What is Agile?

Agile is a **time-boxed**, **iterative** approach to software delivery that builds software **incrementally** from the start of the project, instead of trying to deliver it all at once near the end.

– Agile in a Nutshell (www.agilenutshell.com)

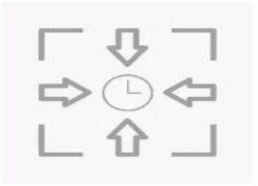
Iterative, Incremental & Time-boxed

Iterative

Work on the project is done in a series of iterations. Each iteration is a mini development cycle, with its own analysis, design, build, test and deployment phases

Incremental

The product is developed as a set of features in each iteration. Every feature delivered at the end of an iteration is potentially deployable in a live Production environment



Time-boxed

Each iteration is of a fixed duration that is constant throughout the execution of the project. The duration can range from hours to weeks, depending on the size of the project executed

Agile Software Development

- Overview: Iterative approach emphasizing collaboration and rapid releases.
- Popular Frameworks: Scrum, Kanban, XP, SAFe, Lean.
- Advantages: Increased productivity, Enhanced team collaboration, Better risk management.

What Is Agility?

- Definition: Ability to adapt and respond quickly to changes in a dynamic environment.
- Key Aspects: Flexibility, Continuous Feedback, Iterative Approach.
- Benefits: Faster response, Increased collaboration, Higher customer satisfaction.

Predictive vs. Descriptive

- Predictive: Linear, structured process, Fixed requirements.
- Descriptive: Adaptive planning, Iterative approach.

Examples:

Predictive - Waterfall model.

Descriptive - Agile Scrum.

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals & Interactions	over	Processes & 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.

The Agile Alliance

February 2001

Agile Manifesto

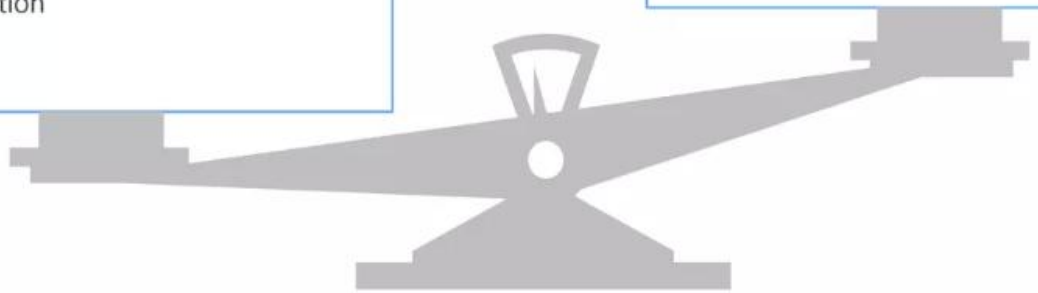
Agile Manifesto | Individuals & Interactions over Processes & Tools

Individuals & Interactions

- Recognizes people as the core driving force of any project
- Healthy interaction fosters mutual respect and trust
- Interactions aimed at improving collaboration among stakeholders
- Emphasis on spontaneous and open communication

Processes & Tools

- Facilitators for achieving project objectives
- Provide a framework to the way projects are executed
- Must be balanced with other aspects of the project such as people and delivery
- No single process or tool is self-sufficient



Agile Manifesto

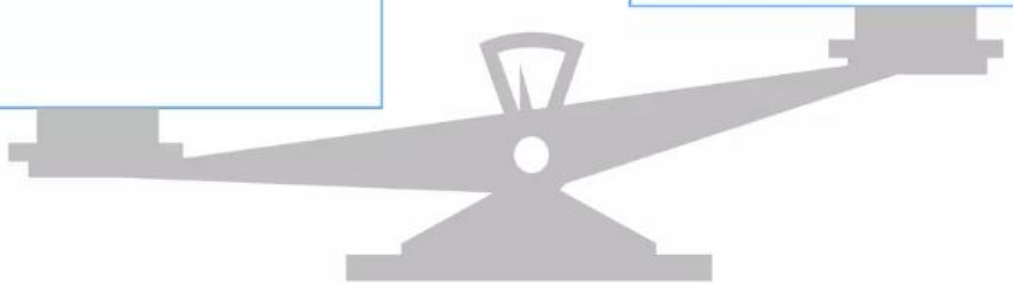
Agile Manifesto | Working Software over Comprehensive Documentation

Working Software

- Primary aim of the development effort
- Tangible output of the project
- Drives business outcomes and thus, generates value for the customer
- Can be maintained with 'just enough' documentation

Comprehensive Documentation

- By-product of the development effort
- Only when there is a strong need for future reference
- The more detailed, the more efforts it takes, driving up the cost factor
- Additional maintenance needed
- Useful in cases where team member turnover is high



Agile Manifesto

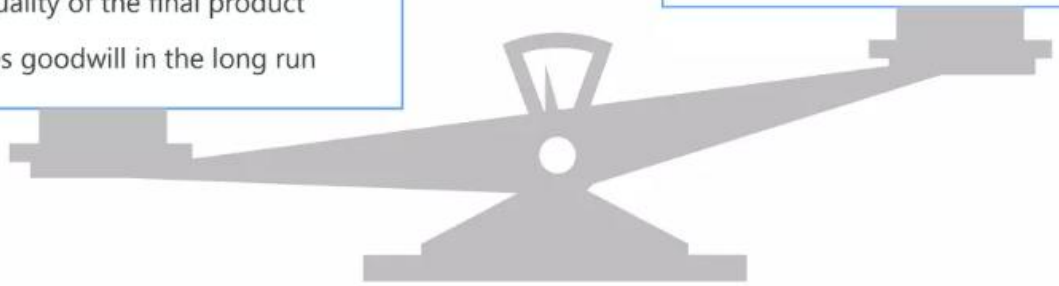
Agile Manifesto | Customer Collaboration over Contract Negotiation

Customer Collaboration

- Recognizes customer as the most important stakeholder on the project
- Improved collaboration with customer builds trust over time
- Customer's trust encourages prompt and honest feedback during development
- Customer engagement at every step improves quality of the final product
- Accumulates goodwill in the long run

Contract Negotiation

- Primarily focused on profitability and legal aspects of the venture
- Might not involve all relevant business stakeholders at crucial stages of drafting
- End-user needs might be overlooked
- Brings in elements of rigidity and formality to they way the project is executed



Agile Manifesto

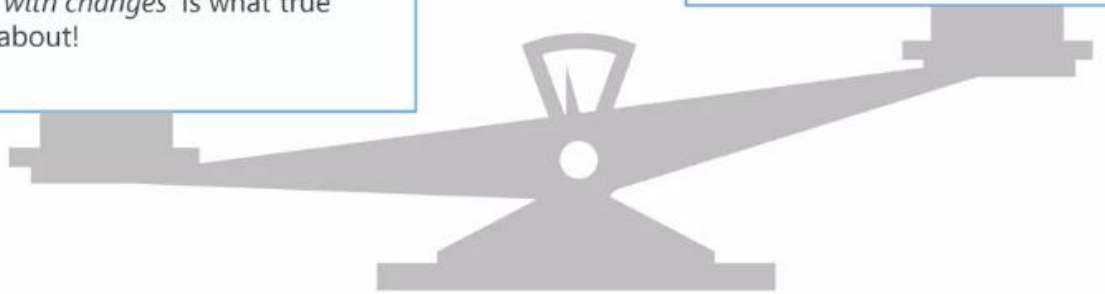
Agile Manifesto | Responding to Change over Following a Plan

Responding to Change

- Quick response to change is crucial for building the right product
- Gives a competitive edge to the customer in capitalizing on changing market trends
- Development efforts geared towards making changes available as early as the next iteration
- '*Keeping up with changes*' is what true agility is all about!

Following a Plan

- Lends a sense of rigidity to project execution
- Preferred when chances of requirement changes are very low to none
- Accommodating change can be expensive
- Quality processes can add more cost to the project



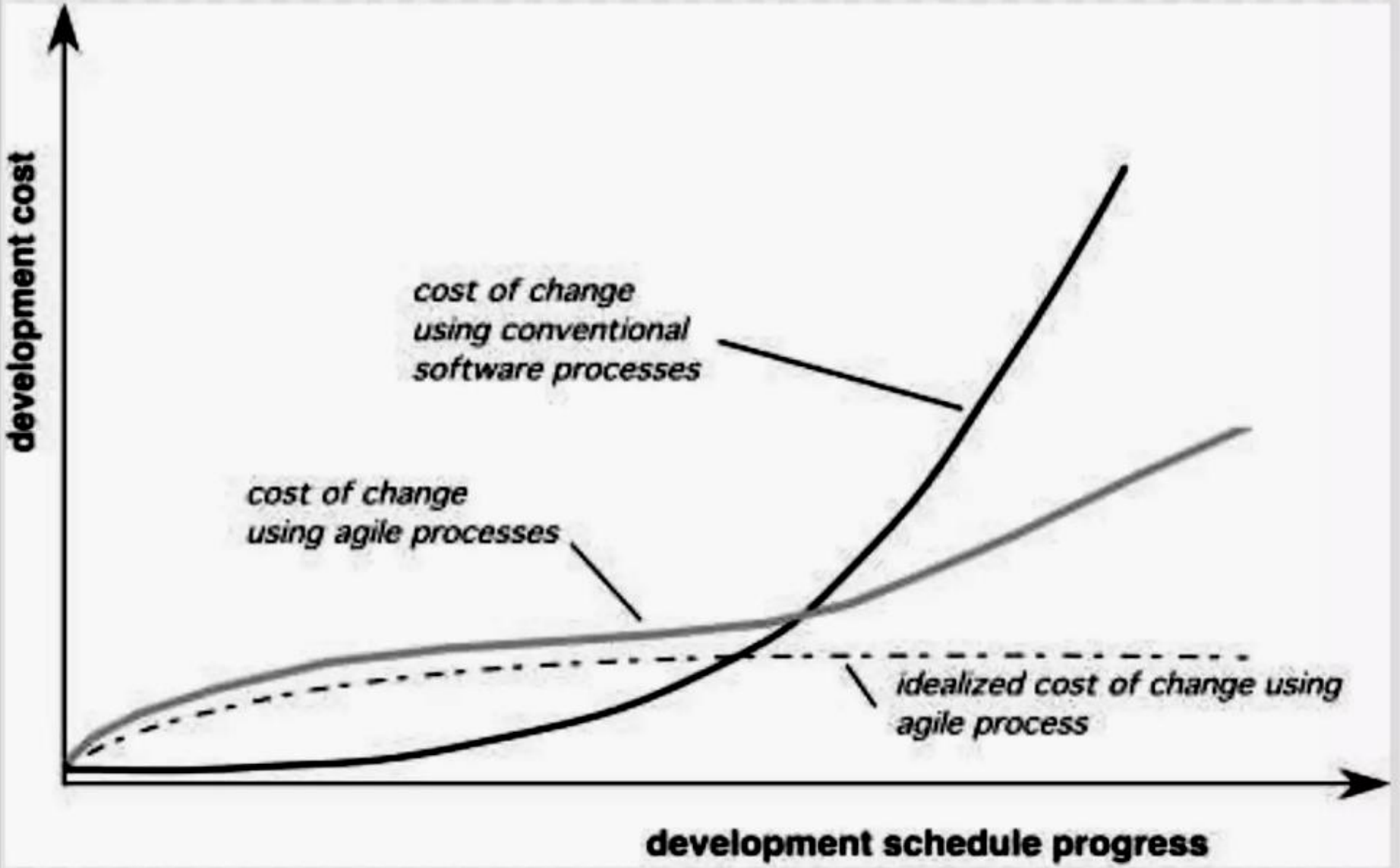
Agility and the Cost of Change

- Traditional: Higher costs for late changes.
- Agile: Reduces costs through early testing and continuous feedback.

Example:

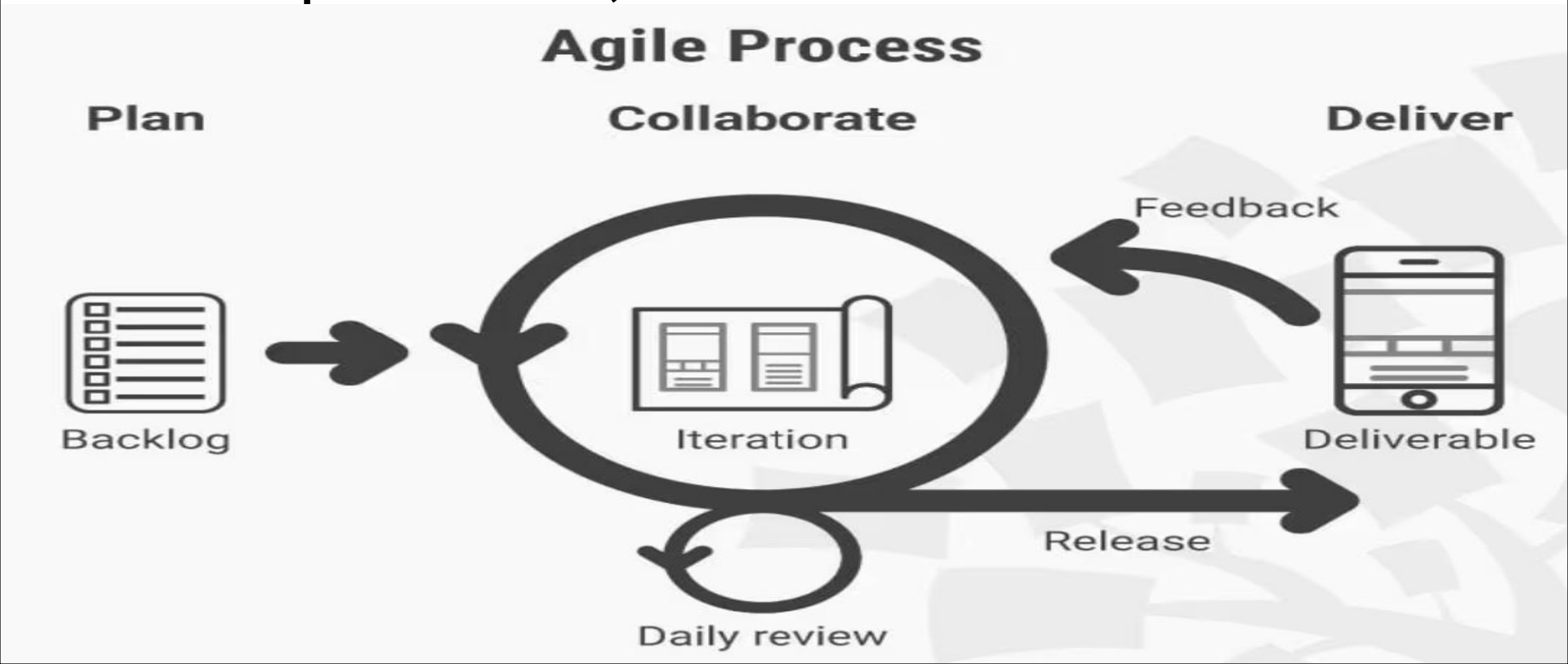
- Traditional - Delays and cost spikes.
- Agile - Incremental updates reduce risks.

Agility and the Cost of Change



What Is an Agile Process?

- **Definition:** Iterative, incremental approach focusing on collaboration and adaptability.
- **Steps:** Define requirements, Plan iteration, Develop and test, Review and refine.



Characteristics of AgileDevelopment

1. Fixed-Length Sprints – Agile teams work in predefined time-boxed iterations (sprints) to ensure consistent and predictable progress.
2. Always Deliver Working Software – The primary goal is to deliver functional software in every sprint, ensuring incremental improvements.
3. Value is Priority – Agile focuses on delivering high-value features first to maximize customer satisfaction and business impact.
4. Rolling Wave Planning – Instead of upfront detailed planning, Agile uses continuous planning, refining requirements iteratively.
5. Multi-Level Planning – Agile integrates planning at different levels: product roadmap, release planning, iteration planning, and daily tasks.

Characteristics of AgileDevelopment

6. Relative Estimation – Instead of fixed estimates, teams use techniques like story points and T-shirt sizing for effort estimation.
7. Features Begin at High-Level – Agile starts with broad feature definitions and refines them into detailed user stories as development progresses.
8. Continuous Testing – Automated and manual testing is integrated into the development cycle to ensure software quality.
9. Continuous Improvement – Agile encourages teams to reflect on their work through retrospectives and continuously enhance processes.
10. Small, Cross-Functional Teams – Agile teams are small, self-organizing, and cross-functional, ensuring faster collaboration and efficiency.

Agility Principles

Our highest priority is to satisfy the **customer** through early and continuous delivery of valuable software.	Welcome changing requirements, even late in development. Agile processes harness **change** for the customer's competitive advantage.	Deliver **working software** frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	**Business people** and developers must work together daily throughout the project.
Build projects around **motivated individuals**. Give them the environment and **support** they need, and **trust** them to get the job done.	The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.	**Working software** is the primary measure of progress.	Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Continuous attention to **technical excellence** and **good design** enhances agility.	**Simplicity** – the art of maximizing the amount of work *not* done – is essential.	The best architectures, requirements, and designs emerge from **self-organizing teams**.	At regular intervals, the team **reflects on** how to become **more effective**, then **tunes and adjusts** its behavior accordingly.

Agility Principles

Agile Principles | 1 of 12

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- Commitment to customer satisfaction takes the highest priority in Agile development
- Produce tangible results as early as towards the end of the first iteration
- Near continuous delivery through shorter iterations
- Software delivered at the end of each iteration should hold value for the customer, i.e., it should be directly usable in the customer's business process

Agility Principles

Agile Principles | 2 of 12

Welcome changing requirements, even late in development. Agile processes harness **change** for the customer's competitive advantage.

- Acknowledge that change is integral to any business environment and must be catered for during development
- Incremental development provides greater flexibility to accommodate changes
- Just-in-time planning addresses changes by allowing them to be taken up in the next immediate iteration
- Ability to quickly adapt to changes lends the customer a competitive advantage
- Changes are only taken up during iteration planning, and not during an iteration (*unless driven by acute business urgency*)

Agility Principles

Agile Principles | 3 of 12

Deliver **working software** frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

- Software that works and thus, adds value to the customer should be delivered at regular intervals
- The shorter the iteration, the more frequent the value delivered
- Shorter iteration lengths also provide frequent opportunities for stakeholder feedback

Agility Principles

Agile Principles | 4 of 12

Business people and developers must work together daily throughout the project.

- Involvement of business along with the developers helps generate timely feedback during development
- Most design defects can be avoided
- Daily involvement keeps business updated of the most current status
- Boosts collaboration and mutual trust within the team
- Requirement changes can be handled more efficiently

Agility Principles

Agile Principles | 5 of 12

Build projects around **motivated individuals**. Give them the environment and **support** they need, and **trust** them to get the job done.

- Team members should have a sense of motivation to deliver what is expected to the best of their abilities
- Trust and mutual respect form the basis of high-performing teams
- An open culture fosters transparency and free expression of opinions and ideas
- Constructive criticism and objective evaluation enable improvement
- Teams must be protected from excessive bureaucracy and red-tape

Agility Principles

Agile Principles | 6 of 12

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Face-to-face conversation is free of most barriers to effective communication
- Improved accuracy in transmission and reception of ideas
- Non-verbal communication such as e-mail or text messages more susceptible to communication gaps
- Co-location is highly recommended for Agile teams
- Frequent video conferencing, if possible, may be employed if co-location not possible

Agility Principles

Agile Principles | 7 of 12

Working software is the primary measure of progress.

- Only software that performs its expected function adds value to the customer
- Effectiveness of the development process is measured by its ability to produce working software at each iteration
- Testing data can provide metrics to assess 'workability' of the software
- New releases should not break prior versions (regression, security and backward-compatibility)
- Strong emphasis on frequent demos of the work completed
- Involvement of business during testing enhances reliability and efficiency of the process

Agility Principles

Agile Principles | 8 of 12

Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- After a few iterations, the pace of development should stabilize as the teams mature and collaboration improves
- '*Committed*' team members are 100% invested in the project
- '*Involved*' team members such as business users should be able to consistently devote a fair share of their time to the development process
- Incremental and iterative development reduces burnout and stress due to timelines, thus, improving productivity
- Productivity improvement, in turn, promotes a sustainable pace

Agility Principles

Agile Principles | 9 of 12

Continuous attention to **technical excellence** and **good design** enhances agility.

- Design should consider present and future scenarios for the feature being worked on
- Quality processes such as peer reviews should be built into the team culture
- Technical debt should be minimum and cleared at the earliest
- Adoption of latest technological trends and best practices to be considered as and when possible
- Time dedicated to quality may be factored in during iteration planning for more accurate estimation
- Improvement is a continuous process and must be a shared responsibility of everyone on the team

Agility Principles

Agile Principles | 10 of 12

Simplicity – the art of maximizing the amount of work *not* done – is essential.

- Work that does not deliver any value to the customer should not be undertaken
- Suggestions from within the team for enhancement of features in scope must first be evaluated by the business
- *'Gold-plating'* should be avoided
- Information radiators should be used to identify and arrest scope creep as early as possible
- The simpler the solution, the more effective it is

Agility Principles

Agile Principles | 11 of 12

The best architectures, requirements, and designs emerge from **self-organizing teams**.

- Mutual trust and collaboration are essential factors for building self-organizing teams
- Respect for the individual and collective decision-making result in an open work culture
- An open culture fosters free exchange of constructive ideas that contribute to improvement of the product
- Self-organizing teams are strongly focused on a common goal of delivering a product with the highest quality
- Over time, as the team matures, it adapts more quickly to the needs of a changing business environment

Agility Principles

Agile Principles | 12 of 12

At regular intervals, the team **reflects on** how to become **more effective**, then **tunes and adjusts** its behavior accordingly.

- Introspection and retrospection are key tools for continuous improvement leveraged by self-organizing teams
- Data points for identifying improvement areas can be obtained from various information radiators updated on a daily basis
- End of an iteration is a preferred retrospection point
- Improvement measures should be unanimously adopted
- Lessons learned and implemented may be formally captured for future reference

Planning

Why Plan?

- Helps in decision making
- Helps in reducing risks
- Establishes trust
- Provides basis for checking progress

Common Planning Mistakes

- Uncertainty is ignored - Trying to plan too much when too little is known
- Delay in the beginning itself - Too much time and effort goes in planning
- Rigid Plans – trying to fit the work around plan rather than updating plan based on real time changes
- Plans are activity driven rather than feature
  - Incorrect measurement x% complete is not same as x% value
  - Parkinson's law – Work expands to fill the time available
  - Activities are often dependent so delay in one cause delay for others
- Features are not developed by Priority

Agile Planning

- "Planning is everything, plan is nothing"
  - Focus on iterative planning rather than trying to plan everything up front and then trying to stick with initial plan
- Inspect and adapt - Gives a plan that can be changed whenever we learn something new
- Planning happens at every level including product strategy, product roadmap, release planning and iteration/sprint planning.
- Agile planning is around working product rather than around activities.

It's better to be roughly right than precisely wrong."

John Maynard Keynes

Agile Planning- Onion

Strategy & Portfolio

- Organization Level Strategic planning
- Product selection inline with org strategy

Product

- Product requirements met in form of releases.

Iteration

- Short fixed length 1-4 weeks long timeframes.
- Planning focus here is to deliver potentially shippable product in each iteration.

Day

- Every day the team plan how to complete the highest priority features and deliver working software.

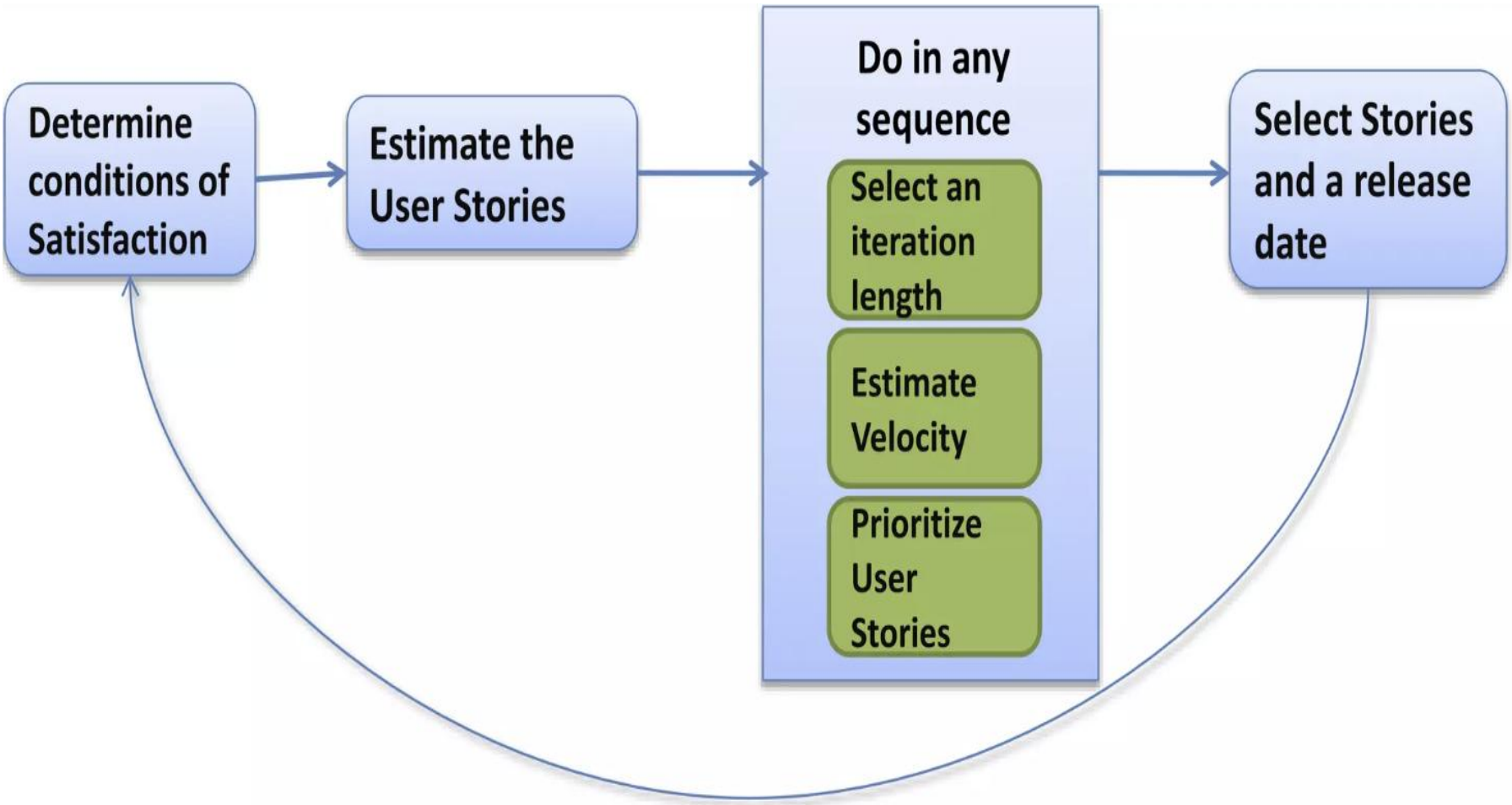
Release

- Release contains number of iterations.
- Release planning focused on identifying and delivering minimal set of features that will help to go in market as early as possible.

Release Plan

- It helps the product owner and the whole team to decide how much must be developed and how long that will take before they have a releasable product.
- It feeds into strategic planning activities
- Serves as a guidepost towards which project team can progress.
- Two types of plan
  - Feature Driven – Scope is fixed. Schedule is flexible.
  - Date-Driven – Time is fixed. Scope is flexible.

Release Planning Steps



Ref: Mike Cohn's "Agile Estimation and Planning"

Release Planning-Iteration Length

One of the key element of release planning is to decide iteration length. Factors in deciding iteration length:-

- Length of release – Shorter iterations are preferred for shorter releases as that allows more regular feedback
- Level of uncertainty/Risks – Keep shorter iteration if risk is high
- How long priority can remain unchanged
- The ease of getting feedback
- Overhead of iterating
- Feeling of urgency is maintained

Release Planning

How to select stories for release:

- use the story map,
- make feedback loop short,
- take stories that help to learn, bring value to users or market or mitigate risk. Or take stories that are dependant on each other.

There are two types of release planning: 1) fixed scope, variable time (how long will it take?), 2) fixed time, scope changes (what can we deliver?)

Release Planning

Fixed scope planning

How to calculate the duration of release and the number of sprints:

1. Determine sprint length.
2. Calculate velocity range.
3. Total up an estimate for selected stories.
4. The number of sprints is the total estimate divided by velocity.
5. Duration is the number of sprints multiplied by sprint length.

Fixed date planning

How to calculate what you are going to deliver:

1. Groom backlog.
2. Determine sprint length.
3. Calculate velocity range.
4. Calculate the number of sprints.
5. Calculate release capacity by multiplying the number of sprints by velocity (multiply two times to get points range)
6. Include items from the top of the backlog until total points exceed points range (will have, might have).

Release Plan: Type

Fixed scope planning

How to calculate the duration of release and the number of sprints:

1. Determine sprint length.
2. Calculate velocity range.
3. Total up an estimate for selected stories.
4. The number of sprints is the total estimate divided by velocity.
5. Duration is the number of sprints multiplied by sprint length.

Release Plan: Type

Fixed date planning

How to calculate what you are going to deliver:

1. Groom backlog.
2. Determine sprint length.
3. Calculate velocity range.
4. Calculate the number of sprints.
5. Calculate release capacity by multiplying the number of sprints by velocity (multiply two times to get points range)
6. Include items from the top of the backlog until total points exceed points range (will have, might have).

Release Plan

Inputs, attendees, and outputs of a release planning session

Inputs	Attendees	Outputs
• Prioritized product backlog with Estimates • Product vision • Team velocity • Agenda • Date (optional)	• Product owner or customer • Delivery Team(s) • Agile project manager • Team Leads (optional) • Stakeholders (optional)	• Release plan • Assumptions • Risks • Action Items • Dependencies • Release backlog

Agile Estimation

- Effort vs Duration.
- Accuracy vs Precision.
- Relative vs Absolute.

Agile Estimation

Concept 1. Effort vs. Duration. *Duration* — is the number of calendar days needed to finish the task, including waiting periods. *Effort* — is the amount of time (days) needed to perform the task, not including waiting periods. *Ideal days in Agile* — how many work days will it take to complete a story, if you work on it uninterrupted.

Concept 2. Accuracy vs. Precision. We prefer accuracy. Precise estimation takes too much time. So you can group tasks in approximate ideal days.

Popular scales:

- 1, 2, 4, 8
- Fibonacci series: 1, 2, 3, 5, 8, 13, ...
- 1, 5, 10, 20
- XS, S, M, L, XL
- Small, Medium, Large.

Agile Estimation

Concept 3. Relative vs. Absolute.

Relative sizing examples:

- If a story A is 1 point, then story B is 5 points, story C is 10 points.
- Small, Medium, Large.
- XS, S, M, L, XL.

Absolute sizing examples:

- Story A will take 1 day, Story B will take 5 days.
- Story A will take 1 hour, Story B will take 10 hours.

Absolute sizing can take longer, but it doesn't take that long if you are using buckets.

Estimation Styles

- Planning Poker
- Card Sorting
- Velocity Calculation.

Estimation Style: Planning Poker

1. Give everybody estimation cards
2. Explain the story
3. Discuss to understand the story
4. Estimate and put one card down
5. Open cards
6. If consensus, move to the next item. If not, go back to step 3.

Cons of planning poker: Time-consuming

Pros of planning poker:

- Uncovers misunderstandings
- Collective ownership
- Engaged
- Good for a backlog grooming session

Estimation Style: Card Sorting

Step 1. If you don't have the buckets, place the smallest story on the left and the largest on the right.

Step 2. Everybody silently places stories between these two cards. If you already have buckets, sort stories in the buckets. If you don't have buckets yet, you will create them in step 4.

- Separate column for stories with questions.
- Stack the same size stories vertically.
- Disagree? Move the story silently.

Step 3. Discuss the changes

- Discuss stories in the column with questions and put them on the board.
- Take another look.
- Discuss disagreements and move stories if needed.

Step 4. Create buckets.

Step 5. Assign size to buckets.

Velocity

Velocity is the amount of work the team gets done in a sprint. It changes based on team, project and other factors. Velocity is used for predicting velocity in the upcoming sprint.

Velocity

Let's imaging this is the table with velocities of your last sprints:
Sprint 21 — velocity 15.
Sprint 22 — velocity 17.
Sprint 23 — velocity 54.
Sprint 24 — velocity 25.
Sprint 25 — velocity 25.
Sprint 26 — velocity 27.
Sprint 27 — velocity 23.

To calculate the velocity for planning upcoming sprint you can use:
- Last sprint velocity. Then the velocity of the sprint #28 will be 23.
- Average of last X sprints (let's say we will use 3). Then the velocity of the sprint #28 will be (25+27+23):3=25.
- Velocity range of last X sprints (let's say we will use 3). Then the velocity of sprint #28 will be from 23 to 27.

When calculating skip anomalies. For example, skip sprint 23, if you use 5 last velocities.

Velocity

If you are calculating velocity for the first sprint:

- Take sample stories from the backlog that the team think the can deliver in a sprint.
- Task out the stories to understand the work involved.
- Sum up the estimate of stories.

If more people are joining the next sprint, or leaving the team, or going on vacation:

- Continue to use the velocity of previous sprints. Exception — when most of the team is going.

If your estimates were off:

- Don’t worry, estimations will average out.
- If a lot of stories are going to another estimate bucket, update estimates.

Release Planning-Estimate Velocity

Team velocity is needed for release planning to estimate how much can be delivered in every iteration.

Three options available to calculate team velocity:-
1) Use historical values 2) Run an iteration 3) Make a forecast

Use Historic Values

Good to have historic values but as every project & team is unique, these have limited value

Run an Iteration

Ideal way if feasible. After each iteration the range of possible velocity figures will converge

Make Forecast

Estimate based on team's capacity

Release Planning- Velocity Forecast

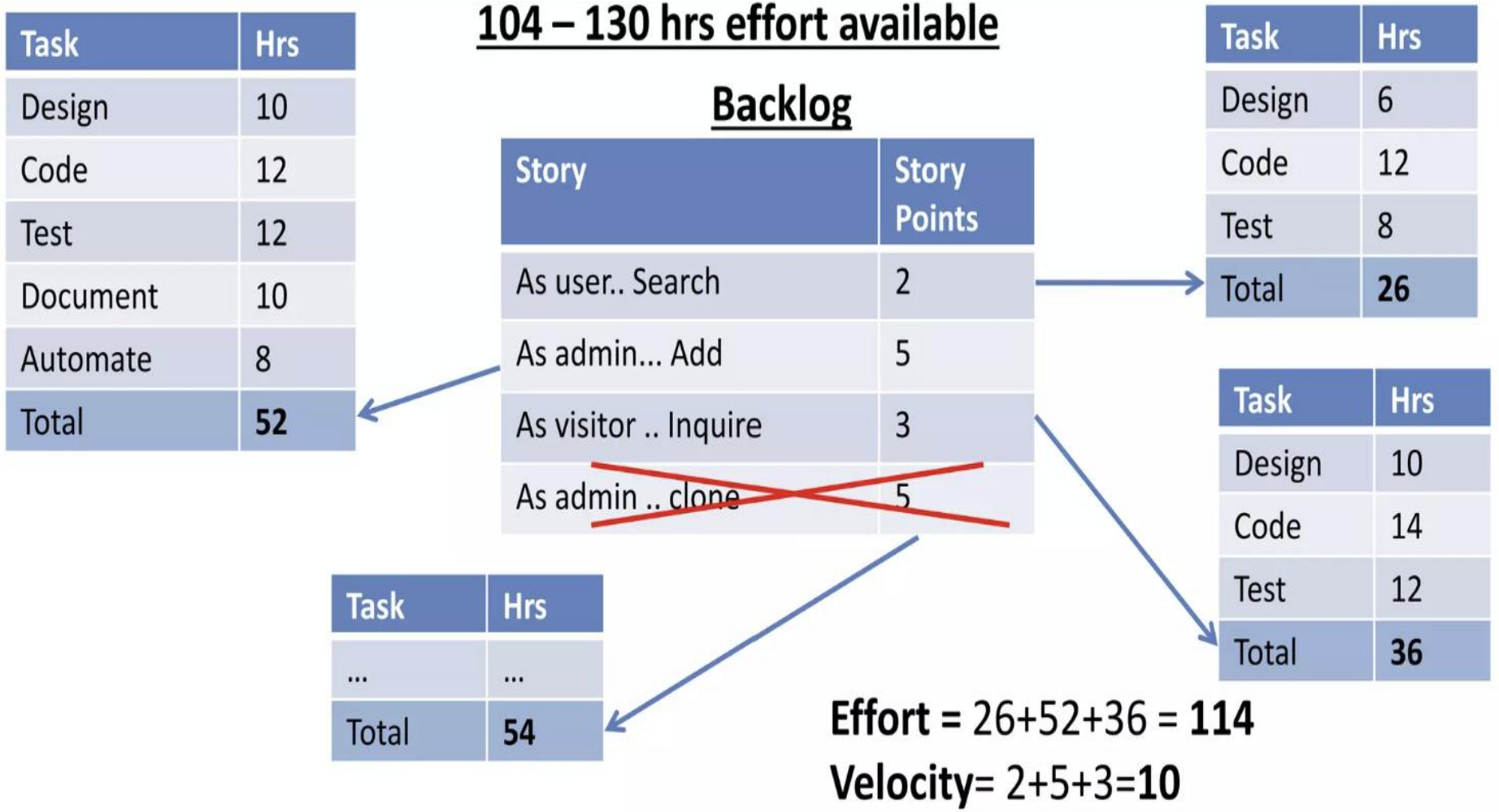
- Estimate number of hours effort available based on team size & team members availability.
- Pick few stories, break these into tasks and estimate these. Identify enough stories and tasks to fill the number of hours available.
- Based on story points of selected stories, velocity can be determined. Instead of using a single value, it's better to determine velocity in range.

Calculate Team's capacity – An example

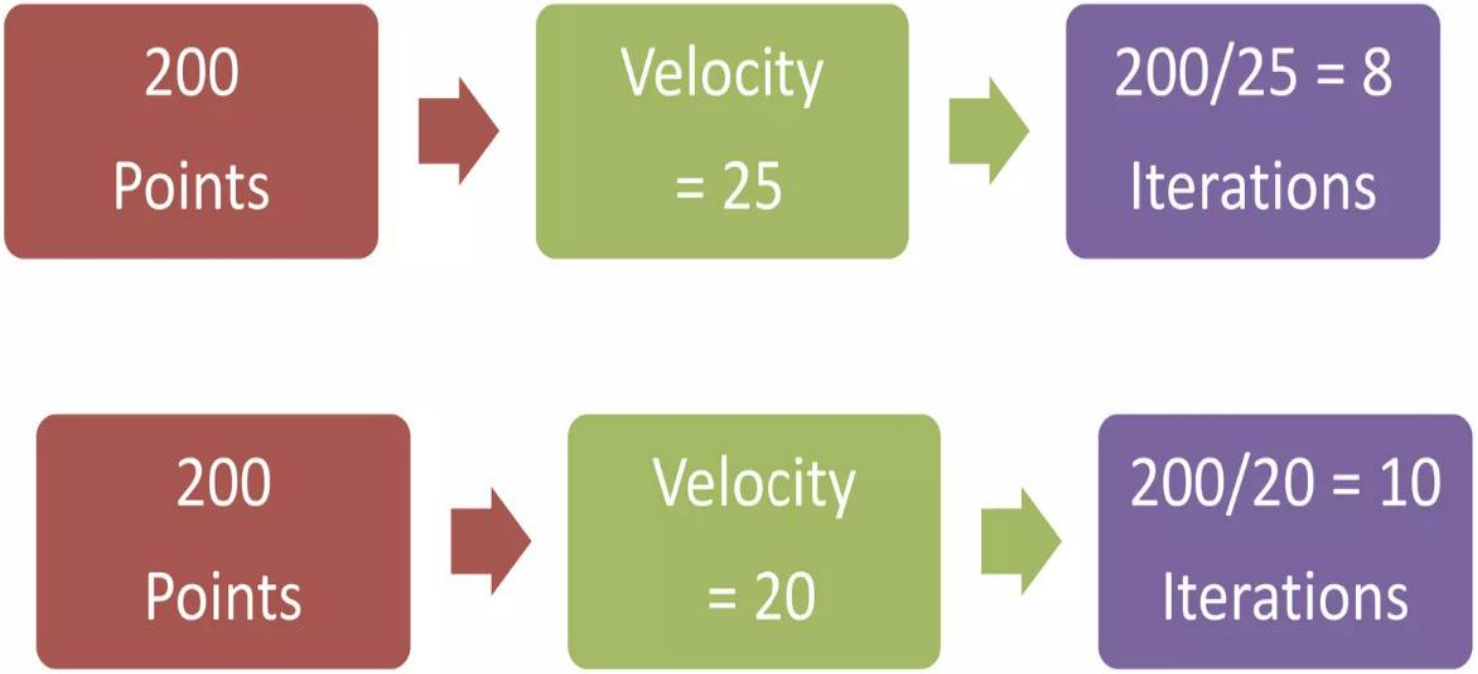
Person	Hours Available per iteration
Tom	32 – 40
Harry	36 – 40
Rita	20 – 28
Han	16 – 22
Total Hours	104 – 130

104 – 130 hrs effort available

Release Planning- Velocity Forecast



Using Velocity for Planning



Update release plan with some regular frequency. For e.g. if there is any major difference between velocity assumed initially and actual velocity.

User Stories

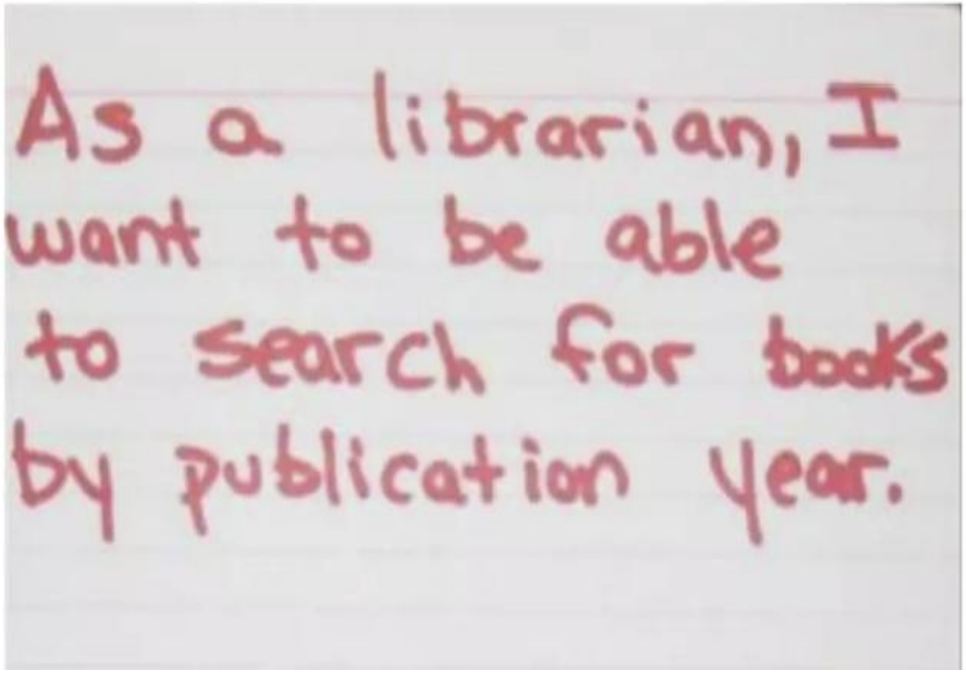
- Definition: Short, simple description of a feature from the user's perspective.

  Importance: Ensures customer needs are met, Provides clear requirements for development.

- Example: 'As a user, I want to reset my password so that I can regain access.'

User Stories

- A user story is one or more sentences in the everyday or business language of the end user that captures what the user wants to achieve. User stories are used with Agile software development methodologies for the basis of what features that can be implemented.
- Each user story is limited, so it fits on a small paper note card to ensure that it does not grow too large.



User Story

- User stories are a quick way of handling customer requirements without having to elaborate vast formalized requirement documents and without performing overloaded administrative tasks related to maintaining them.
- Story must describe the What not How.
- The intention of the user story is to be able to respond faster and with less overhead to rapidly changing real-world requirements.

Characteristics of Good User Stories

USER STORY SHOULD BE

- **Independent** - stories should not be dependent on one another. Dependencies between stories lead to prioritization and planning problems.
- **Negotiable** - They are not written contracts or requirements that the software must implement. Story cards are short descriptions of functionality, the details of which are to be negotiated in a conversation between the customer and the development team.
- **Valuable** - each story must bring some business value.
- **Estimable** – the scope of as story must be observable.
- **Small** – to estimate, to track progress
- **Testable** - must be a criteria of "done"

User Story Author

- The user stories should be written by the customers for a software project and are their main instrument to influence the development of the software.
- Business analyst can be a surrogate of a customer.
- User stories could also be written by developers to express non-functional requirements.

User Story Template

user story template

WHO are we building it for? Who is the user?	As a <type of user>
WHAT are we building? What is the intention?	I want <some goal or objective>
WHY are we building it? What is the value for the customer?	So that <benefit/value>

User Story Template

- As a **<role>**, I want **<goal/desire>** so that **<benefit>**.

  - Example:
  - As a **user** I want **to create organizer records**, so that I can **store my plans scheduled somewhere**.

User Story Writing Workshop

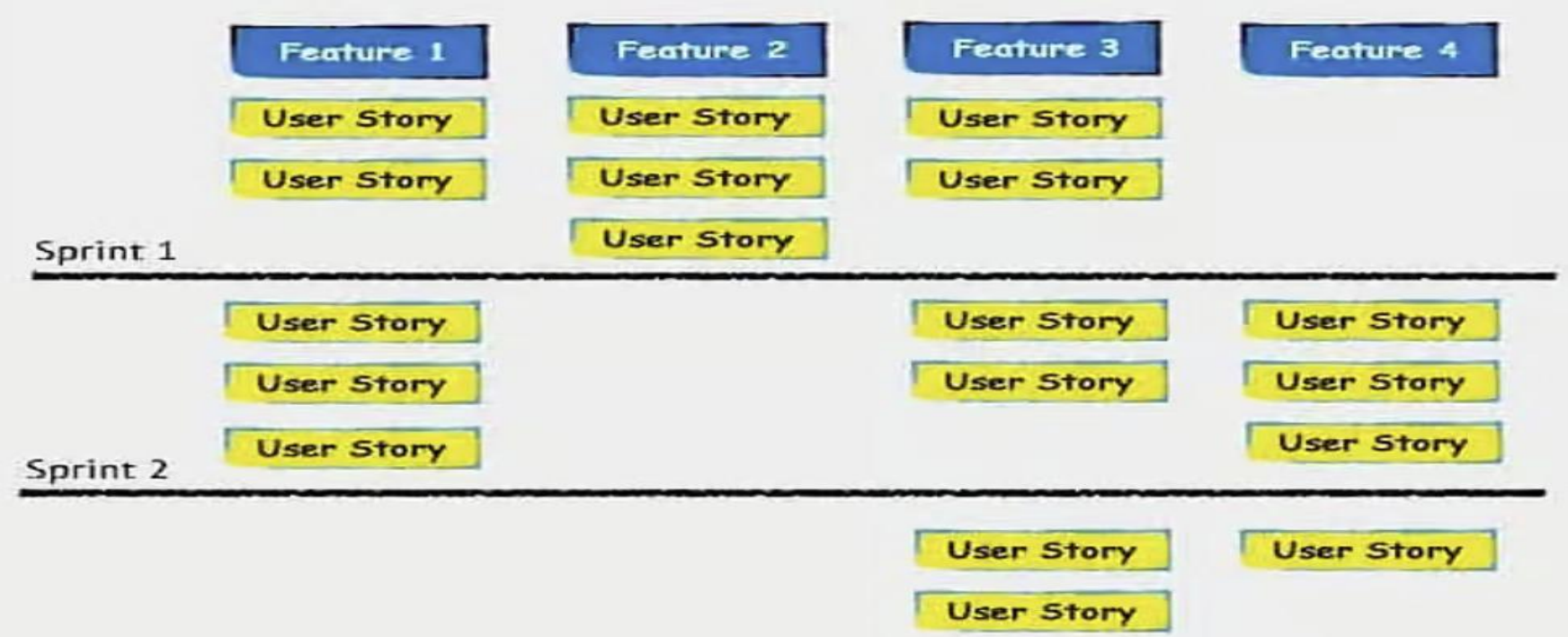
- Definition: Collaborative event where teams refine user stories.
- Participants: Product Owners, Developers, UX Designers, Stakeholders.
- Activities: Brainstorming, Story Splitting, Prioritization.

User Story Writing Workshop

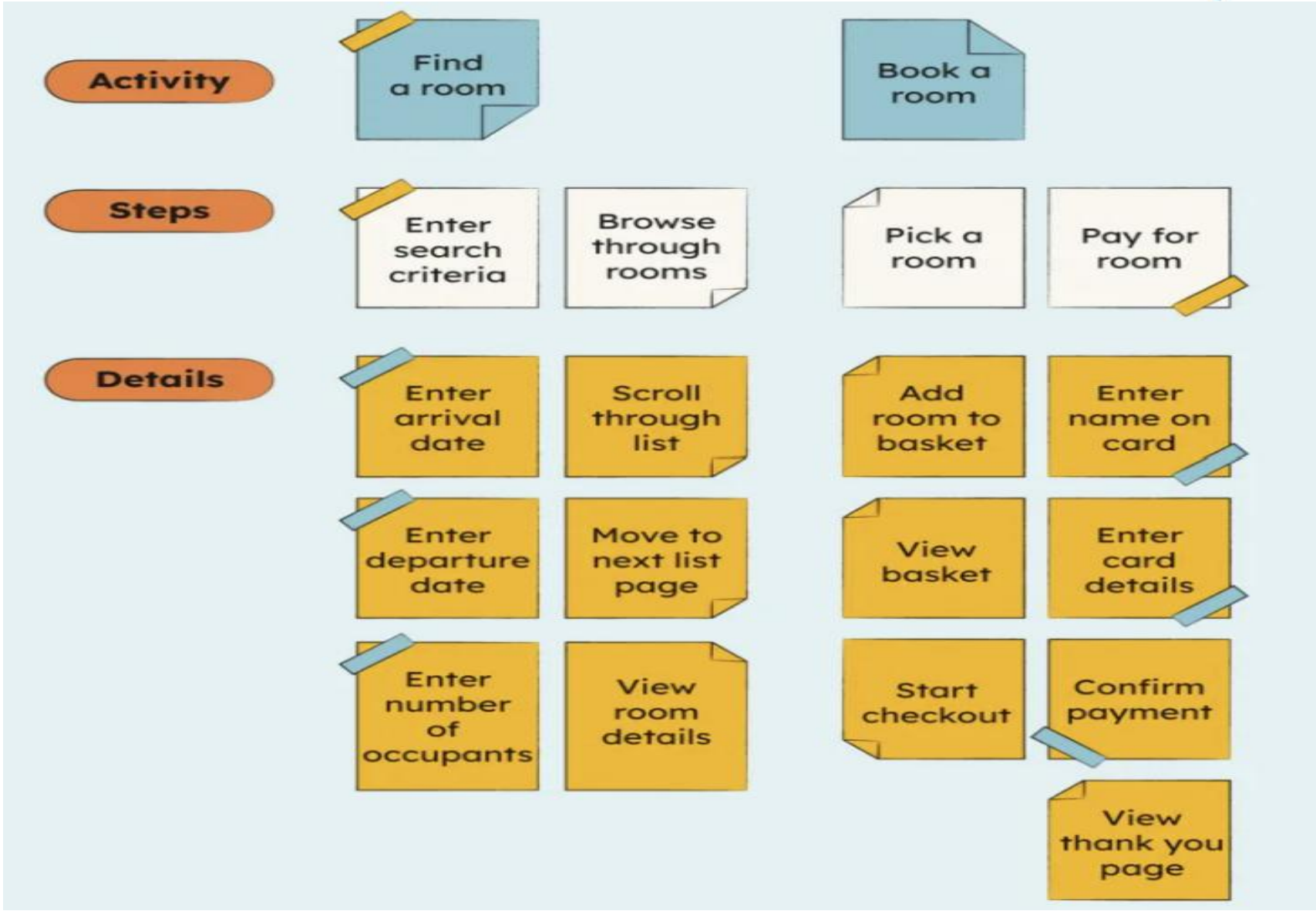
Use story no	As a <type of user/ persona>	I want to <goal/ objective>	So that <benefit/ result/ some reason>
1	Project team member	know all my tasks in advance	I can prepare and plan my time properly
2	Content manager	get a weekly report of content analytics	I can monitor the effectiveness of content writer
3	CEO	get a weekly report from all department heads on their team goals	I know whether my strategy is working
4	Middle-aged woman on a sabbatical	eam a certificate for digital marketing from coursera	I can restart my career
5	Software developer	reskill myself by attending the training program organized by my employer	I can stay employed

Story Mapping

- **Definition:** Technique to visualize user interactions and prioritize tasks.
- **Steps:** Identify activities, Break into tasks, Organize into roadmap.



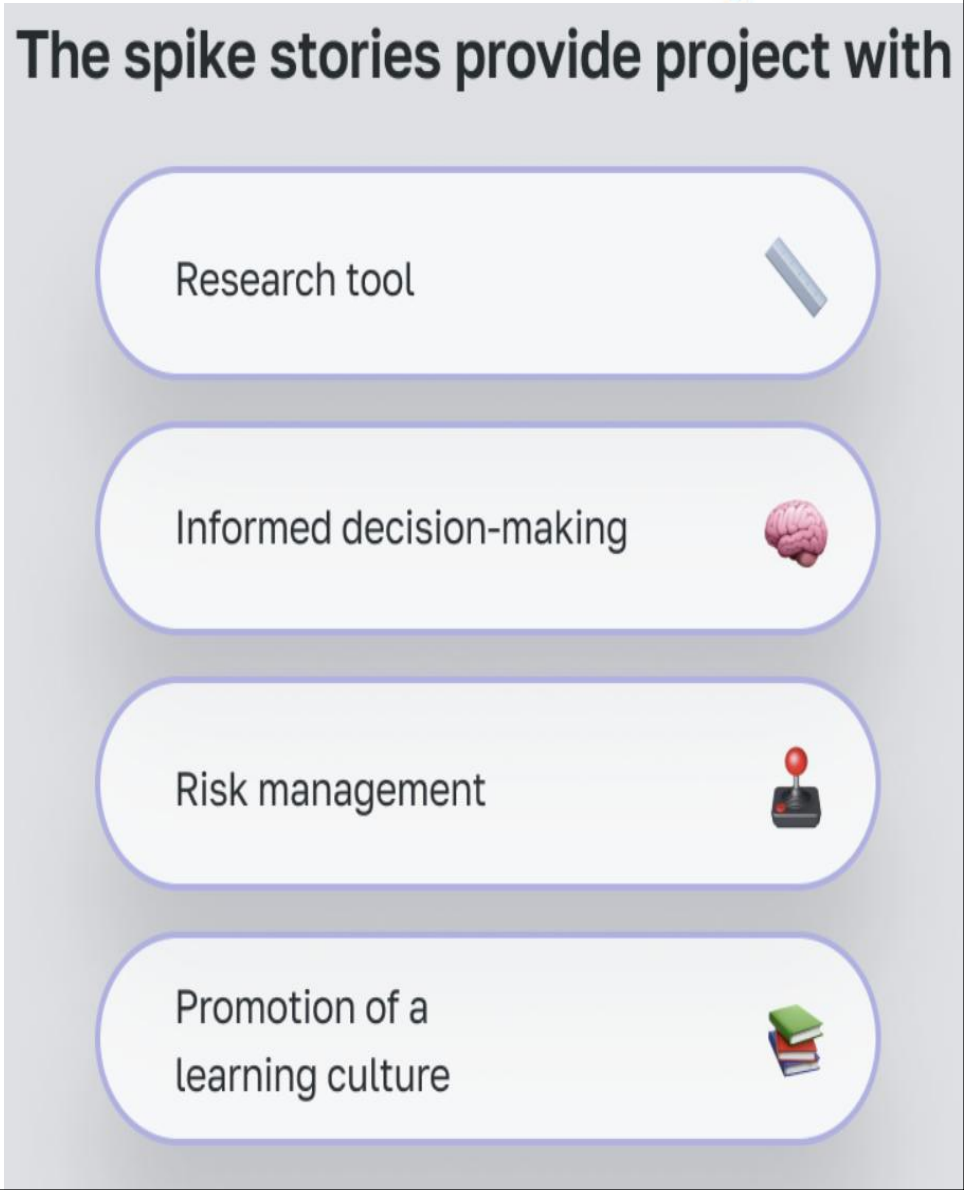
Story Mapping



Spike

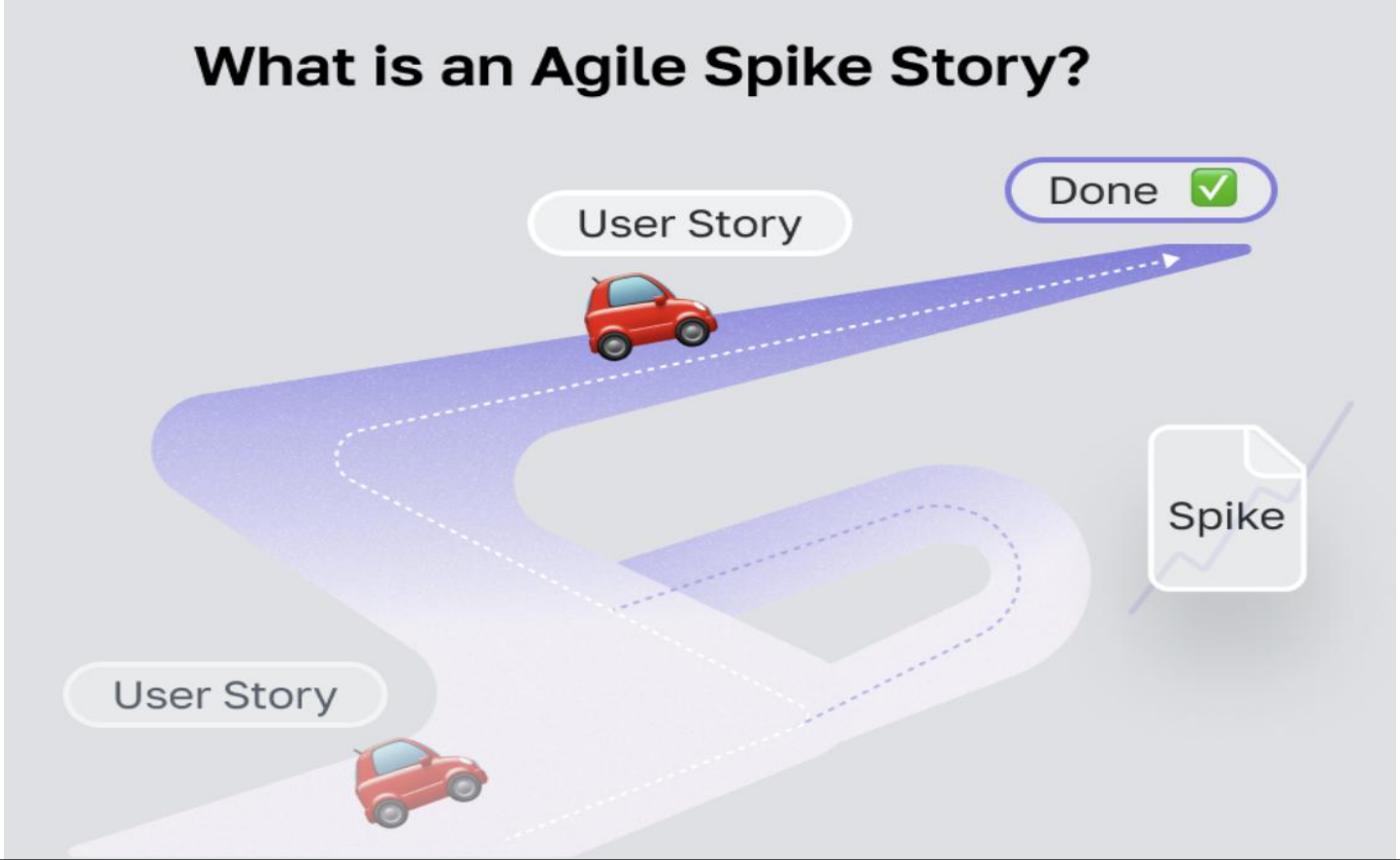
- **Definition:**

A spike story in Agile is a user story that needs more information so the team can estimate how long the story will take to complete.



Spike

Spike stories are like the secret agents of Agile software development. When you stumble upon a thorny problem or a murky area in the project, you send in a spike story to gather intelligence and pave the way for smoother progress.



Kanban

- Kanban is a visual work management system that provides a clear representation of work progression within a process.
- It offers a visual depiction of both the workflow process and the tasks in transit.
- The primary aim of Kanban is to pinpoint any possible obstructions in your process and resolve them, enabling work to move efficiently at an optimal pace, thus ensuring cost-effectiveness.

Kanban

Five Core Principles of Kanban

1. Visualize Work Flow

2. Limit Work in Progress

3. Measure and Manage Flow

4. Make Process Policies Explicit

5. Use Models to Recognize Improvement Opportunities

Kanban Board

Scrum Model

- Scrum is an Agile framework for managing and organizing work on complex projects, primarily used for software development but applicable to various fields.

- It was originally formalized for software development projects in the early 1990's and has since gained widespread adoption in various industries.

Components of Scrum Model

1. Roles:

- **Product Owner:** Represents the stakeholders and is responsible for defining and prioritizing the product backlog.
- **Scrum Master:** Facilitates the Scrum process and ensures that the team adheres to its practices.
- **Development Team:** Cross-functional and self-organizing group responsible for delivering the product increment.

2. Artifacts:

- **Product Backlog:** A prioritized list of features, enhancements, and bug fixes that need to be addressed in the product.
- **Backlog:** A subset of the product backlog selected for a specific sprint, containing tasks the team commits to completing.
- **Product Increment:** The sum of all the completed product backlog items at the end of a sprint.

Components of Scrum Model

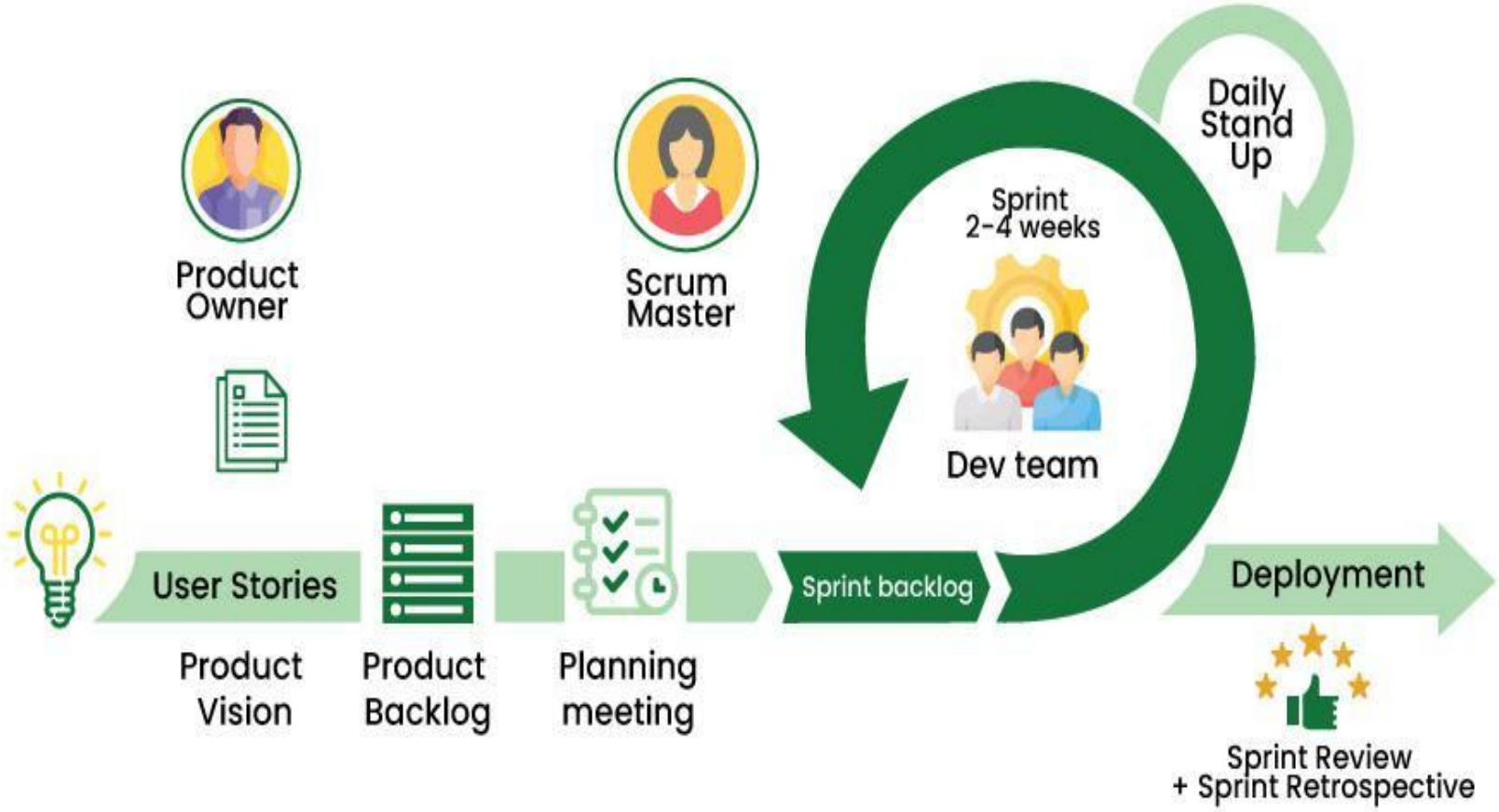
3. Events:

- **Sprint:** A time-boxed iteration (usually 2-4 weeks) during which a potentially shippable product increment is created.
- **Sprint Planning**: A meeting at the beginning of each sprint where the team plans the work to be done.
- **Daily Scrum (Stand-up):** A short daily meeting where team members discuss progress, plan for the day, and identify and address impediments.
- **Sprint Review:** A meeting at the end of each sprint to review the completed work and gather feedback.
- **Sprint Retrospective:** A meeting at the end of each sprint for the team to reflect on their processes and identify improvements.

4. Rules:

- Scrum emphasizes transparency, inspection, and adaptation.
- The product is built incrementally in fixed-length iterations (sprints).
- Changes are only made between sprints unless there is a compelling reason to make a change during a sprint.

Scrum Process Model



Scrum Process

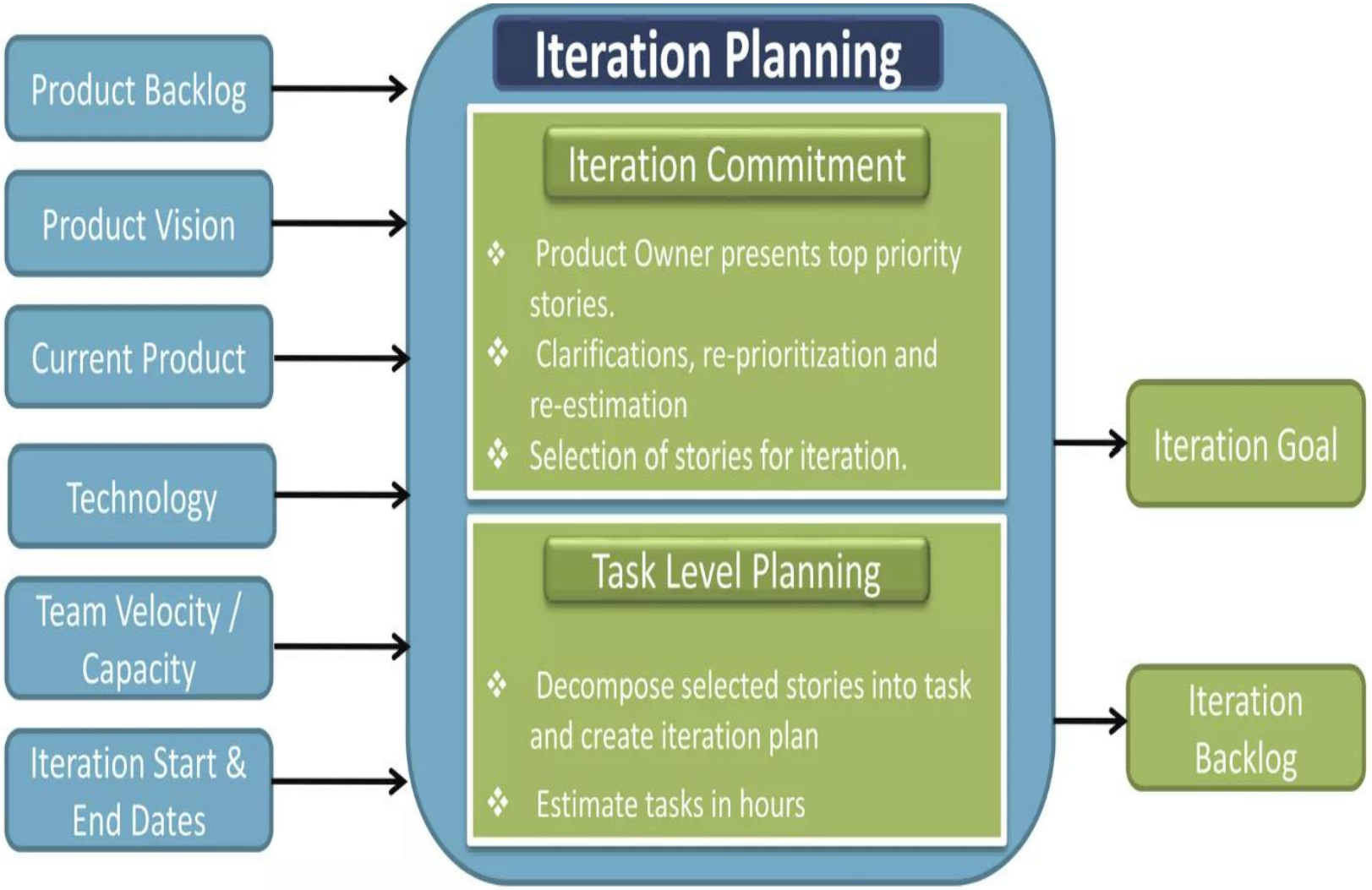
Scrum Process Model

1. **Product Backlog:** The Product Backlog is a prioritized list of features, user stories, enhancements, and bug fixes that need to be addressed in the product. It is managed by the Product Owner and serves as the source of work for the Development Team.
2. **Planning Sprint:** Sprint Planning is a key event at the beginning of each sprint. During this meeting, the Scrum Team, including the Product Owner, Scrum Master, and Development Team, collaboratively selects items from the Product Backlog to work on during the upcoming sprint. The team defines the Sprint Goal and creates the Sprint Backlog, detailing the tasks required to complete the selected items.
3. **Sprint Meeting:** The Sprint Meeting, often referred to as the Daily Scrum or Daily Stand-up, is a brief daily meeting where team members provide updates on their progress, discuss what they plan to work on next, and highlight any impediments. The goal is to synchronize the team's activities and ensure everyone is on the same page.

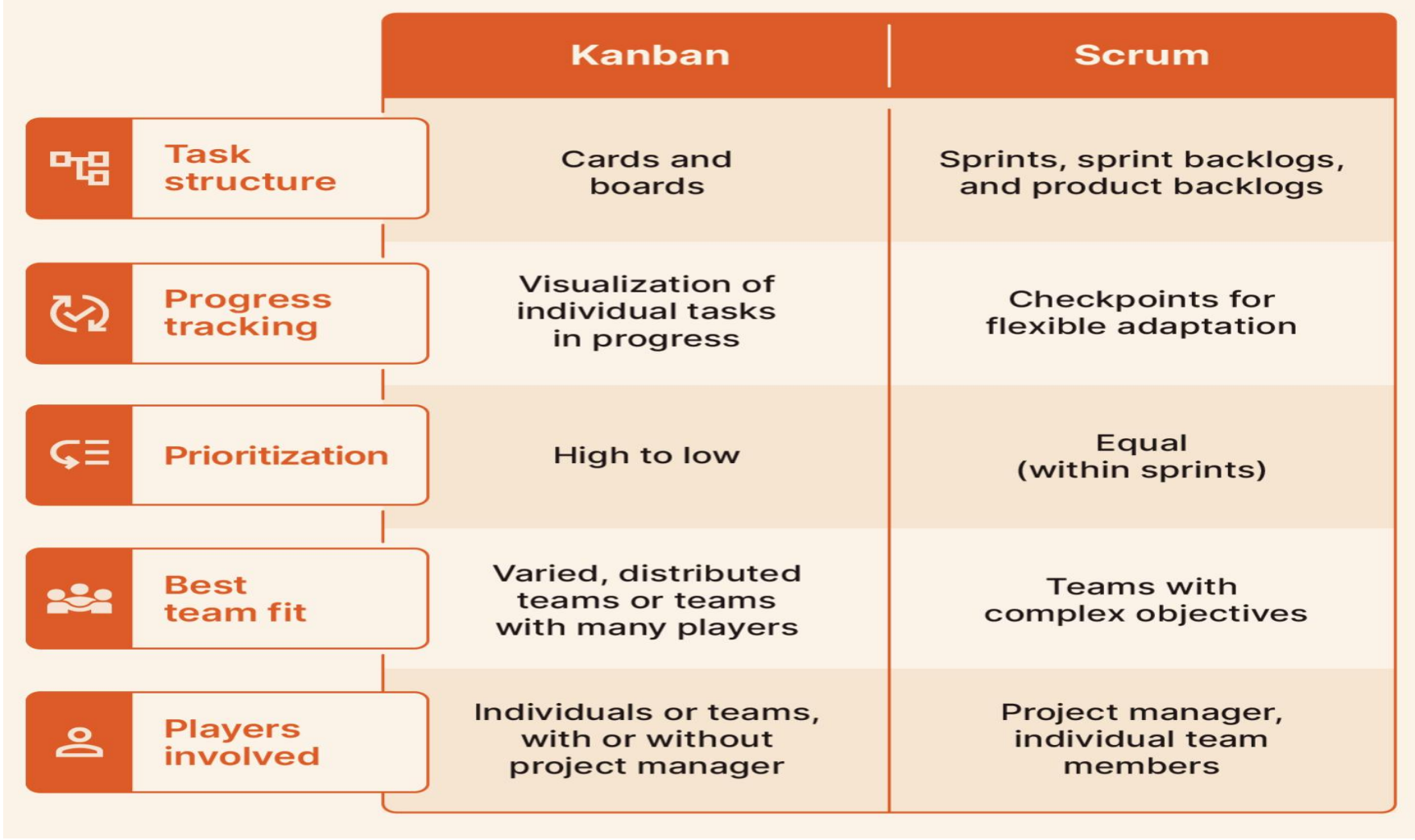
Scrum Process Model

4. **Sprint Review:** The Sprint Review is held at the end of each sprint. The Scrum Team, stakeholders, and the Product Owner come together to review the completed work. The Development Team demonstrates the product increment, and stakeholders provide feedback. This session informs future planning and adjustments to the Product Backlog.
5. **Sprint retrospective:** The Sprint Retrospective occurs after the Sprint Review and involves the Scrum Team reflecting on the previous sprint. The team discusses what went well, what could be improved, and any action items for enhancing their processes. The focus is on continuous improvement.
6. **Repeat:** After the Sprint Retrospective, the cycle repeats with a new Sprint Planning meeting, followed by another sprint of development, daily stand-ups, Sprint Review, and Sprint Retrospective. This iterative process continues throughout the project, allowing the team to adapt to changing requirements, continuously improve, and deliver increments of the product at the end of each sprint.

SCRUM: Sprint Planning



SCRUM vs KANBAN



	Kanban	Scrum
Task structure	Cards and boards	Sprints, sprint backlogs, and product backlogs
Progress tracking	Visualization of individual tasks in progress	Checkpoints for flexible adaptation
Prioritization	High to low	Equal (within sprints)
Best team fit	Varied, distributed teams or teams with many players	Teams with complex objectives
Players involved	Individuals or teams, with or without project manager	Project manager, individual team members