the plan and goals which needs to be achieved for the upcoming Release.

There are multiple releases planned for the project based on the project objective. The release plan establishes the goal of the release, high-priority Product Backlog items, major risks and overall features and functionalities that the release will contain. A probable delivery date is agreed upon. The stakeholders can inspect the progress and make changes to the release plan on a Sprint-by-Sprint basis.

Following are the key points discussed in the Release planning meeting

- Estimate Product Backlog
- Decide the Sprint Length
- Calculate the number of sprints
- Fix the release Date

2.2.2 Sprint Planning

In the Sprint Planning meeting, the Team collaboratively plans and decides the work to be accomplished in the upcoming Sprint. It is a time boxed event consisting of two halves. In the first half, which is of three to four hours of duration (for a three/four week Sprint), the Team discusses 'WHAT' to be delivered. In the next half of similar duration, the Team discusses and decides 'HOW' to deliver the same in the Sprint. The Sprint planning meeting is proportionately shorter in case Sprint duration is less than three weeks.

Sprint Planning – First Half

In the first part of Sprint planning, the Team focuses on understanding what the Product Owner wants and what would be delivered as part of the Sprint. The Product Owner provides Product Backlog items in a prioritized way to the Team. The entire Team work together on understanding the scope of the Sprint. The inputs for this part of the meeting are

- The Product Backlog
- Projected capacity of the Scrum Team during the Sprint
- The latest product Increment (if available)
- Past performance of the Scrum Team (If available)

The Product Owner presents the prioritized Product Backlog to the Team. The Team discusses the requirements with the Product Owner and understands the acceptance criteria for each item in scope. The Team decides on the Sprint Goal in agreement with the Product Owner. The Sprint Goal ensures the Team is focused to achieve it at the end of the Sprint. If there are issues on the items in scope for the Sprint, Team discusses with the Product Owner for the reprioritization.

Sprint Planning – Second Half

After the team understands on what needs to be delivered in the Sprint, the remaining part of the Sprint Planning Meeting is to address the question of how it is going to build this functionality. The team discusses how to achieve the Sprint goal and deliver the product increment. Key discussions are on:

- Discuss the rough architecture
- Make design decisions
- Identify tasks the team needs to do
- Identify components and interfaces
- Identify dependencies
- Identify the external integration points
- Identify Data to be used and data sources
- Discuss Architectural patterns to be applied
- Discuss Testing strategy

Team starts by planning their capacity for the new Sprint i.e. by estimating how much time each member has for Sprint-related work. Based on the capacity, the team determines how many Product Backlog items they can finish in that Sprint, and the approach for completing them. The team decomposes the Product Backlog items into tasks based on the understanding of the overall design. The Team starts with the first item on the Product Backlog, such as the highest priority items and decomposes it into individual tasks, which are recorded in a document called the Sprint Backlog. The Sprint Backlog consists of the prioritized Product Backlog items for this Sprint and the plan for delivering them. Tasks must be decomposed to the granular level so they can be completed in less than one day.

During the second part of the Sprint Planning Meeting, the Product Owner presence is optional. It is required only if any clarification is required on the prioritized Product Backlog items and to help make any trade-offs on them.

The Team may renegotiate the items with the Product Owner if in case it has too much or too little work in hand. The Team may also invite other people outside the scrum team i.e. SMEs (Subject Matter Experts) to attend in order to provide technical or domain advice. By the end of the Sprint Planning meeting, the Team produces a list of all the tasks with estimates and a plan on how to accomplish the work selected.

In Agile projects, 'Definition of Done' determines conditions to be considered when a specific task or event (i.e. User Story or a Sprint or a Release) is completed, validated and is ready for acceptance. The team agrees on and displays prominently a list of criteria which must be met before a task is considered "Done". If this criterion is not met, it implies that the work should not be considered for any acceptance/approval. For completeness and integrity of deliverables, project team would generally define various definitions of done. This could be for a User Story, a Sprint and a Release, or for a technical/functional doneness.

Example: 'Definition of Done' for a User Story could have the following activities:
→ All the unit tests are implemented for new functionality and are in green
→ Code review completed, source code committed and all tests are green in CI server
→ Acceptance tests are written, executed and passed and Acceptance criteria is met
→ New integration have not caused any regression or impact and code coverage is above X%

2.2.3 Daily Stand-up Meeting

The Daily Scrum meeting is a 15-minute time-boxed event for the Team to synchronize activities and create a plan for the next 24 hours. During the meeting, each Team member explains:

- What he or she has accomplished since last meeting
- What he or she is going to do before the next meeting
- What issues/obstacles are in his or her way

Scrum Master facilitates the Daily Scrum meeting and helps the Team to get rid of the obstacles. Product Owner's presence is optional for this meeting.

The Daily Scrum Meeting helps the Team in the following way

- Promotes "Openness" as everyone shares information
- Helps to synchronize the activities and effectively plan for next 24 hours
- Helps to focus by creating an "anticipating culture"
- Helps Team to respect each other for their knowledge
- Provides enough data through updated Sprint Burn Down chart on daily basis
- Reinforces commitment

2.2.4 Sprint Review

This Meeting is held at the end of every Sprint. During the Sprint Review, the Team exhibits to all the relevant stakeholders about what was done (i.e. developed) in that particular Sprint and overall progress of the Release.

The following elements are included in that Sprint:

- The team explains (primarily to the Product Owner) what has been **"Done"** and what has not been "Done"
- The Team demonstrates what has been accomplished in the Sprint
- The Product Owner discusses the Product Backlog as it stands
- The entire group discusses on what to do next, which in turn helps the subsequent Sprint Planning Meetings

The Product Backlog may get revised as an outcome of the Sprint Review meeting.

2.2.5 Sprint Retrospective

During the Sprint Retrospective meeting, the Scrum Team inspects the performance of the Sprint which is just concluded and creates a plan on what needs to be improved for the upcoming subsequent Sprints. This Retrospective meeting is done post the Sprint Review meeting and before the next Sprint begins. The purpose of having a Retrospective meeting at the end of each Sprint is to:

- Inspect the Sprint which is just concluded with regards to process, tools, people, and relationships
- Identify and prioritize the practices that went well and those practices that if done differently, could make things even better. These include team composition, tools, definition of “Done”, meeting arrangements, Communication methods and effectiveness, and processes for turning Product Backlog items into something “Done.”
- Plan for implementing the identified improvements

Following are the steps for the Sprint Retrospective:

- Setting the Stage
- Gather Data
- Generate Insights
- Decide What to Do
- Close the Retrospective

2.3 Scrum Artifacts

Scrum artifacts represent work or value in various ways that are useful in providing transparency and opportunities to inspect and adapt. Following are the Scrum Artifacts:

2.3.1 Product Backlog

The first step in Scrum is for the Product Owner to articulate the product vision. This evolves into a Product Backlog (PB) which is a refined and prioritized list of features. PB can also be defined as “An ordered and emerging list of user needs (User Stories) plus anything else that is required to fulfill the Product Vision”. The Product Owner is responsible for the Product Backlog, its contents, its availability, and its prioritization. Only a single Product Backlog exists and is refined and updated through the lifetime of the project.

The Product Backlog includes the requirements of the project. It can be articulated in any way that is clear and sustainable. User stories” are commonly used to describe the Product Backlog items in terms of their value to the end user of the product. The Product Backlog is continuously updated by the Product Owner to reflect changes in the needs of the customer, new ideas, issues and risks that appear.

A Good Product Backlog Should be DEEP:

- Detailed enough – To make the requirements very clear.
- Emergent – The Product backlog keeps emerging through new ideas and continuous feedbacks from the customers.
- Estimated – The User stories must have the relevant points already.
- Prioritized – The User stories must be ordered based on their importance / priority.

2.3.2 Release Burn down

The Release Burn down graph records the sum of remaining Product Backlog estimated work across time. The X-axis contains the number of Sprints for the release whereas the Y axis contains the effort remaining in the release. The Release Burn down depicts the progress of the release on a real time basis.

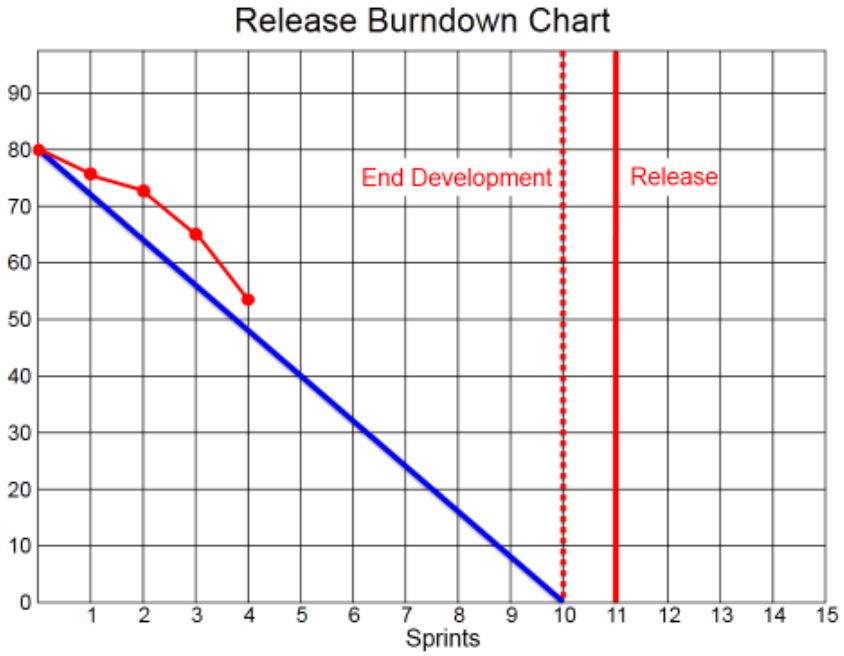


Figure 2: Release Burndown (*Reference from Scrum Primer V1.2 by Pete Deemer)*

2.3.3 Sprint Backlog

The Sprint Backlog is a highly visible, real-time picture of the work that the Team plans to accomplish during the Sprint. The Sprint Backlog is the set of Product Backlog items and associated tasks the Team performs to turn Product Backlog items into a “Done” increment. It is all of the work that the Team identifies as necessary to meet the Sprint goal.

The Sprint Backlog gets updated by the Team throughout the Sprint by updating the remaining effort for the tasks, completion of the tasks and modification of tasks

.At any point in time in a Sprint, the total work remaining in the Sprint Backlog items can be summed. The Scrum Team tracks this total work remaining at least for every Daily Scrum.

2.3.4 Sprint Burndown

Sprint Burndown is a graph that represents the amount of Sprint Backlog work remaining in a Sprint across number of days in the Sprint.

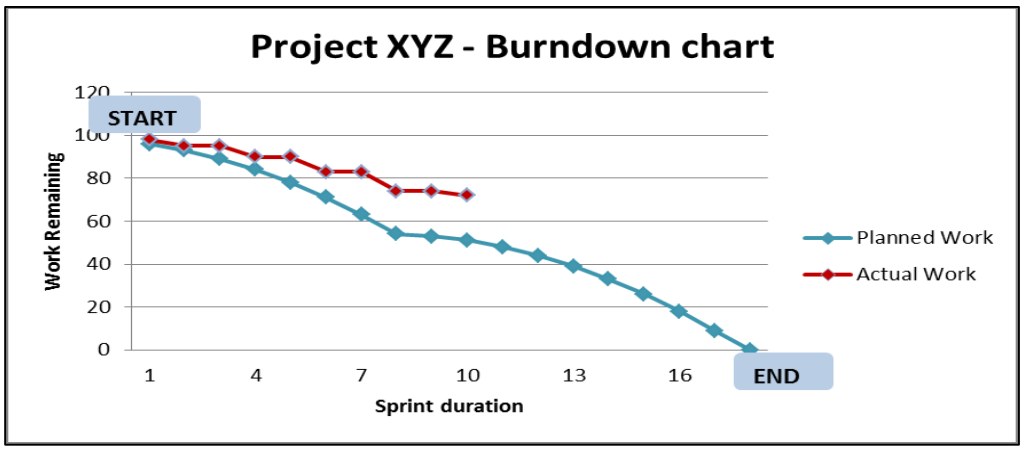


Figure 3: Sprint Burndown *(Reference from Scrum Primer V1.2 by Pete Deemer)*

The “Y” axis shows the remaining effort required to complete the work planned and the “X” axis contains the number of days until the iteration deadline. The remaining effort is determined by summing the time estimates for incomplete tasks.

Team members must update their actual amount of time remaining for the completion of their task in the Sprint Backlog. The Sprint Burndown Chart is plotted accordingly based on this update. It shows the Team their progress towards their goal in terms of how much work remains in the future. If the actual line of the Burndown chart is behind (i.e. Lagging) the Ideal line, then it indicates that the team is not on schedule or as per the plan. The Team needs to adjust, such as to reduce the scope of the work or to find a way to work more efficiently while still maintaining a sustainable pace

3 Extreme Programming (XP)

Extreme Programming (XP) is one of the popular Agile methodologies and it is mostly used along with Scrum framework to create a blend of both engineering and management practices in Agile projects. The term XP is conceptualized from the execution of traditional software engineering practices at extreme levels.

For example, code review is a practice which improves code quality. To manifest this practice at an extreme level, code review is performed as and when the code gets developed. Pair Programming, refactoring and test-driven development are some of the practices which have been evolved from these concepts.

Extreme Programming is widely known for its technical practices which can help developers to build high quality software, more productively. There are 12 practices which fall under four key areas:

1. Fine Scale Feedback
2. Continuous Process
3. Shared Understanding
4. Programmer Welfare

Proper adoption of all Extreme Programming practices requires great level of discipline, teamwork and skill.

Extreme Programming recommends short Sprints, which span around 1 to 4 weeks, and the project artifacts include story cards, code and unit tests.

3.1 Extreme Programming Practices

3.1.1 Fine Scale Feedback

- **Planning Game:** The key planning process of Extreme Programming is called planning game to ensure work is planned incrementally. It is divided into two parts –
    - Release Planning
    - Iteration Planning

    ➔ **Release Planning:** This is a planning meeting to shortlist requirements to be included in the releases planned and timeline of each release. It has three phases:

    ➔ **Iteration Planning:** This is a planning meeting by the Team to decide upon the tasks and activities, to accomplish requirements planned for the iteration.

- **Pair Programming:** Extreme Programming emphasizes on each practice done at its extreme. Coding is generally followed with code review, and in the extreme case, coding and review can happen in parallel. Pair

Programming practice entails the same wherein two persons work on the same code. While one programmer is doing the coding, focusing on the code and program level details, the other programmer has the big picture and continuously reviews the code. The pairs are not fixed and keep on changing thereby helping everybody to be aware of the entire system.

- **Test Driven Development:** Test Driven development is an Extreme Programming practice in which, developer writes the automated test cases before the actual code is written. Automated unit test cases should inevitably fail at the first attempt of run. If it does not fail, it indicates the feature already exists or there are some problems with the automated test case. Then the code is written and the automated test suite is run. Based upon the result, the code is modified. The units are usually kept small and code is refactored as and when required.

- **Whole Team:** In Extreme Programming, the entire Team functions as a whole. Team members are encouraged to be more generalized than specialized. Customers are included as part of the Team who are always available to respond to the queries and provide clarifications.

3.1.2 Continuous Process

- **Continuous Integration:** Continuous Integration (CI) is an Extreme Programming practice where the code base is integrated on a frequent basis from the start of the feature development. It is an automated process in which builds are created from the common source code repository. Continuous Integration helps in identifying integration issues much ahead in the lifecycle and hence, reduces much of the rework cost. Tools are available to facilitate the CI process by accepting the checked-in latest code, prepare the build and monitor the build duration and failures, if any. This helps in enabling the delivery of Short or Small releases to the clients/customers.

- **Refactoring:** It is the practice of regular improvement of existing design and code without affecting the functionality. Agile development involves frequent releases over a span of time which may introduce design flaws, duplicate code and so on. Therefore, refactoring as a practice helps to improve the overall code quality. This practice prompts developers to proactively improve the product as a whole and this also increases the developer's knowledge of the system.

- **Small Release:** Frequent and small releases is another XP practice which is common across various Agile development methodologies. The overall software development does not happen in one go; rather smaller releases are planned which provide customer the confidence of the overall progress. Also, it helps customer to provide suggestions for any changes and helps to meet the overall goal. The term "Small" here means "Less functionality". Less Functionality means releases happen more frequently to the customers.

3.1.3 Shared Understanding

- **Coding Standards:** In order to reduce defects in coding, protocols consisting of style and format for the source code are agreed upon by the Team. Rules need to be as per the standards defined by language vendors and/or customized by the Team. From developer's perspective, the coding style should be the same even when two or more programmers are being involved in the code development which means it should never be possible to identify "who coded which or what piece of code".

- **Collective Code Ownership:** Pair Programming practice of XP brings in the collective ownership of the overall code. Since the Team is cross-functional and entire team works in pairs which keep changing, all parts of the code are known to everybody. In case of error, any programmer is allowed to change the code. It may also introduce error because of unforeseen dependencies. To take care of such issues, there should be exhaustive well-defined unit tests available. This promotes developers to take responsibility for the system as a whole and not limit themselves to parts of the system.

- **Simple Design:** XP recommends the simple and best way to implement code. Refactoring also facilitates this process of achieving simple design.

- **Metaphor:** It is usually a naming convention which makes all the stakeholders understand what the functionality is about.

3.1.4 Programmer Welfare

- **Sustainable Pace:** Team members should work on a sustainable pace to meet project deadlines. It helps to achieve repeatable, predictable and consistent delivery

3.2 XP Roles

Customer:

- Creates user stories and prioritizes user stories for each release
- Addition/Modification/Deletion of user stories for each release

Programmer:

- Estimates along with the Team
- Builds the users stories as per the standards

Coach:

- Monitors XP process implementation and issue resolution on XP practices
- Mentors team members

Tracker:

- Monitors the progress and alerts the Team

3.3 XP Activities

There are four basic activities recommended by XP for software development which are coding, testing, listening and designing.

Coding

- Most important activity of Software Development
- Used to find out the software solution
- Used to communicate thoughts among programmers
- Working product is nothing but code

Testing

- Focus on Test Driven Development
- Through test of every piece of code through automated Unit Testing
- Continuous Integration Testing
- Acceptance Testing

Listening

- Listen to clients to understand business logic
- Effective Listening through planning game

Designing

- Creating the design structure to organize logic of the system
- Good design practices

4 KANBAN

A Kanban is basically a signaling device that instructs the moving of parts in a 'pull' production system, developed as part of the TPS (Toyota Production System). The main aim of Kanban is to minimize WIP (Work-In-Progress), or inventory, between processes by making sure upstream process creates parts only if its downstream process needs it.

The Kanban method is an approach to evolutionary and incremental systems and process change for organizations. Work-in-progress limited pull system is used as the central mechanism to uncover system operation (or process) complications and encourage collaboration to improve the system continuously.

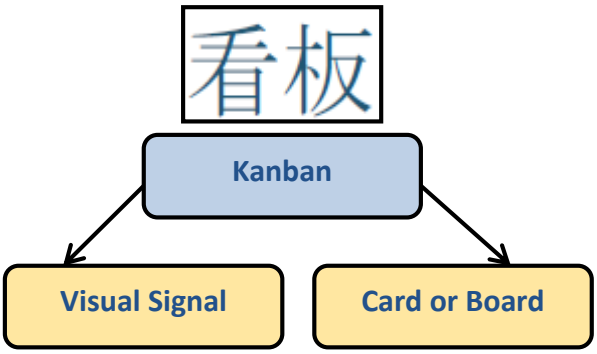


Figure 4: KANBAN

Kanban technique is based on the below mentioned principles:

- Begin with whatever you do currently
- Agree to pursue evolutionary and incremental changes
- Admire current roles, process, responsibilities & titles
- Leadership at all levels

4.1 Kanban Practices

Following is a brief of Kanban Practices as per literature:

Limit WIP

Limiting Work-In-Process (WIP) implies that a pull system is executed on either parts or the whole workflow. It (PULL system) will act as one of the key stimuli for incremental, continuous and evolutionary changes to the system. Limit WIP assigns explicit limits to the number of items that may be in progress at each workflow state.

Visualize

Visualizing the work-flow and making it visible is essential in order to understand how work proceeds. Without understanding the flow of work, incorporating the right changes is difficult. Usually, a card wall with columns and cards is used to visualize the flow of work. Different states or steps in the workflow are represented by the columns on the card wall.

Manage flow

Flow of work through every state in the workflow should be observed, measured and informed. By managing the flow vigorously, the incremental, continuous and evolutionary modifications to the system can be assessed to have negative or positive effects on the system.

Improve Collaboratively, Evolve Experimentally

Kanban encourages small incremental, continuous and evolutionary changes. Whenever teams have a common understanding of concepts about work, process, workflow and risk, they are more likely to be able to form a shared understanding of a problem and suggest enhancement actions which could achieve a consensus.

Implement Feedback Loops

Early feedbacks from client and the pull system are important in Kanban. Feedback from different stakeholders and processes helps to eliminate risk and optimize delivery process.

Make Policies Explicit

Until the mechanism of a process is not made clear, it is difficult to hold a debate and discuss ways to improve it. Without a clear understanding of how work is truly done and how things actually work, any conversation of complications tends to be anecdotal, emotional and subjective. With clear understanding, it is possible to hold a more rational, empirical, objective discussion of issues. It is more likely to facilitate consensus around improvement suggestions.

4.2 Principles of Kanban

Kanban is based on four key principles which are mentioned below.

Start with existing process

It is a change management method which starts with the existing process. Changes are done in the system in incremental and evolutionary way. Unlike Scrum, there is no specific process or roles defined in Kanban.

Agree to pursue incremental, evolutionary change

After starting with the existing process, the team must agree for continuous, incremental and evolutionary change. The changes should be small and incremental. Rapid and substantial changes may be effective but it will be subjected to larger resistance as well by the Team.

Respect the current process, roles, responsibilities and titles

Though Kanban suggest continuous incremental changes in the process, it respects current roles, responsibilities and job titles. This helps for the Team to gain the confidence as they get started with Kanban.

Leadership at all levels

Kanban does not expect leadership from a specific set, rather the actions of leadership at all levels in the organization, are very much encouraged.

4.3 Kanban Approach

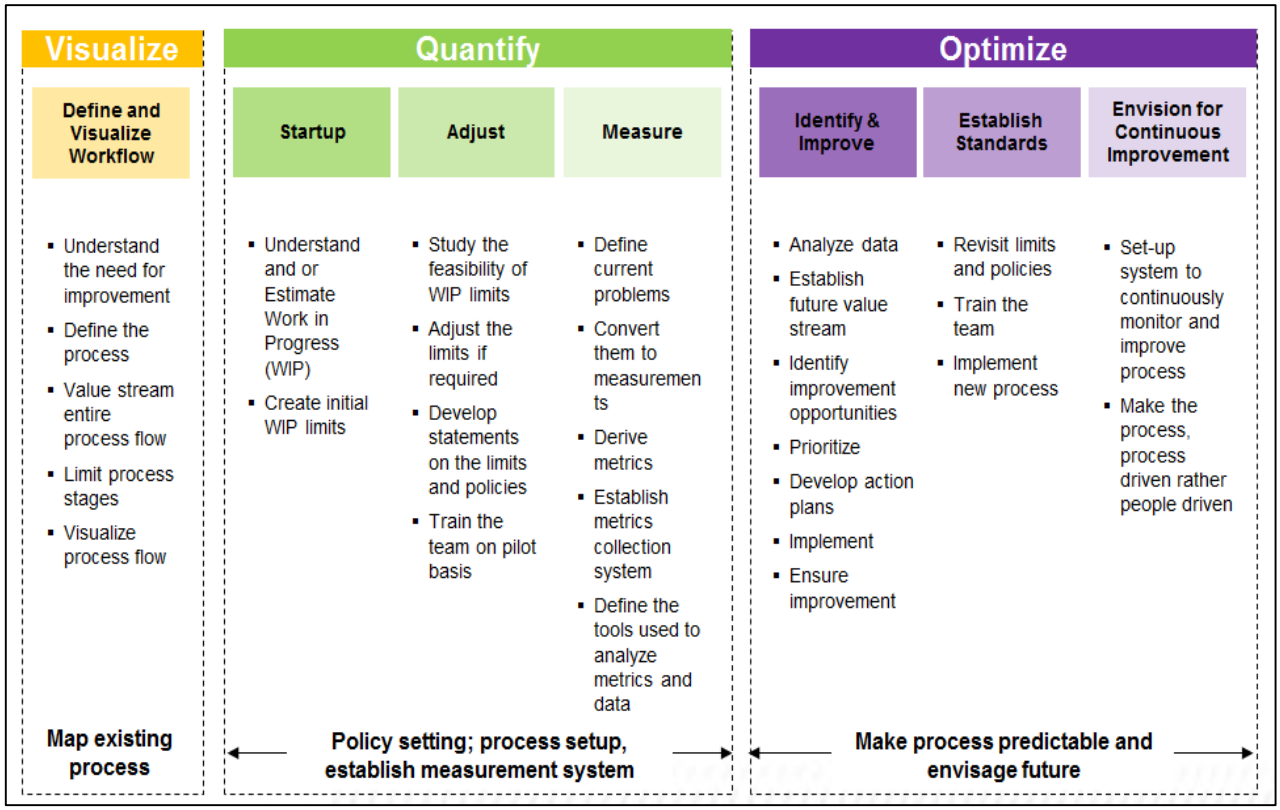


Figure 5: Kanban Approach

4.4 Kanban Benefits

- Introduction of process, visualization and optimization
- Limiting WIP and establishing policies will bring greater focus on quality thus achieving customer satisfaction
- Support for Just-In-Time planning and execution for application maintenance teams
- Predictable lead times based on WIP (Work in progress) limits allow committing to SLAs (service-level agreements) and make realistic release plans, based on critical business needs
- Focus on maximizing the flow of work, based on continuous planning and execution
- Use transparency to drive process improvement
- Organizational maturity improves leading to improved decision making and better risk management
- Minimized Waste
- Less process overhead
- More accurate and predictable pace ensures team members are never overloaded
- Enables fast reprioritization so as to accommodate changes according to the market demand
- Better estimated lead times and thereby effort required
- Flexibility to react to continuous changing priorities of work items

5 Case Study

Project Description

- The project involved an asset management web application
- The application had multiple business groups as stakeholders and users across the world
- The application was built on legacy platform that required high maintenance
- The project ran in waterfall model and all new features were delivered based on the schedule defined using traditional waterfall model. The four major stakeholders had different expectations from the applications as well as future requirements road map.

Project Delivery Process

- Project followed standard waterfall model
- The release cycles were between 4-6 months, based on amount of work needed
- Often the business would change requirements based on new product releases or feedback from application users worldwide
- Business(i.e. Client) were to be involved in giving initial requirements and then during UAT

Issues and Business Concerns

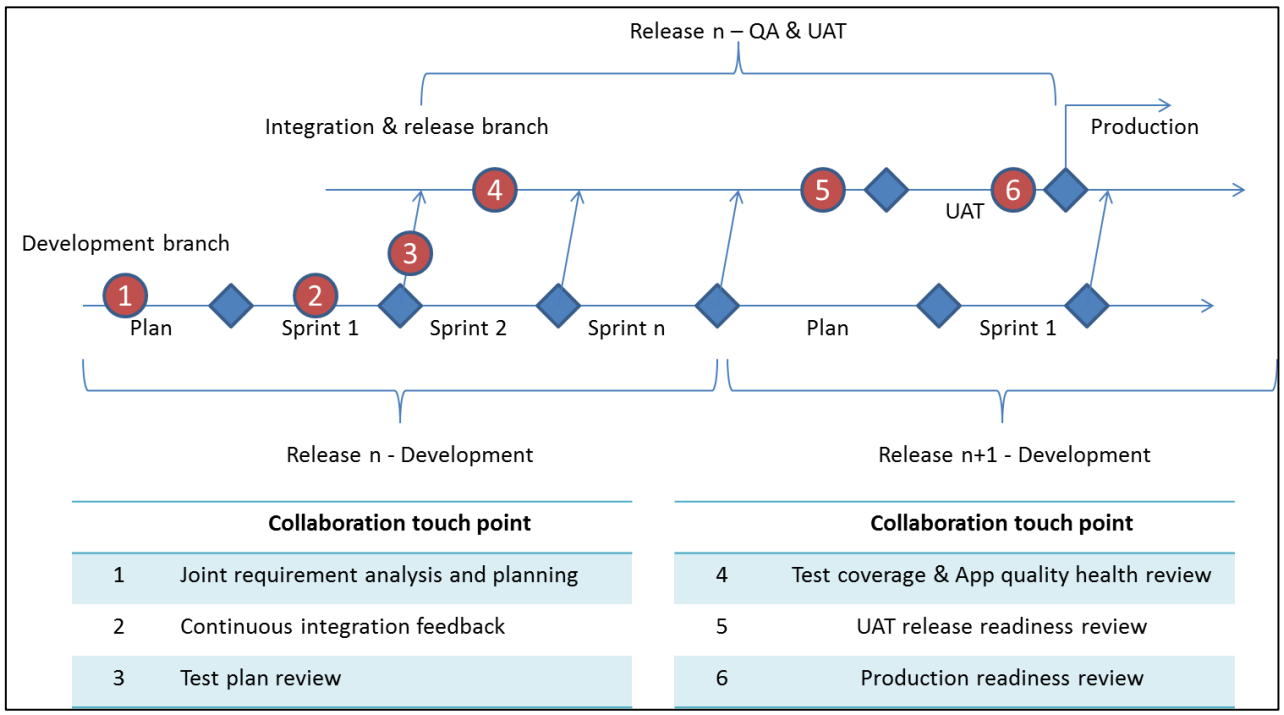
- Release cycles were long and this sometimes delayed the release of key and useful features
- Business feedback was minimal during application change process and Business would often give feedback during UAT that would lead to more changes and further delays
- QA was involved at very later stages and often needed handholding by development teams to understand scope of changes and to plan testing and regression for existing features
- As multiple stakeholders provided their respective requirements in isolation, the application comprised many duplicate and overlapping features that caused unnecessary increase in clutter

Transformation to Agile

- Transition to Agile was the only way out to bridge the gap between what business needed and when
- The journey of transition started with a plan to build a new platform with participation of business through a Product Owner, who would define and drive business requirements
- The transition was done in the following steps:
    - Scrum Master was identified for the project
    - Agile trainings/workshops were conducted for both, Scrum team (Developers, QA, UI leads) as well as Business Users
    - All current features were documented as User Stories in JIRA, to build a Product Backlog (PB)
    - All new requested features were also added as User Stories in JIRA, to enhance the PB
    - 3 weeks workshop was conducted with Business Users of all 4 groups along with Product Owner and Scrum Master, to prune the PB and remove duplicate requirements, consolidate them as well as remove based on future roadmap
    - Team used MoSCoW methodology to prioritize ‘Must have’, ‘Should have’ and ‘Nice to have’ requirements and the ones that need not be considered at all

- After this workshop, Team had a prioritized Product Backlog and stories that could be taken up by the scrum team.
- Two-week sprints were planned, with demos to business after every two weeks to collect early feedback
- Scrum team, in conjunction with Product Owner, decided on stories to be taken for each sprint and Product Owner clarified all by taking inputs from Business users and helped finalize ‘Done’ criterion for each story

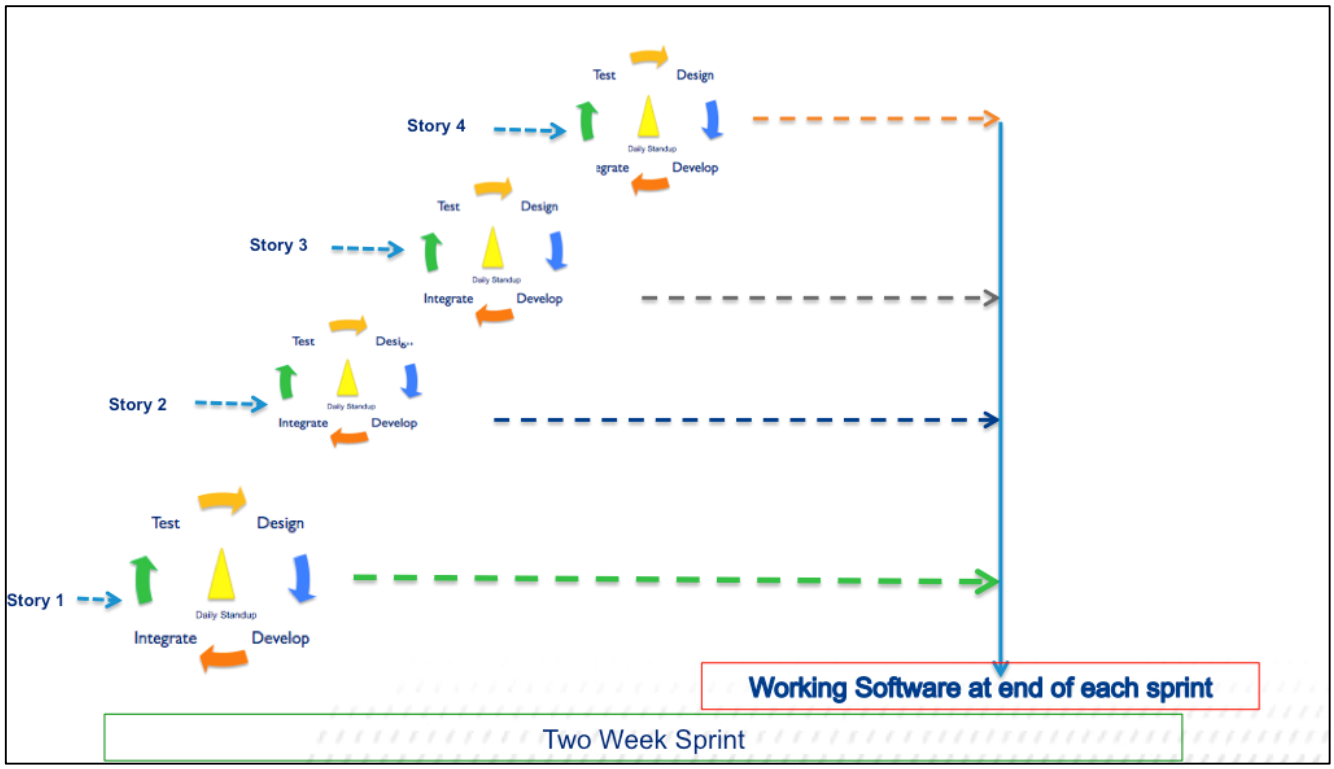
- QA team was involved in requirement clarifications and discussions on design. This early involvement helped the QA team to write more effective test cases



	Collaboration touch point		Collaboration touch point
1	Joint requirement analysis and planning	4	Test coverage & App quality health review
2	Continuous integration feedback	5	UAT release readiness review
3	Test plan review	6	Production readiness review

Figure 6: Dev and QA collaboration Model

- QA team could do early automation and build a strong regression suite as well as test each story to build a stable application and provide early feedback to developers



Advantages of changing the approach

1) Release cycles became more predictable, frequent and in line with business requirements
2) Clients were involved early into the application development, which enabled soliciting timely feedback
3) Testing was done in a more effective manner that helped reduce issues and delays
4) With Product Owner as the single entity to approve requirements or changes, requirements management became easier
5) It also helped build transparency. With team's velocity per sprint available to Product Owner, he and business users had better understanding of what can be delivered in sprint and prioritization happened accordingly

6 Practice Questions

Question 1

What do most of the Agile methodologies have in common?

a) Focus on iterative and incremental development
b) Early feedback is key for the project success
c) Ensure transparency for the work in progress
d) All of the above

Question 2

Which of the following are not Scrum values?
a) Vision
b) Respect
c) Motivation
d) Commitment
e) Focus
f) Openness
g) Courage

Question 3

What should an Agile project do in order to communicate well?

a) Keep team-size large, to avoid stakeholders from feeling left out
b) Split the Agile project team into small, mixed-skill, self-organizing teams
c) Operate with one team of less than 10 people
d) Have separate Daily Standup meetings for the Developers and Testers

Question 4

A _____ is decided during the first half of the Sprint planning meeting and a _____ is created during the second half of the Sprint planning meeting.

a) Sprint Backlog, Collection of tasks
b) Product Backlog, Collection of tasks
c) User stories to be delivered, Sprint Backlog
d) Product Backlog, Sprint Backlog

Question 5

What does incremental delivery mean?

a) Team delivers non-functional increments in the Sprint retrospectives
b) Team deploys functional increments over the course of the project
c) Team does testing only after completing the entire development i.e. at the end of the Release
d) Team follows the same Agile process throughout the Project without any alterations

7 References

- Agile Scrum Primer 1.2
  - http://www.goodagile.com/scrumprimer/scrumprimer.pdf

- http://sparsh/v1/myUnit/QD/QD_Quality_Academy_Agilecertification_StudyMaterial.htm
  - Agile Overview and Methodologies
  - Introduction to Agile Practices

- Mike Cohn, *'Succeeding with Agile – Software Development using Scrum'*, Pearson, 2013

- Websites:
  - http://www.extremeprogramming.org/
  - http://en.wikipedia.org/wiki/Extreme_programming
  - http://en.wikipedia.org/wiki/Kanban_(development)
  - http://en.wikipedia.org/wiki/MoSCoW_method

www.infosys.com