NAVIGATE YOUR NEXT

Infosys Global Agile Developer Certification Refresher Part – 1

Infosys®
Navigate your next

Sections Covered (Agile Generic)

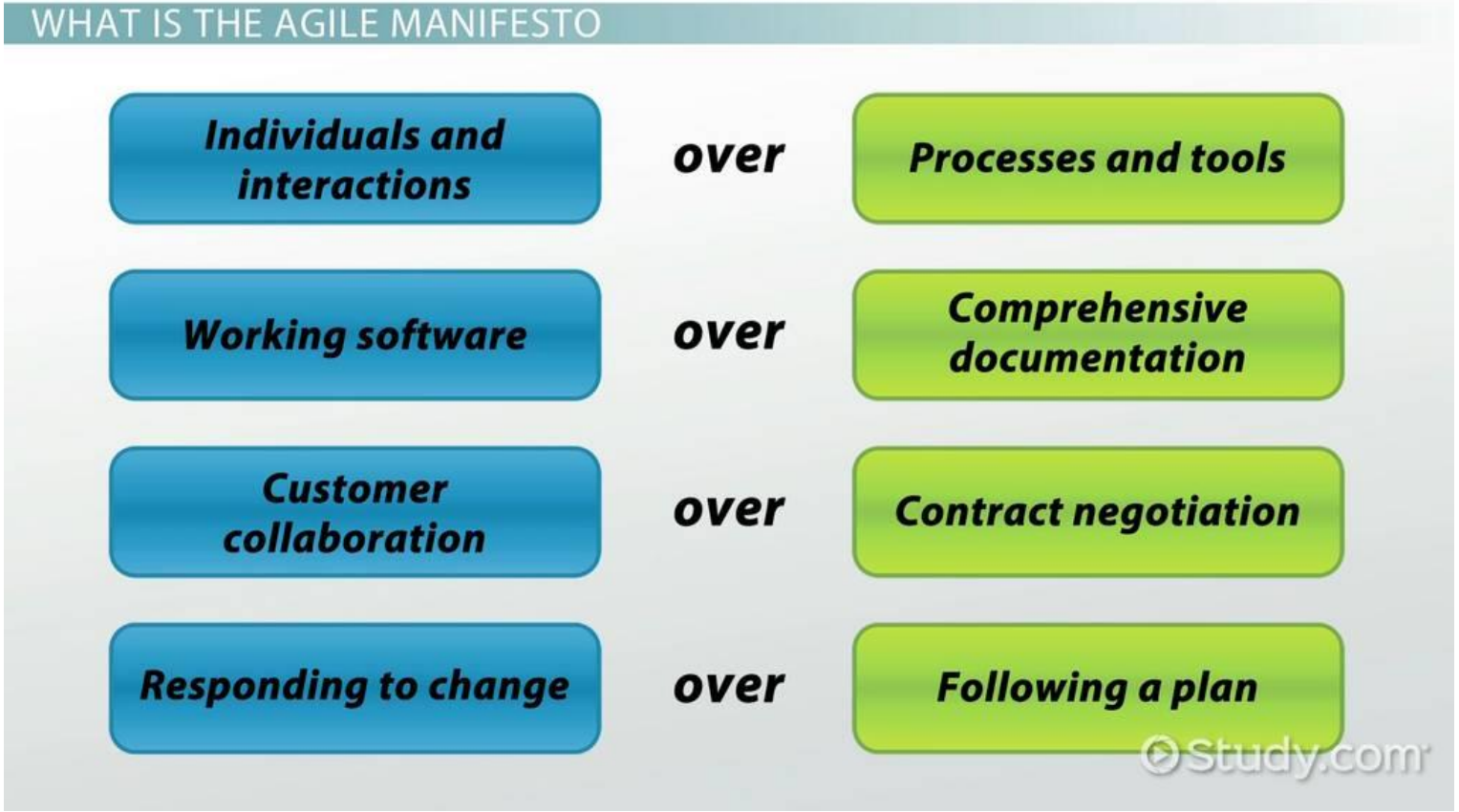
Following study contents cover the basic concepts of Agile, Planning and Estimation techniques in Agile and Soft Skills which are key for Agile project execution.

Study Content for Generic Modules in Agile		
Overview	Planning & Estimation	Soft Skills

Technical aspects play an important role for the disciplined delivery in Agile and maintain the quality of the output. Following study contents provide the key engineering practices which the Team must be conversant with while getting started in the Agile project.

Study Content for Engineering Modules in Agile		
Requirements, Architecture & Design	Test Driven Development	Refactoring
Automation	Continuous Integration and Tools	Testing Techniques

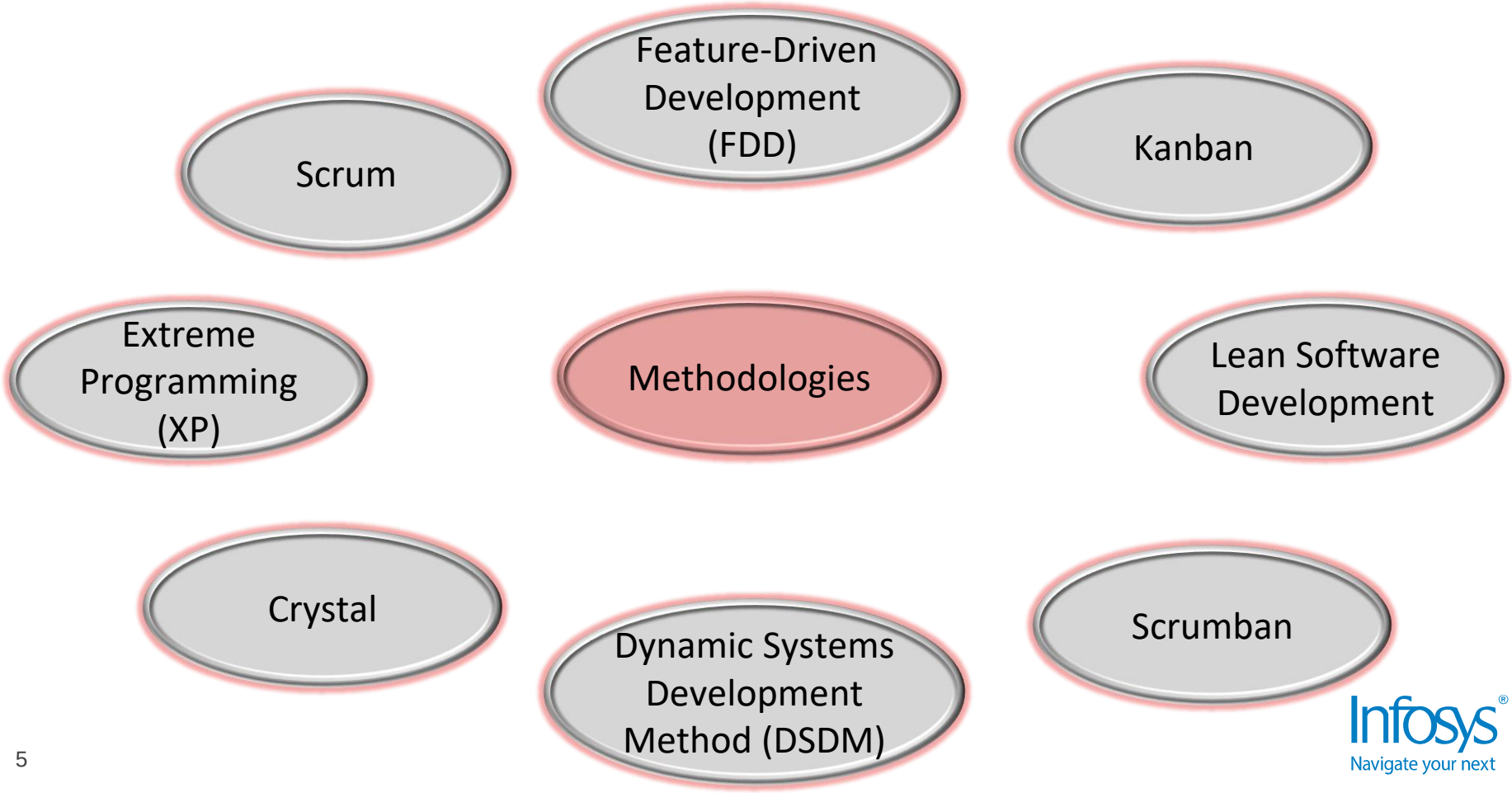
Agile Manifesto



Agile Principles

- **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**

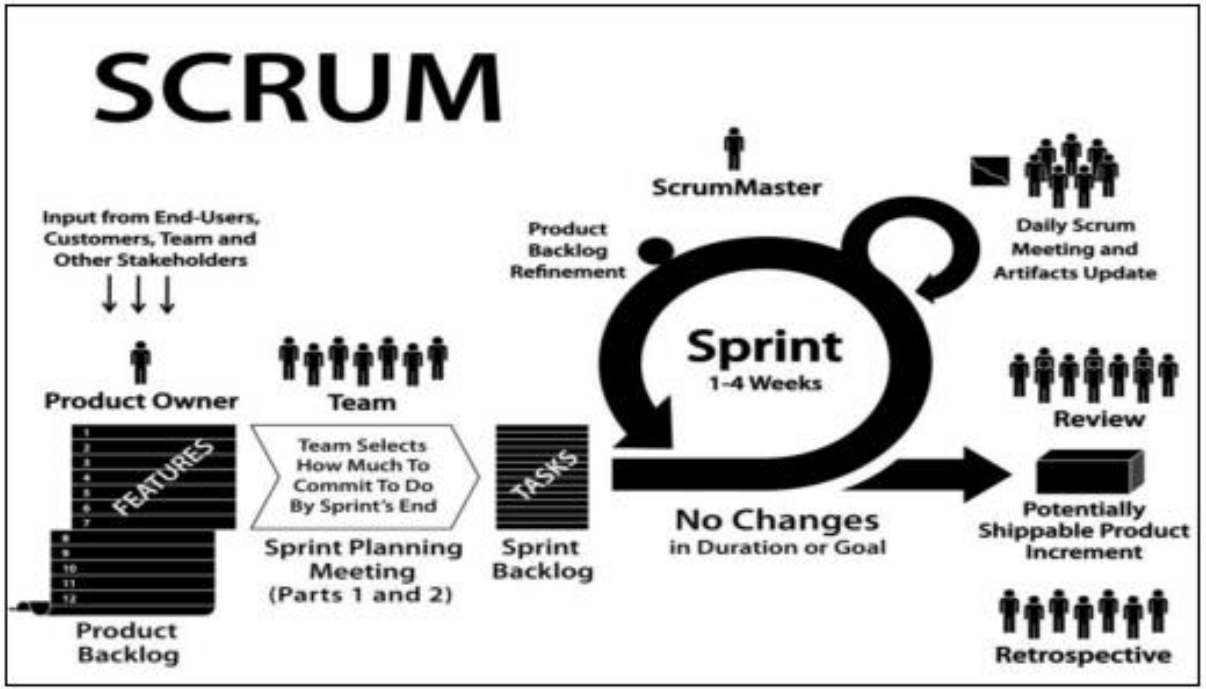
Agile Software Development Methodologies



Agile methodologies

Scrum

- Iterative and incremental framework for application or product development
- Achieved through iterative cycles known as Sprint
- At the end of each Sprint, Scrum emphasizes for an fully tested integrated working software and potentially shippable product
- Sprints are strictly time boxed and occur sequentially
- End date of a Sprint does not get extended, irrespective of the completion of the work initially planned



Infosys®
Navigate your next

Scrum Roles

Product Owner

- Providing the vision of the project to the team
- Maximizing the value of the project and the work of the Scrum Team
- Managing the Product Backlog
- Clearly expressing Product Backlog items
- Prioritizing the items in the Product Backlog
- Ensuring that the Product Backlog is visible, transparent and clear
- Change to a Product Backlog item's priority has to discuss with the Product Owner

The Team

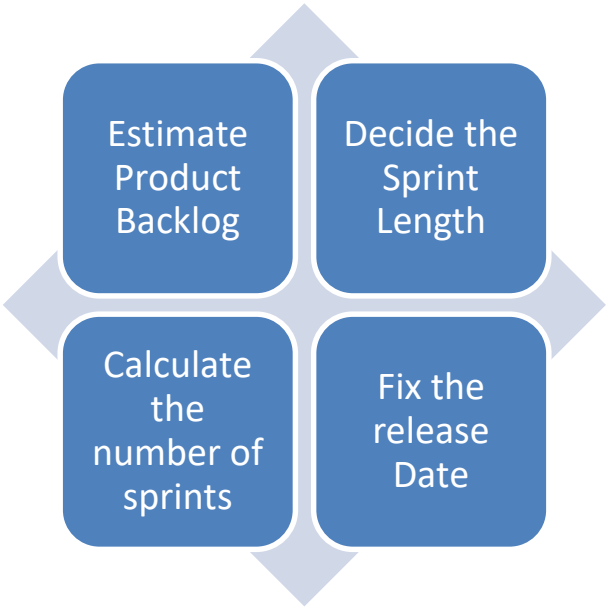
- Team size should ideally be around 7 persons (+/- 2)
- Recommended to be cross-functional with skills
- Adheres to Scrum principles
- Self-organized and decides what to commit and how best to accomplish that commitment
- Accountability of the work product belongs to the team as a whole

Scrum Master

- Helps the Scrum team to adopt Scrum
- Helps the team to learn and apply Scrum to achieve the desired objective of the project
- Does not tell people what to do or assign tasks, but facilitates the process
- Serves the Team, protects them from outside interference, educates and guides the Product Owner and the Team in use of Scrum
- 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

Scrum Events: Release Planning Meeting

- Establishes the plan and goals which needs to be achieved for the upcoming Release
- Turn the vision into a successful project in best possible way
- Establishes the goal of the release, high-priority Product Backlog items, major risks and overall features and functionalities
- A probable delivery date is agreed upon; the stakeholders can inspect the progress and make changes to this release plan on a Sprint-by-Sprint basis



Scrum Events : Sprint Planning

- Work to be performed in a Sprint is planned
- Plan is created by collaborative work of the entire Scrum Team
- Consists of two parts, each one being a time-box of half of the Sprint Planning Meeting duration

Inputs:
- 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)

Output:
- list of all the tasks with estimates
- plan on how to accomplish the work selected

First Half: WHAT to deliver

- Product Owner presents the prioritized Product Backlog to the Team
- Team discusses the requirements with the Product Owner
- Understands the acceptance criteria for each item in scope
- Team decides on the Sprint Goal in agreement with the Product Owner
- If there are issues in scope for the Sprint, Team discusses with the Product Owner for the reprioritization

Second Half: 'HOW' to deliver

Key Discussions:

- 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 estimates how much time each member has for Sprint-related work
- ✓ Determines how many Product Backlog items they can finish in that Sprint & the approach for completing them
- ✓ Decomposes the Product Backlog items into tasks based on the understanding of the overall design
- ✓ Tasks are decomposed to granular level to be completed in less than one day
- ✓ Sprint Backlog consists of the prioritized Product Backlog items for this Sprint
- ✓ Product Owner presence is optional
- ✓ Team may renegotiate the items with the Product Owner if in case it has too much or too little work in hand
- ✓ Team may also invite people outside the scrum team i.e. SMEs to attend in order to provide technical or domain advice

Scrum Events: Daily Scrum

- 15-minute time-boxed event for the team
- Identify and remove impediments to development

During the meeting, each Team member explains:

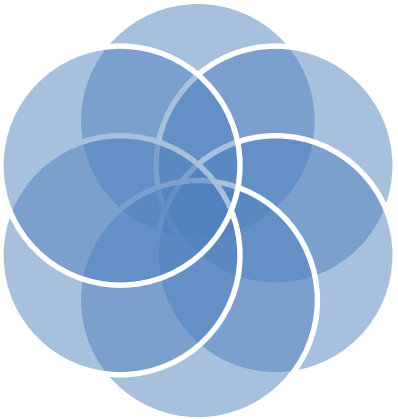
✓ What he or she has accomplished since last meeting

✓ What he or she is going to do before the next meeting

✓ What issues/obstacles are in his or her way

Promotes “Openness” as everyone shares information

Reinforces commitment



Helps to synchronize the activities and effectively plan for next 24 hours

Provides data through updated Sprint Burn Down chart on daily basis

Helps to focus by creating an “anticipating culture”

Helps Team to respect each other for their knowledge

Scrum Events: Sprint Review & Sprint Retrospective

Sprint Review

- Held at the end of the Sprint
- Scrum Team demo's what was developed in the sprint to the stakeholders
- Receive feedback on the progress made so far and decide the next course of action
- Product Backlog may get revised as an outcome of the Sprint Review meeting

The Sprint Review includes the following:

✓ Product Owner identifies what has been 'Done' and what has not been 'Done'
✓ Team demonstrates the work that has been 'Done' and answers questions about the increment
✓ Product Owner discusses the Product Backlog as it stands
✓ The entire group discusses on what to do next

Sprint Retrospective

- Inspect the last Sprint with regards to people, relationships, process and tools
- Discuss what went well during the Sprint, what problems were faced and how they were solved
- These include Scrum Team composition, meeting arrangements, tools, 'Definition of Done' and methods of communication
- Create a plan for implementing improvements to the way the Scrum Team works
- Scrum master & team attend this meeting

Steps:

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

Scrum Artifacts: Product Backlog

Product Backlog				
Product Owner articulates product vision at the start of the project	This evolves into a refined and prioritized list of features or requirements	An ordered and emerging list of user needs (User Stories) plus anything else that is required to fulfill the Product Vision" is called Product Backlog	Continuously updated by the Product Owner	Only a single Product Backlog exists for a project

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

Scrum Artifacts: Release Burndown

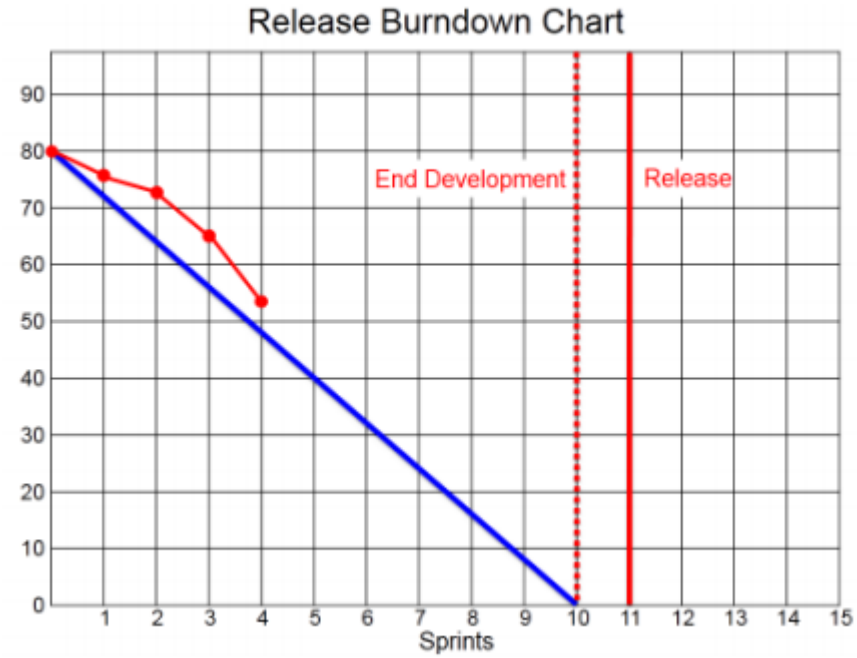
Release Burndown

Chart reflects the remaining Product Backlog items across the releases

X-axis contains the number of Sprints for the release whereas the Y axis contains the effort remaining in the release

Depicts the progress of the release on a real time basis

Available to all stakeholders of the project



Infosys®
Navigate your next

Scrum Artifacts: Sprint Backlog

Highly visible, real-time picture of the work that team plans to accomplish during the Sprint

Subset of Product Backlog items

Comprises of all the tasks that the Team identifies as necessary to meet the Sprint goal

Gets updated by the Team throughout the Sprint by updating the remaining effort for the tasks

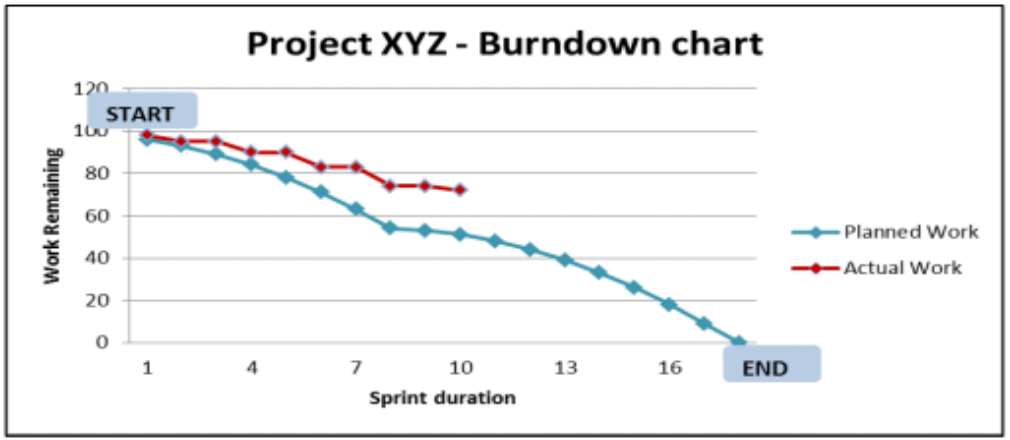
At any point in time in a Sprint, total work remaining in the Sprint Backlog items can be summed

Scrum Artifacts: Sprint Burndown

Sprint Burndown

- Represents the amount of Sprint Backlog work remaining in a Sprint across number of days in the Sprint
- "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
- 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, it indicates that the team is not on schedule or as per the plan. Team needs to adjust, such as to reduce the scope of the work or to find a way to work more efficiently



Infosys® Navigate your next

Extreme Programming (XP): Fine Scale Feedback

The term XP is conceptualized from the execution of traditional software engineering practices at extreme levels.

Following are its features:

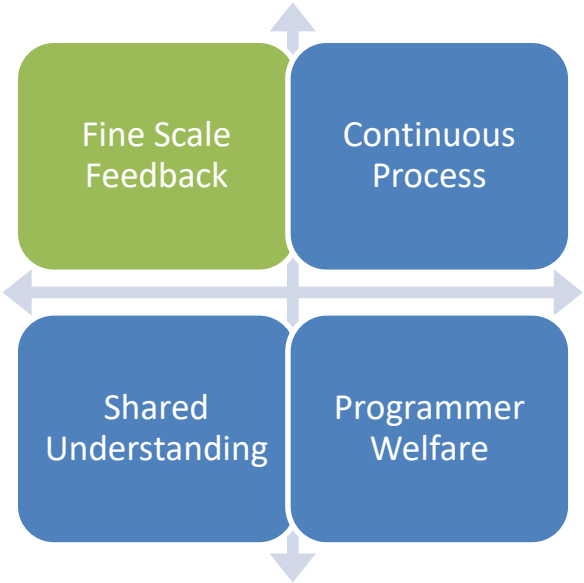
1. Planning Game: to ensure work is planned incrementally. It is divided into two parts:

Release Planning: meeting to shortlist requirements to be included in the releases planned and timeline of each release. It has three phases:

Exploration	Commitment	Steering
Create User Story	Team commits to date and functionality	Adjust plan and requirement change

Iteration Planning: meeting by the Team to decide upon the tasks and activities, to accomplish requirements planned for the iteration

Exploration	Commitment	Steering
Create task cards	Task estimation and allocation	Perform the tasks



Infosys®
Navigate your next

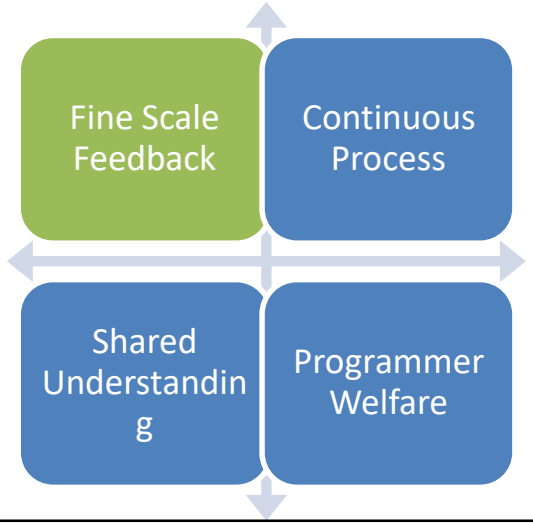
Extreme Programming (XP): Fine Scale Feedback

2. Pair Programming:

- Coding is generally followed with code review, and in the extreme case, coding and review can happen in parallel.
- 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

3. Test Driven Development:

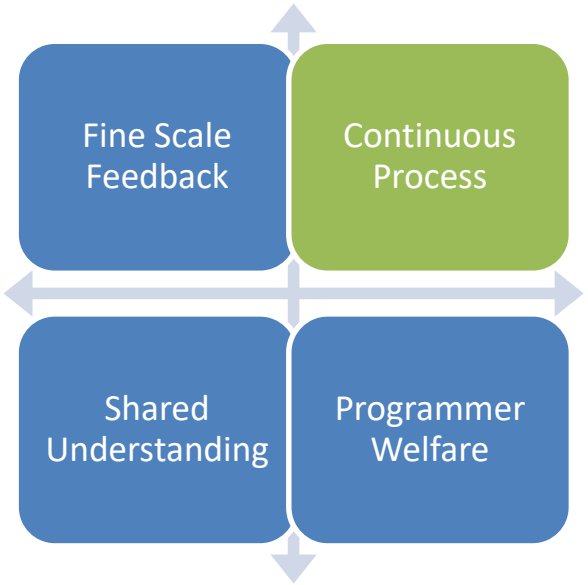
- Developer writes the automated test cases before the actual code is written
- Automated unit test cases should inevitably fail at the first attempt of run. If it does not fail, it indicates the feature already exists or there are some problems with the automated test case
- The code is written and 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



4. Whole Team:

- 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.

Extreme Programming (XP): Continuous Process



1. Continuous Integration:
- Code base is integrated on a frequent basis from the start of the feature development
- An automated process in which builds are created from the common source code repository
- Helps in identifying integration issues much ahead in the lifecycle which reduces much of the rework cost.

2. Refactoring:
- Regular improvement of existing design and code without affecting the functionality
- Helps to improve the overall code quality

3. Small Release:
- Smaller releases are planned which provide customer the confidence of the overall progress
- Helps customer to provide suggestions for any changes and helps to meet the overall goal

Extreme Programming (XP): Shared Understanding

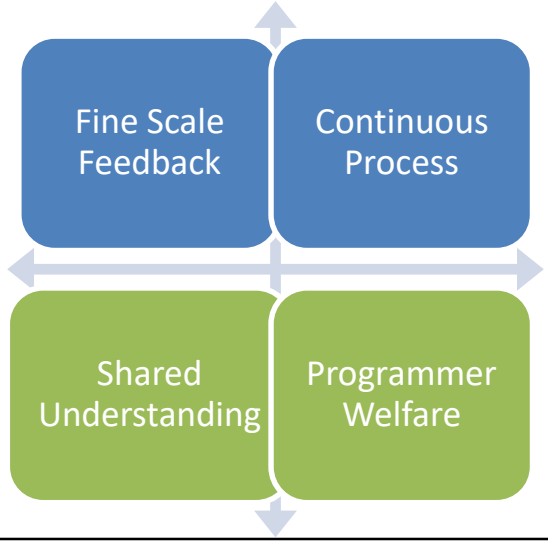
The term XP is conceptualized from the execution of traditional software engineering practices at extreme levels. Following are its features.

Coding Standards:
Rules need to be as per the standards defined by language vendors and/or customized by the Team

Collective Code Ownership:
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

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

Metaphor:
A naming convention, which makes all the stakeholders understand what the functionality is all about in detail



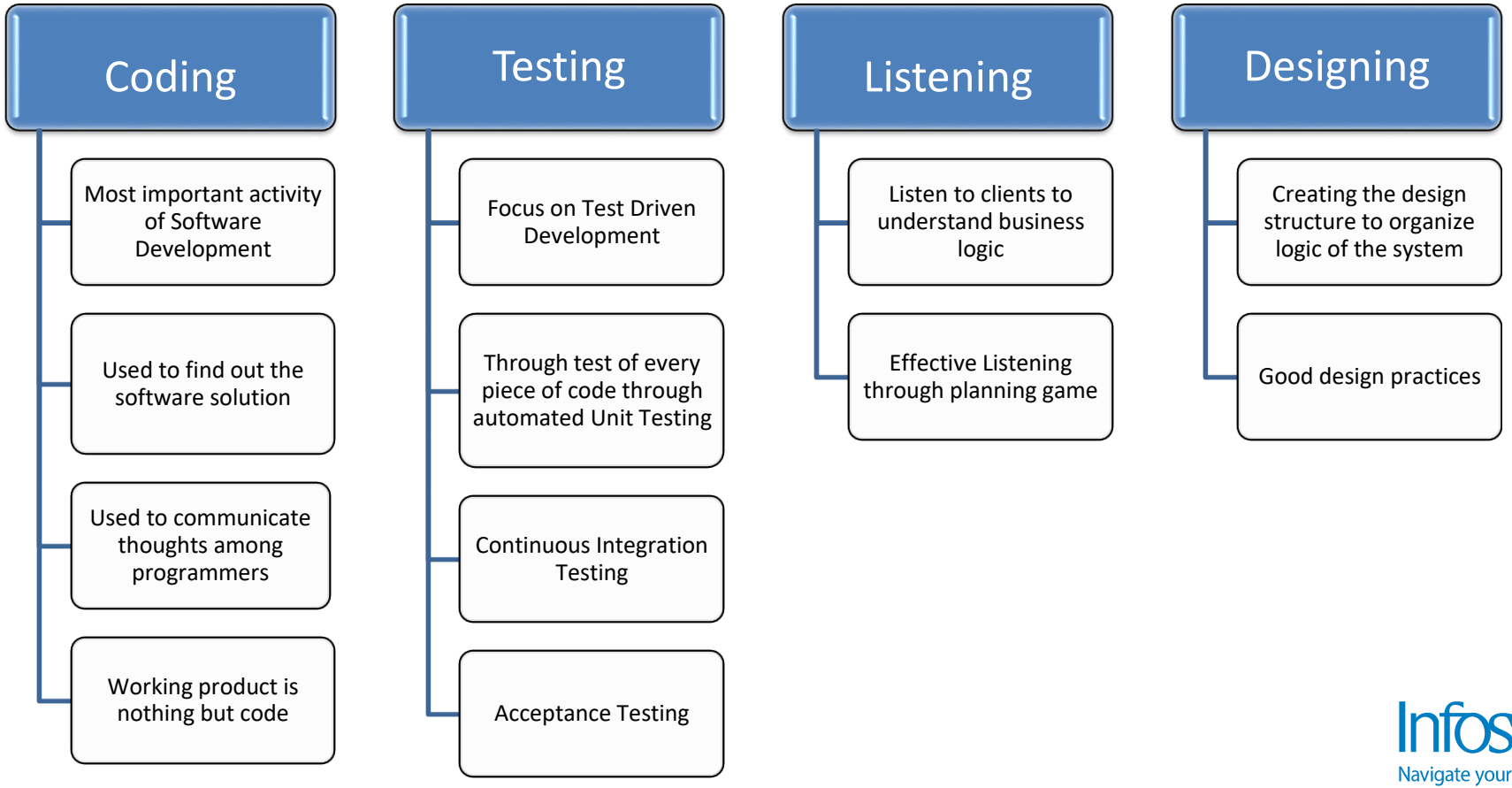
Programmer welfare:

- Team members need not work for more than 40 hours per week (ideally) to meet project deadlines as it hampers repeatable, predictable and consistent delivery

Extreme Programming (XP) : Roles

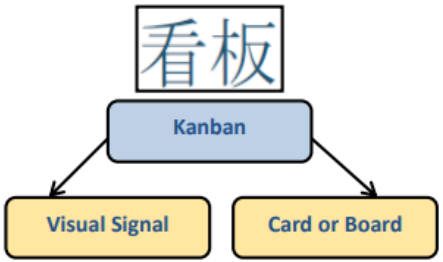
Customer	• Creates user stories and prioritizes user stories • Addition/Modification/Deletion of user stories
Programmers	• Estimates for user stories along with the Team • Builds the users stories as per the standards
Coach	• Monitors XP process implementation • Issue resolution on XP practices • Mentors team members
Tracker	• Monitors the progress • Alerts the Team

Extreme Programming (XP) : Activities



Kanban: Principles

Kanban is a Japanese word where Kan means “Visual” and ban means “Card” or “Board”

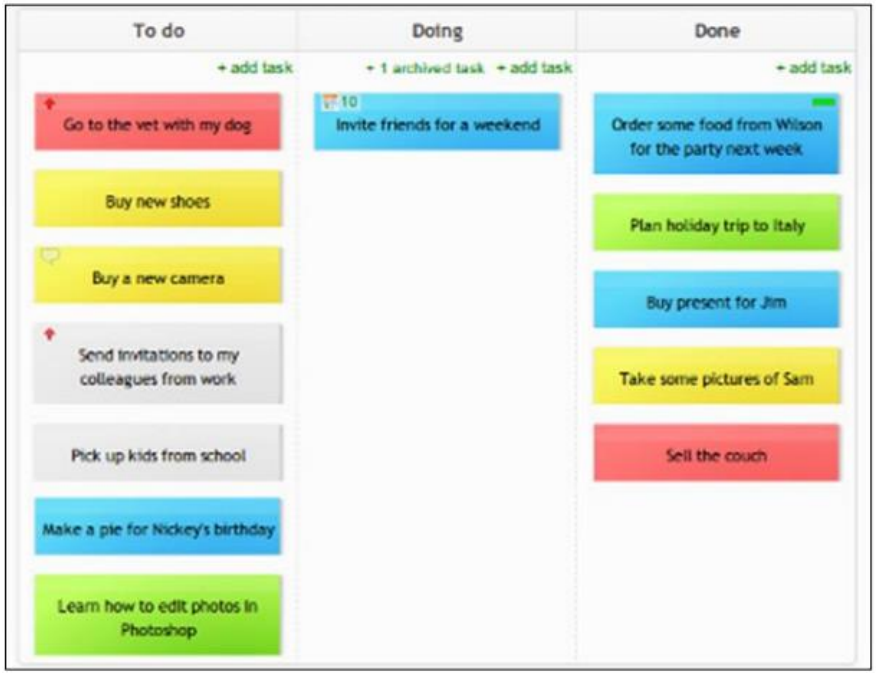


Kanban is basically a signaling device that instructs the moving of parts in a ‘pull’ production system

developed as part of the TPS (Toyota Production System)

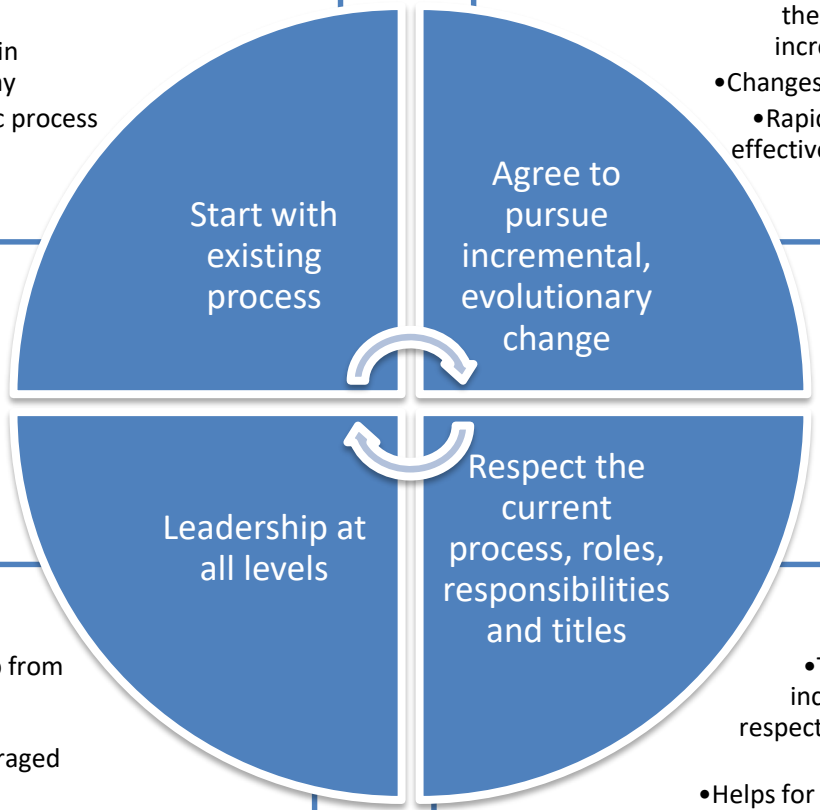
Main aim is to minimize WIP (Work-In-Progress), or inventory by making sure upstream process creates parts only if its downstream process needs it

Sample Kanban Board



Kanban Principles

- 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



- After starting with the existing process, the team must agree for continuous, incremental and evolutionary change
- 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

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

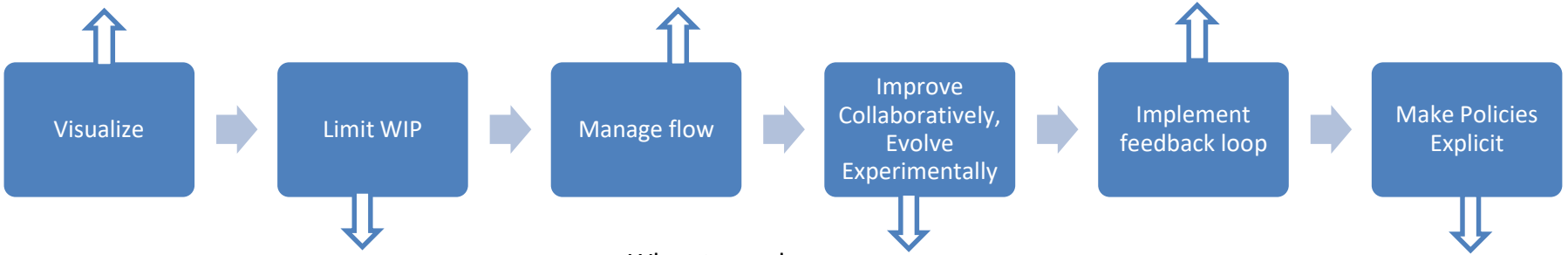
- Though Kanban suggest continuous incremental changes in the process, it respects current roles, responsibilities and job titles
- Helps for the Team to gain the confidence as they get started with Kanban

Kanban: Practices

- Visualizing the work-flow and making it visible to understand how work proceeds
- 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

- Flow of work through every state in the workflow is observed, measured and informed
- Incremental, continuous and evolutionary modifications to the system can be assessed to have negative or positive effects on the system

- Early feedbacks from client and the pull system
- Feedback from different stakeholders and processes helps to eliminate risk and optimize delivery process

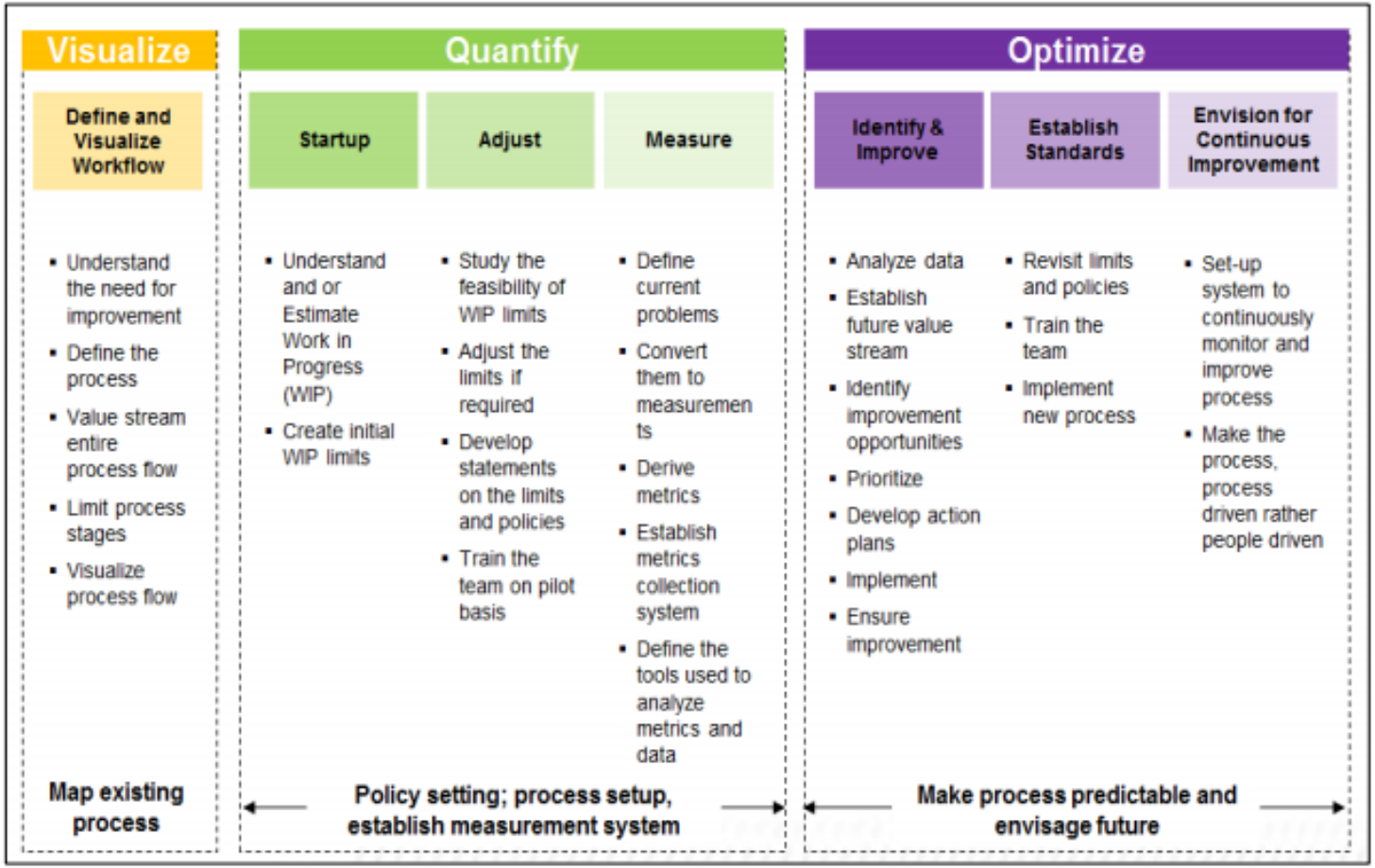


- Limiting WIP implies that a pull system is executed on either parts or the whole workflow
- Assigns explicit limits to the number of items that may be in progress at each workflow state

- When teams have a common understanding of concepts about work, process, workflow and risk, there is a shared understanding of problem
- Suggest enhancement actions which could achieve a consensus

- Mechanism of a process of how work is truly done and how things actually work
- With clear understanding, it is possible to hold a more rational, empirical, objective discussion of issues.
- More likely to facilitate consensus around improvement suggestions.

Kanban Approach



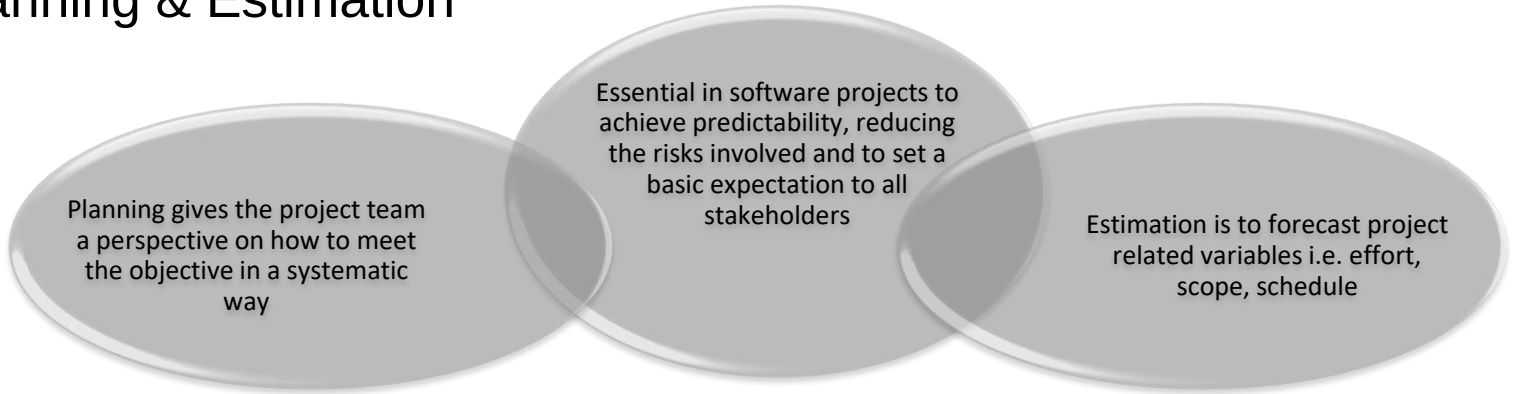
Infosys®
Navigate your next

Kanban Benefits

- Introduction of process, visualization and optimization
- Limiting WIP resulting in 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

Planning & Estimation

Planning & Estimation



Agile planning balances the effort and investment in planning with the knowledge that we will revise the plan through the course of the project

This is done to avoid the weaknesses like:

- Concentrating on activities rather than delivered feature
- Ignoring the prioritization
- Ignoring the existence of uncertainty
- Using estimations as commitments

Software projects are typically controlled by four major variables i.e. Schedule, Scope, Cost, and Effort

Estimation is a process to forecast these variables to develop or maintain software

3 main challenges faced during estimation:

- Uncertainty
- Self-knowledge
- Consistency of Method used for Estimation

Standardized and scientific estimation methods:

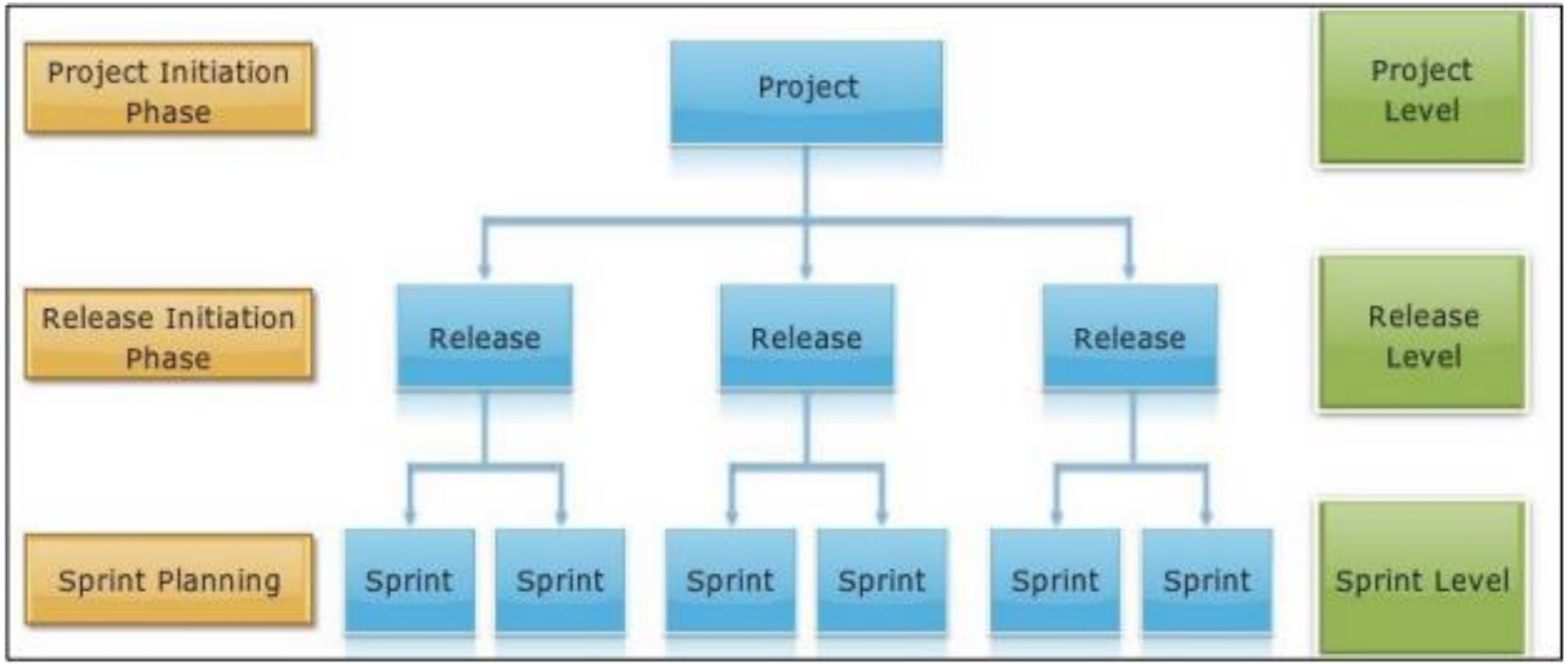
- Helps towards maintaining minimal variance between the planned estimates and actual values for maximum estimation accuracy
- Provides better client experience

No single method fits all estimation needs for a project

Different methods of estimation for different stages

Planning & Estimation Stages

- Proposal or Project Level
- Release Level
- Sprint Level



Planning & Estimation: Project Level

Project Level planning is required to understand:

- Overall vision
- Roadmap
- Value delivery
- Planning for better risk coverage
- Creation of Product Backlog
- Defining the Releases

- Highest level of planning that focuses on impending activities of the project and its ultimate goal
- Creates the macro image of the project and its importance for the stakeholders
- Helps project teams to create priorities and estimations
- Product Owner (or Client/Business manager) is the primary stakeholder involved in this activity
- Planning at Project level involves infrastructure planning, quality management planning, environment setup planning, tools and reuse planning, build automation and continuous integration planning

Important aspects of planning includes

1. **Product Backlog**: explained in earlier slide

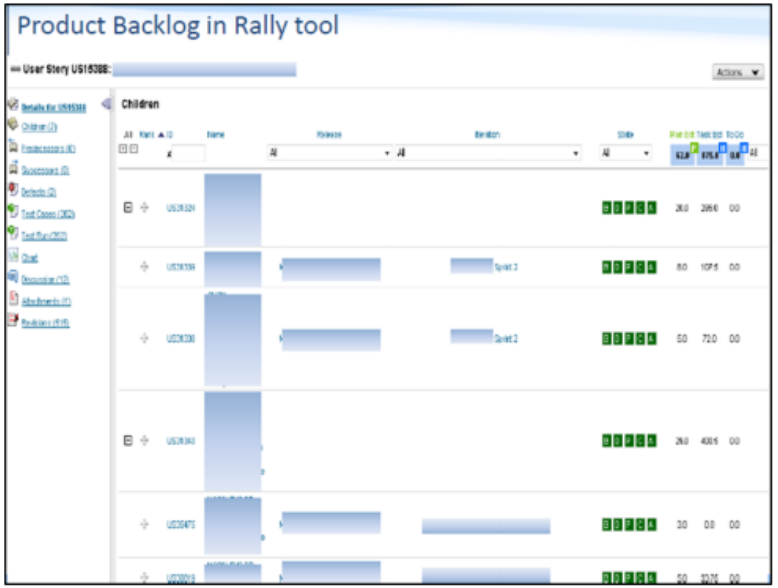
2. **Workshop**:
- main purpose of conducting workshops is for gathering, understanding and prioritizing requirements, mainly during the Initiation phase
- group of people collaborate to present different ideas to reach a predetermined objective supported by an impartial facilitator
- there should be a clear idea on what is required and what should be the outcome
- usually conducted to compress timelines (i.e. defining the Releases) and reach a consensus/decision faster
- not the same as brainstorming
- workshops are worthwhile to do once in every 6 months, just to review the direction of the project

Planning: Project Level

3. **Application Life Cycle Management**:
- aids in managing Agile projects by providing mechanisms to deliver software more predictably and to drive business value
- supports development process, project management, metrics and dashboards, compliance reports, sharing artifacts within team
- used for planning and estimating user stories and sharing it within team
- used for building a Product Backlog, Sprint Backlog, establishing team commitment and velocity, visualizing team activity and project progress via Burn Down charts and reporting on team progress
- efficiently capture, self-assign and manage their tasks

ALM tool facilitates the following activities of the distributed Agile project:

- Product and Release Planning i.e. for planning and managing Agile requirements, epics, stories, and goals across multiple projects, teams, and portfolios
- Sprint planning and Tracking i.e. for iteratively planning user stories, defects, tasks, tests, and impediments in a single environment
- Managing project artifacts (Product and Sprint backlog items, Source Code, Development and Testing artifacts, Test reports)
- Managing reports (Release and Sprint Burn Down charts)
- Tracking process information (who does what and when)
- Managing communication tools (discussion forum, chat, information-sharing, notifications to users etc



Estimation: Project Level

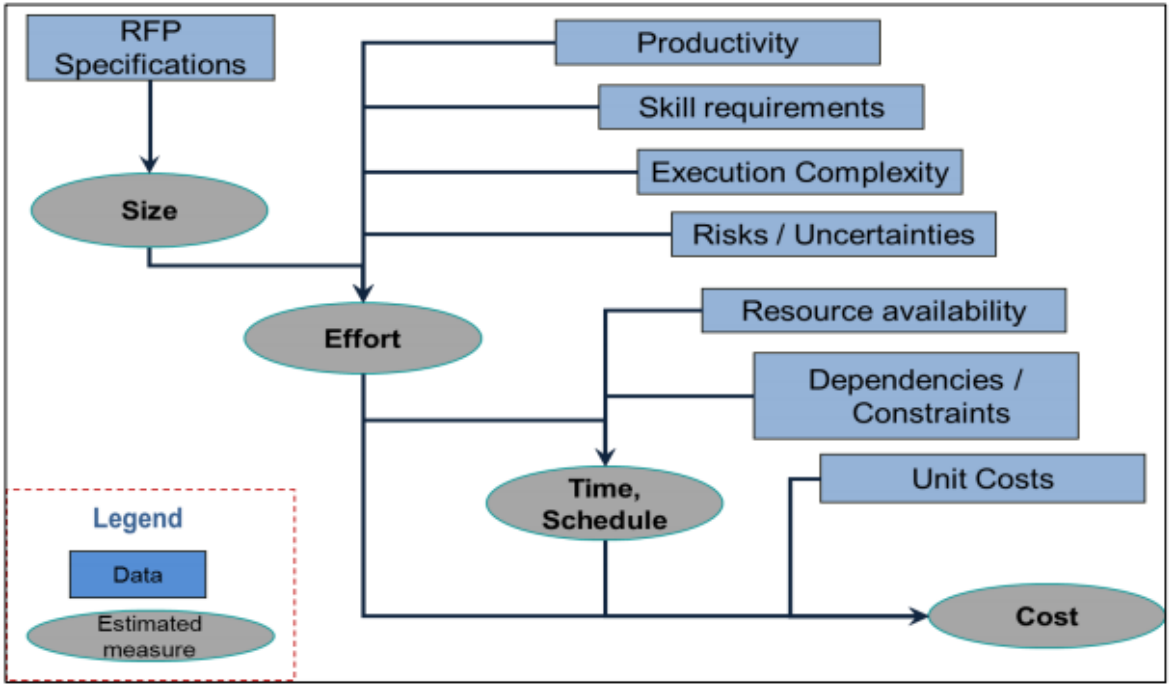
- Functional requirements are at high level and are still at Epic or Feature level
- Estimation is required at this stage, as this will help in prioritization and planning the Releases
- Estimation during Project Initiation stage should be either a group activity, results should be compared and differences to be resolved
- project assumptions and risk should be highlighted clearly
- Epics are sized using “Quick Function Point Estimation” (or any similar technique

Product Owner	•Briefs the entire team about the project objectives, business initiatives and the corresponding timelines or deadlines for the completion of the project
Clients & Business Stakeholders	•Ensures the software developed will meet the requirements and the correct/expected functionality •Responsible for participating in the estimation meetings, provide clarification to any doubts/queries that arise during this activity
Project Manager	•Responsible for performing estimation along with the developers and designers
Developers and Designers	•Design and build the application to meet the requirements
Testers	•Validates the system from functional and non-functional requirements perspective

Estimation: Project Level

The Project Level estimate will give details about the

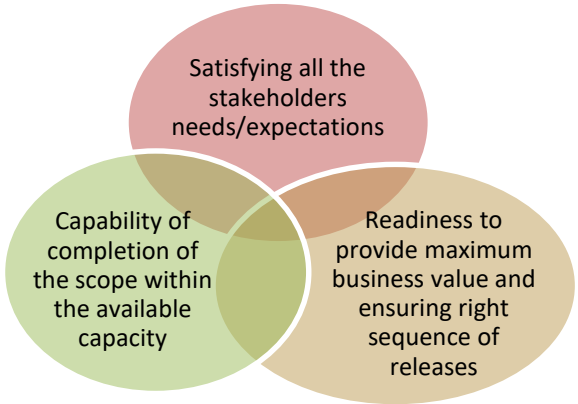
- Approximate number of Releases/Sprints required before deployment
- Approximate end date of project
- Approximate staffing plan
- Approximate effort plan etc.



Planning: Release Level

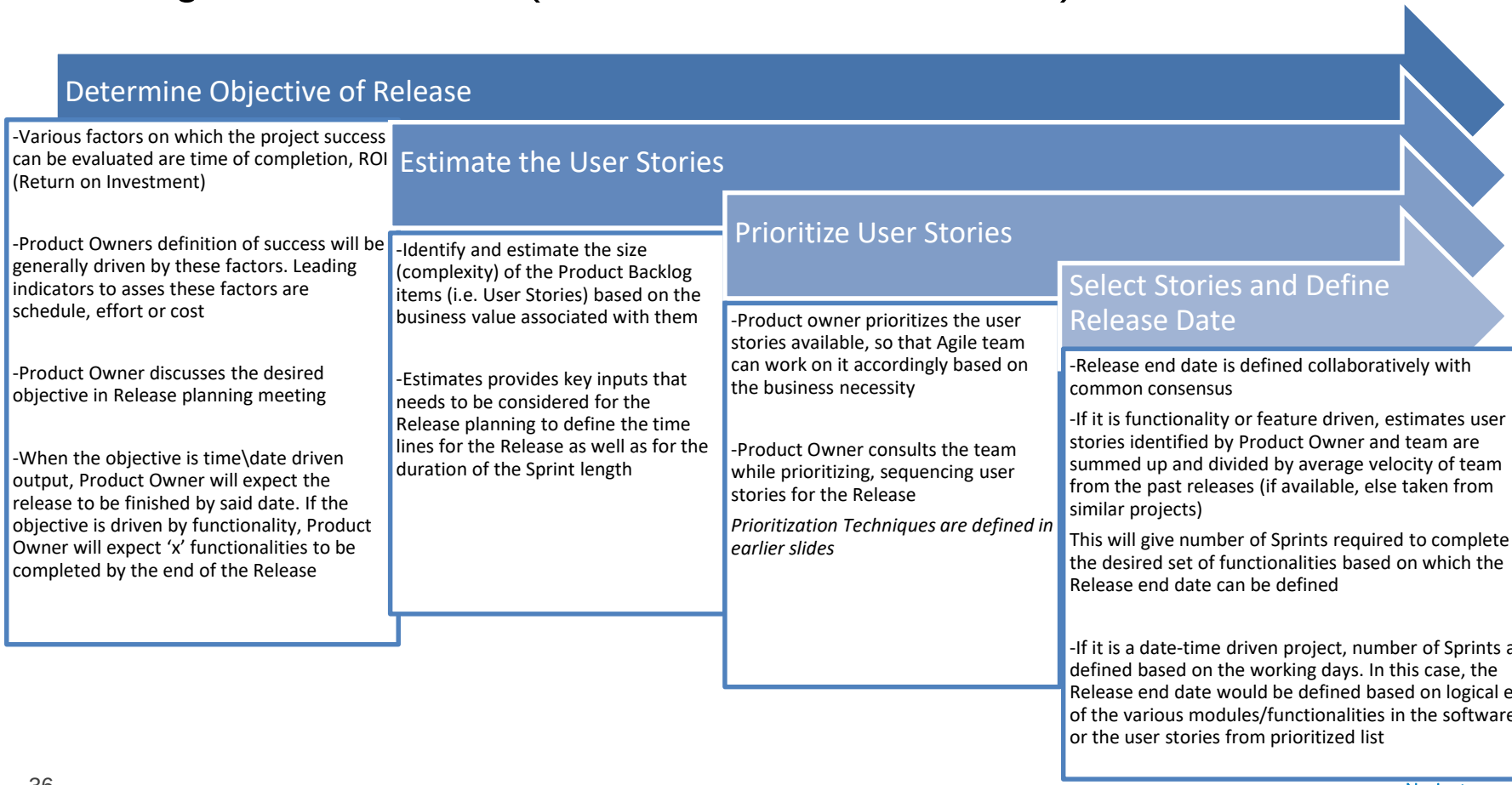
- Organizing and scheduling user stories (i.e. functional/non-functional requirements) that are to be converted into the working software by the team and delivered for that release
- Team decides number of Sprints, for each of the Release in a time-boxed manner
- Release plan can also mention how long a Sprint will be, how big the team will be, who will be working in the team, when the first Sprint will start, when the last Sprint will finish, number of Sprints, for each of the Release, duration, goal and content for each Sprint and Release
- Releases are recommended to be planned anywhere between three to six months based on business context for the annuity (long duration i.e. more than one year) projects and one to three months for the non-annuity projects

To achieve a highly efficient Release plan, it should be established based on

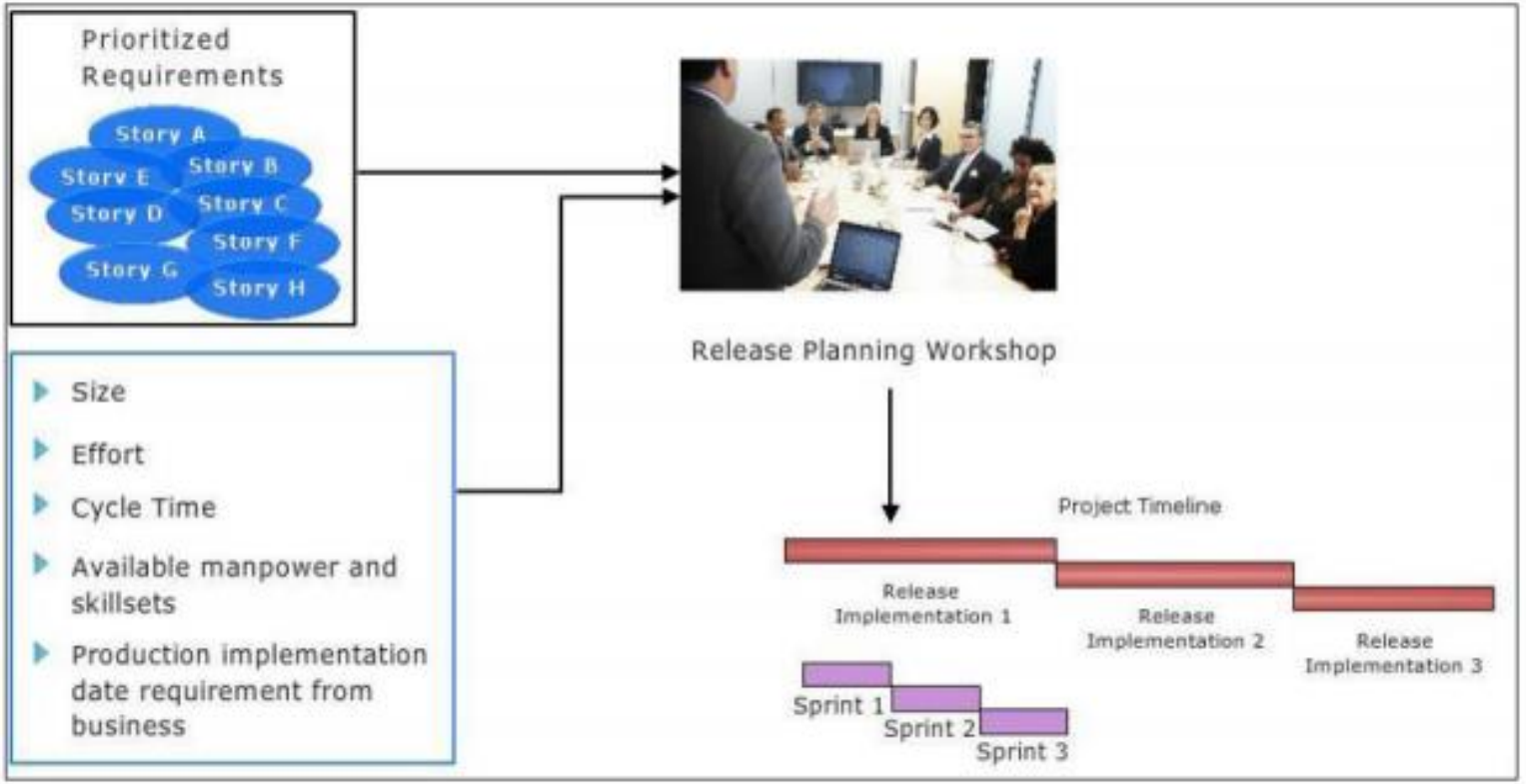


- Helps the team to deliver the expected project output in an incremental basis to the client which helps them to get an early feedback
- In the beginning of each Release and Sprint, the planning is revisited and scope is revised due to changing requirements
- The project team can decide to have additional Sprints or Releases during the course of the project depending on the dynamics at that point of time

Planning: Release Level (Various activities involved)



Estimation: Release Level



Infosys®
Navigate your next

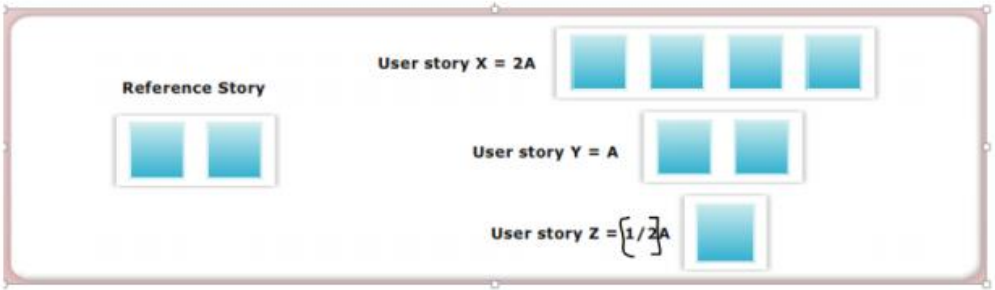
Estimation types

Wideband Delphi

- The coordinator speaks to the team and the Product Owner about the User Stories.
- Individual members in the team estimate the number of person days required to develop each functionality.
- The coordinator compares the individual estimates and discusses it with the team.
- Process is repeated until there is no significant difference in the estimation by the group.
- Typically this exercise can be completed within 3 rounds.

Estimation by Analogy

- Estimation is done for a user story based on its relationship with one or more reference
- Similar sized user stories are grouped and estimation is done for them together
- For example, a user story “Ability to add payee” is estimated for 40 hrs. If the complexity of user story “Ability to edit payee” is half of add payee, then the effort required for this story would be 20 hrs



Estimation types

Planning Poker:

Uses estimation cards, which is based on Fibonacci series. It is a relative sizing technique and based on consensus of the team.

Following are the steps followed

- Each member / estimator has a deck of cards, with each card having a valid estimate
- Moderator (who generally does not play) reads a story and it is then discussed briefly
- Product Owner answers any questions that are raised
- Each estimator selects the card representing their estimate and keeps it facing down (hidden)
- Cards are simultaneously turned over so all can see them
- If estimates differ, then the highest and lowest estimator provides justification
- Re-estimate until consensus is reached on the story points

Relative Sizing

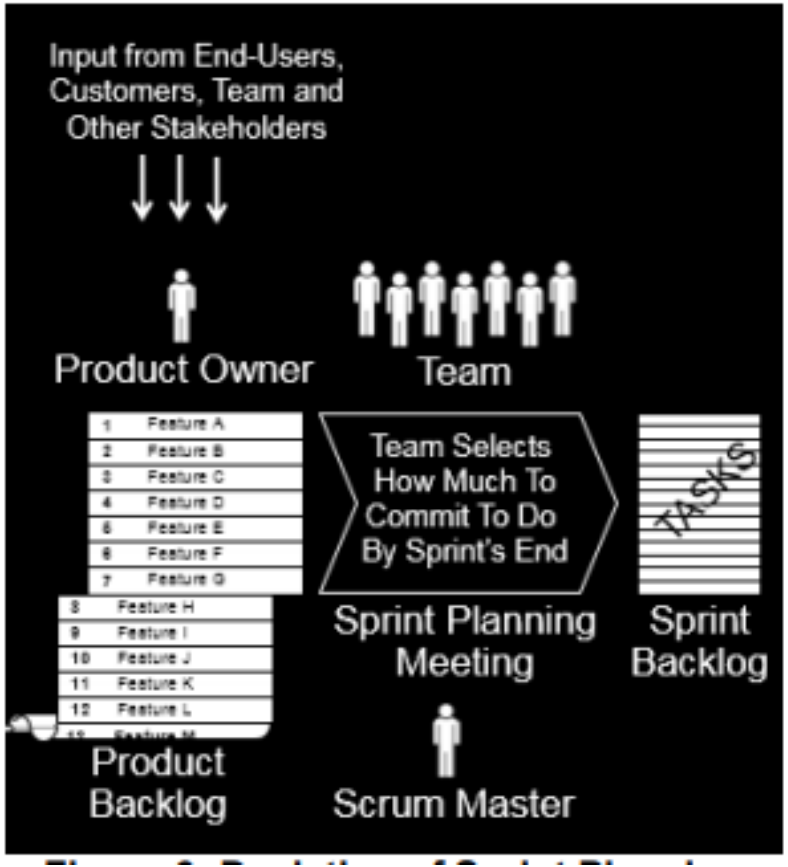
- T-shirt sizing – Each User Story is classified based on t-shirt sizes as XL (eXtra Large), L (Large), M (Medium), S (Small), XS (eXtra Small) etc.
- Fibonacci series – Relative sizing is done based on Fibonacci series 0, 1, 2, 3, 5, 8, 13, 20, 40, 100,

Planning : Sprint Level

- Starts the first day of the Sprint and allows the team to be more specific on what they are going to deliver at the end of the Sprint
- Team takes inputs from the Scrum Master and prioritized list from Product Owner
- Team decides on what they can take from the prioritized backlog for this Sprint, detail out the tasks for each user story and assign it to themselves

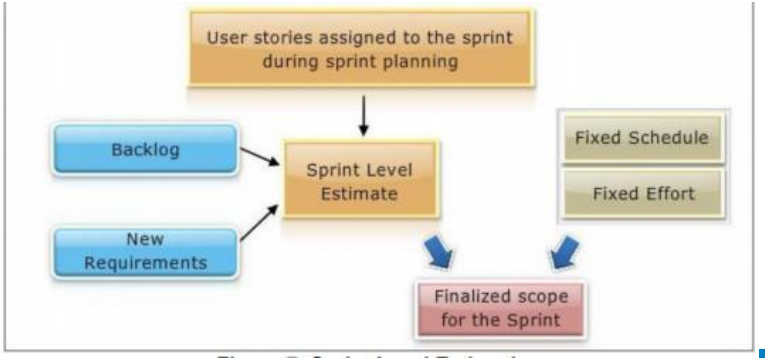
The Sprint Planning meeting consists of two parts. Explained in Slide no. 10

Ideally, for a Sprint with 4-weeks duration, the Sprint Planning meeting happens for one day i.e. 8 hours out of which first four hours are for the Part-1 and second four hours for the Part-2.



Estimation: Sprint Level

- Bottom-up estimation is used in Sprint Planning and tasks are estimated in days / hours, total of which gives the actual effort for the User Story
- Tasks can be defined related to design, coding, test scenarios, unit testing, test case execution
- Sprint success means all the parameters are set as part of Definition of Done
- Team members have to own the tasks and estimate for them separately
- Time can be categorized as programming time and non-programming time
- During Release Planning, estimation is done based on relative sizing and gets automatically considered. During Sprint planning, each task should be estimated by respective owners
- Tasks are estimated in ideal time. Estimation of ideal time for Tasks translates the size (measured in story points) to a detailed estimate of effort. This effort is typically measured in terms of actual days or actual hours. Task estimation is meant for Sprint Backlog and its existence is within the Sprint



Example

Team picks a User Story which may be sized to 5 Story Points. Break it down into a number of smaller working tasks like designing, implementing and testing.

Team members are asked to sign up for the tasks and estimate the actual effort, measured in hours or days, for their tasks

When a team uses ideal time for estimating, they refer exclusively time required to get a feature or task done by the programmer compared to other features or tasks

The effort remaining is displayed in a common place (i.e. Burndown Charts) so that the team is aligned to meet the estimated goal

For ideal time estimation of tasks, individual team members pick up an activity and provide estimates. If there is a disagreement in these estimates among the team members, then they discuss it and come to consensus.

Good Practices

1. Estimation for User Story Points:

- Story Point Estimations are reflection of 3 factors
  - Volume of Work
  - Complexity
  - Doubt (i.e. uncertainty/ambiguity in solution)
- Ensure that an estimate reflects the collective understanding of team

2. Team Planning

- Involve the whole team for planning
- With higher involvement, commitment is more from team members
- Team inclined for higher value delivery

3. Frequency of Estimation

- If estimation is difficult, Agile team increases the feedback from the Product Owner to get more clarifications in the following ways:
  1. Shorten the time for estimation and increase it to take feedback about accuracy of estimate
  2. Increase the frequency of estimating
  3. Sketch out options and get feedback from the client before doing detailed estimation

4. Re-Plan

- Re-plan often, at start of Sprint or Release
- Relevance of the plan has to be evaluated at start of each Release or Sprint and plan is updated
- Keeping the relevance of the current plan helps achieve the target

Good Practices

5. Develop Multiple Estimates:

If requirements and assumptions are unclear, the team develops multiple estimates

- Communicate the assumptions or constraints of the estimates to the relevant stakeholders rather than just the numbers
- Discuss the constraints/assumptions with the relevant stakeholders (i.e. Client or Product Owner)
- Client can provide feedback to better align the team's understanding with the business drivers

6. Consider Learnings while Planning:

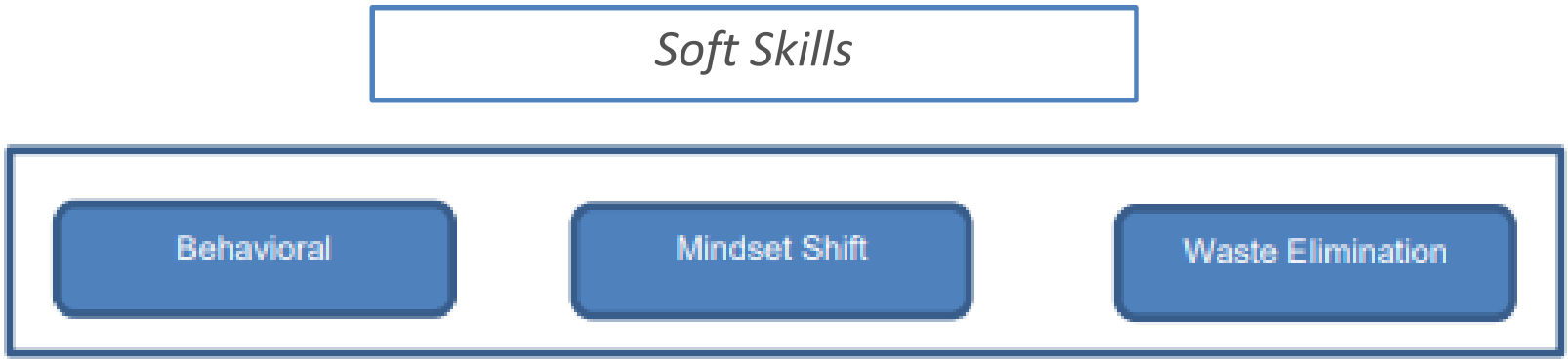
- At Sprint retrospective, there is learning about the project which is discussed by the team
- Team must acknowledge learning and consider it while planning

7. Validate Estimates:

- Team to validate estimates by comparing them with other estimates/experiences, use simple rules, and intuitive decision making
- Once the team has agreed on an estimating unit, they should ensure to implement it consistently and stick to its original definition
- Especially in the projects initial Sprints, everyone should resist the urge to try mapping these units to time units with any exact precision

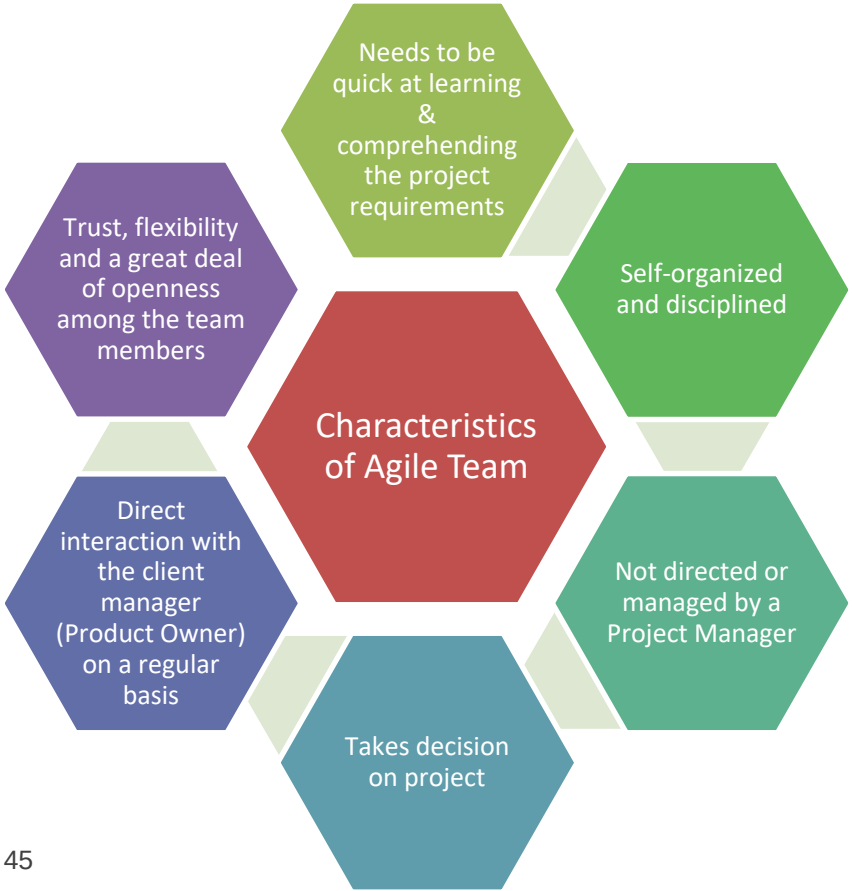
8. Leave some slack while planning:

- Recommended not to plan 100% of every team member’s time on development activities
- Team spends time in non-development activities as well like detailing out user stories whenever required, collaborating with other team members, attending daily stand up meetings, participating in backlog grooming sessions, clarifying requirements, updating ALM tool, Sprint Retrospective and Sprint Review
- Plan for approximately 6 hours of effort for development activities and the rest for non-development activities



Infosys®
Navigate your next

Characteristics of Agile Team



Scenarios where certain soft skill and mindset related aspects can **cause failure of Agile projects**

- Lack of learning inquisitiveness among Team members
- Lack of collaboration among stake holders
- Lack of ownership
- Interpersonal Conflicts
- Unwillingness to share and take additional responsibility
- Absence of open and transparent working attitude

Soft skills of a Agile Team member

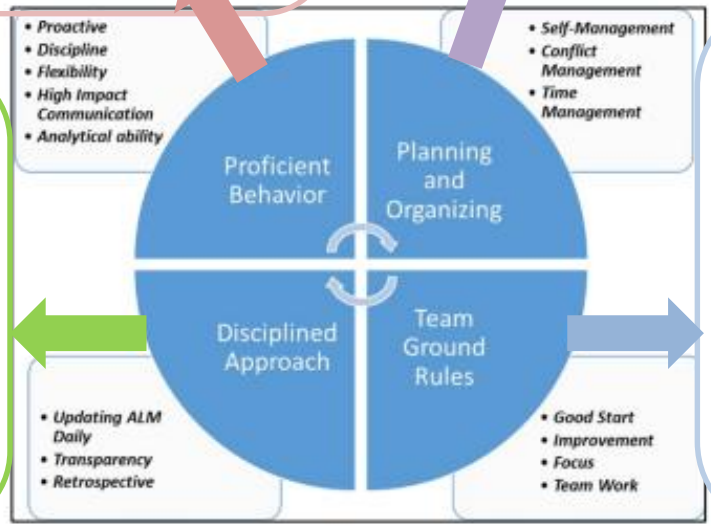
- Transparency, Discipline and commitment to work
- Ready to speak about the road block
- Self-organized and motivated
- Strong at collaboration and communication
- Very high degree of learnability
- Ready to accept the change
- Believe in Team work
- Able to negotiate and mange interpersonal conflict

Behavioral Aspects

- Manager doesn't drive the team. So proactive in raising any concern at any point of time
- Sprints are time boxed and also all the meetings. Important to be regular, disciplined and time bound completion of activities
- Should be flexible to interchange the roles they play based on the requirement for the Sprint
- Two imp pillars: Listening and Questioning. Articulation skills, clarity of speech, initiate ideas, clarify them, bring other members into discussion (Specially new & less experienced)
- Good analytical skills for problem solving by root cause analysis, quick, fast and accurate in the analysis

- Show high degree of learnability and ready to accept change, proactive in taking responsibilities, being self-managed and collaborative
- Agile coach (or the Scrum master) helps out the team and resolves conflicts in a constructive way as a Negotiator. Before going to the coach, team member should ask 2 questions:
  1. Have I shared my concerns with person I am in conflict with?
  2. Do I need an intermediary person to help me sort the issues?
- Plan approximately 6 hours of every day work for the committed user stories. Rest for issue resolution and planned/unplanned meetings. Team should handle non-value adding activities like:

Procrastination	Unnecessary long breaks
Unofficial calls	Visitors



- Application Lifecycle Management tool used for planning and tracking Agile projects. Common features are Product Backlog, Sprint Backlog, Release and Sprint Burndown charts. Team should update remaining effort for the tasks they are working upon.
- Each member knows the strength and weakness of the team, visual way of tracking for transparency
- Helps to plan better and improve in the next iteration. What went well', 'What didn't go well' & 'What can be improved'

- Stand-up meeting should give energy. Clear sense of the purpose and a clear understanding what needs to be done. distinguish this from 'false urgency'
- Meetings expose problems in the project to the entire team, provides better opportunity on improvement. Be proactive in identifying better ideas and share with each other
- Meetings to be conducted by doing focused and constructive discussions so that Team member's attention to the end objective is never lost
- Communicate regularly and supportive towards each other, and work effectively

Mind Set Shift

	Traditional Mind set	Agile Mind shift
Embracing Change	• The traditional mindset is to follow all processes and methods to achieve the goal • No changes are encouraged by the team during the course of implementation • Lot of constrains in terms of working collaboratively and innovatively	• Mind shift is needed, when working in Agile to move from constraining change to embracing and leading change • Practices such as incremental delivery and continuous feedback • Show regular progress so that necessary corrections are done during the course • Win win situation for both client and the project team
Self-Organizing Teams	• Team members used to focus on assigned tasks for completion • No awareness on the bigger picture or overall purpose of the project • Disconnect in understanding the client/business objective, less motivation and no innovation	• Self-organized and self directing teams. • Each team member is accountable for self-assignment and completing the assigned work • Team jointly accountable for solving the business problem and project issues • Self-driven and self-motivated team • Higher connect with client/business, higher motivation, and increased innovation
Collaboration	• Practices based mindset	• Focus on collaboration between all the stakeholders and delivering valuable business outcome • Behavior based mindset • Shifting towards embracing the change • Joint accountability on all the activities • Focusing on value based outcomes

Infosys®
Navigate your next

Transforming from a Traditional Project Team Member to Agile Project Team Member

- Agile methods break Releases into small **Sprints** which involve minimal planning
- Sprints are short time frames (time-boxed)
- Last from one to four weeks
- Multiple Sprints would be required by the team to release a working software or new features

- Agile development has **Daily Stand-up meetings**
- Max. for 15 mins
- What they did the previous day
- What they intend to do today
- What their roadblocks are

- Each Sprint involves a **cross-functional team** working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing.
- At the end of the Sprint, a working product is demonstrated to stakeholders (**Sprint Review**)
- Helps project team to adapt to changes quickly based on the feedback from the stakeholders at the end of the Sprint

- Agile team has a customer representative, e.g. **Product Owner**
- Available for the team to answer mid-Sprint questions
- Responsibility of each team member to use the availability or time of the Product Owner effectively for any clarifications/misinterpretation of the functional requirements

- **Feedback** during the incremental development process increases awareness
- Helps the project team to develop solutions with positive alternatives
- Team members provide feedback to each other that contains a clear purpose, which is specific and descriptive, and offers positive alternatives

To **improve product quality** and enhance project agility implement:

- Continuous Integration
- Automated Unit Testing
- Pair Programming
- Test-Driven Development
- Design Patterns
- Code Refactoring

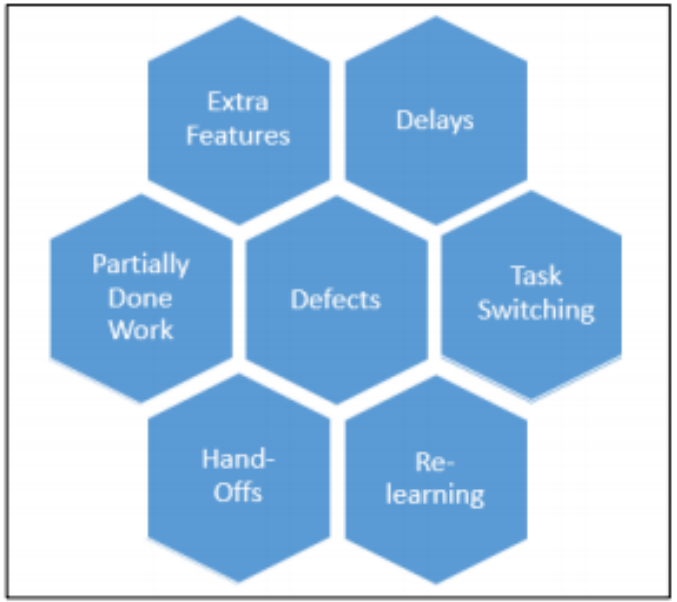
- **Motivated** team and has continuous interaction with the Product Owner
- Technical expertise needed to gather requirements, and develop and test new product line
- **Soft skills, leadership competencies**, and an understanding of how to apply those skills in a more malleable, people-focused setting

During **Sprint Retrospective**:

- What to start in the subsequent Sprint
- What to stop from the subsequent Sprint onwards
- What should be continued from the previous Sprint to the next Sprint

Waste Elimination

- ‘Waste’ is from the Lean principles
- Introduced in the manufacturing industry
- In Software terms, any feature or functionality or process step that neither adds value nor is used are considered as waste
- Should be eliminated from the system/product/process
- Everything is waste which does not add value to the client



Waste Elimination

Type	Description	Ways to eliminate
Extra Features	• Providing or developing more functionality than what is required or asked for • Occurs because of improper understanding of the product vision, or scope of the project by the team	✓ Team must ensure that they get the right information and clarity on scope and prioritization of these features (User Stories) ✓ Team members to participate in the Release Planning meeting and provide their inputs from the feasibility, interpretations and implementation of these stories ✓ Team should take frequent feedback from the relevant stakeholders to validate their understanding of the requirements and providing the right solution or functionality
Delays	• Anything that takes or causes more time for a value added activity • Caused due to 1. unwanted processes 2. too many things in progress 3. dependencies (internal/external) 4. assumptions/clarifications	✓ Team should identify only the mandatory process which would be needed for that Sprint ✓ Should not take any activities which they cannot understand or have no knowledge ✓ All dependencies should be sorted out by continuous collaboration with the required stakeholders. If there is any dependency with the Architect for example, for any important review during the Sprint, it has to be identified upfront and planned appropriately. When the team members are distributed across locations, they should proactively plan for all logistics and backup plans for any Video/Audio conferences so that they can avoid any delays due to infrastructure constraints ✓ Must get all their assumptions clarified at the right time

Waste Elimination

Type	Description	Ways to eliminate
Partially done work	• User Stories which does not meet the criteria of 'Definition of Done' or has incomplete tasks are considered as partially done work • Occurs due to technical complexity that was not appropriately anticipated during the Sprint Planning meeting, or wait time (long gap) between the tasks that were identified for the story completion, or inadequate tasks identification	✓ Complexity of User Stories should be assessed appropriately so that the team can pick it up accordingly. If needed, request the Product Owner to explain the stories in further detail ✓ When stuck somewhere, proper support/aid should be looked for. Cross-functional teams should be leveraged ✓ Take inputs from the team members while decomposing the user story into tasks
Defects	• indicates erroneous functionality that produces in-correct output • Caused due to 1. improper understanding of the User Story 2. failing acceptance criteria 3. team member's technical skill incapability 4. involvement of testing activities	✓ Get the complete understanding of the user story by getting clarifications from Product Owner or required stake holders ✓ Acceptance criteria against each User Story is must. If ignored leads to inconsistencies in the final acceptance ✓ Right skills while working on the project and if required undergo trainings to get equipped with right skill sets ✓ Involve testers for planning, and strategizing the testing activities right from the Release stage and during every Sprint Planning stage
Hand-offs	• passing the work or tasks from one person to another • occurs due to nature of tasks required for the user story, or if teams are distributed between different locations, or lack of visibility of the information	✓ Have cross-functional teams ✓ Usage of flowcharts and wireframes ✓ Executing the tasks at one location as much as possible

Waste Elimination

Type	Description	Ways to eliminate
Context Switching	• happens when team members have not completed the present activities and starts working on another activities • Occurs due to 1. shared team, i.e. same members are asked to work on another project in parallel 2. various interruptions on the ongoing activities 3. improper coordination between the team members	✓ External dependencies related to infrastructure, or environment dependencies should be identified up front during sprint planning. Impediments discussed in daily stand-up meeting and let the Scrum Master work on them ✓ Ensure that the User Stories are prioritized during the Sprint Planning meeting else team members would be juggling between stories during the Sprint ✓ Team members allocated to a project must be dedicated teams. In Agile, assigning the team member's effort across various projects would lead to lower productivity of the individuals as well as for the overall team
Relearning	• reinventing the wheel and not using the existing knowledge repository available amongst the team • Caused due to 1. improper knowledge-sharing practices with in the team 2. lack of needed content or documentation 3. because of distributed teams	✓ Regular activities for sharing knowledge amongst themselves. Team members to participate in all the team meetings ✓ Plan for creating the just enough documentation as this is one of the important artifact for knowledge sharing especially if there are exits or replacements within the team ✓ When the teams are distributed, it is very important to leverage the technology aids i.e. WebEx, Skype, Video/Audio Conferencing, etc. as per feasibility for interacting with different team members across location and share the knowledge

THANK YOU

Infosys®
Navigate your next