
SmartCliff
CAREER MOBILITY SOLUTIONS
We are on a mission to address the digital skills gap for 10 Million+ young professionals, train and empower them to forge a career path into future tech

SmartCliff
Career Mobility Solutions

Agile Methodolgies

JUL, 2025

Contents

Agile Methodology Overview

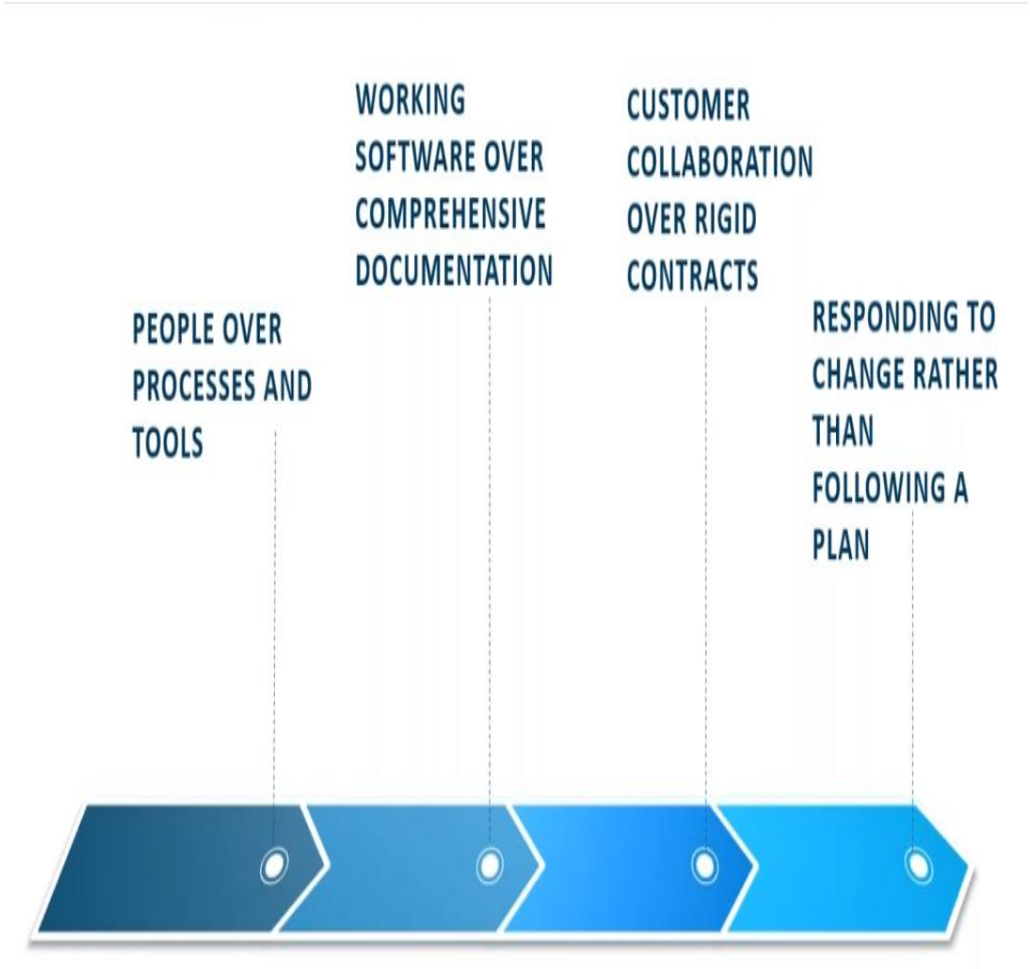
Introduction

- While Agile is a general approach used for software development, it relies heavily on teamwork, collaboration, time boxing tasks, and the flexibility to respond to change as quickly as possible.
- Agile is a software development methodology that focuses on iterative development, customer collaboration, and responding to change over following a rigid plan.
- **Agile emphasizes "working software over comprehensive documentation."**

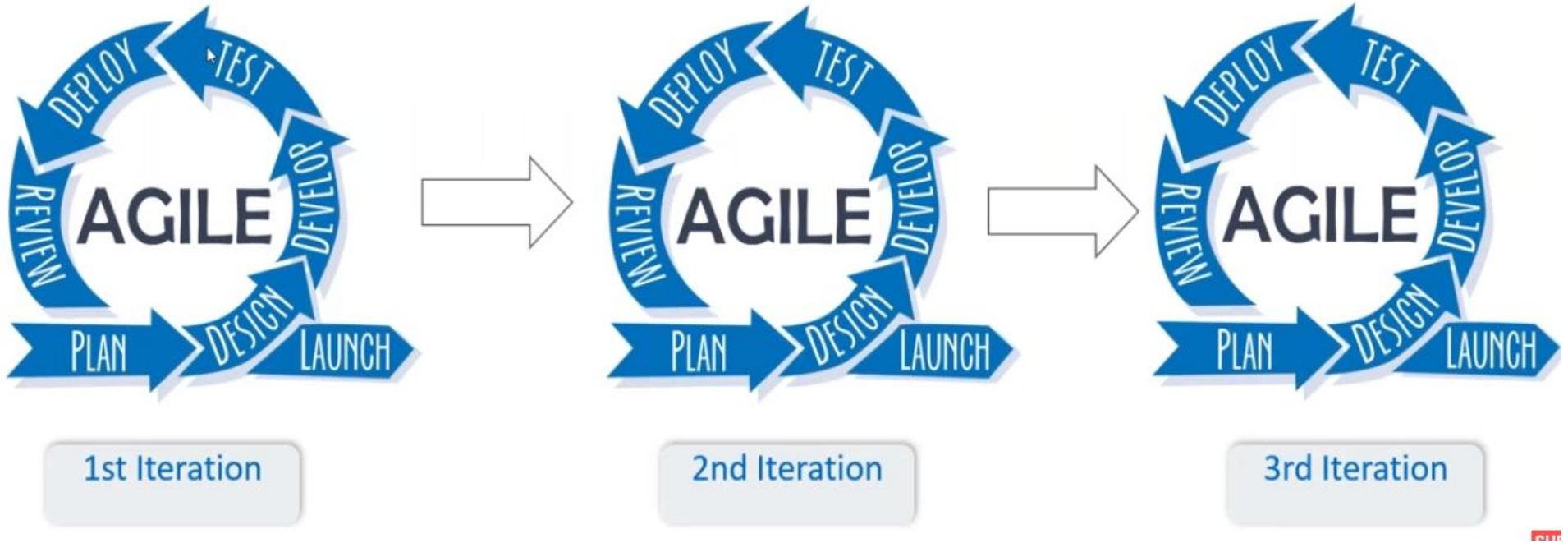
Introduction

The agile manifesto has four important values:

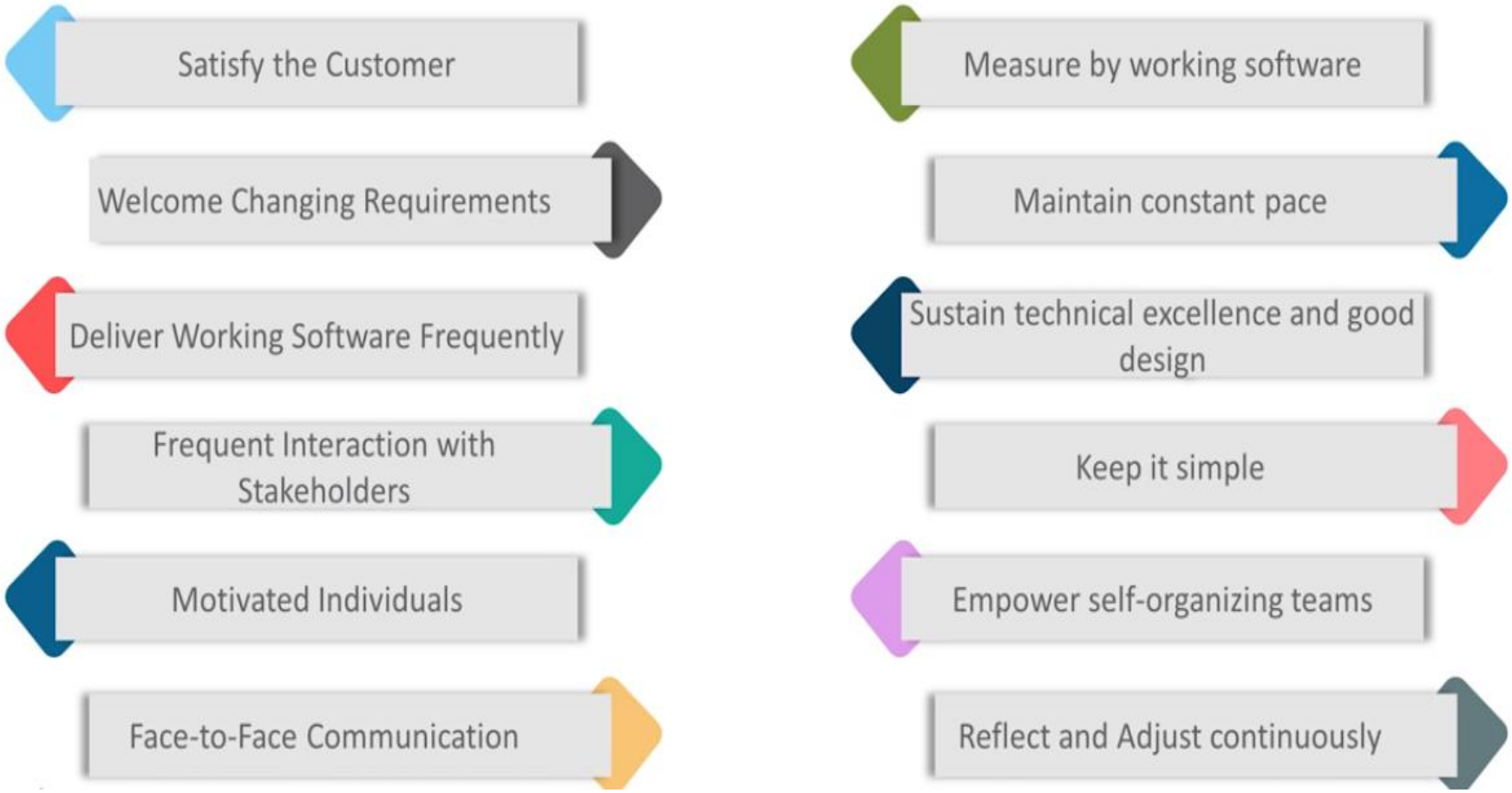
- More focus on individuals and interactions than processes and tools
- Working software is more important than comprehensive documentation
- Customer collaboration is more vital than negotiation
- The process should respond to change rather than blindly following a plan



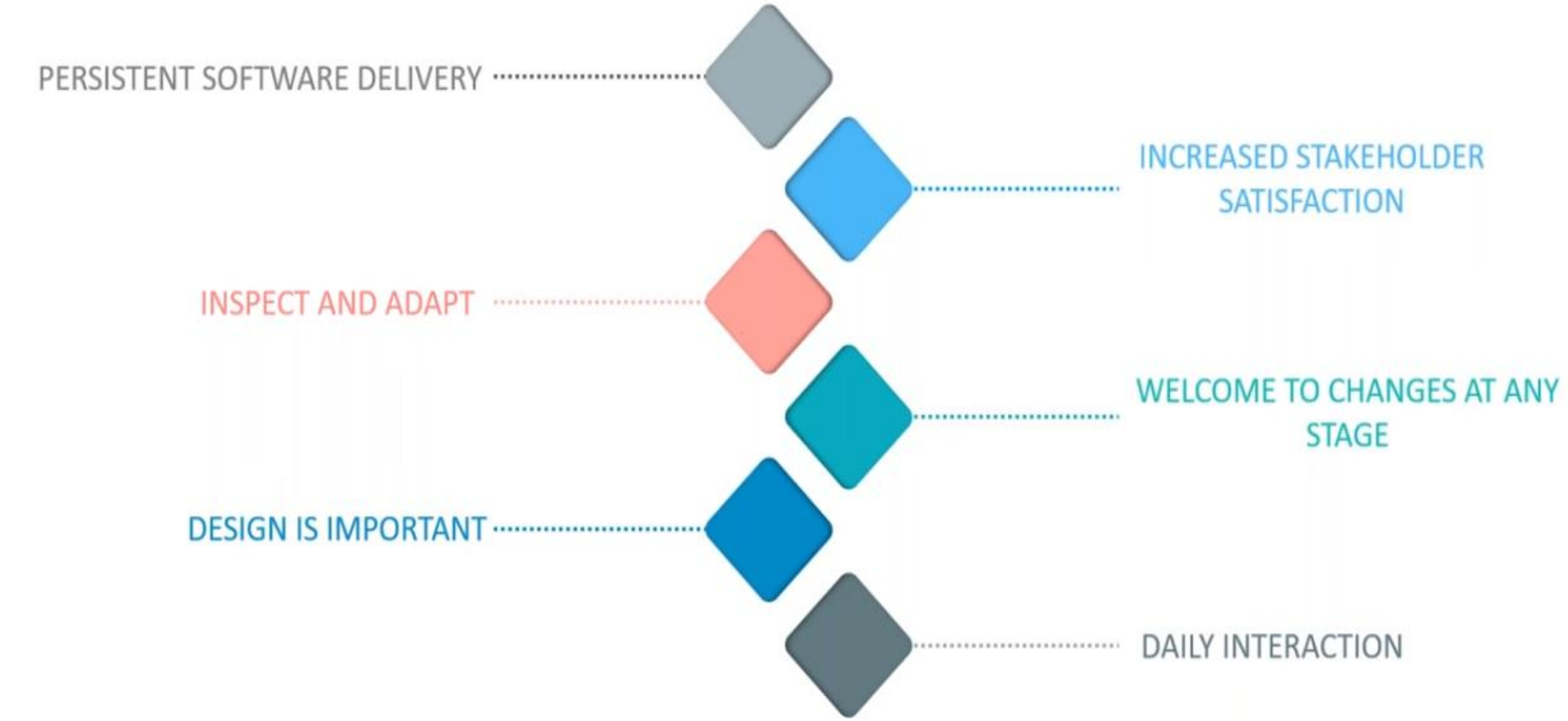
Agile - time boxing tasks as iteration



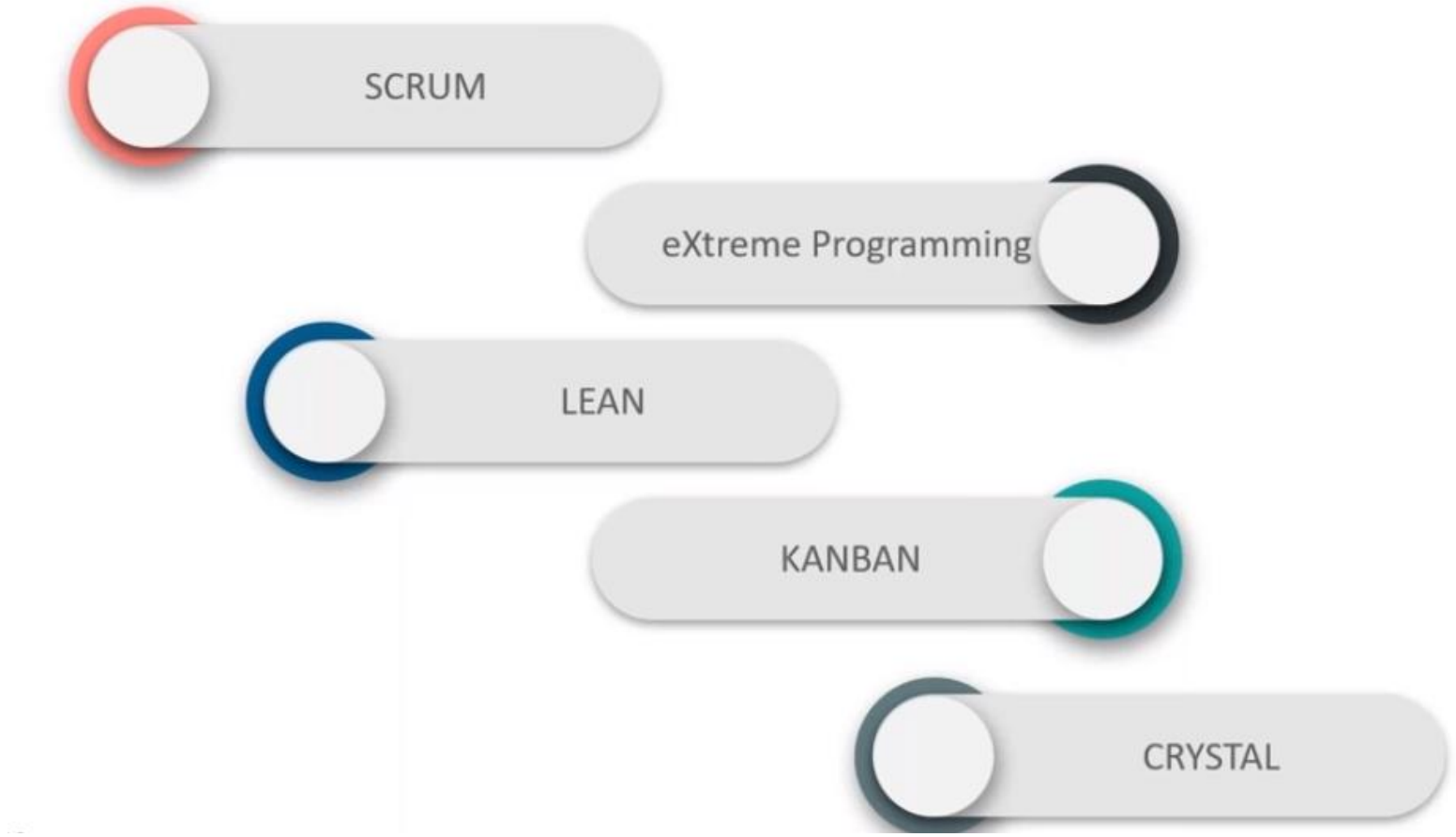
Principles of Agile



Advantages of Agile



How to implement agile ?



SCRUM

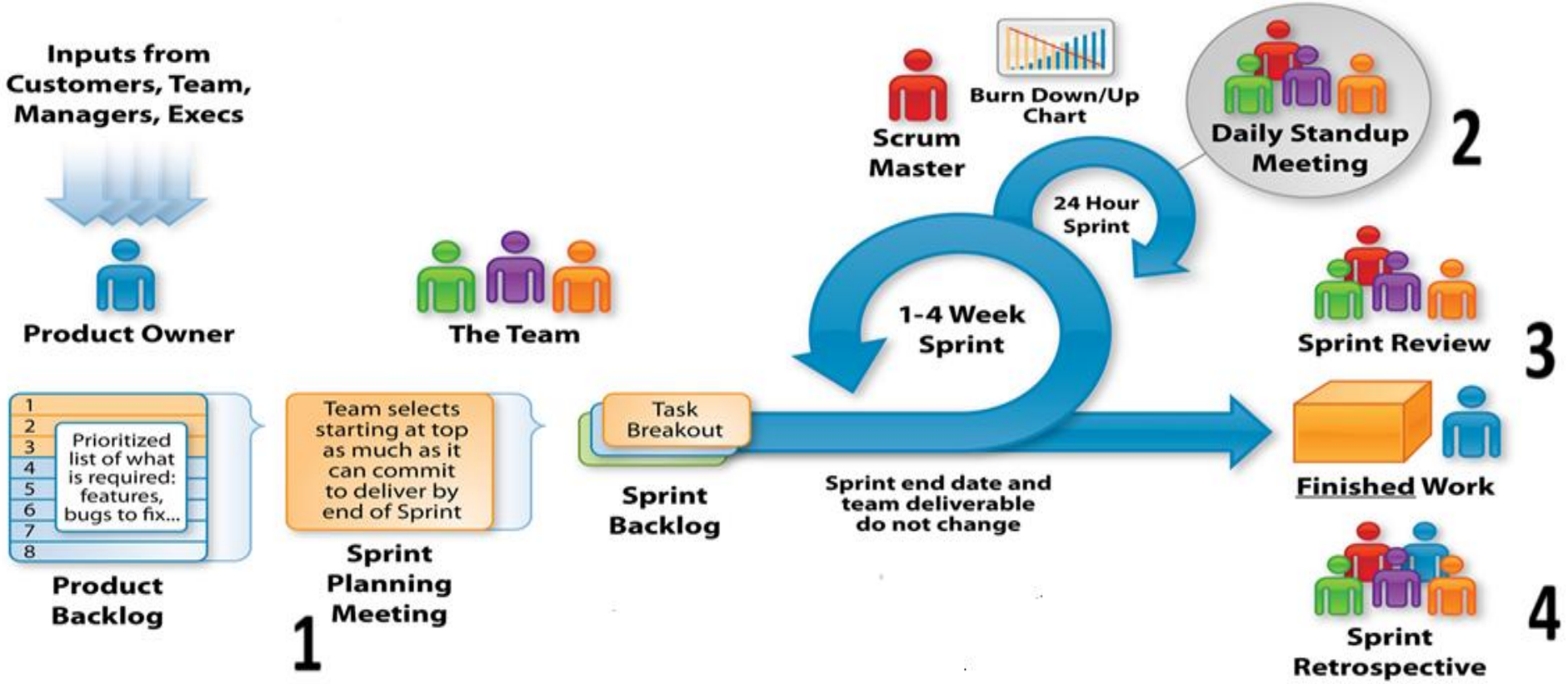
- Scrum is a lightweight, Agile framework used to develop, deliver, and maintain complex products. It promotes iterative and incremental development, continuous feedback, and team collaboration.
- It’s based on empirical process control, meaning decisions are made based on experience and observation, not assumptions.

Key Characteristics of Scrum:

- Time-boxed iterations called Sprints (1–4 weeks)
- Cross-functional teams (developers, testers, etc.)
- Regular ceremonies for planning, review, and improvement
- Continuous delivery of working, valuable software

SCRUM PROCESS FLOW

- Scrum is a lightweight, Agile framework used to develop, deliver, and maintain complex products.



SCRUM – Core Components

1.Product Backlog:

- A prioritized list of features, bugs, technical tasks, or enhancements.
- Owned by the Product Owner.
- Continually refined and updated.

Example:

- Search flights
- Filter hotels
- Add to cart
- Payment integration

SCRUM Roles and Responsibilities

- Scrum has three defined roles, each with clear responsibilities:

Role	Responsibilities
Product Owner (PO)	Owns the **Product Backlog**, defines the product vision, prioritizes features
Scrum Master (SM)	Acts as a **servant leader**, removes blockers, ensures Scrum is followed
Development Team	Cross-functional members who **design, build, test**, and deliver increments

SCRUM Roles and Responsibilities

Product Owner:

- Represents stakeholders and customers.
- Writes user stories and defines acceptance criteria.
- Decides what gets built and when.
- Collaborates with the team on priorities and expectations.

SCRUM Roles and Responsibilities

Scrum Master:

- Coaches the team in Scrum practices.
- Helps resolve impediments (e.g., delays in requirements).
- Facilitates Scrum ceremonies.
- Shields the team from outside interruptions.

SCRUM Roles and Responsibilities

Development Team:

- Usually 3–9 people.
- Responsible for delivering a potentially shippable product increment at the end of each sprint.
- Self-organizing and accountable.

Team task accountability *

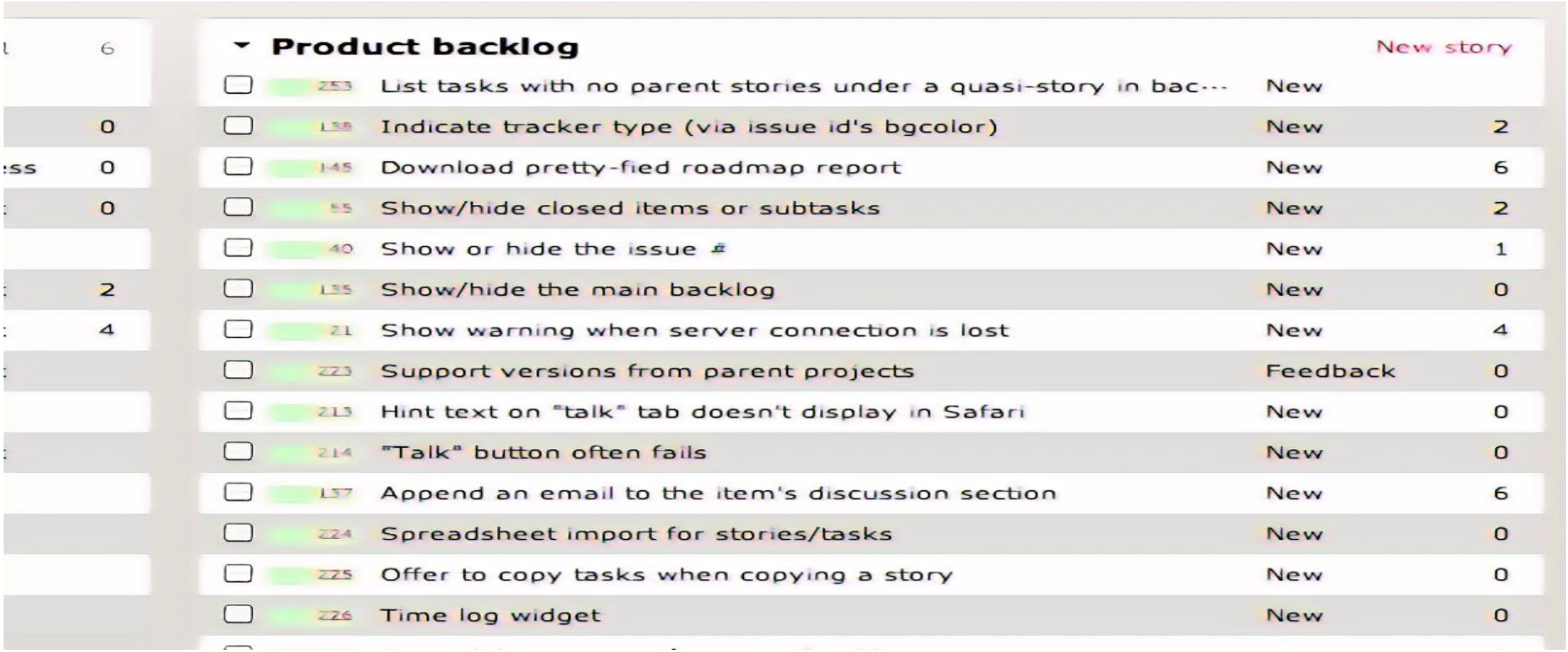
- **Developers are accountable for :**
- Creating sprint backlog, sprint plan, sprint goal.
- Instilling quality
- Adhering to definition of done
- Adapting their plan each day towards the sprint goal,
- Holding each other accountable as professionals

- **Product Owner is accountable for**
- Maximising value of sprint
- Developing and explicitly communicating product goal
- Ordering product backlog items
- Ensuring that the product backlog is transparent, visible and understood.

- **Scrum Master is accountable for**
- Establishing scrum by helping everyone understand scrum theory & practice, both within the team and the organization.
- Coaches team in self management and cross functionality.
- Helping the team focus in delivering high value. (protecting it from interruption)
- Causing removal of impediments
- Ensuring scrum events take place – positive, productive and time boxed.
- Helps Product owner communicate vision to team
- Sets up collaboration channels with stakeholders for PO and team.

SCRUM – Core Components

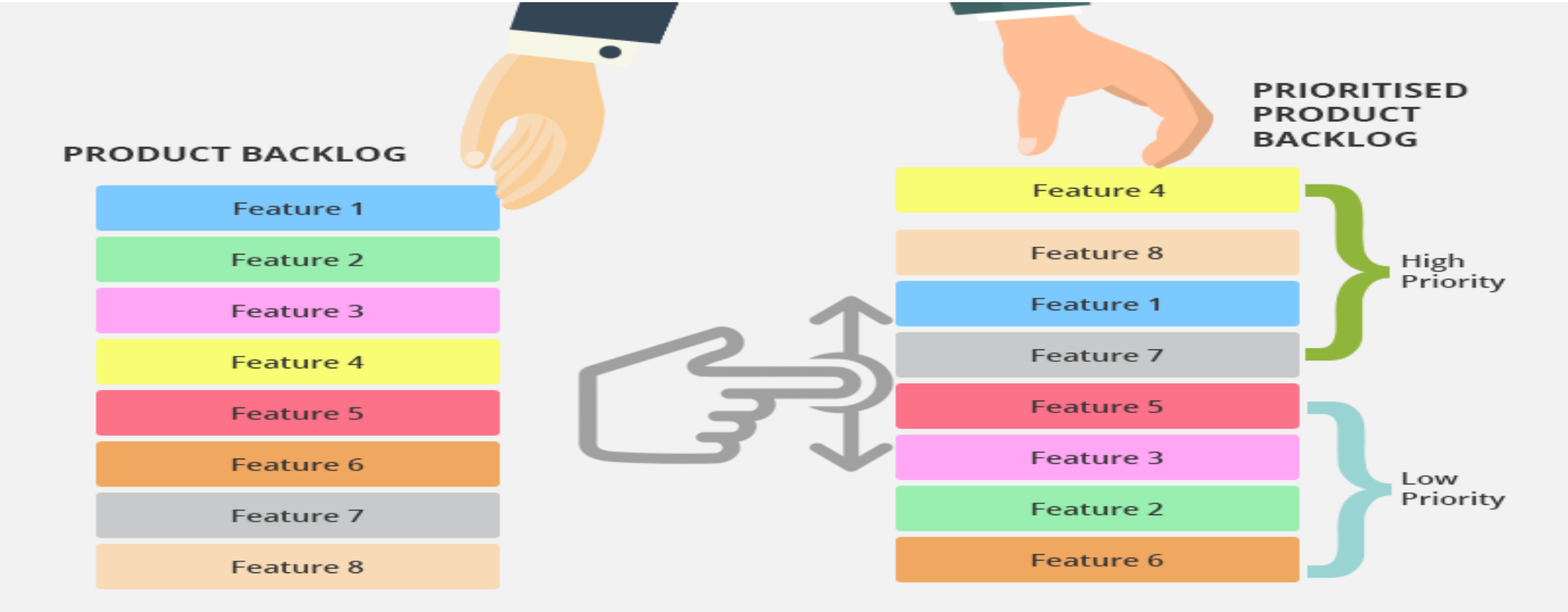
1.Product Backlog: Example



		Product backlog		New story
	253	List tasks with no parent stories under a quasi-story in bac···	New	
0	138	Indicate tracker type (via issue id's bgcolor)	New	2
0	145	Download pretty-fied roadmap report	New	6
0	55	Show/hide closed items or subtasks	New	2
	40	Show or hide the issue #	New	1
2	135	Show/hide the main backlog	New	0
4	21	Show warning when server connection is lost	New	4
	223	Support versions from parent projects	Feedback	0
	213	Hint text on "talk" tab doesn't display in Safari	New	0
	214	"Talk" button often fails	New	0
	137	Append an email to the item's discussion section	New	6
	224	Spreadsheet import for stories/tasks	New	0
	225	Offer to copy tasks when copying a story	New	0
	226	Time log widget	New	0

SCRUM – Core Components

Product Backlog Refinement:



SCRUM – Core Components

2. Sprint Backlog:

- A list of items selected from the Product Backlog to work on in the current Sprint.
- Created during Sprint Planning.
- Includes tasks and effort estimations.

Example:

- For a 2-week sprint, the team might select:
  - Add hotel filter by star rating
  - Implement login functionality
  - Fix payment validation bug

SCRUM – Core Components

2. Sprint Backlog:

User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information.	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests ...	12	12	10	6		
	Code the ...	8	8	8	4		

SCRUM – Core Components

3. Increment:

- An Increment is the sum of all completed Product Backlog Items (PBIs) during a Sprint, plus the value of the increments from all previous sprints.
- It must be:
    - Usable
    - Potentially releasable (shippable)
    - Meet the Definition of Done (DoD)
- Every Increment is a step toward the final product.

SCRUM – Example Scenario – E-Commerce App (Sprint Increments)

Sprint 1 Increment:

Feature Completed:

- User registration
- Email verification

Increment:

- A working flow where a new user can sign up and verify their email.
  - ☑ Fully tested
  - ☑ Meets DoD
  - ☑ Usable and ready to be deployed

SCRUM Ceremonies

- Scrum ceremonies (also called Scrum events) are structured meetings in the Scrum framework that facilitate transparency, inspection, adaptation, and collaboration within the team.
- Each ceremony serves a specific purpose in the Sprint cycle and is crucial for delivering value effectively.
- Scrum ceremonies are:
    1. Sprint Planning
    2. Daily Scrum (Standup)
    3. Sprint Review
    4. Sprint Retrospective
    5. Backlog Refinement (optional but recommended)

SCRUM Ceremonies

Ceremony	Purpose	When It Happens	Time-boxed Duration
1. Sprint Planning	Plan what will be delivered and how	Beginning of the Sprint	2–4 hours (for 2-week sprint)
2. Daily Scrum (Standup)	Sync up on progress and identify blockers	Every day of the Sprint	15 minutes
3. Sprint Review	Demonstrate the completed work and get feedback	End of the Sprint	1–2 hours (for 2-week sprint)
4. Sprint Retrospective	Reflect on the Sprint and improve as a team	After Sprint Review	1–1.5 hours
5. Backlog Refinement (optional but recommended)	Update and clarify backlog items	Mid-Sprint or ongoing	5–10% of Sprint time

SCRUM Ceremonies – Sprint Planning

Purpose:

- To plan the work for the upcoming sprint. The team decides what to work on and how to achieve it.

Who Attends:

- Product Owner, Scrum Master, Development Team

Key Activities:

- Product Owner presents prioritized backlog
- Team selects user stories for the sprint
- Break stories into tasks
- Define a Sprint Goal

SCRUM Ceremonies – Sprint Planning

Output:

- Sprint Backlog (list of committed stories + tasks)
- Sprint Goal

Example:

- Sprint Goal: "Implement user login and registration"

SCRUM Ceremonies – Daily Scrum (Daily Stand-up)

Purpose:

- A short daily meeting to help the team stay aligned and make quick adjustments.

Who Attends:

- Development Team (Scrum Master & PO may attend as observers)
- Duration:15 minutes max
- Each member answers:
  - What did I do yesterday?
  - What will I do today?
  - Are there any blockers?

Example:

- "Yesterday I fixed the checkout bug. Today I'll start payment integration. Blocked by missing API documentation."

SCRUM Ceremonies – Sprint Review

Purpose:

- To demo the Increment to stakeholders and gather feedback.

Who Attends:

- Scrum Team, Stakeholders, Customers/Product Owners

Key Activities:

- Demonstrate completed user stories
- Get feedback from stakeholders
- Adjust backlog based on feedback (if needed)

Example Output:

- "Demoed new booking flow. Stakeholders requested adding a summary screen before final payment."

SCRUM Ceremonies – Sprint Retrospective

Purpose:

- To reflect on the last Sprint and identify what went well, what didn’t, and how to improve.

Who Attends:

- Scrum Master, Product Owner, Development Team

Key Activities:

- Discuss positives and negatives
- Identify process or communication gaps
- Create actionable improvement steps

Example Actions:

- “Add checklist to clarify Definition of Done”
- “Start writing test cases during sprint planning”

Retrospective

Set the Stage

Check-in activities to engage the team

Gather and Share Data

- ✓ Team Performance metrics

Generate Insights

- ✓ What went well?
- ✓ What didn't go well?
- ✓ What to do more ?
- ✓ Root cause analysis
- ✓ Any Suggestions ?

Make Decisions

Agree on a few improvements or changes to try in the subsequent iteration

Close

- ✓ New information
- ✓ Appreciation
- ✓ Thanks

SCRUM Ceremonies – Backlog Refinement (Grooming)

Purpose:

- To keep the Product Backlog updated, clear, and ready for upcoming Sprints.

Who Attends:

- Product Owner (main driver), Development Team, Scrum Master (optional)

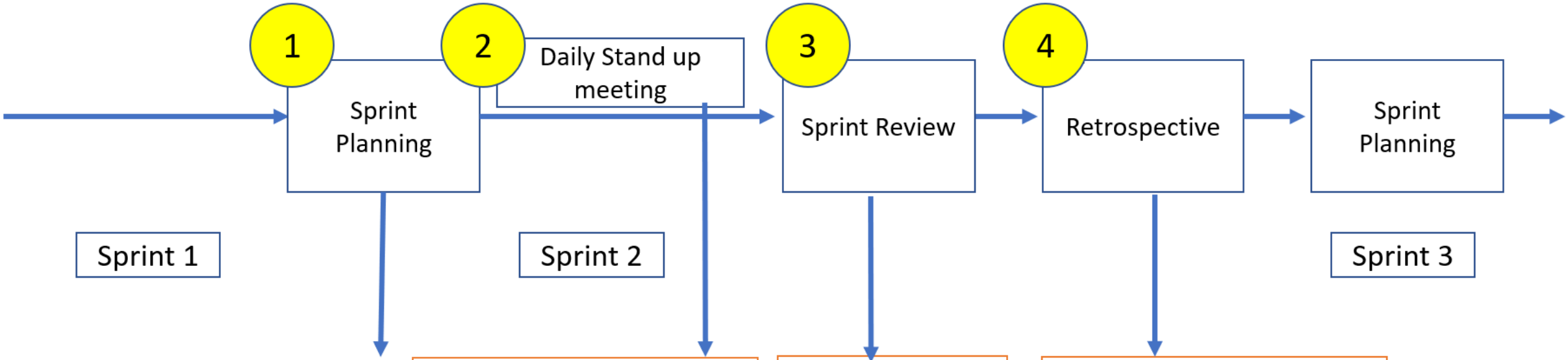
Key Activities:

- Break large stories (Epics) into smaller ones
- Add acceptance criteria
- Estimate effort (e.g., using story points)
- Prioritize items

Best Practice:

- Spend 5–10% of Sprint time on grooming.

SCRUM EVENTS



- 4 hours for selecting user stories
- 4 hours for estimating duration and effort of tasks
- Team agrees on iteration goal
- Team does not waste too much time on task assignments. They unfold as iteration progresses.

Will last not more than 15 minutes. Each team member answers :

1) What did I do yesterday
2) What do I plan to do today
3) What are the impediments

Team demonstrates working software

Team identifies problems or opportunities, gathers data, reflects on how to improve team performance

SCRUM - Writing User Stories and Acceptance Criteria

User Story:

- A User Story is a short, simple description of a feature told from the perspective of the user or customer.

Format (INVEST model):

As a [user], I want [feature], so that [benefit]

Example User Stories:

- As a user, I want to filter hotel results by price range, so that I can find affordable options.
- As an admin, I want to add new hotels, so that they are visible to customers.

SCRUM - Writing User Stories and Acceptance Criteria

Acceptance Criteria:

- Acceptance Criteria are the conditions that a user story must meet to be considered complete and accepted.They define the “done” for a story.

Example (For Hotel Search Story):

User Story:

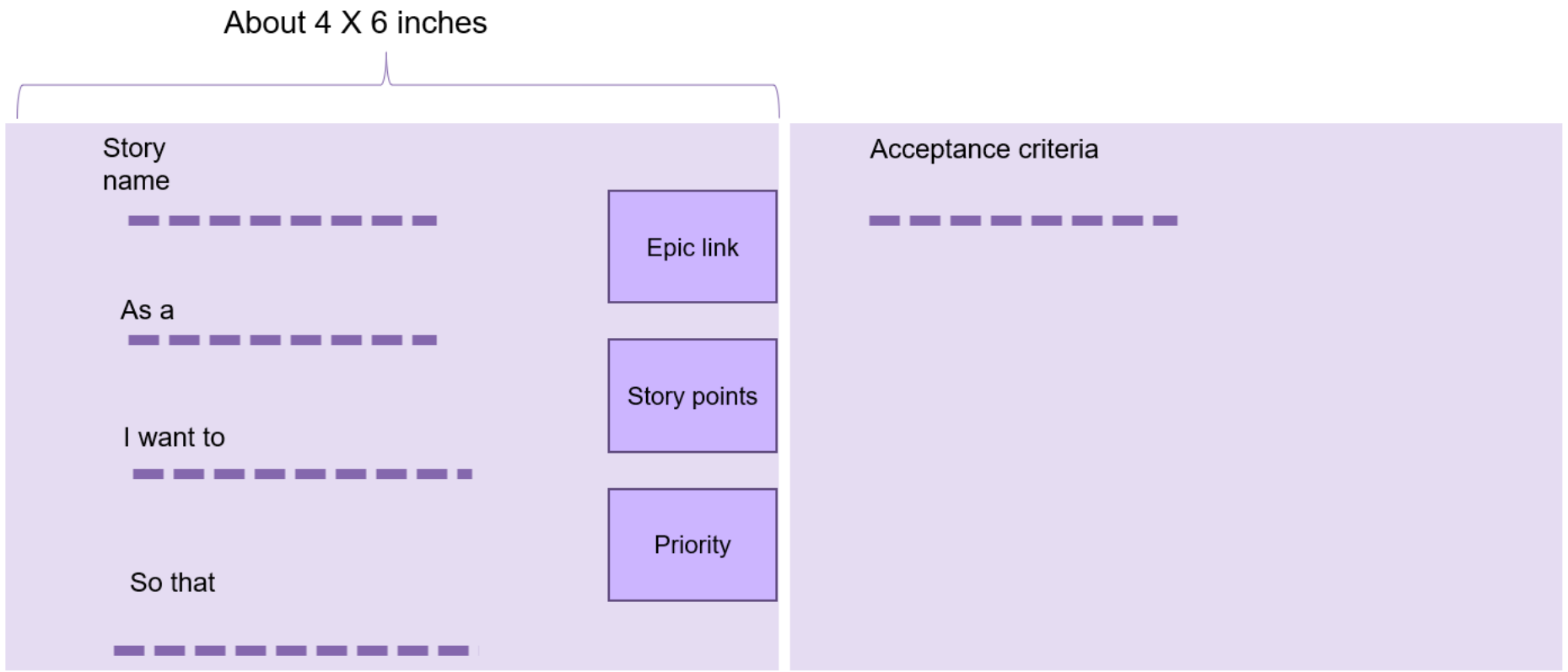
As a user, I want to search for hotels in a city so that I can book a place to stay.

Acceptance Criteria:

- ☑ User can enter city and check-in/check-out dates.
- ☑ Results show hotels matching the entered city.
- ☑ Hotels show name, price, and rating.
- ☑ If no hotels are found, show "No results found".
- ☑ Results update within 3 seconds.

User story card

3Cs – Card, Conversation and confirmation.



Estimation Techniques – Agile projects

Story Pointing

- Agile projects avoid using absolute time estimates.
- They adopt relative sized estimating (ie. How difficult or easy it is in relation to another)
- Teams use story point estimates. Story point number tells how difficult or easy a user story is in relation to another
- Teams create benchmark stories against which all user stories are estimated in story points.
- During iteration planning story points lose relevance and estimates are made in time units.

'T' SHIRTS

FIBONACCI NUMBERS

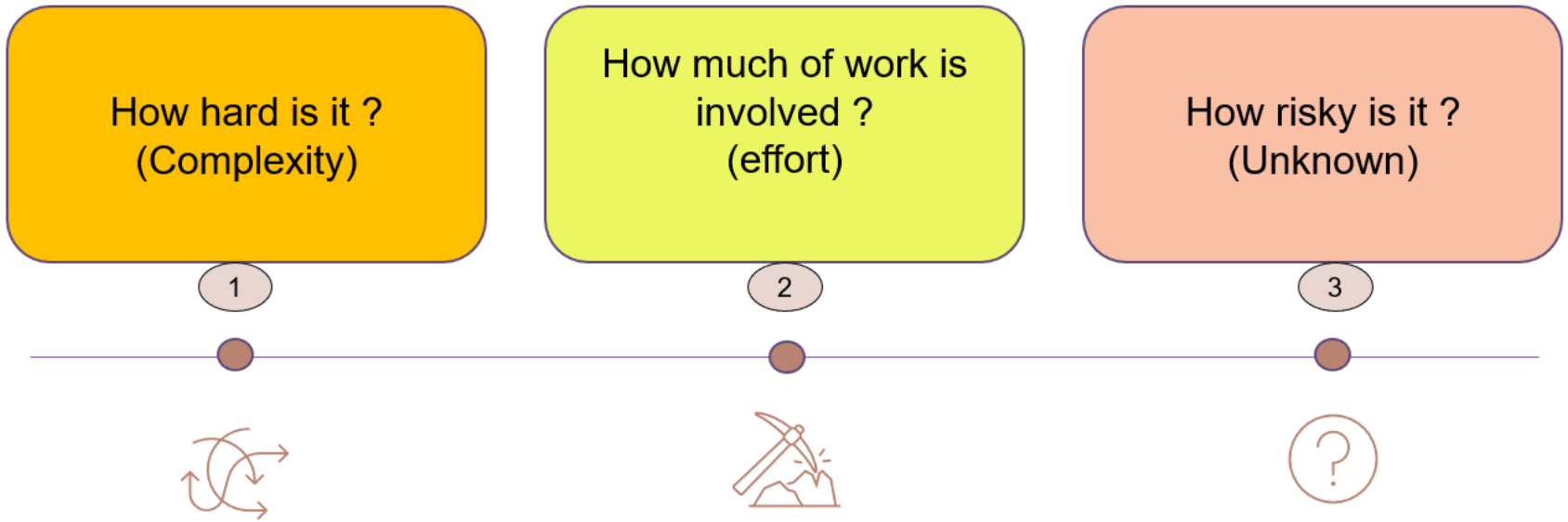
0-1-2-3-5-8-13-21

Explain user stories, story points etc.

STORY POINT

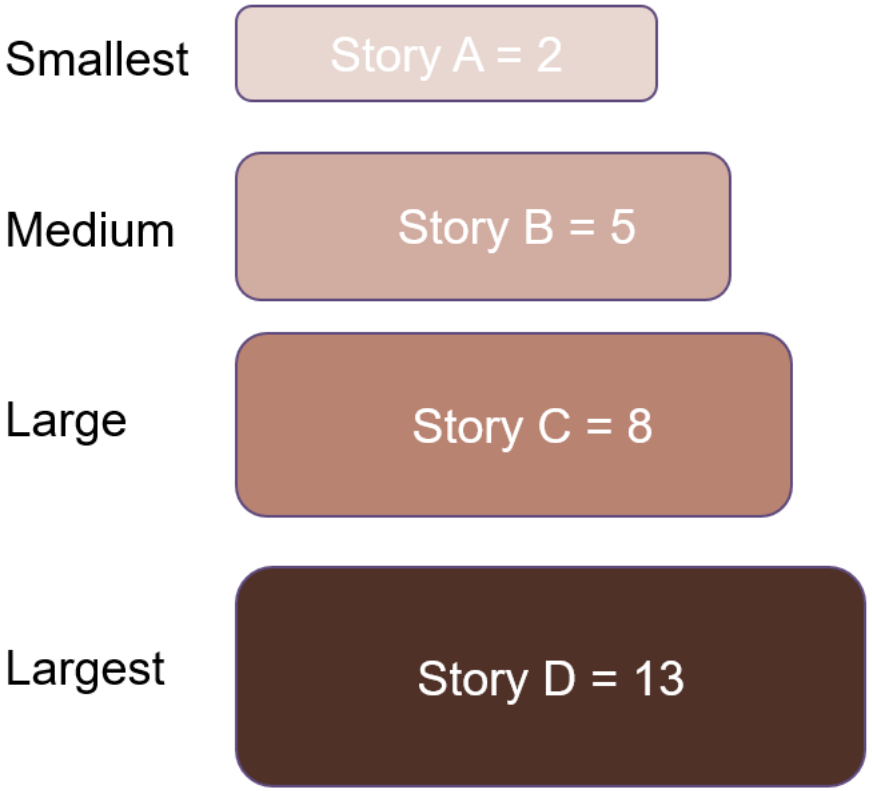
A Story point is a number that tells the difficulty level of the story, relative to other user stories.

- Size of story depends on three factors :



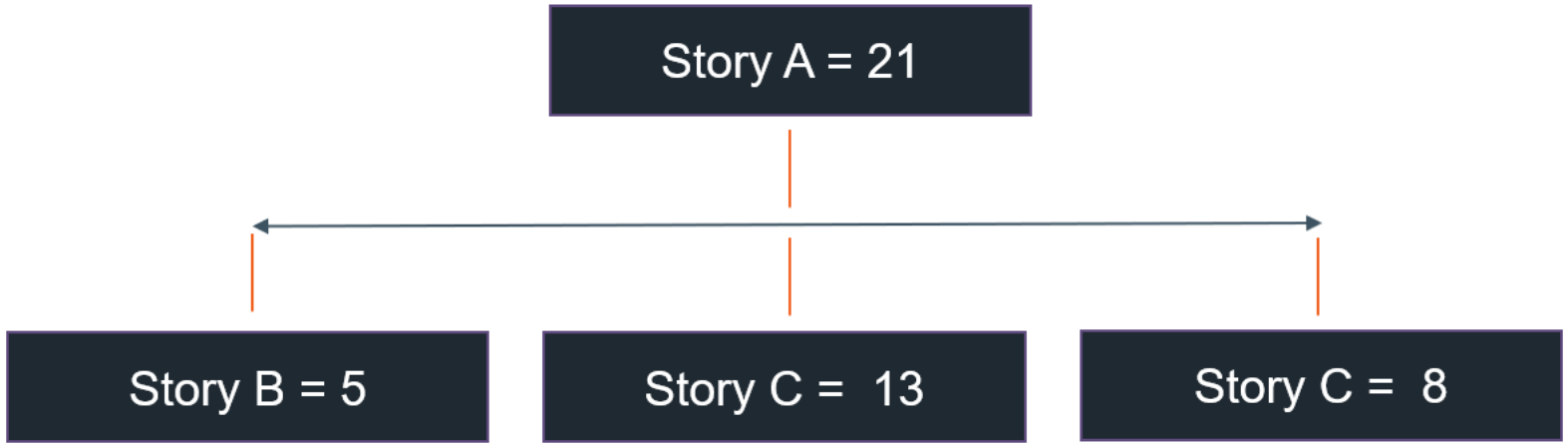
Begin with establishing benchmarks for user stories

- Select the smallest story and estimate it to be 2 points
- Select a medium sized story and assign it 5 points
- Select a large sized story and assign it 8 points
- Select the largest sized story and assign it 13 points



Too big user stories

- Teams must think about a threshold too big to include in an iteration story as a story with 21 points.
- These work items should be broken into smaller pieces or refined into smaller user stories.



Choosing not to over-estimate or under-estimate

A baseline user story has a size of 3 points. During estimation session the team comes across a story they feel is 3 times the baseline user story. Which option should they select.

a) 5 story points
b) 3 story points
c) 8 story points
d) 13 story points

Answer: Option C – 8 story points as this value is closest to 3 times the baseline user story. Assigning 5 story points is an under estimation, while assigning 13 story points is an over estimation.

PLANNING POKER

1. Each estimator receives a deck of cards numbered in Fibonacci series
2. PO reads user stories and answers team questions, if any.
3. Estimators estimate effort of user story and select a card representing the size
4. Scrum Master requests everyone to hold up the numbered card with their estimation at the same time.
5. If there are outliers estimators explain why their estimates are high or low
6. After discussions team members re-estimate and repeat until consensus is reached

Planning Poker example

- The PO reads the user story and 4 team members provide their estimate using planning poker cards.
- Sample user story : As an insurance agent, I want to create a quote, so that I can share with the customer.
- Ken, Mike, Esther & Mary choose 3,8,2 & 5 in Round 1.
- They choose 5,5,5 & 8 in the second round.
- At this point estimates are somewhat close.
- PO asks if they go with majority or a conservative option of 8.
- If team members desire they can again re-estimate and come to consensus.

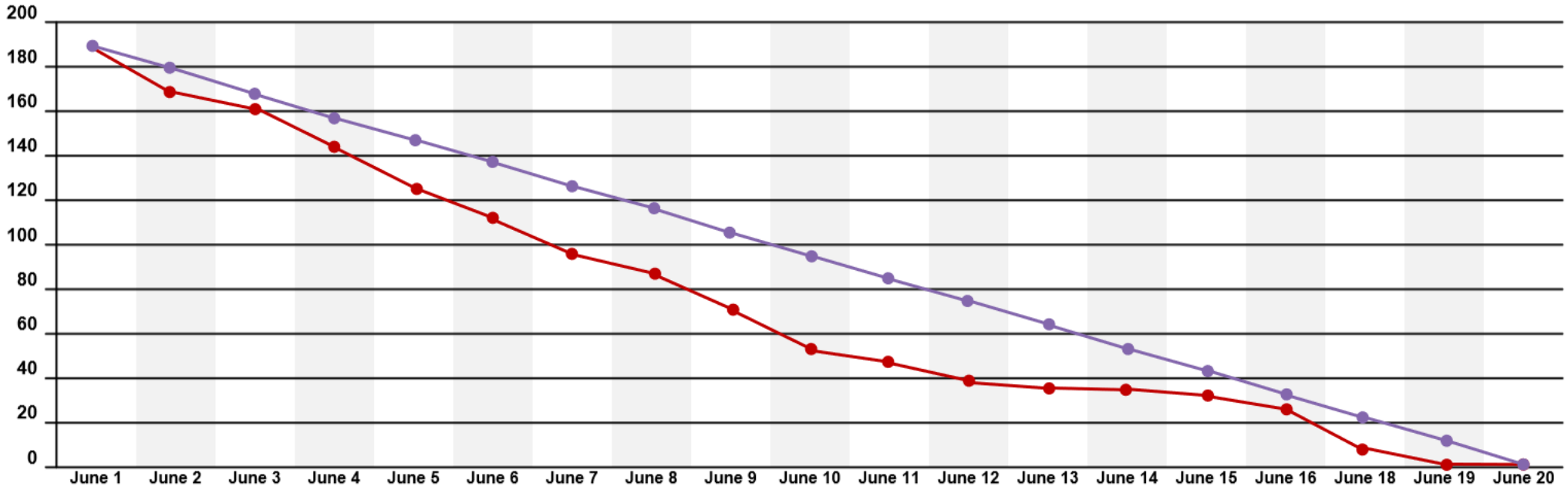
Estimator	Round 1	Round 2	Round 3
Ken	3	5	5
Mike	8	5	5
Esther	2	5	5
Mary	5	8	5

Burn Charts

Burndown (Iteration)

- Tracks the work to be completed in the iteration
- Used to analyze variance to ideal burndown of work committed to during planning

Diagonal line is ideal burndown against which daily actual remaining is charted.



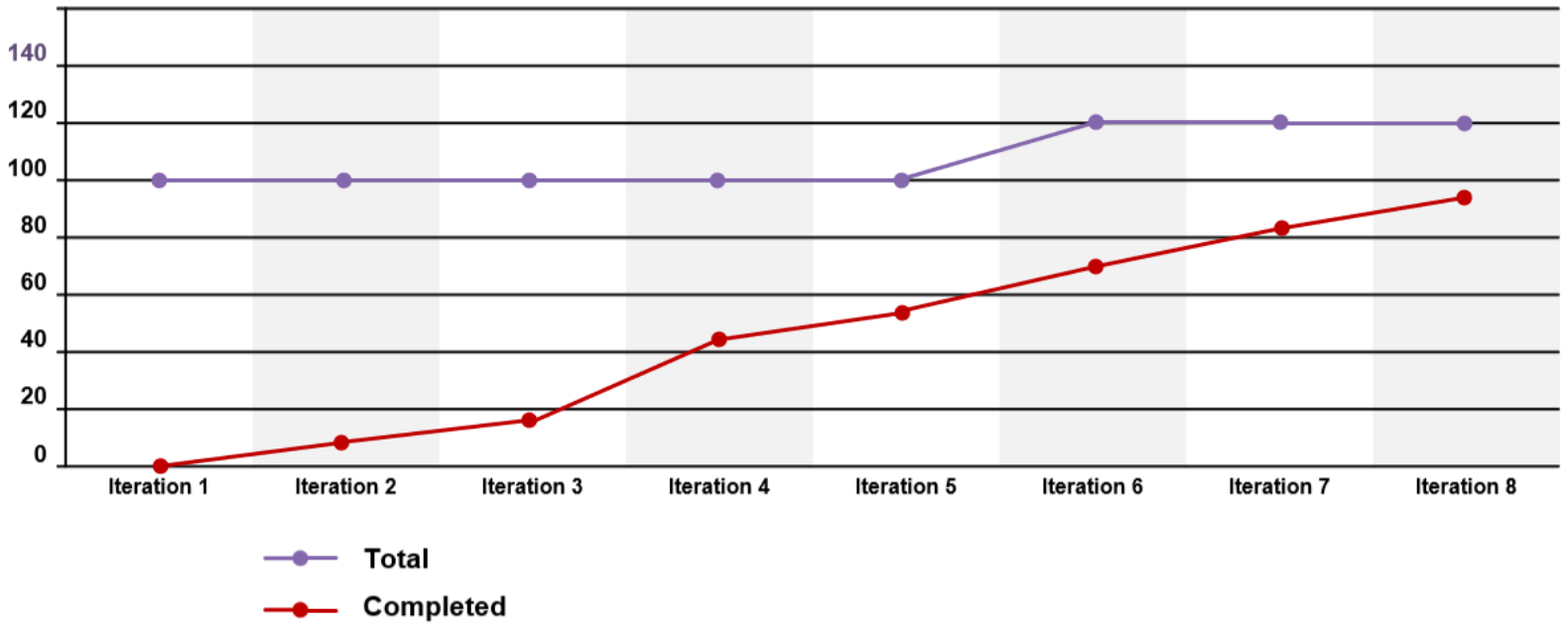
Burn Charts

https://age-of-product.com/burn-down-charts/

Burnup (Release)

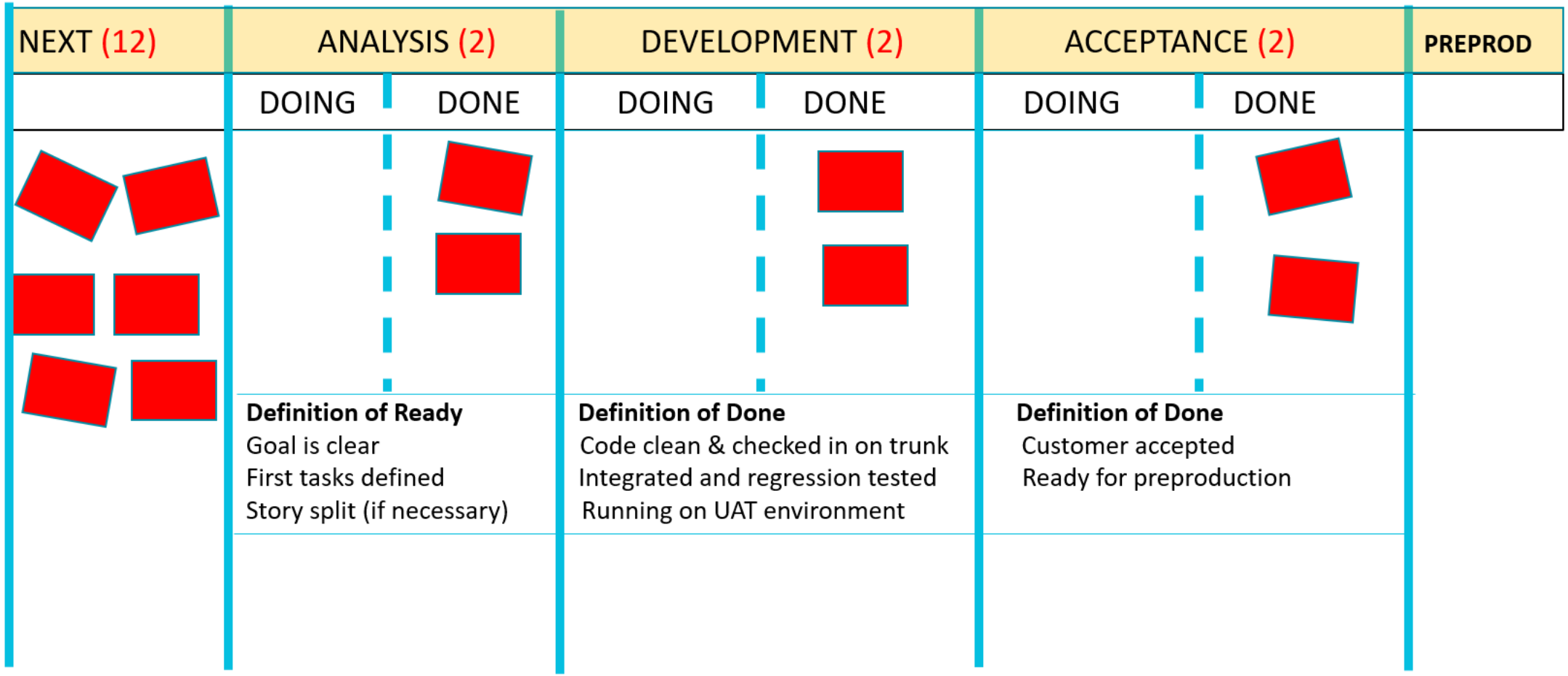
aka Feature Complete Graph

- Show accumulated progress of completed work
- Update after each iteration



TASK BOARDS

- Organize work into tasks on cards
- Display task information at every stage of the workflow
- Tailor your task board workflow stages

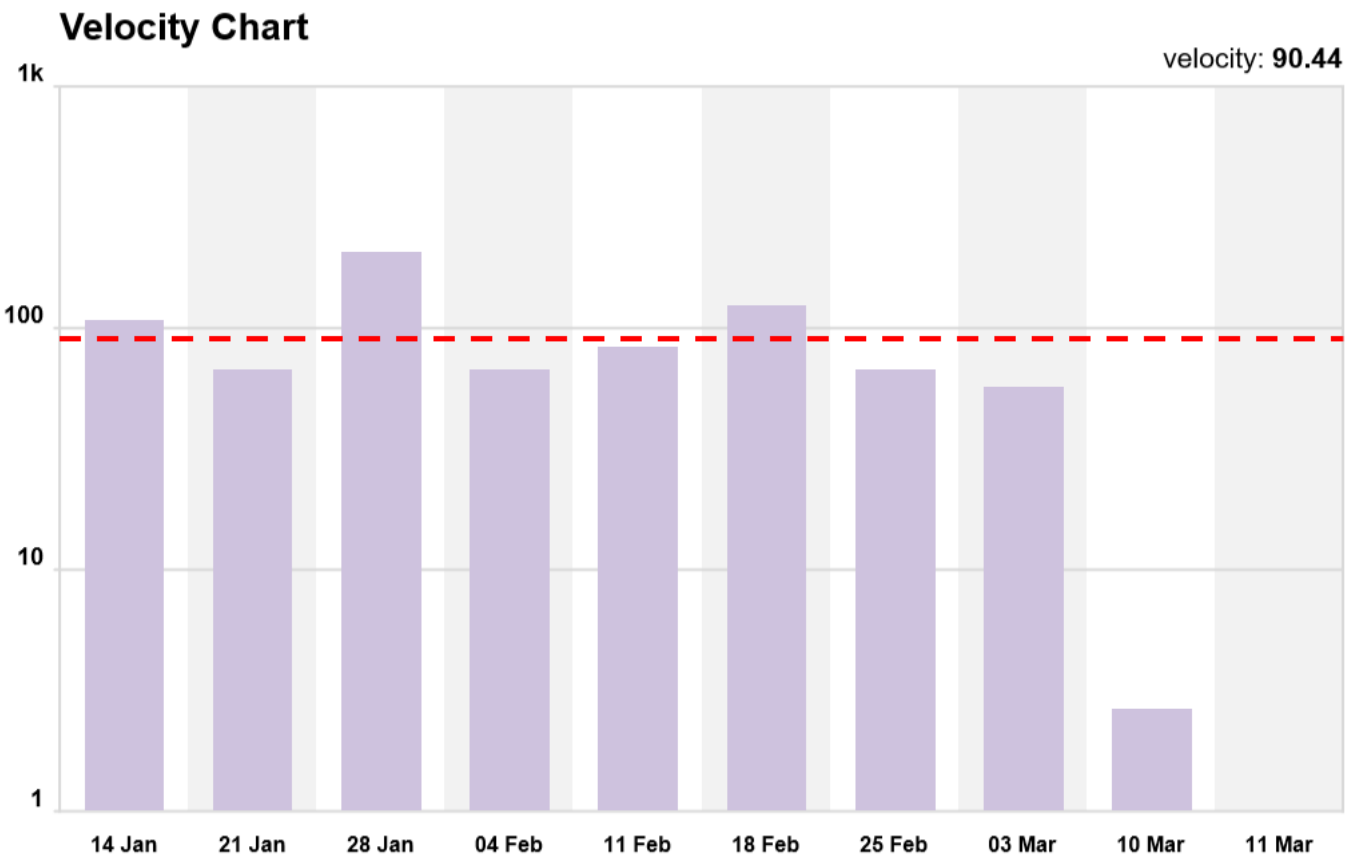


Estimate Velocity

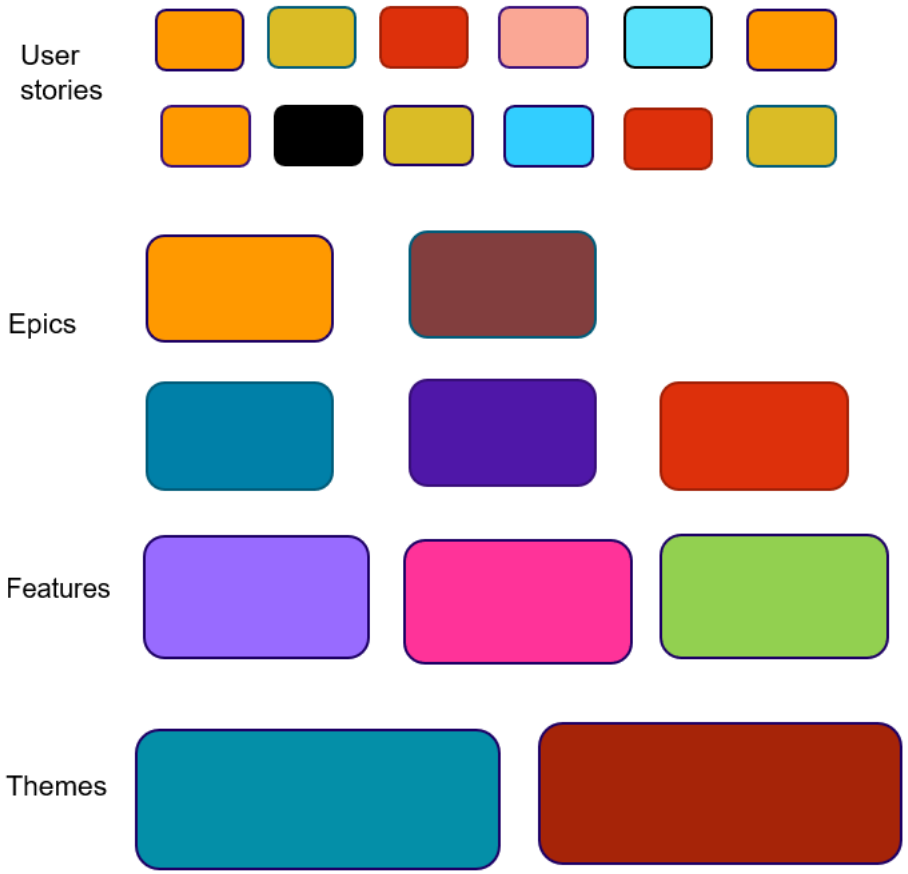
Aim for Constant Rate (with optional discussion)

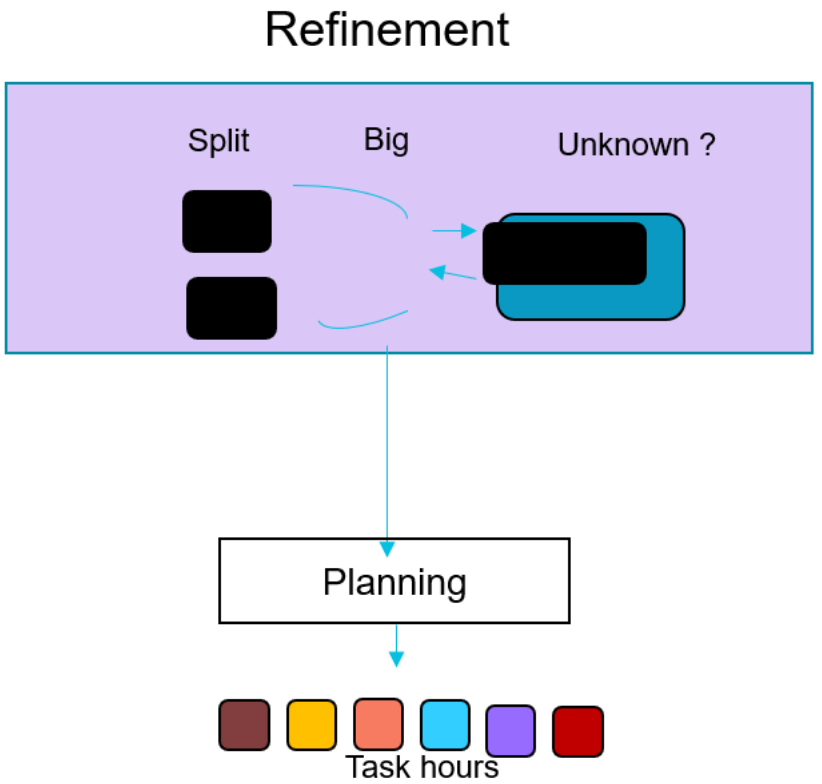
- Team's estimated rate of progress of completed work
- Calculate by estimating number of story points that can be completed during an iteration
- Then modify during subsequent iterations
- Goal: Achieve constant velocity from one iteration to the next

Velocity is a unique metric to a project; it can't be used to compare the performance of teams.



PRIORITIZE, RIGHT SIZE, SEQUENCE





Quiz

1) In an Agile project, what is the best way to handle a change in requirements during a sprint?

a) Pause the sprint and update the backlog

b) Add it to the current sprint immediately

c) Reject the change to maintain sprint focus

d) Add it to the product backlog for future sprints

Answer : Option d)

Quiz

2) A Product Owner wants to maximize value delivered. What should they do during backlog refinement?

a) Increase sprint length

b) Assign tasks to developers

c) Prioritize backlog items based on business value

d) Finalize sprint scope

Answer : Option c)

Quiz

3) A developer says, "I couldn't finish the task due to unclear acceptance criteria." What should have been done earlier?

a) Add more tasks in the sprint

b) Discuss user story in Sprint Planning

c) Increase the team size

d) Skip the story

Answer : Option b)

Quiz

4) Which role is responsible for removing impediments that block the Scrum team?

a) Product Owner

b) Scrum Master

c) Agile Coach

d) Team Lead

Answer : Option b)

Quiz

5) A user story reads: "As a user, I want to filter products by price." What's missing if it's hard to test?

a) Business context

b) Acceptance criteria

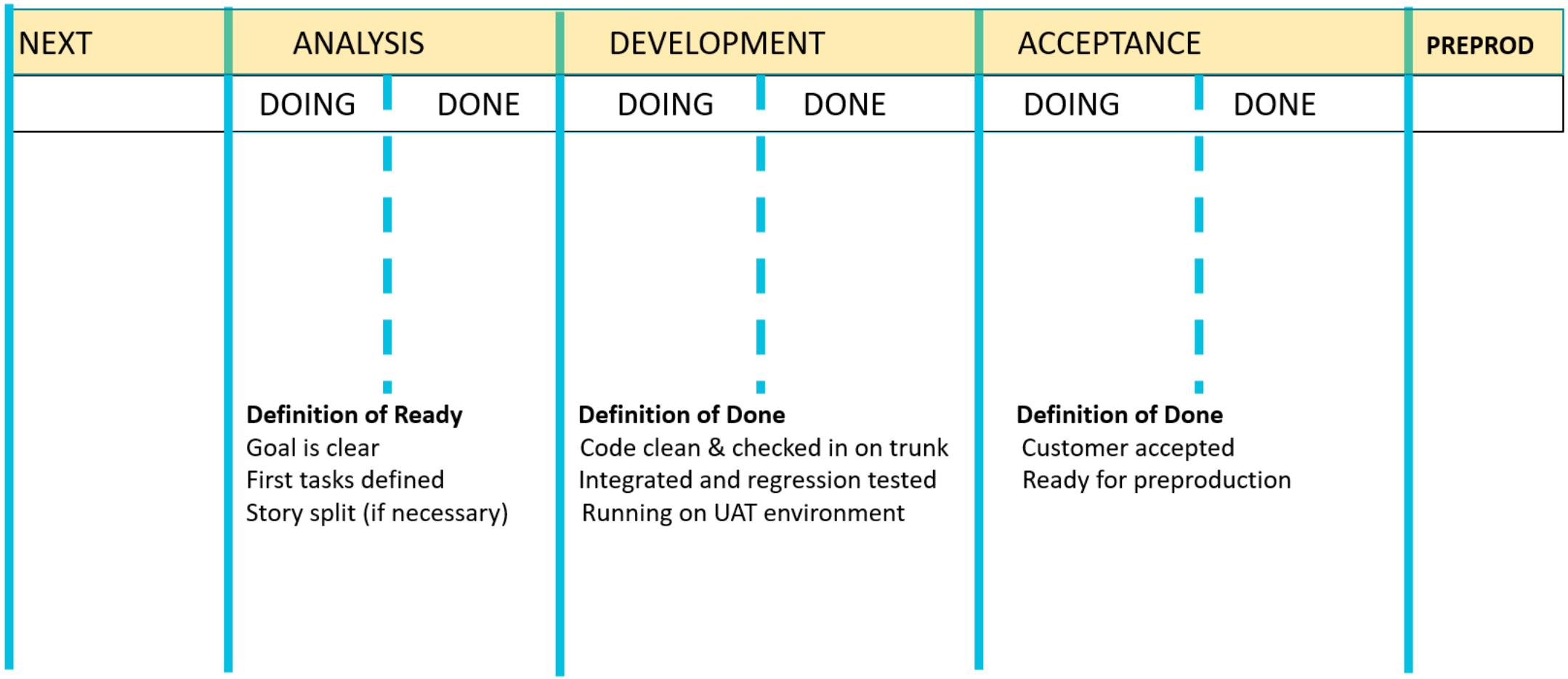
c) Story points

d) Task assignment

Answer : Option b)

Kanban

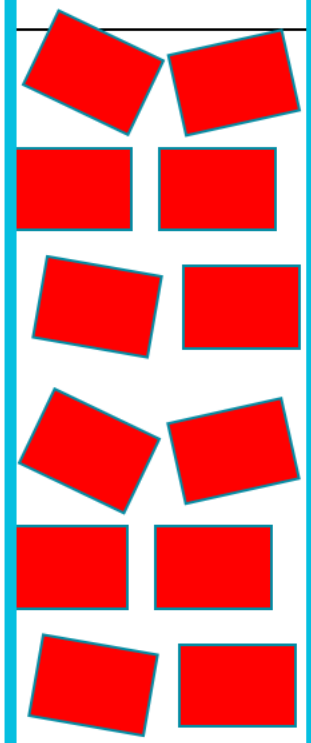
LAY DOWN POLICIES



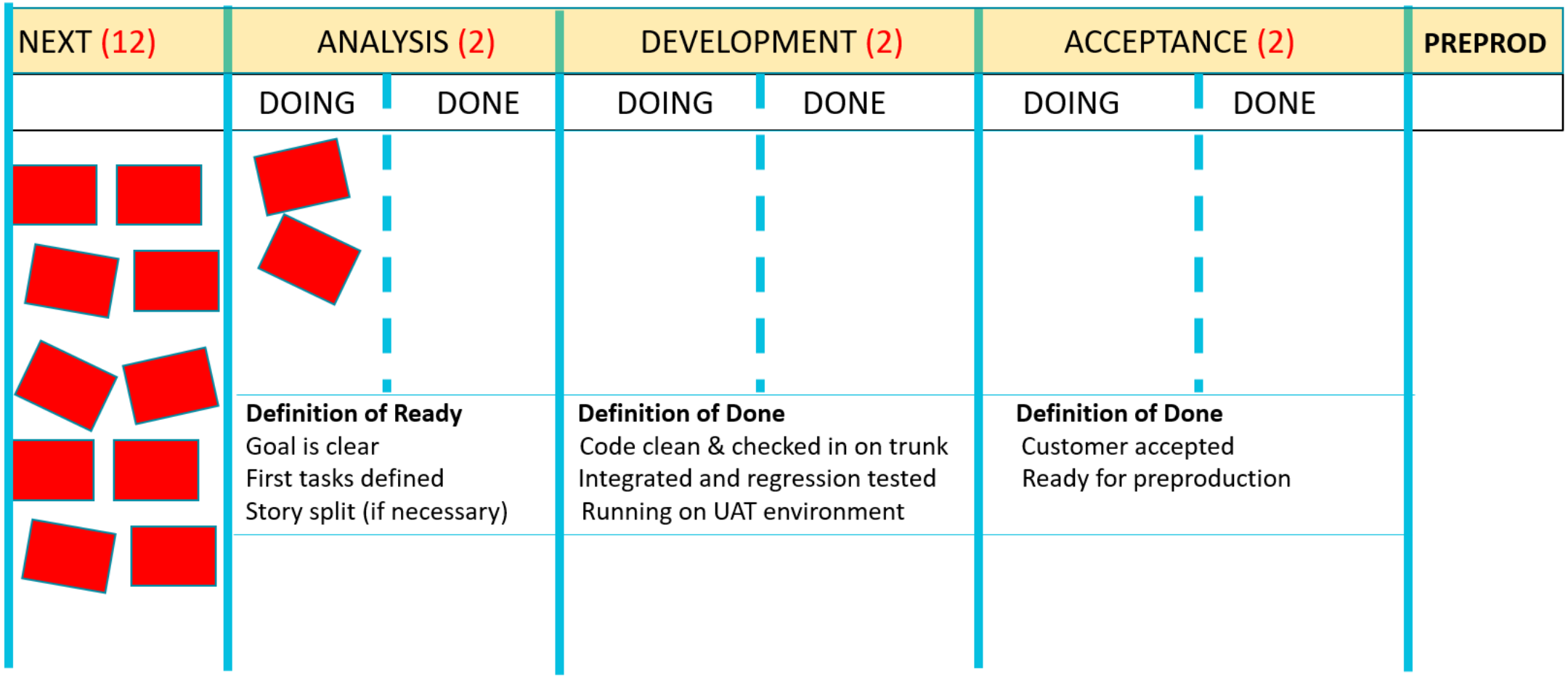
DEFINE WIP LIMITS

NEXT (12)	ANALYSIS (2)		DEVELOPMENT (2)		ACCEPTANCE (2)		PREPROD
	DOING	DONE	DOING	DONE	DOING	DONE	
	Definition of Ready Goal is clear First tasks defined Story split (if necessary)		**Definition of Done** Code clean & checked in on trunk Integrated and regression tested Running on UAT environment		**Definition of Done** Customer accepted Ready for preproduction		

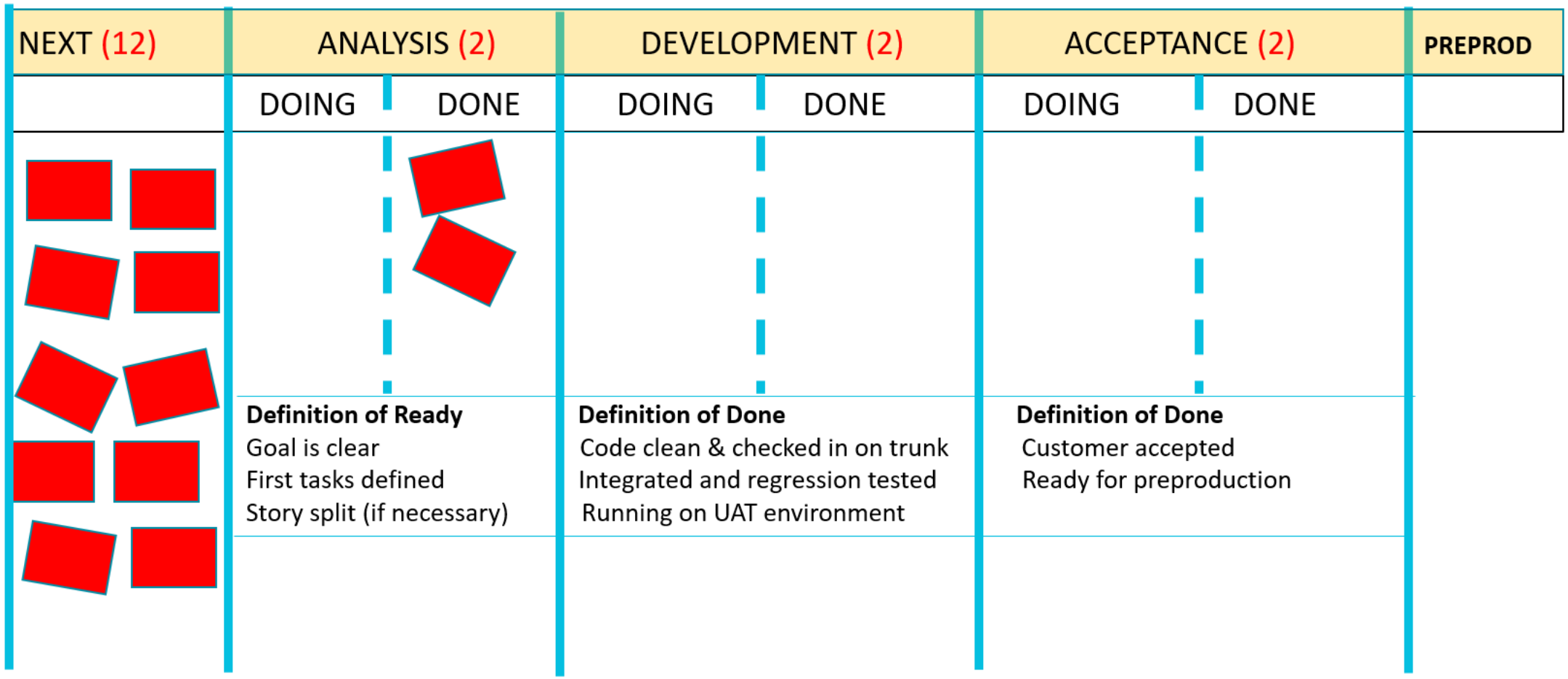
MANAGE FLOW

NEXT (12)	ANALYSIS (2)		DEVELOPMENT (2)		ACCEPTANCE (2)		PREPROD
	DOING	DONE	DOING	DONE	DOING	DONE	
	**Definition of Ready** Goal is clear First tasks defined Story split (if necessary)		**Definition of Done** Code clean & checked in on trunk Integrated and regression tested Running on UAT environment		**Definition of Done** Customer accepted Ready for preproduction		

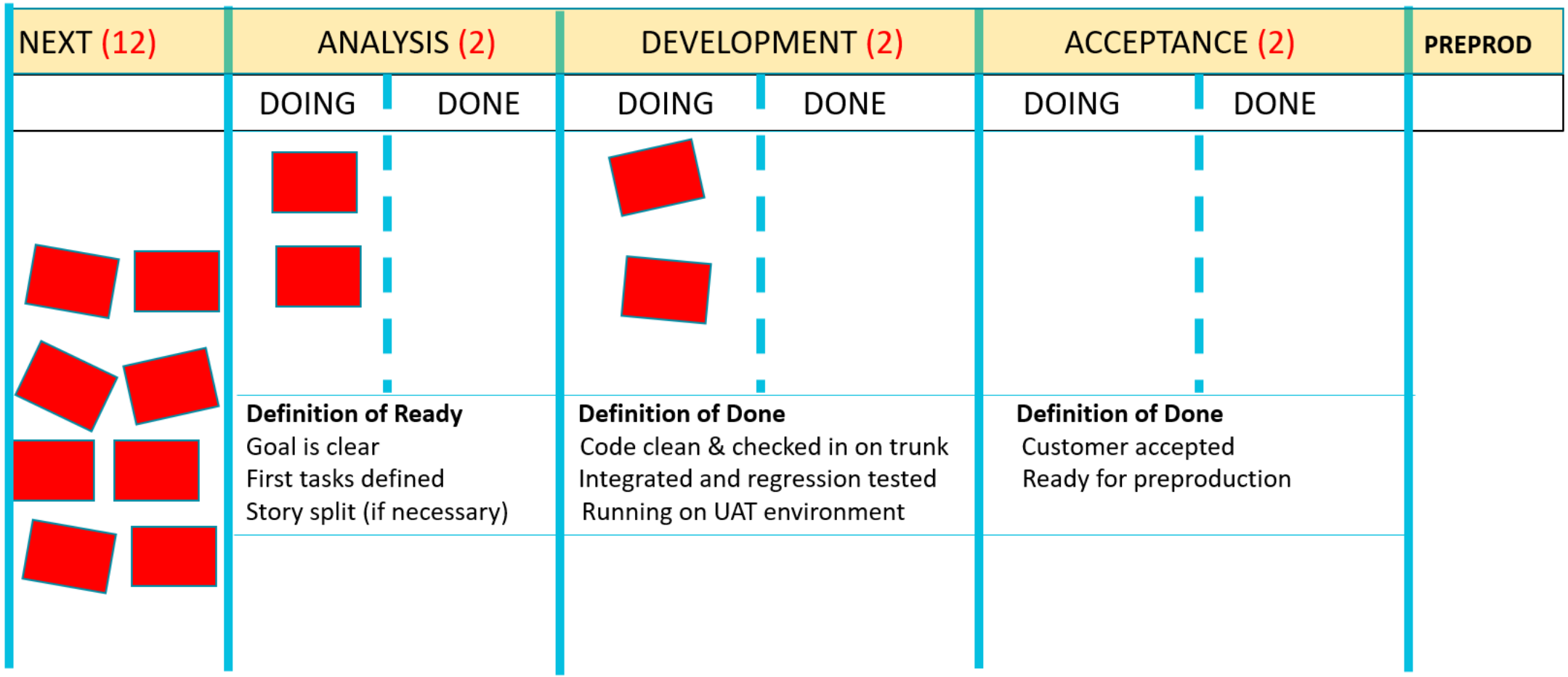
MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

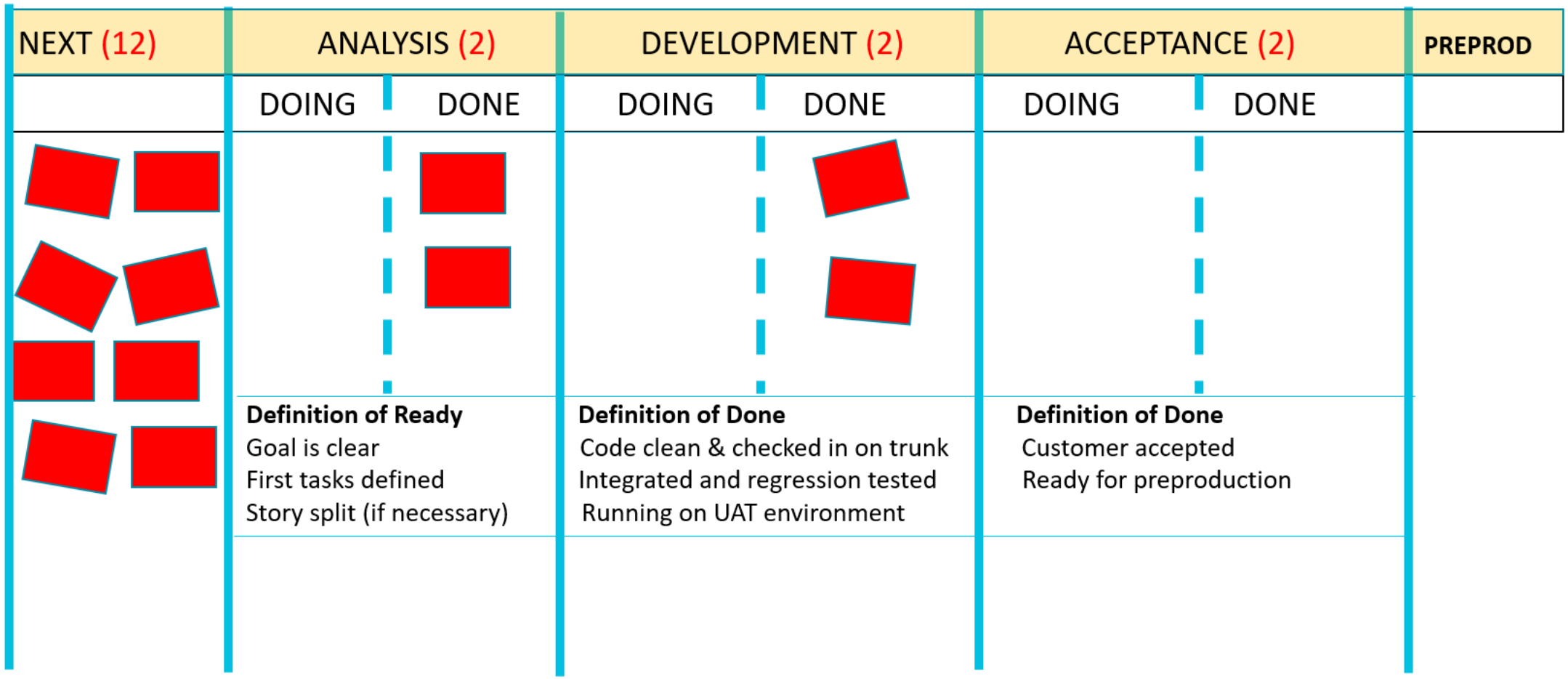
MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

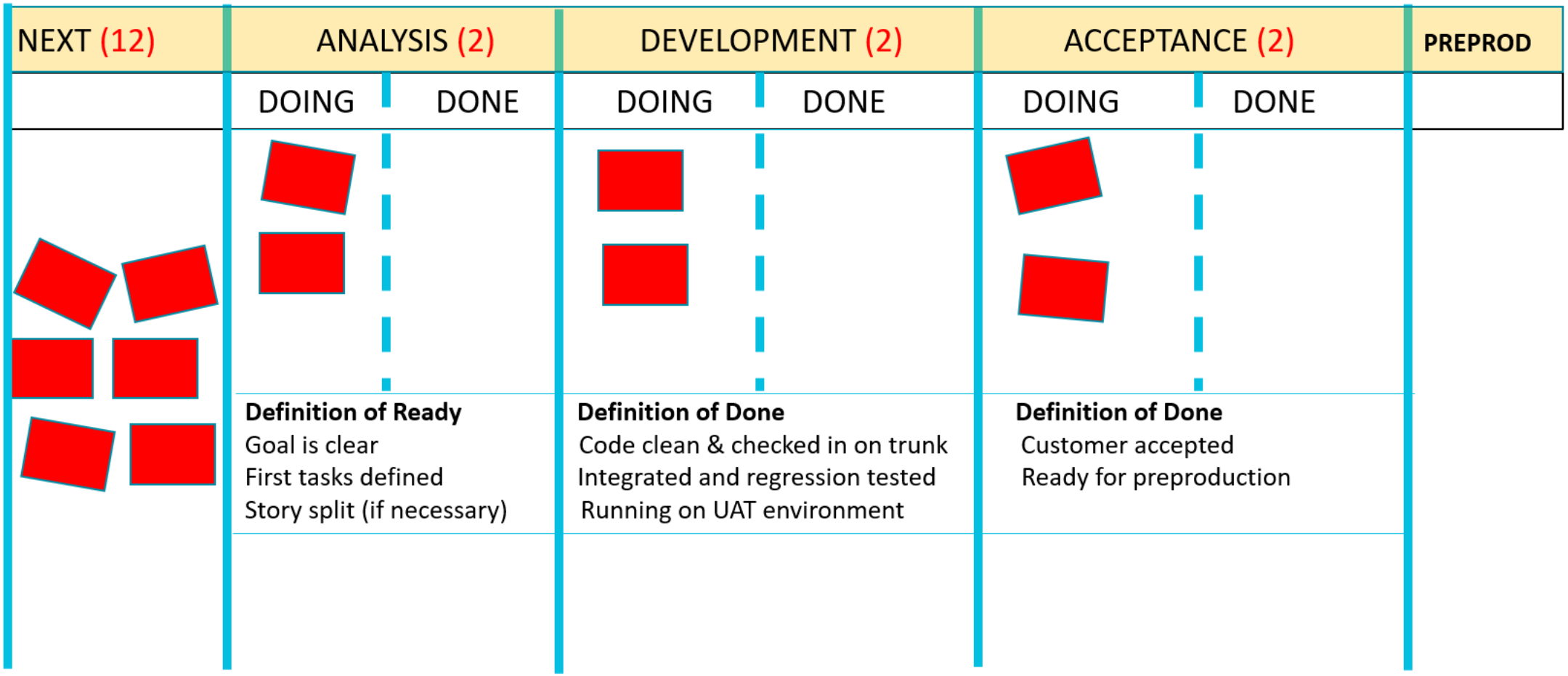
MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

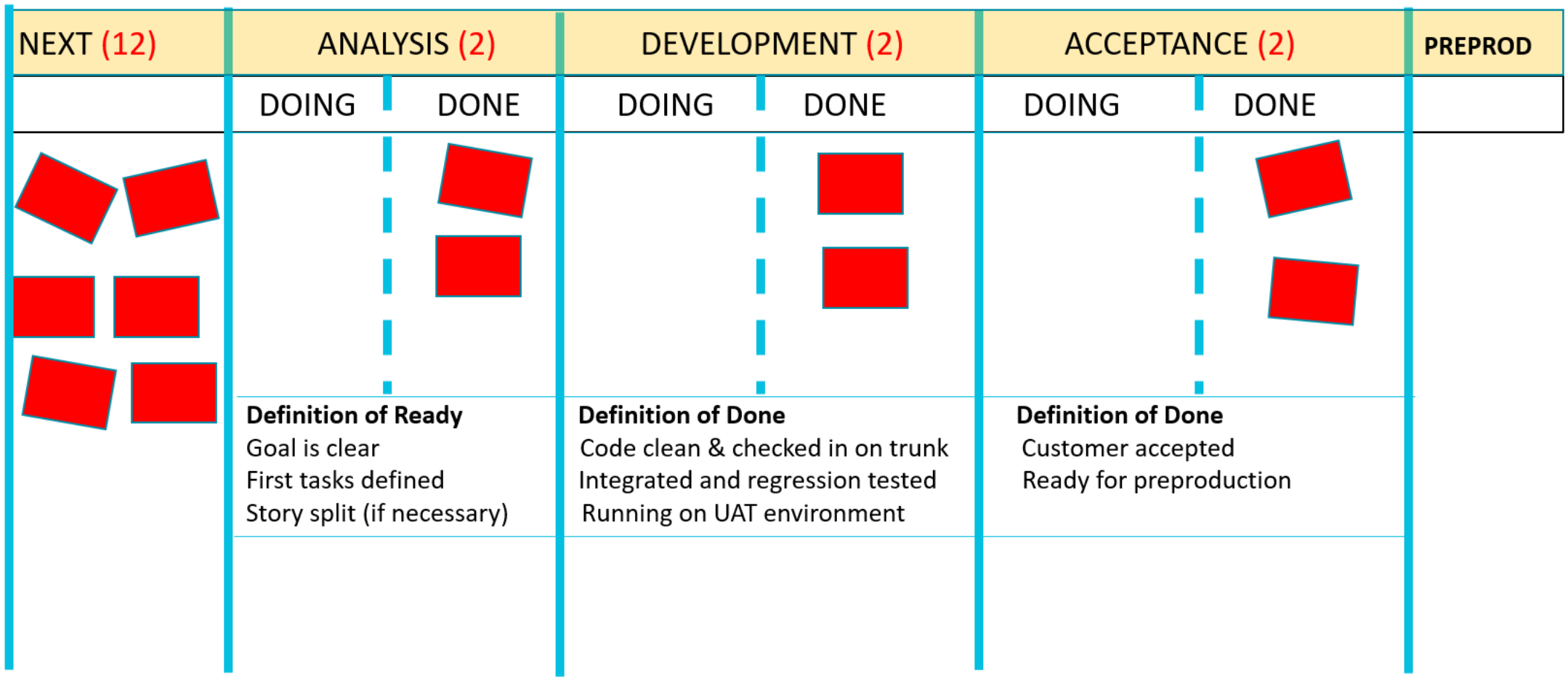
MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

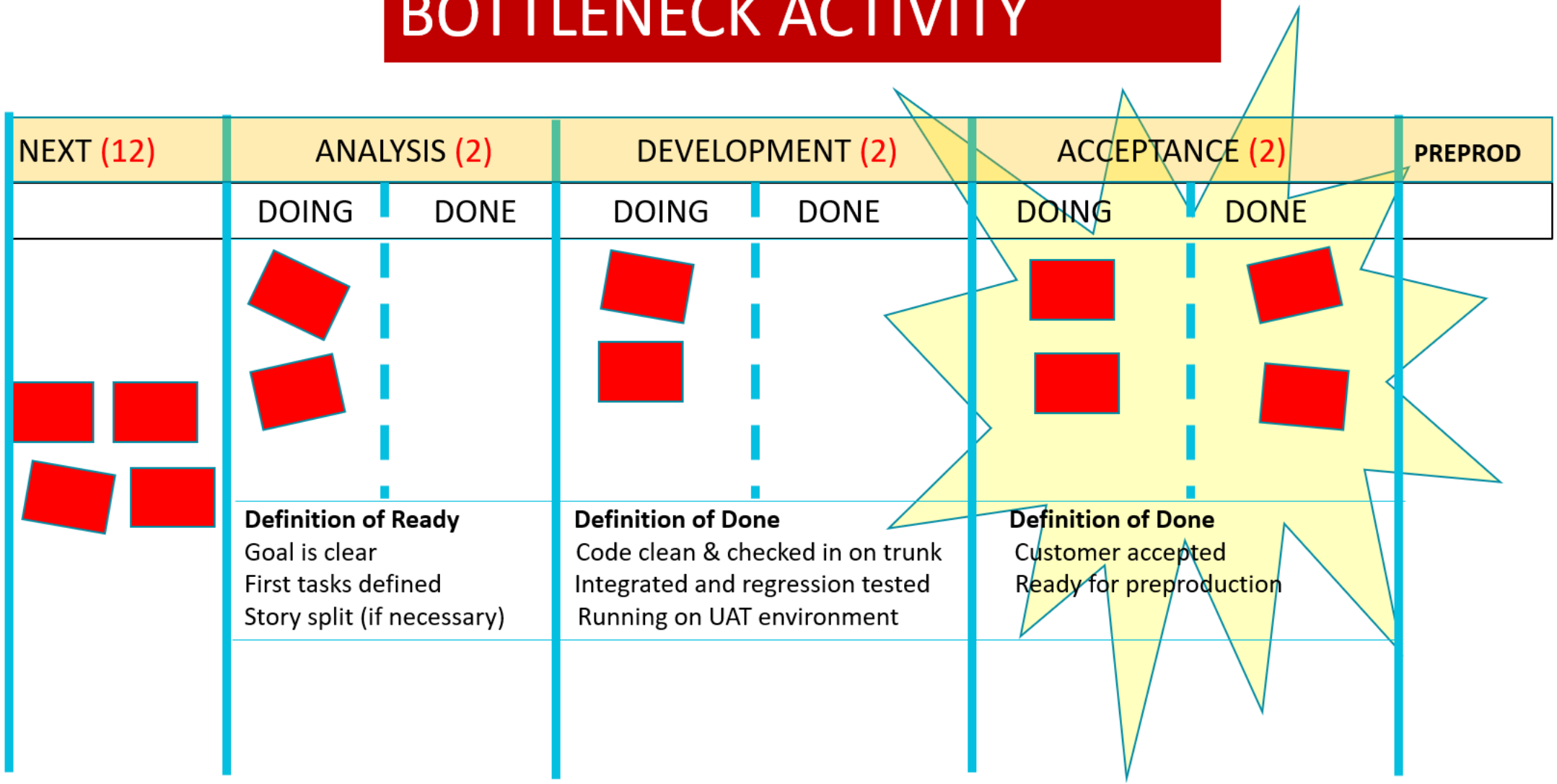
MANAGE FLOW

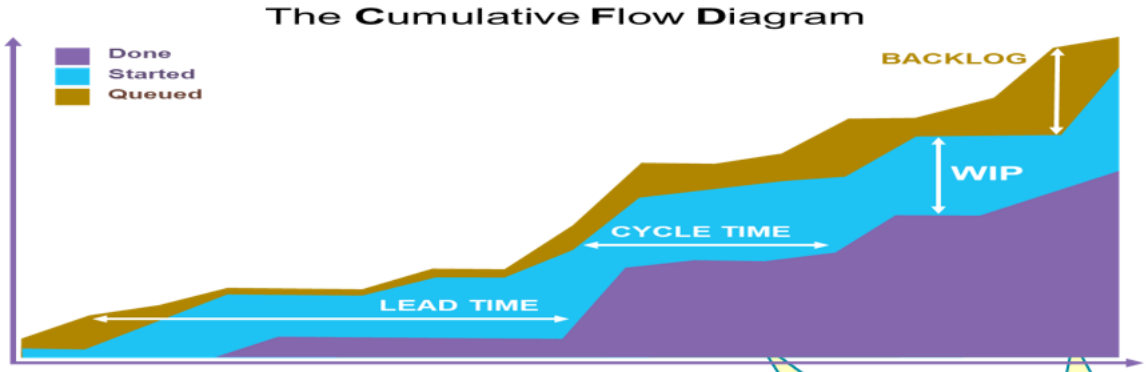

NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

BOTTLENECK ACTIVITY





- Slack downstream
- Queue upstream

NEXT (12)	ANALYSIS (2)		DEVELOPMENT (2)		ACCEPTANCE (2)		PREPROD
	DOING	DONE	DOING	DONE	DOING	DONE	

How can we help

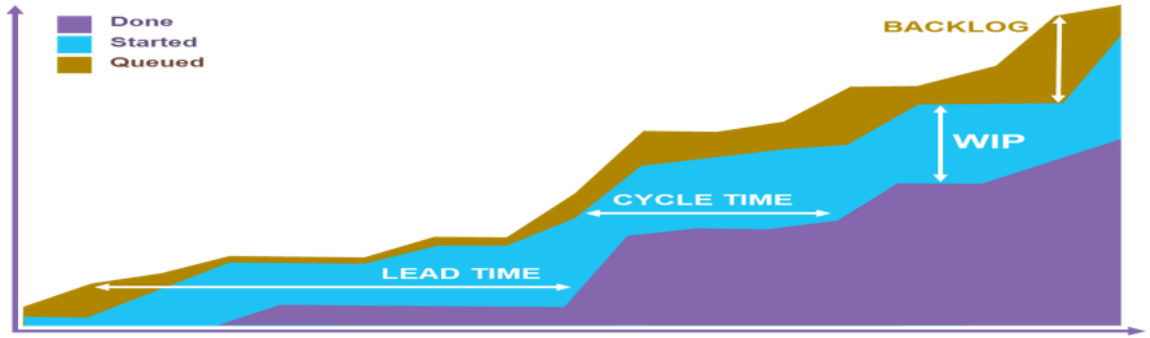
How can we help

Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)

Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment

Definition of Done
Customer accepted
Ready for preproduction

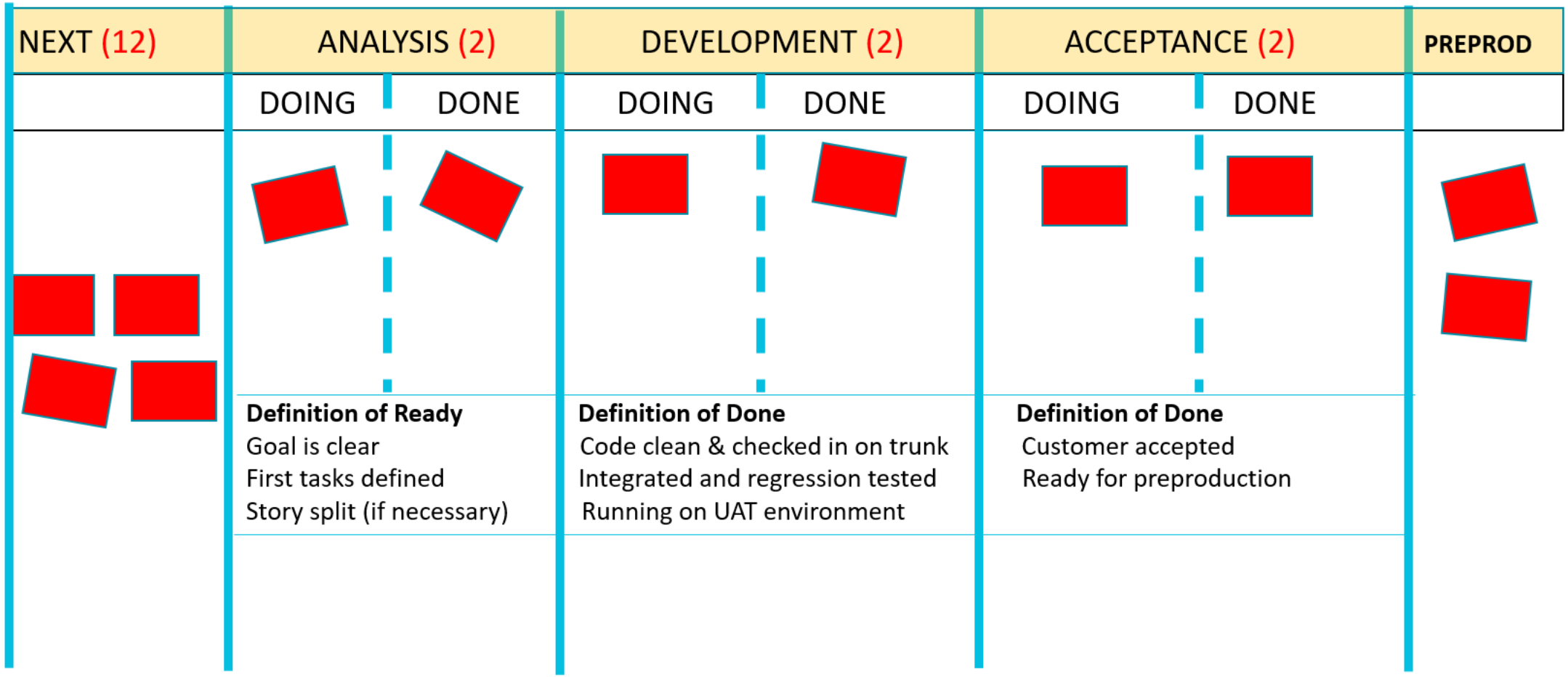
The Cumulative Flow Diagram



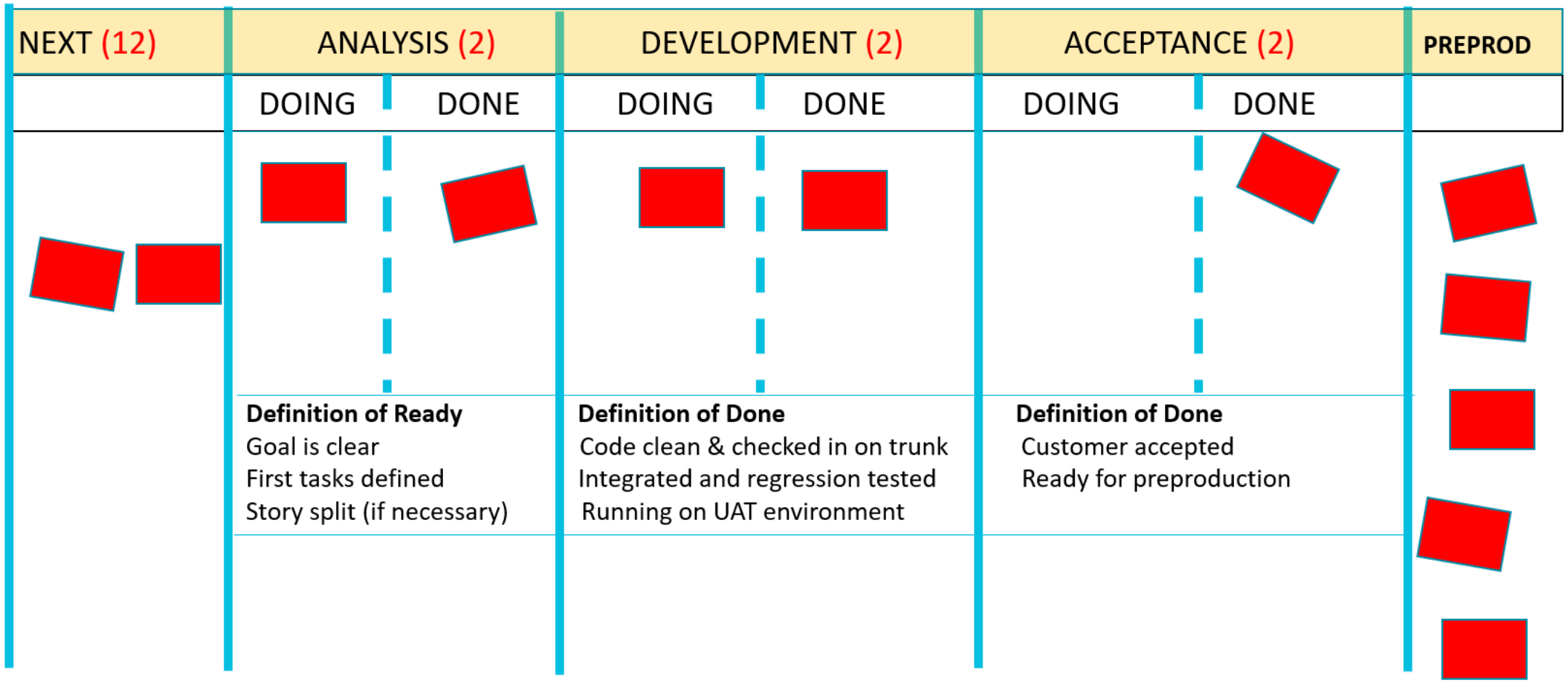
- Slack downstream
- Queue upstream

NEXT (12)	ANALYSIS (2)		DEVELOPMENT (2)		ACCEPTANCE (2)		PREPROD
	DOING	DONE	DOING	DONE	DOING	DONE	
	Definition of Ready Goal is clear First tasks defined Story split (if necessary)		**Definition of Done** Code clean & checked in on trunk Integrated and regression tested Running on UAT environment		**Definition of Done** Customer accepted Ready for preproduction		

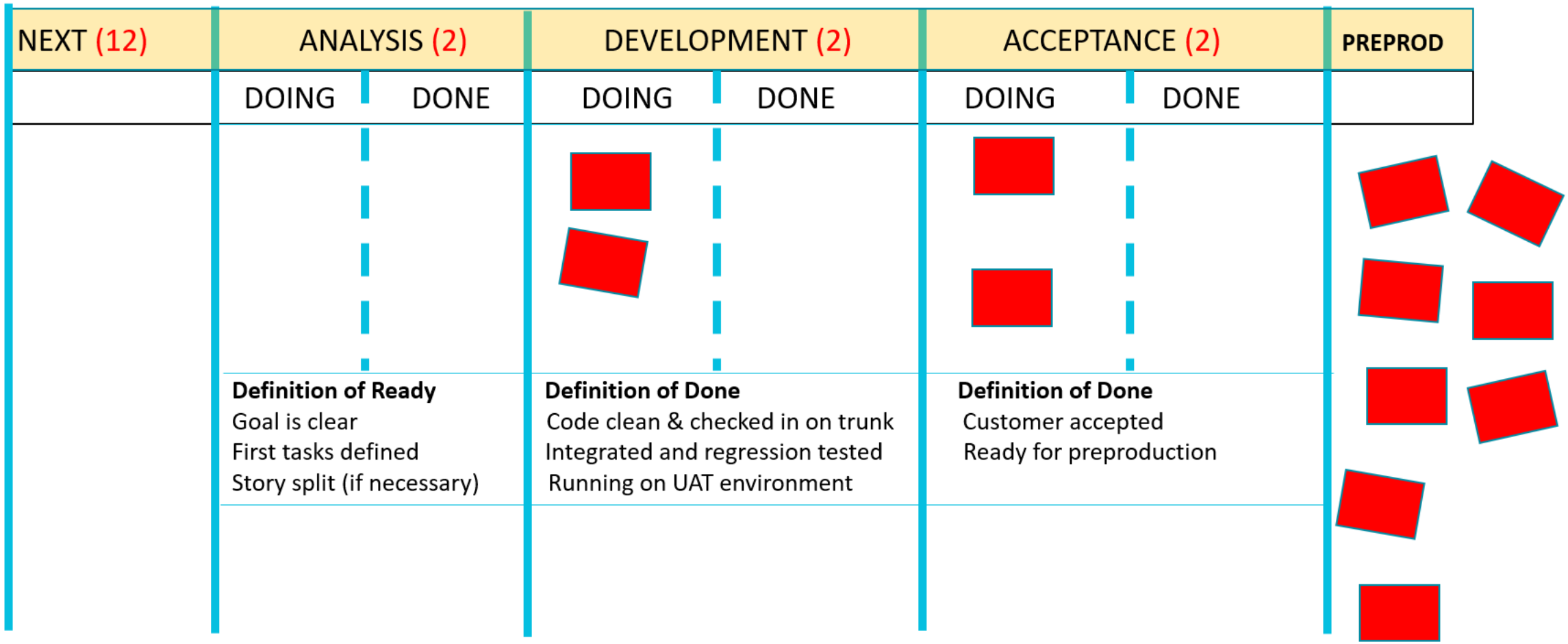
MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

MANAGE FLOW

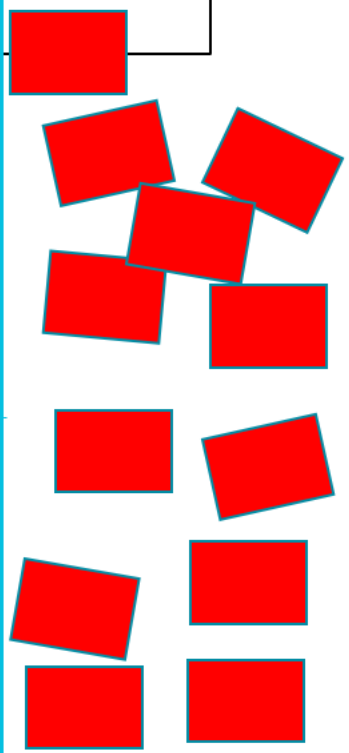

NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

MANAGE FLOW


NEXT (12)
ANALYSIS (2)
DEVELOPMENT (2)
ACCEPTANCE (2)
PREPROD
DOING
DONE
DOING
DONE
DOING
DONE
Definition of Ready
Goal is clear
First tasks defined
Story split (if necessary)
Definition of Done
Code clean & checked in on trunk
Integrated and regression tested
Running on UAT environment
Definition of Done
Customer accepted
Ready for preproduction

MANAGE FLOW

NEXT (12)	ANALYSIS (2)		DEVELOPMENT (2)		ACCEPTANCE (2)		PREPROD
	DOING	DONE	DOING	DONE	DOING	DONE	
	Definition of Ready Goal is clear First tasks defined Story split (if necessary)		**Definition of Done** Code clean & checked in on trunk Integrated and regression tested Running on UAT environment		**Definition of Done** Customer accepted Ready for preproduction		



THE CYCLE TIME WILL BE 4 DAYS

CYCLE TIME = WIP / THROUGHPUT

(This is called Little's Law)

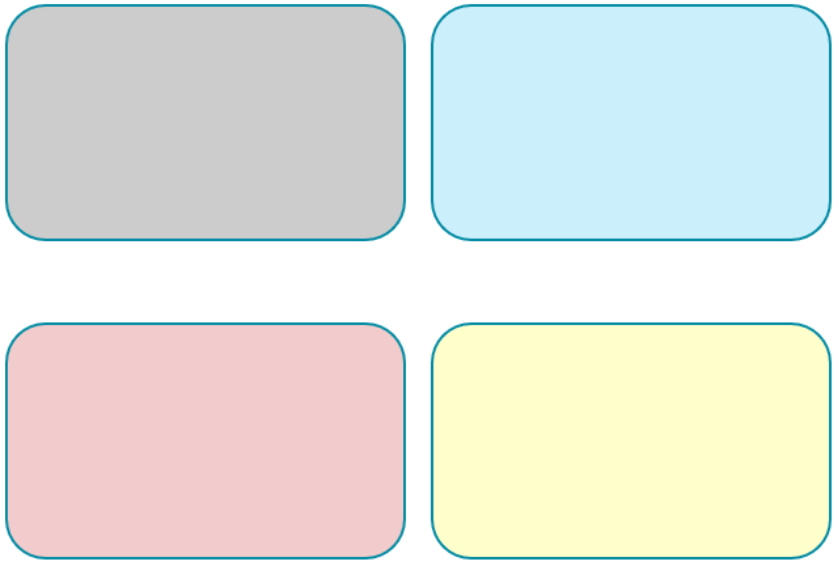
Cycle time increases as WIP increases.

Example: If WIP increases to 8, cycle time will increase from 4 to 8 days

To decrease cycle time, you must decrease WIP, increase team or production capacity

IF WIP IS 4

WIP=4



IF ONE UNIT GETS "DONE" EVERY DAY (THROUGHPUT)

Agile Metrics

- In Agile product development, metrics help teams track progress, measure performance, and improve processes.
- These metrics provide insights into how efficiently a team is working, the quality of the product, and whether customer needs are being met.
- While Agile values flexibility and collaboration, metrics play a key role in improving transparency, predicting outcomes, and optimizing workflows.

Most common and useful Agile metrics

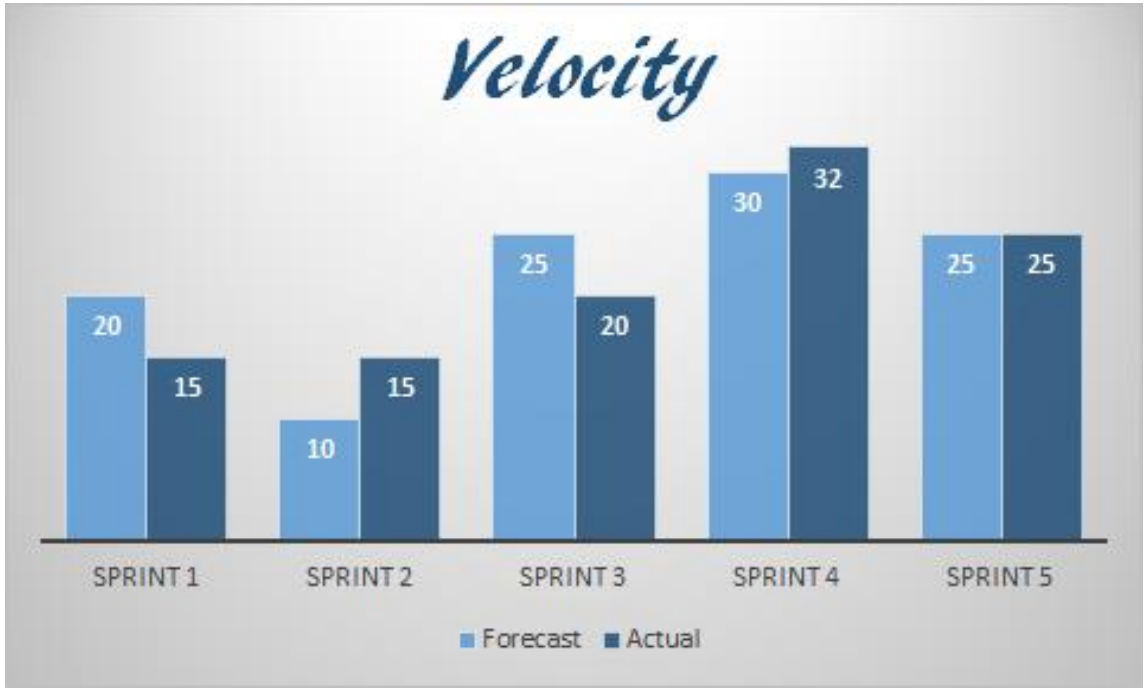
1. Velocity

- **What It Measures:** The amount of work a team can complete in a given iteration or sprint.
- **How It's Measured:** Velocity is typically measured in story points, which are units of effort assigned to user stories or tasks. For example, if a team completes 30 story points in a sprint, their velocity for that sprint is 30.
- **Why It's Important:** Velocity helps teams understand their capacity for future sprints. Over time, it allows the team to predict how much work they can take on in upcoming sprints.
- **Limitations:** It's a relative measure and should not be compared between teams, as story points can be defined differently. It also doesn't capture the quality or impact of the work.

Most common and useful Agile metrics

1. Velocity

- Velocity is an indication of the average amount of Product Backlog turned into an Increment of product during a Sprint by a Scrum Team, tracked by the Development Team for use within the Scrum Team.

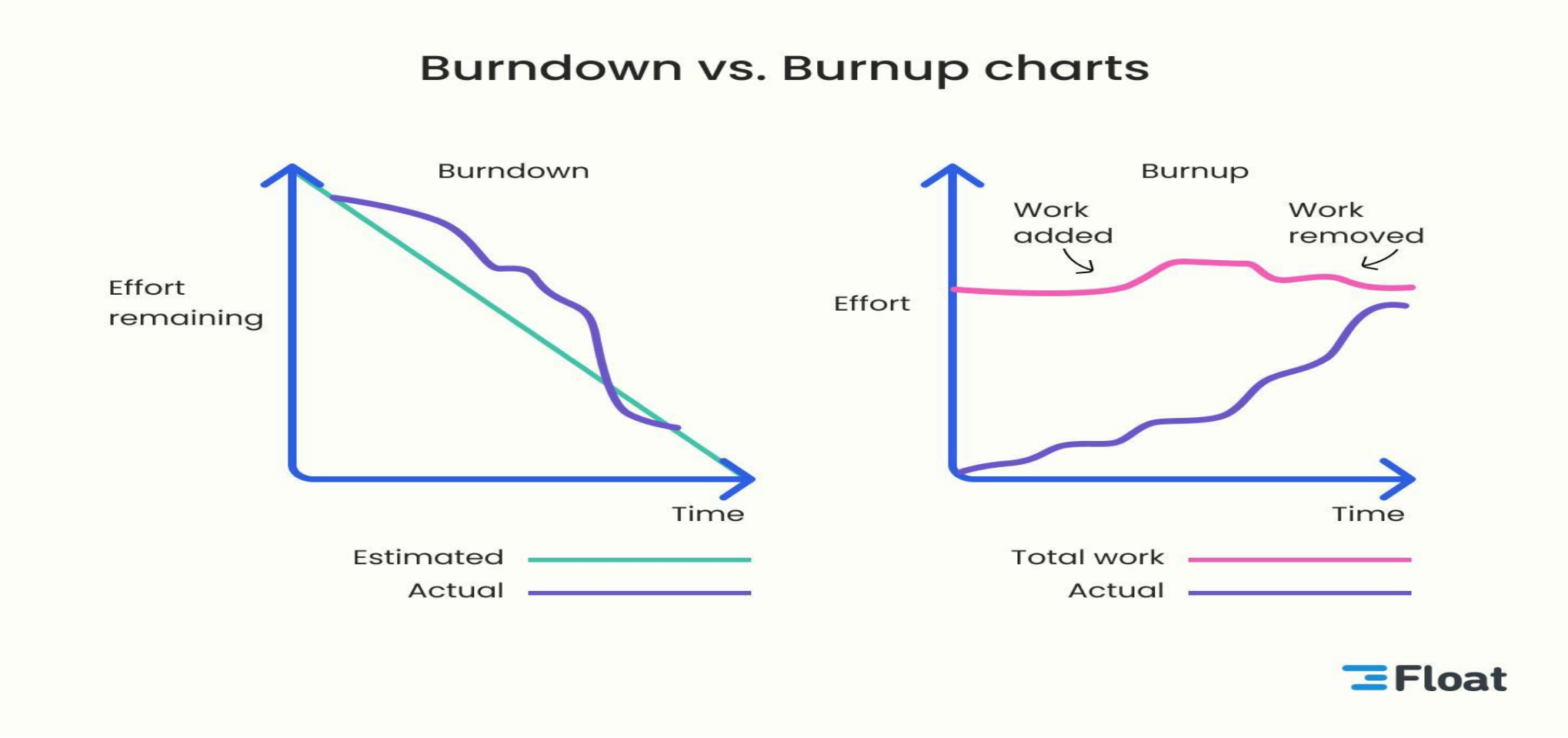


Most common and useful Agile metrics

2. Burn-down Chart

- **What It Measures:** The amount of work remaining in a sprint or project, showing how the team is progressing toward completing the sprint goal.
- **How It's Measured:** A burn-down chart tracks remaining story points (or effort) over time. Ideally, the line should decrease steadily toward zero as the team completes work.
- **Why It's Important:** It provides a quick visual of how much work is left and whether the team is on track to finish by the end of the sprint. It helps identify issues early (e.g., if the team is falling behind schedule).
- **Limitations:** A burn-down chart only shows progress; it doesn't provide insight into quality, customer satisfaction, or the cause of delays.

Most common and useful Agile metrics



Most common and useful Agile metrics

3. Burn-up Chart

- **What It Measures:** Similar to the burn-down chart, but tracks the work completed rather than work remaining.
- **How It's Measured:** The burn-up chart tracks the total amount of work (e.g., story points) over time, with a line showing the work completed and a line showing the total scope of work.
- **Why It's Important:** A burn-up chart provides more context because it shows both the progress toward completion and changes in scope (e.g., if the product backlog increases).
- **Limitations:** Like the burn-down chart, it doesn't provide information on quality or customer feedback.

Most common and useful Agile metrics

4. Lead Time

- **What It Measures:** The total time it takes for a work item (e.g., a user story or feature) to move from the moment it is requested (e.g., added to the backlog) to its completion (e.g., shipped).
- **How It's Measured:** Lead time is measured from the moment a user story is created or prioritized in the backlog until it's delivered to the customer or deployed to production.
- **Why It's Important:** Lead time helps teams understand the overall speed of delivery, from the initiation of an idea to when it's realized. Shorter lead times are typically associated with more responsive teams and faster delivery.
- **Limitations:** Lead time can vary significantly depending on the complexity of the work and may be influenced by factors outside the team's control (e.g., dependencies on other teams).

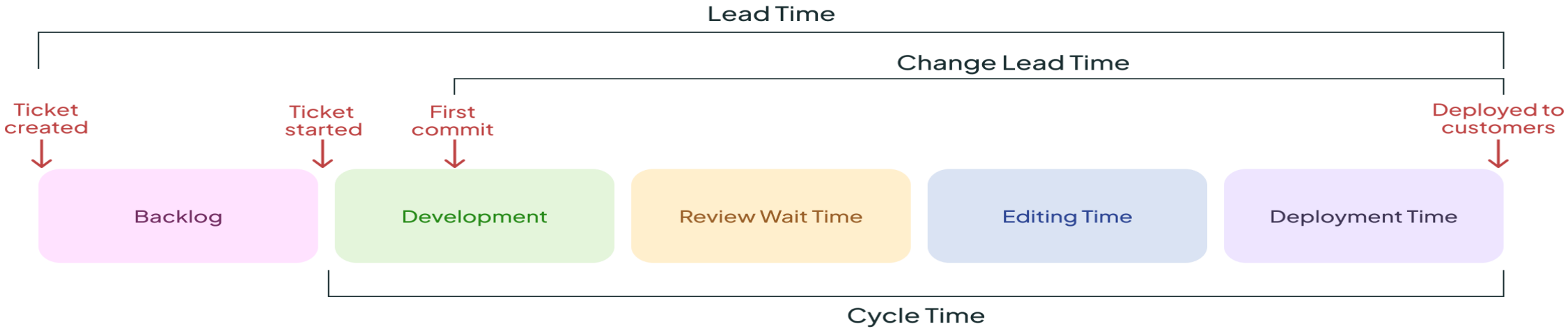
Most common and useful Agile metrics

5. Cycle Time

- **What It Measures:** The time it takes to complete a specific work item once the team starts working on it (i.e., from "In Progress" to "Done").
- **How It's Measured:** Cycle time starts when a work item is actively worked on and ends when it's completed.
- **Why It's Important:** Cycle time provides insight into the efficiency of the team's workflow. A shorter cycle time typically means a team can deliver features more quickly and efficiently.
- **Limitations:** Like lead time, cycle time can be influenced by external factors, and high variability in cycle time can indicate inefficiencies or bottlenecks in the process.

Most common and useful Agile metrics

- **Lead Time** starts when a customer or stakeholder requests something and ends when the result is delivered. It reflects the total wait time from idea to delivery, including time in queues or waiting for prioritization.
- **Cycle Time** starts when the team actively begins working on an item and ends when it's completed. It reflects the time spent in actual development or production.



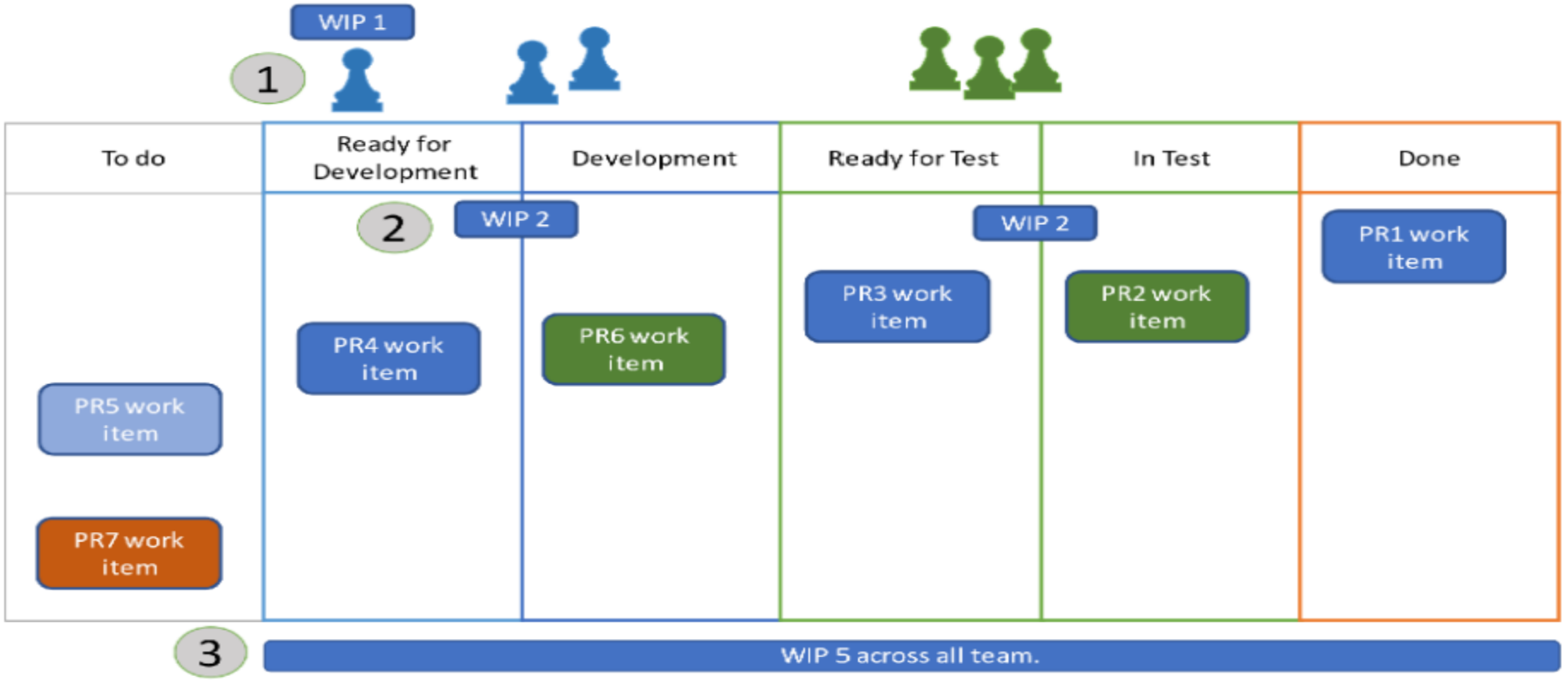
Most common and useful Agile metrics

6. Work in Progress (WIP)

- **What It Measures:** The number of work items (e.g., user stories, tasks, or features) that are currently in progress at any given time.
- **How It's Measured:** WIP is simply the count of tasks in the "In Progress" state at any time. Teams can set WIP limits to control this number, which helps maintain focus and avoid multitasking.
- **Why It's Important:** Limiting WIP helps reduce multitasking and context switching, which can improve productivity and quality. Teams can use WIP metrics to identify bottlenecks and areas for improvement.
- **Limitations:** WIP by itself may not give a full picture of team performance without context, such as how quickly tasks are being completed or how the WIP aligns with the overall flow of work.

Most common and useful Agile metrics

6. Work in Progress (WIP)

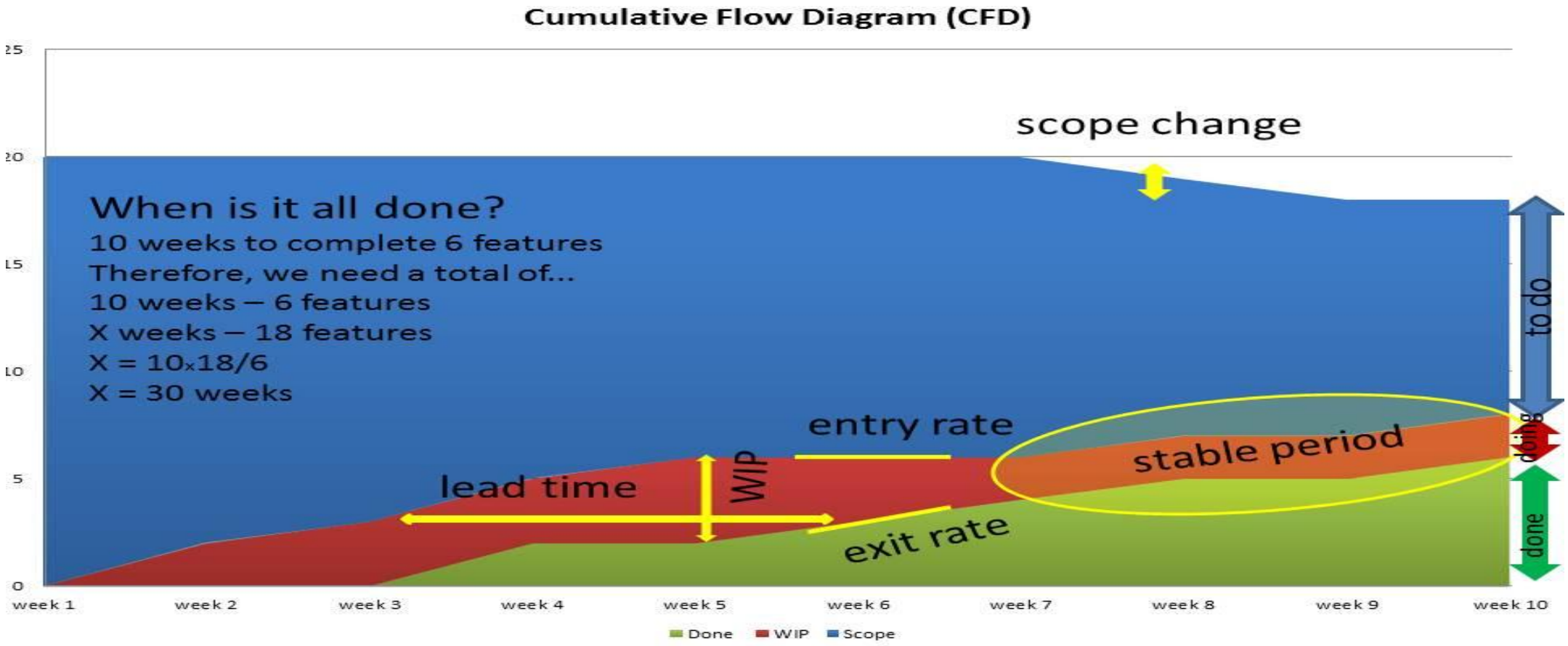


Most common and useful Agile metrics

7. Cumulative Flow Diagram (CFD)

- **What It Measures:** A visual representation of the flow of work through various stages of the process (e.g., To Do, In Progress, Done).
- **How It's Measured:** The CFD shows the amount of work in each state (e.g., backlog, in progress, completed) over time.
- **Why It's Important:** The CFD helps teams visualize bottlenecks and inefficiencies in the process. For example, if the "In Progress" line is growing faster than the "Done" line, it may indicate that work is piling up and not getting completed.
- **Limitations:** CFDs can be complex to interpret, especially in larger projects with many stages, and they require accurate tracking of tasks through each stage.

Most common and useful Agile metrics



JIRA

Introduction

- Jira Software is the #1 agile project management tool used by teams to plan, track, release and support world-class software with confidence.
- It is the single source of truth for your entire development lifecycle, empowering autonomous teams with the context to move quickly while staying connected to the greater business goal.
- Whether used to manage simple projects or to power your DevOps practices, Jira Software makes it easy for teams to move work forward, stay aligned, and communicate in context.

Introduction

- Jira Software launched in 2002 as an issue tracking and project management tool for teams.
- Since then, 65,000+ companies globally have adopted Jira for its flexibility to support any type of project and extensibility to work with thousands of apps and integrations.
- Used by :

1. Agile teams
2. Bug tracking teams
3. DevOps teams
4. Product management teams
5. Project management teams
6. Software development teams

Introduction

1. Jira Software for bug tracking

- Bugs are just a name for to-do's stemming from problems within the software a team is building.
- It is important for teams to view all the tasks and bugs in the backlog so they can prioritize big picture goals.
- Jira’s powerful workflow engine ensures that bugs are automatically assigned and prioritized once they are captured.
- Teams can then track a bug through to completion.

Introduction

2. Jira Software for requirements & test case management

- Jira Software is the central hub for the coding, collaboration, and release stages.
- It integrates with a variety of quality assurance apps, and allows for customizable fields, workflows and screens.
- All of these enable teams to manage manual and automated tests in their software development cycle seamlessly and effectively.

Components of JIRA

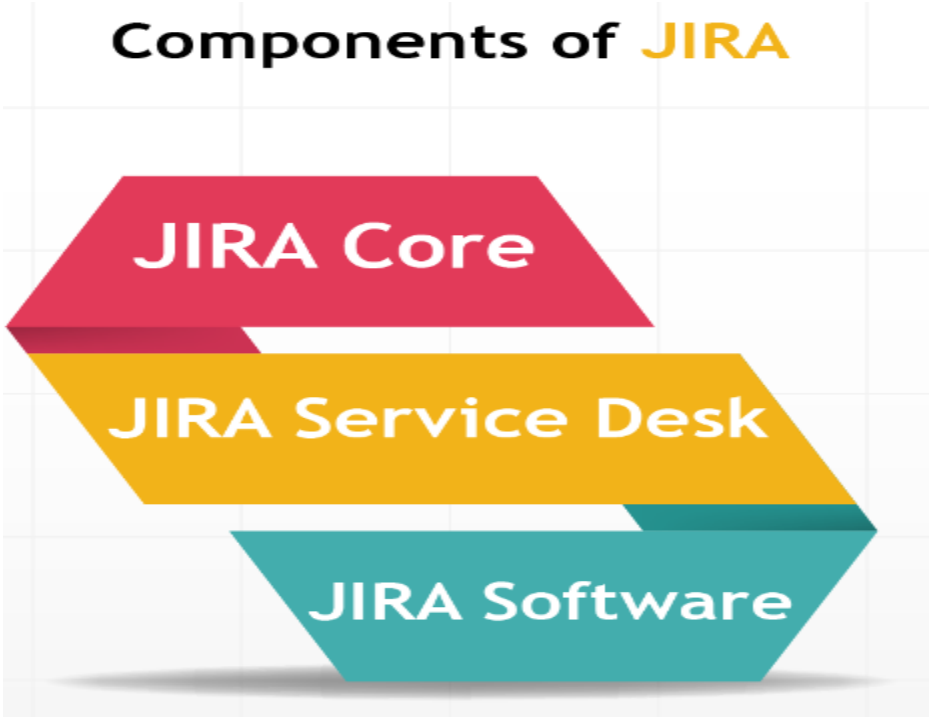
- JIRA is divided into 3 major components and the term JIRA refers to a common platform that these products are developed on. Now, let's have a look at these 3 JIRA components.

1. **JIRA Core** - JIRA Core resembles classic JIRA with its workflow capabilities and field customizations. JIRA Core is ideal for general-purpose task management.

2. **JIRA Service Desk** - JIRA Service Desk is a JIRA Core with the capabilities of a Service Desk. JIRA Service Desk is designed for running JIRA as a support system for ticketing with a focus on customer satisfaction with Service Level Agreement (SLA) goals and with a simple user interface for the end-users.

Components of JIRA

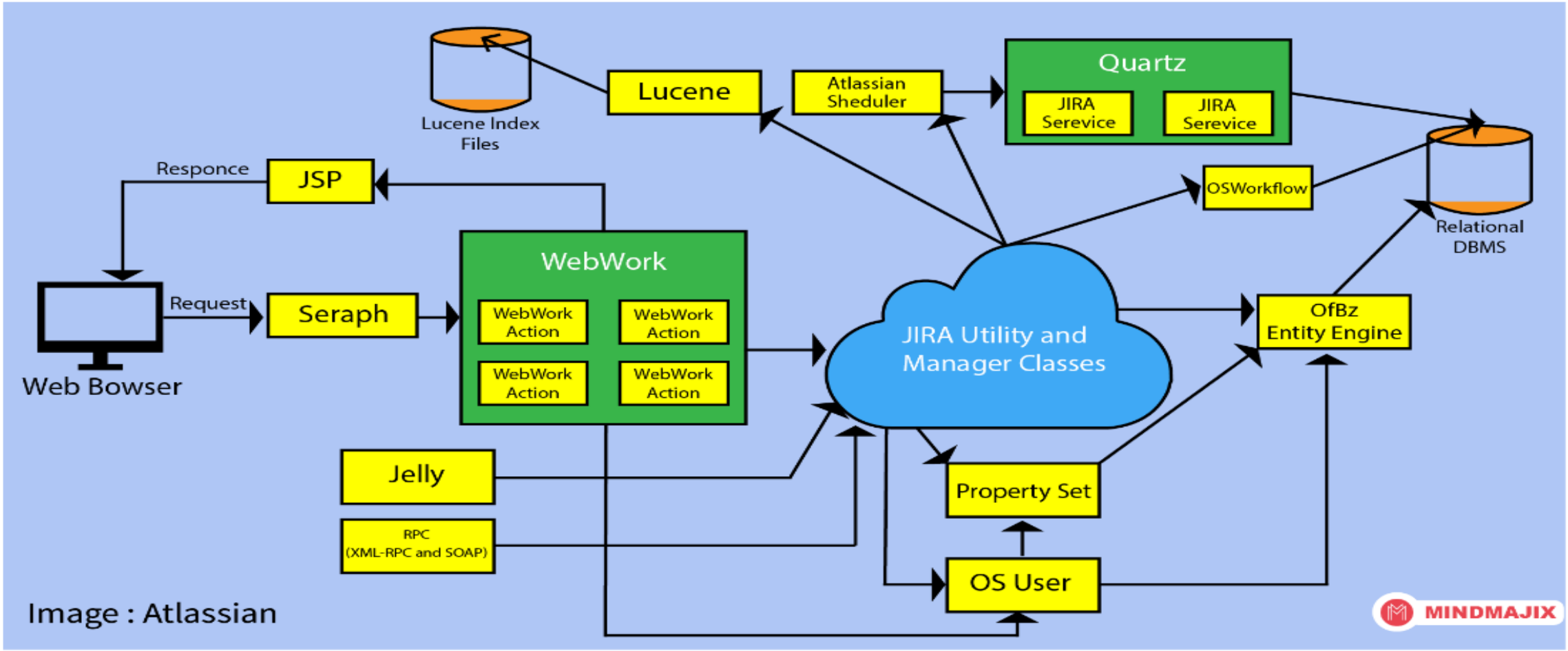
3. JIRA Software - Previously known as JIRA Agile, JIRA Software is JIRA Core with capabilities of Agile. This is highly suited for software development teams that want to utilize Agile methodologies like JIRA Kanban and Scrum.



JIRA Architecture

- JIRA is written in Java and is deployed into a java Servlet Container like Tomcat as a standard Java WAR file.
- Users interact with JIRA with the help of a web browser since JIRA is a web application. JIRA utilizes WebWork 1 for processing web requests that the users submit.
- WebWork 1 is actually an MVC framework that is similar to Struts.
- JSP is used by JIRA for the View layer. When you visit a specific URL in a web application, the scenario that happens is defined by a web application framework.
- In JIRA, almost all authentication is performed via Seraph. Seraph is an open-source web authentication framework.

JIRA Architecture



JIRA Architecture

- JIRA can have new actions that are defined with the help of Webwork.
- The Webwork Sample plugin consists of example classes and actions that can be utilized to understand this topic in a comprehensive manner.
- The Single Sign-On (SSO) and Identity Management tool of Atlassian is known as Crowd.

JIRA software Installation

- https://www.atlassian.com/software/jira/download

- Go the above site and choose appropriate software : JIRA SOFTWARE with DATA Center and BUG management.

JIRA Issues

- The issues in JIRA software help you to keep track of your team, estimate workload, and manage code. In this section, I will let you know what you can do with an issue.

What is an Issue?

- Various companies use JIRA Tool to track different types of issues, which can represent anything such as a leave request form, a project task, or a software bug.
- In any JIRA project, issues can be termed as its building blocks.
- A task, a bug, a story, or any other issue type can be represented as an issue.
- An issue can be stated as a packet of work in JIRA Core.

JIRA Issues

- It can be a large chunk of work or a small task depending on your project and how your team decides to divide work into issues.
- An example of a small issue can be “Remember to order a burger for charity night” and a large issue can be “Build a bridging wall between garage and house.”

Creating an Issue

- Before you begin with issue creation, you require the Create Issue project permission for the relevant project of the issue. Now, let's have a look at the steps involved in creating an issue.
- At the top of the screen, click "create" for opening the "Create Issue" dialog box.
- In the Create Issue dialog box, select the relevant Issue Type and Project.
- For the issue, type a "Summary" and complete any appropriate fields.
- If you want to hide existing fields or if you want to access fields that aren't displayed in this dialog box:
    - You should click the "Configure Fields" button present at the top right of the screen.

Creating an Issue

- Click on “Custom” and choose the fields you want to hide or show by clearing or selecting the relevant check boxes respectively, or for showing all fields, you should click “All.” These selected fields will be displayed when you next create an issue.

- **Optional:** You can select the “Create another” checkbox present at the bottom of the dialog if you want to create a series of similar issues with the same Issue and Project Type.
- In the new Create Issue dialog box, some of the fields may be pre-populated depending on the values you may have specified while creating previous issues, and your configuration. Before creating the next issue, ensure you check they are all correct.
- Click the “Create” button when you are satisfied with your issue’s content.

JIRA Issue Types

- JIRA software allows you to keep track of various things such as helpdesk tickets, tasks, bugs, etc with the help of different tissue types.
- An issue type can also be configured to act differently. This section will let you know the specific issue types that exist within the 3 JIRA applications, for example, to track different pieces of information or to follow a different process flow.

Default Issue Types in JIRA Core

•Task - A task that needs to be done.

•Subtask - A small task existing with a large chunk of work.

JIRA Issue Types

Default Issue Types in JIRA Software

•Story - A story is a functionality request expressed from the user's perspective.

•Bug - A bug is a problem that impairs service or product functionality.

•Epic - An Epic is a large chunk of work that consists of many issues.

Default Issues Types in JIRA Service Desk

•Incident - An incident or a System outage.

•Service request - A service request is a general request from a user for a service or a product.

•Change - A change is a rollout of new solutions or technologies.

•Problem - A problem is the tracking of underlying causes of incidents.

Exporting Issue in JIRA

- Let’s say you already have a JIRA Cloud site and you need to move to JIRA Server, you can actually create a backup of your JIRA Cloud data which you can import into server installation.
- It is to be noted that, for your instance, the Atlassian Cloud takes backups every 24 hours for application recovery purposes.
- The following data can be backed up and exported from your JIRA Cloud site:

  - Issues
  - Users and user group settings
  - Project logos, user avatars, and issue attachments

Sharing Issues in JIRA

- You can use the Share option to email the link to an issue, to other JIRA users.
- This option also enables you to add an optional note to the email.
- For sharing the issue with other JIRA users, first, you have to view the issue that you want to share.
- Then, you should click the Share button present at the top-right.
- After this, you have to specify the JIRA users or type the individuals' email is that which you wish to share the issue. You can add an optional note.
- If you are done with all this, you can click the Share button which is present at the bottom of the window that is displayed.

JIRA Screens

- In JIRA, when an issue is viewed, edited, or created, the resulting collection of fields that appear is defined as a Screen.
- A screen provides you with control over the data you want to get included in your issue based on the type of issue you are editing or creating.
- How to Create a Screen in JIRA?
- To create a screen in JIRA Tool, you must navigate to the Issues Administration page and choose “Screens” from the left sidebar.
- Then click on “Add Screen” and then, name your screen and write a description for it.

JIRA Screens

- See that the name is descriptive and says exactly what the screen is intended for.
- Then, you have to add fields to your screen.
- The required field is only the Summary field but that is not enough for an issue.

JIRA Schemes

- A JIRA Scheme maps a project to any type of concept managed by JIRA.
- By defining schemes, JIRA enables you to display specific pieces of issue information at specific times.
- A screen can simply be stated as a collection of fields.
- When an issue is being edited, viewed, created, or transitioned via a specific step in a workflow, you can choose which screen to get displayed.
- The screen scheme of a project determines which screens have to be displayed for different issue operations such as create, edit, or view.

JIRA Schemes

The various types of Schemes are as follows.

- Issue type of screen scheme
- Issue type scheme
- Workflow scheme
- Issue security scheme
- Screen scheme
- Field configuration scheme
- Permission scheme
- Notification scheme

JIRA Project

- In JIRA Tool, the collection of issues is known as a Project.
- A JIRA project could be used to manage a help desk, track a project, coordinate a product's development, and more, based on your requirements.
- A JIRA project can also be customized as well as configured for suit your needs.

  - Creating a Project in JIRA
  - Click on "Projects" in the header and then on "Create Project."
  - Follow the wizard for project creation.

JIRA Project

Project types:

- More than one project type is available to you based on which JIRA applications you have installed.
- Every project type has a particular set of features.
- All the users on the JIRA instance can see all projects, but the actions they can take and the features they see are determined by their project-specific permissions and application access.

JIRA Project

Shared Configurations:

- A project is created with its own set of schemes when you create it from a template. The schemes are:

  - A field configuration scheme (default)

  - An issue type screen scheme

  - An issue type scheme

  - A workflow scheme

  - An issue security scheme

  - A notification scheme

- - A permission scheme

JIRA Sprints

- A Sprint is defined as a fixed time period wherein teams finish work from their product backlog.
- The time period for sprints may be one, two, or four weeks.
- A team will usually have developed and implemented a working product increment at the end of the Sprint.

JIRA Sprints

How to Create a Sprint in JIRA?

- In order to create a sprint in JIRA, you must navigate to the “Backlog” of your Scrum project and click on the “Create Sprint” button which is present at the top of the Backlog.

Planning Sprints

- Every sprint begins with a planning meeting.
- While planning a sprint, your team would usually commit to providing a set of stories that are pulled from the top of a backlog.
- You see sprints on a board in JIRA software and assign issues to them. With the help of JQL (Sprint field), you can search for any issues in upcoming sprints. This involves starting the sprint, assigning stories to the sprint, and creating a sprint.

JIRA Sprints

How to Start a Sprint?

- In order to start a sprint, you must navigate to the “Backlog” of your Scrum project.
- You should locate the sprint that you wish to begin and click on “Start sprint”.
- Then, you have to update the Sprint name and add a sprint goal if required.
- After that, choose the Start and End dates for the Sprint.
- You will be navigated to the Active sprints where you can view the issues in the newly started sprint.

JIRA Sprints

Viewing Issues in a Sprint

- To view planned sprints, you can utilize the backlog of aboard. You can utilize the Active sprints of a board if you want to see a sprint in progress. JQL can also be used to search for a sprint's issues

JIRA Reports

- JIRA Core offers a range of reports that display statistics for specific versions, people, information about issues.

How to Generate Reports in JIRA?

In order to generate a report, you must navigate to the desired project, and click on “Reports”. Then, you have to select a report from the list of reports. The various types of reports are as follows.

- Average Age Report
- Pie Chart Report
- Created vs Resolved Issues Report
- Resolution Time Report

JIRA Reports

- Recently Created Issues Report
- Time Since Issues Report
- Single Level Group By Report
- User Workload Report
- Time Tracking Report
- Workload Pie Chart Report
- Version Workload Report

JIRA Blocks

Workflows

- Your JIRA issues can follow a process that mirrors the practices of your team.
- A workflow defines a sequence of steps or statuses that an issue will follow.
- Examples of the statuses or steps are “Resolved,” “In Progress,” and “Open.”
- How the issues will transition between steps can be configured by you.
- The workflow scheme of a project determines which workflows will apply to issue types in this project.

JIRA Blocks

Fields

- You can define field behavior in JIRA by using fields.
- Every field can be visible or hidden, plain text or rich text, and optional or required.
- A field configuration determines the requirements, overall visibility, and help-text, and formatting of each field.
- A field configuration scheme in a project determines which field configuration can be applied to issue types in this project.

JIRA Blocks

Roles

• In different projects, different roles can be played by different people.

• For example, the same person can be an observer in one project and the leader of another project. JIRA allows you to allocate specific people to particular roles in your project.

Versions

• In JIRA, a grouping of issues can be done by allocating them to versions.

• Take, for example, if JIRA can be used to manage the build of a house or manage the development of a product, you must define different versions which can help you to track which issues relate to various phases of your build or product. JIRA can help you archive, release, and manage your versions.

JIRA Blocks

Components

• You can define various components for categorizing and managing different issues.

• Take for example, for a software development project, you might define components such as “Documentation,” “Usability,” “Database”.

• For each component, you can choose a default assignee.

• This is useful if you have different people who lead different sub-teams in your project.

Permissions

•JIRA enables you to control who will be able to access your project and what exactly they can do, with the help of project permissions. With the help of security levels, access to individual issues can be controlled by you. Also, you can choose to provide access to specific roles, or groups, or users.

JIRA Blocks

Notifications

• When the occurrence of a specific event happens in your project, JIRA can notify the appropriate people.

• When different events occur, you can select specific roles, groups, or people for receiving email notifications.

Development Tools

• The development tools section is available only on JIRA projects, and it can be viewed only by users of JIRA software.

• This section provides you with an overview of development tools that are connected and specifies which users can utilize the integration features between them.

JIRA Blocks

Epic

• An Epic captures a large chunk of work and it is defined as a large user story that can be divided into a number of smaller user stories.

User story or Story

• A user story is a software system requirement that is stated in a few short sentences with the usage of non-technical language.

• A story is expressed as an issue in JIRA Agile and, within the story, the individual tasks are represented as sub-tasks.

JIRA Blocks

Task

• A unit of work present within a story is known as a task.

• The stories are expressed as parent issues in JIRA Agile while the sub-task issues are the name by which the individual tasks are represented.

Sub-task

• For an issue, a sub-task can be created to allow different aspects of an issue to be assigned to different persons, or to split the issue into smaller chunks.

• To allow the sub-task to be worked on independently, you can convert it to an issue if you find it holding up the issue’s resolution.

JIRA Blocks

Using a Bug Report Template as a Jira User

• You will need to create an issue, select the project under which it falls, and decide on the issue type.

• When you create an issue, there are many details you will need to provide. For bugs, there are some standard things you will have to add.

• Summary: a quick description of the bug

• Description: a more in-depth look at the bug

• Expected and actual results

• Steps to reproduce

• Priority: the severity of the bug

JIRA Blocks

- Environment: the operating system, device, network connectivity, etc.
- Attachment: screenshots, video, and other documents
- Assignee: the person assigned to deal with the issue
- Status: created automatically depending on your workflow
- It is important to remember that Jira lets you assign one or more people to an issue. You also have the option of automating this step so that it can change assignees when going from one status to another.
- Each issue also has an activity section where those assigned to deal with it can leave notes and where you can see the history of all activities.
- Jira also makes it possible for you to link issues together if they are related. If lots of people report the same bug, you can put the reports together.

Quiz

1) In JIRA, which of the following represents a high-level objective that can span multiple sprints?

a) Story

b) Sub-task

c) Task

d) Epic

Answer : Option d)

Quiz

2) A team wants to track how much work remains in the sprint each day. Which chart should they use in JIRA?

a) Burnup Chart

b) Velocity Chart

c) Burndown Chart

d) Cumulative Flow Diagram

Answer : Option c)

Quiz

3) What is the default issue type used to represent a piece of work that delivers value to the end user in JIRA?

a) Epic

b) Bug

c) Story

d) Task

Answer : Option c)

Quiz

4) In JIRA, what is the purpose of a board in Scrum projects?

a) To assign permissions

b) To manage releases

c) To visualize the team's workflow

d) To generate reports

Answer : Option c)

Quiz

5) Who usually has permission to start and complete a sprint in JIRA Scrum board?

a) Product Owner

b) Scrum Master or Board Admin

c) Team Member

d) Any logged-in user

Answer : Option b)

JIRA – Hands-On

STEPS

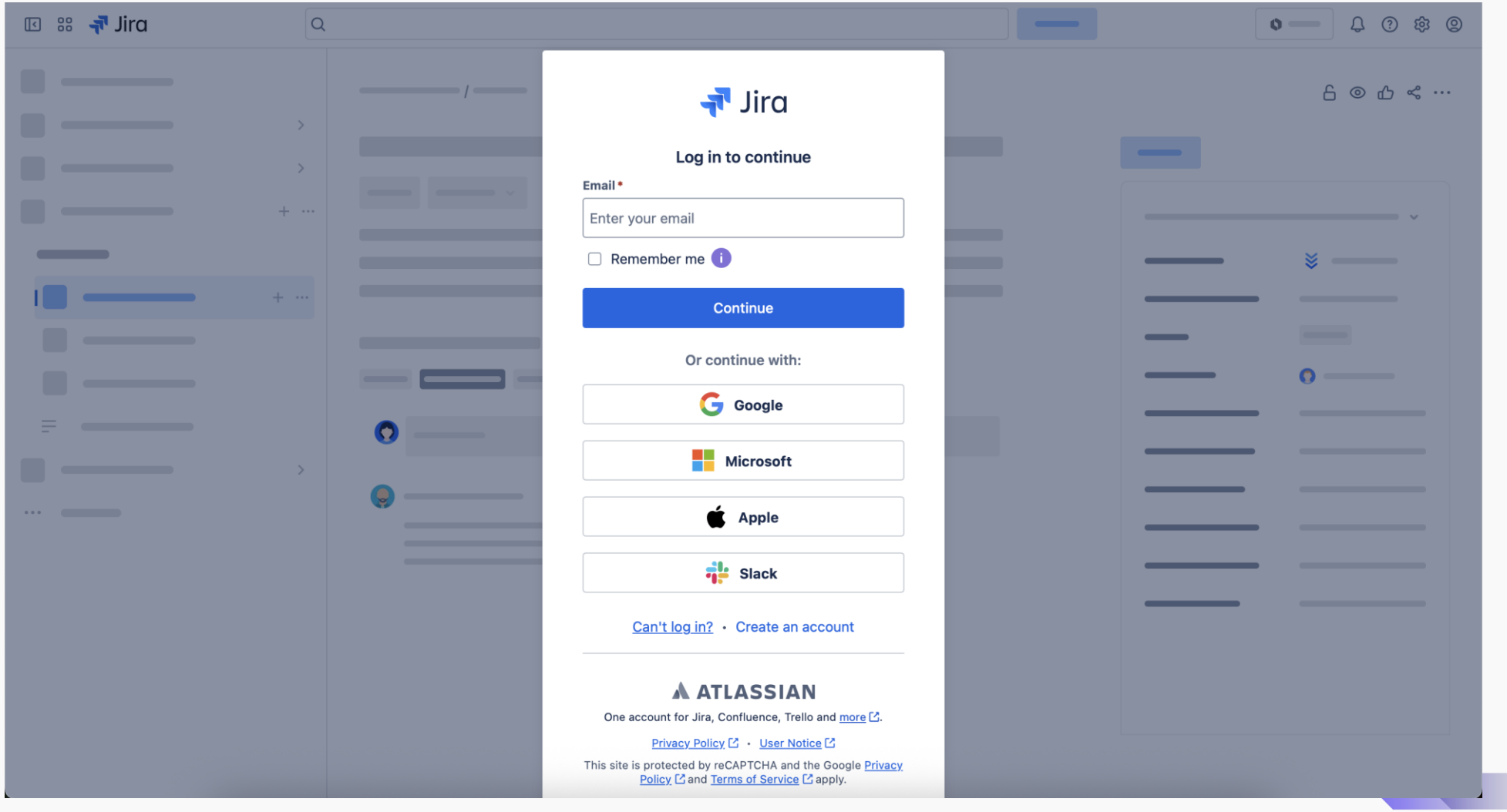
01 Login

02 Project Creation

03 Backlog

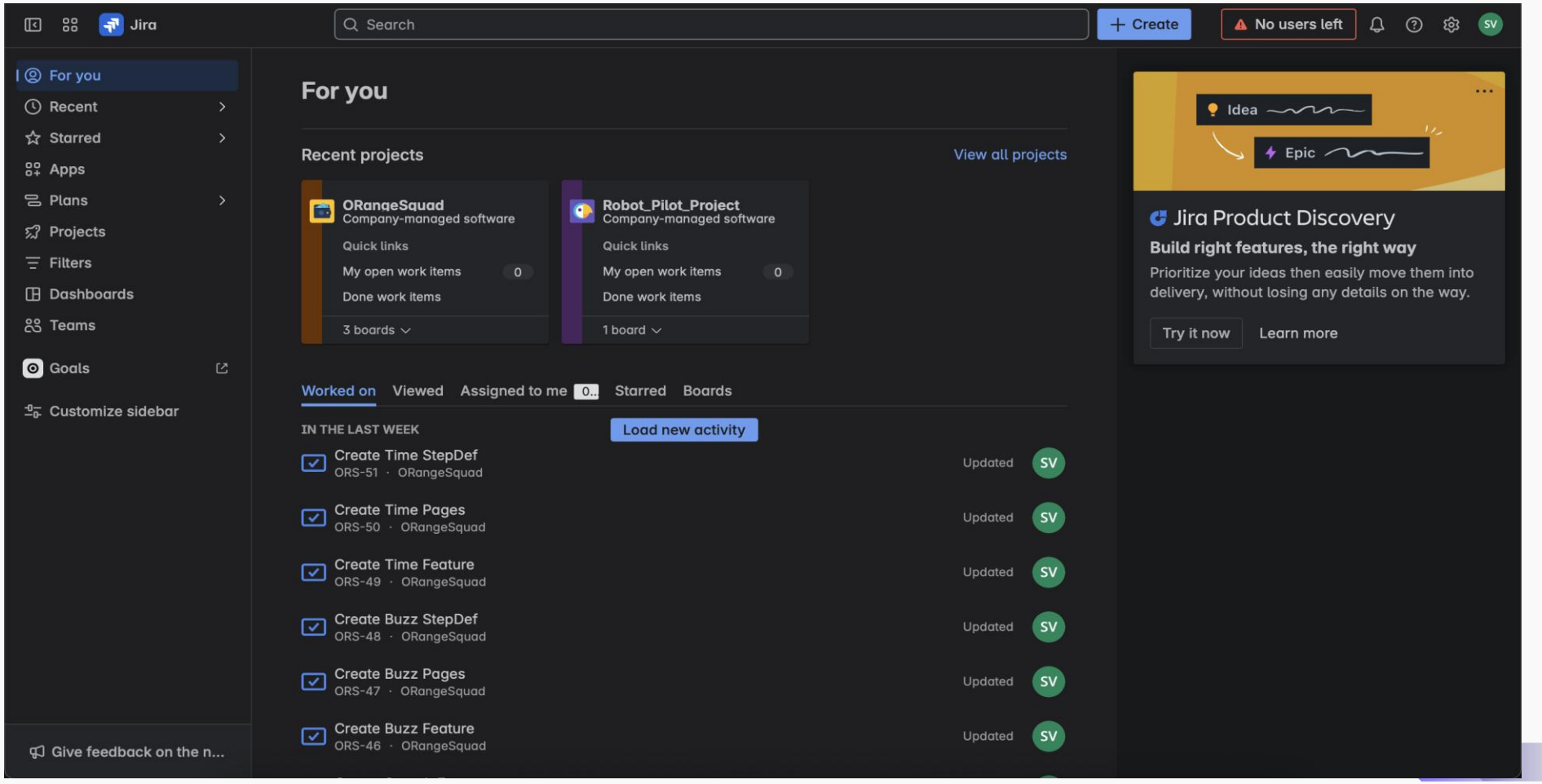
04 Reports

LOGIN https://www.atlassian.com/try/cloud/signup



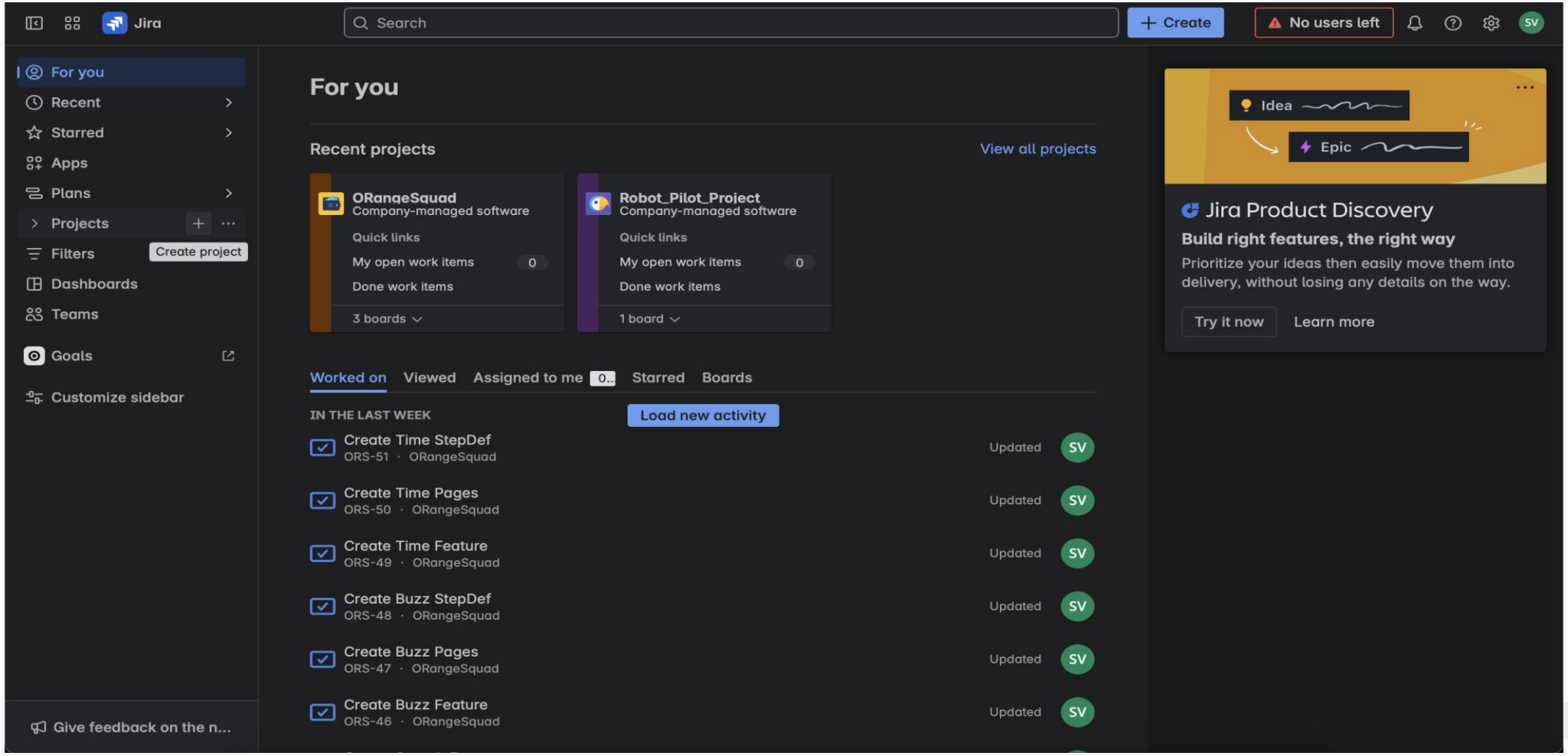
PROJECT CREATION

After login you will be shown with a Homepage



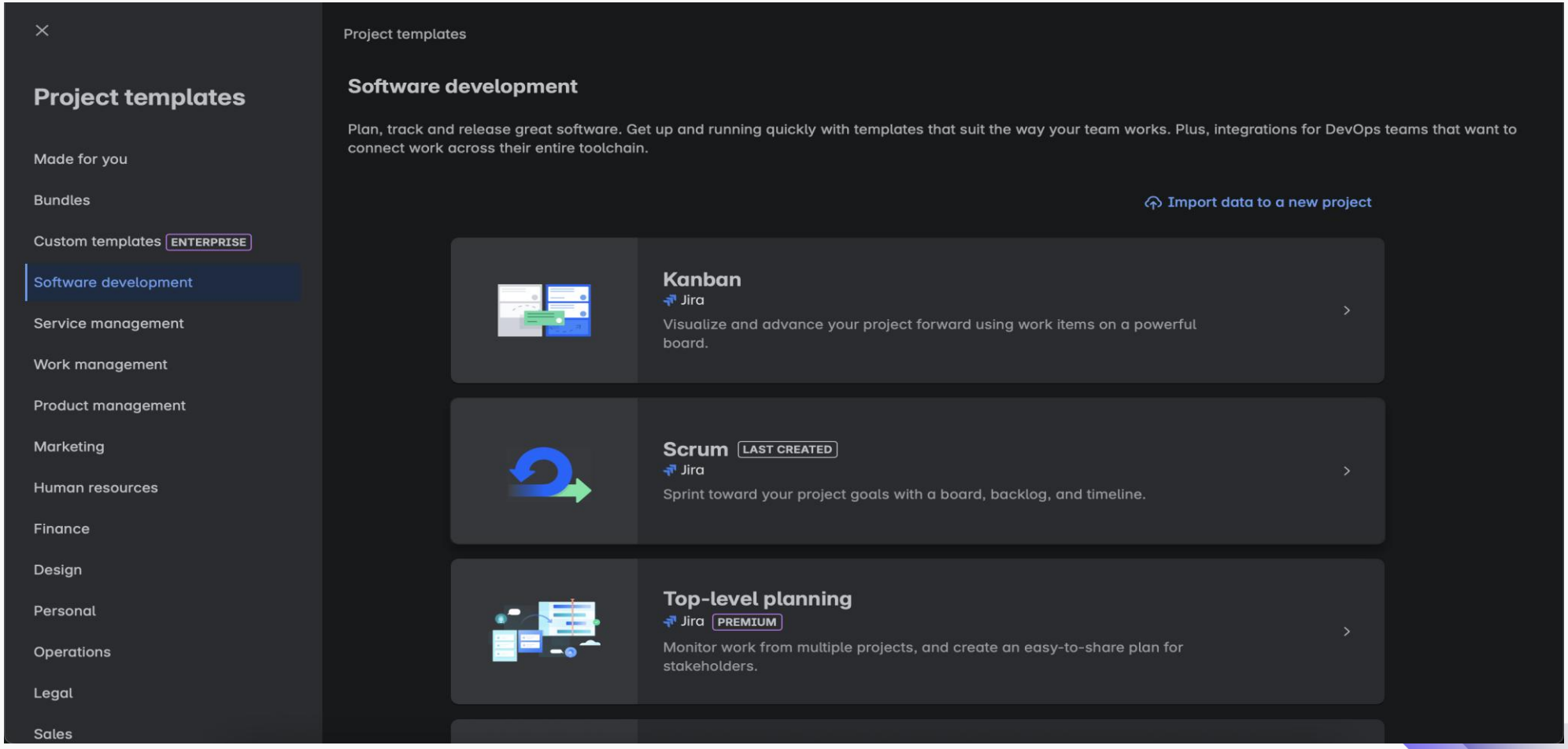
PROJECT CREATION

Click the + icon near the create project



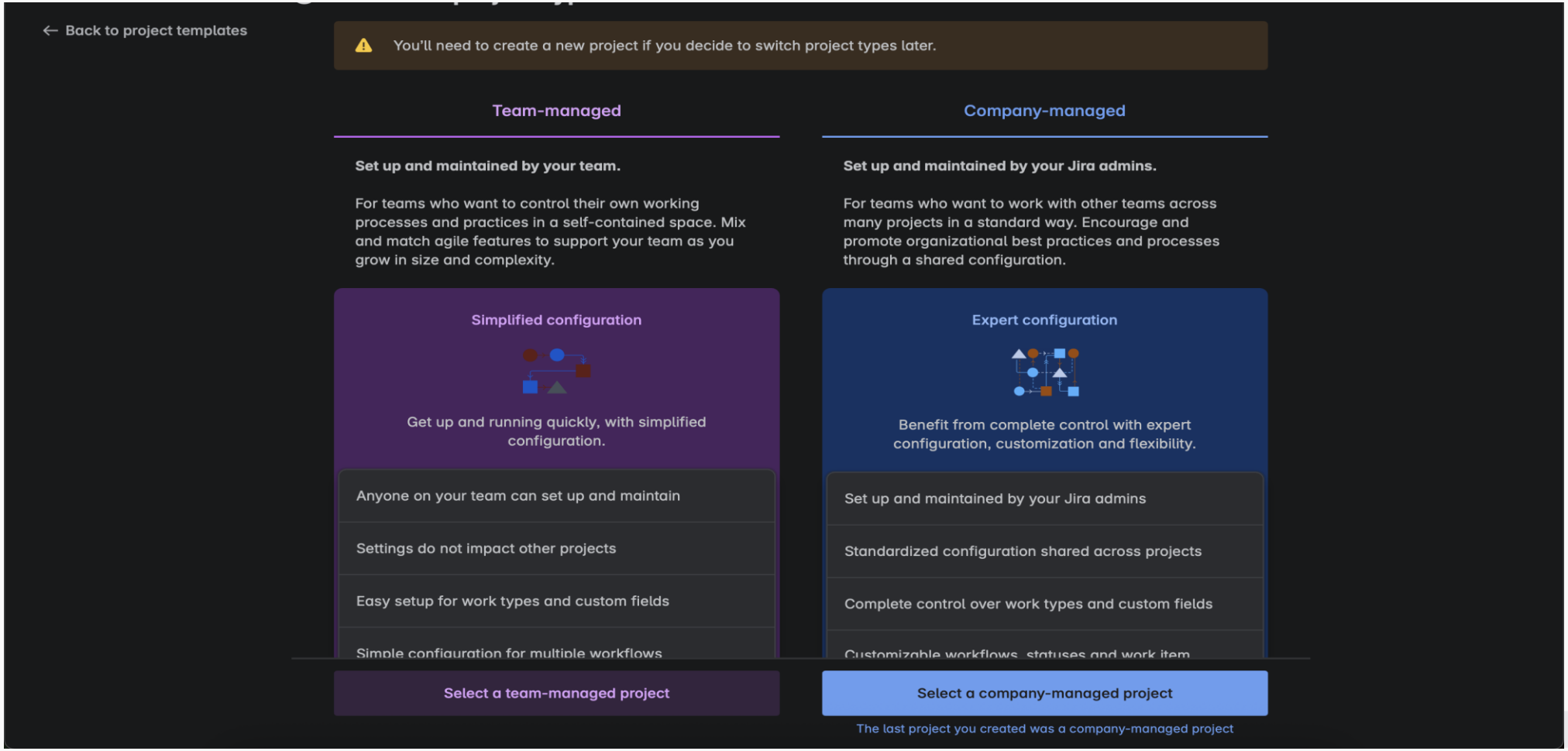
PROJECT CREATION

Click on the Software Development and Scrum



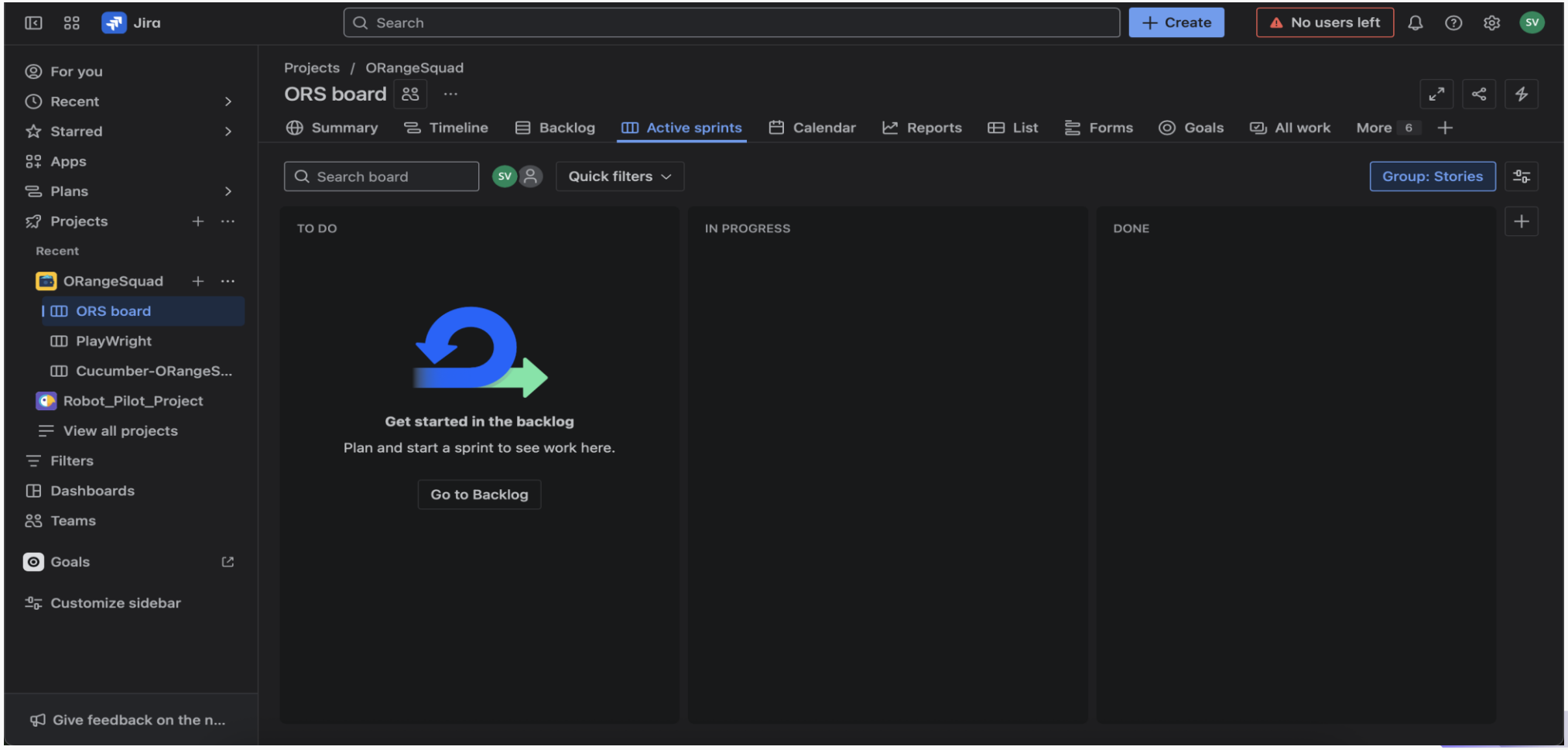
PROJECT CREATION

Click on the Company Managend Project



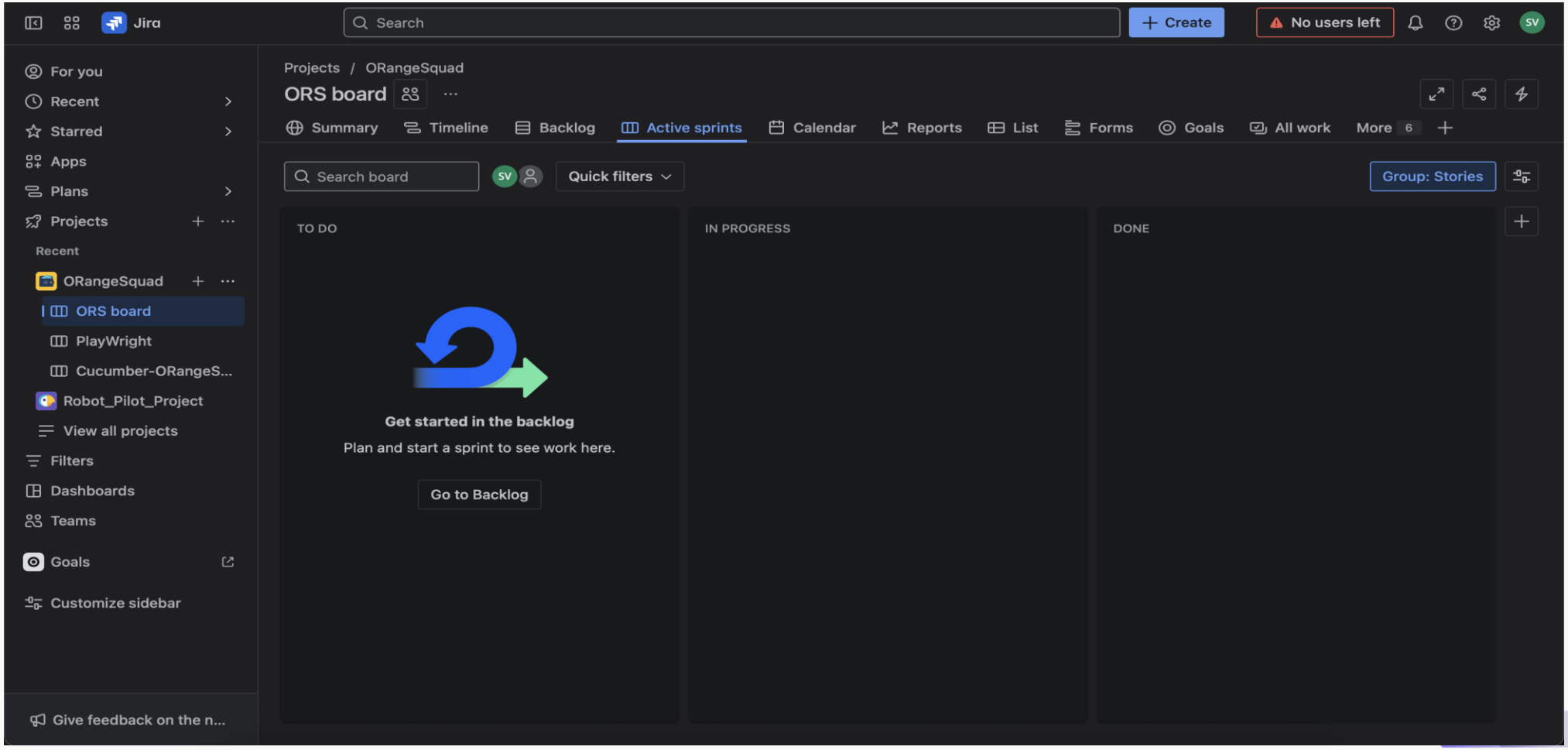
PROJECT CREATION

The dashboard will be displayed



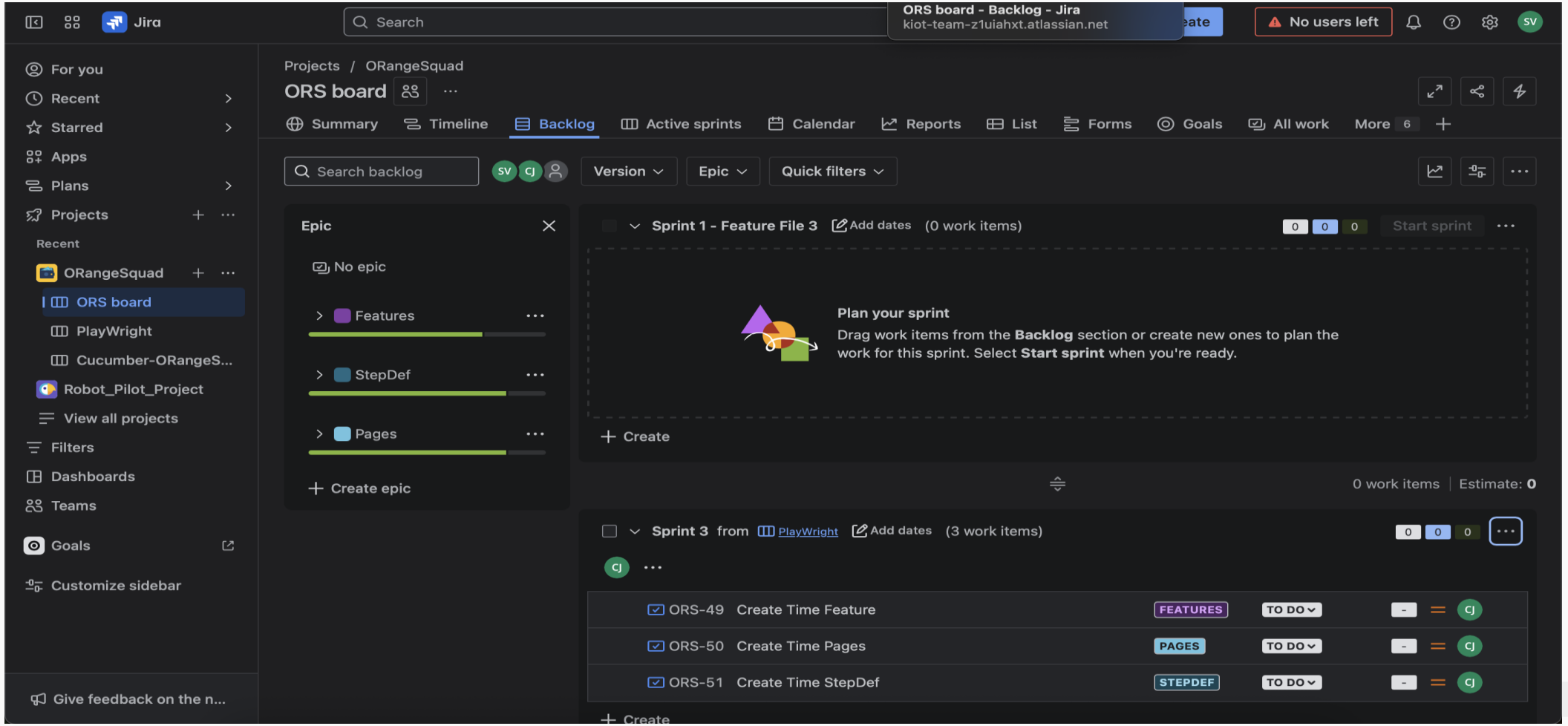
BACKLOGS

The dashboard will be displayed and click on Backlog



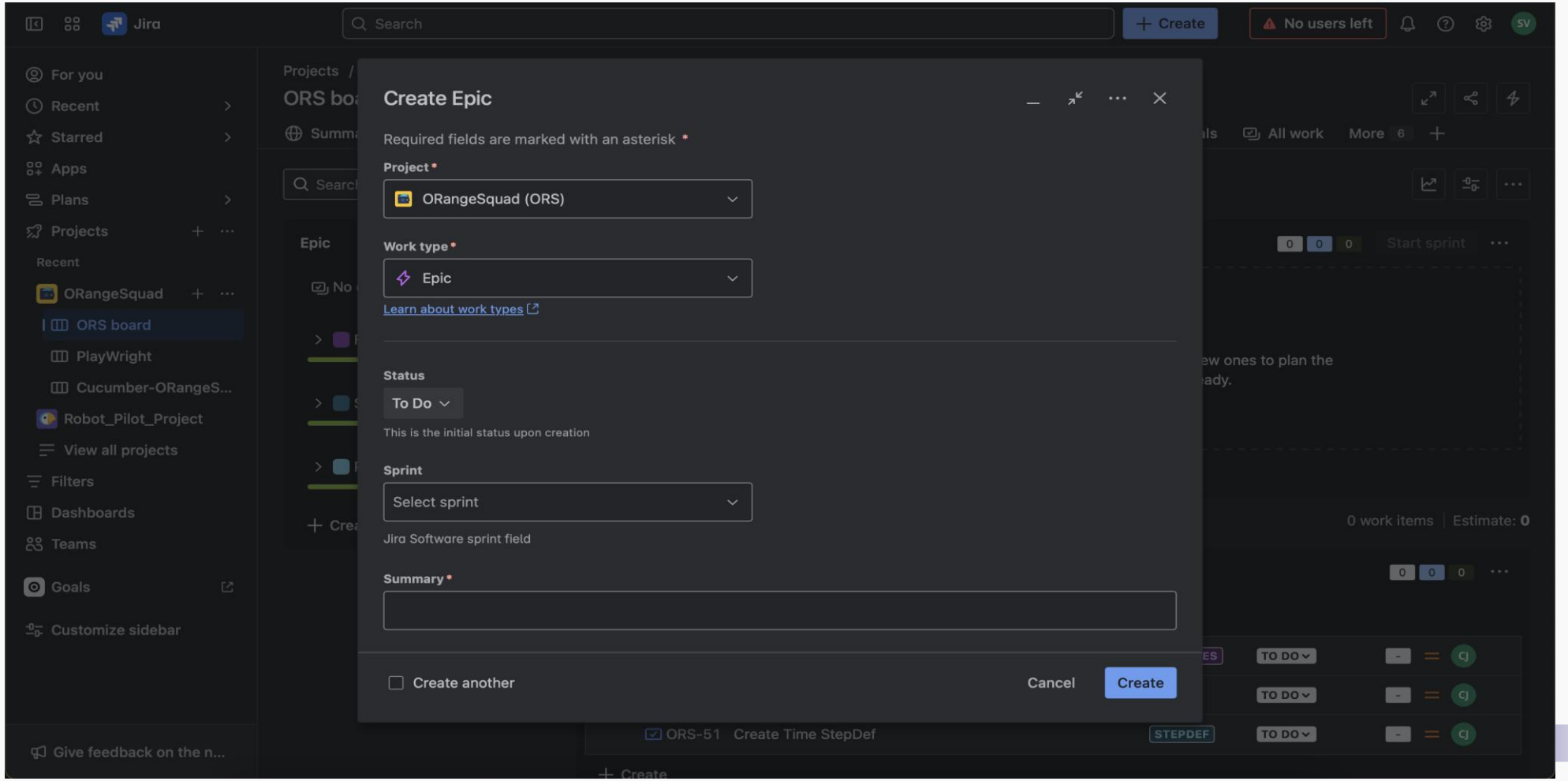
BACKLOGS

Press E key to display the Epics and click Create epic



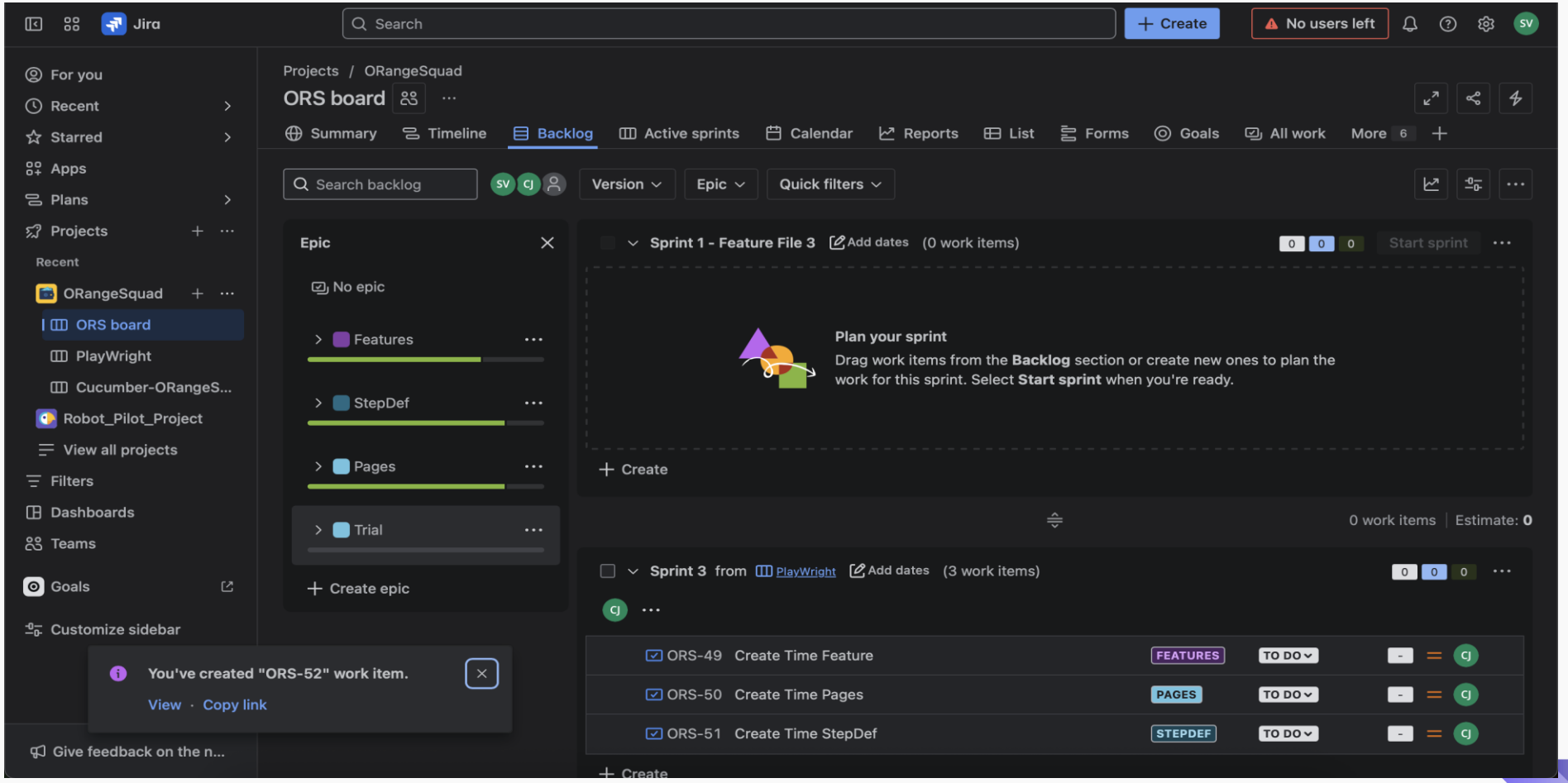
BACKLOGS

Fill the fields to create your epic



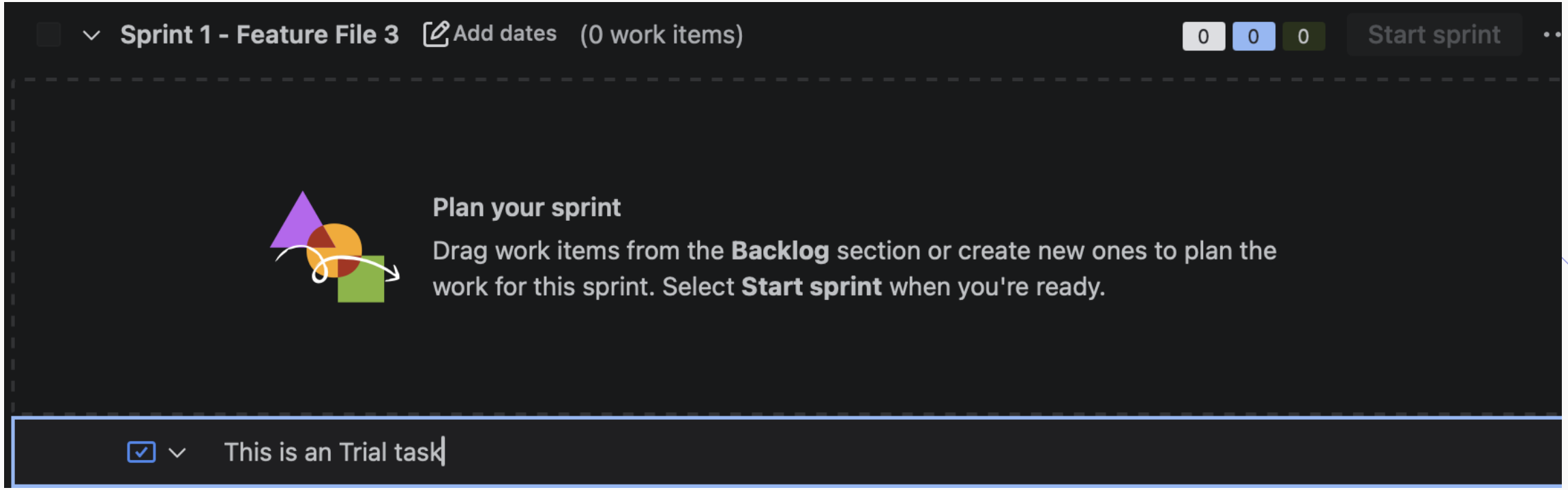
BACKLOGS

Your new epic will be displayed here



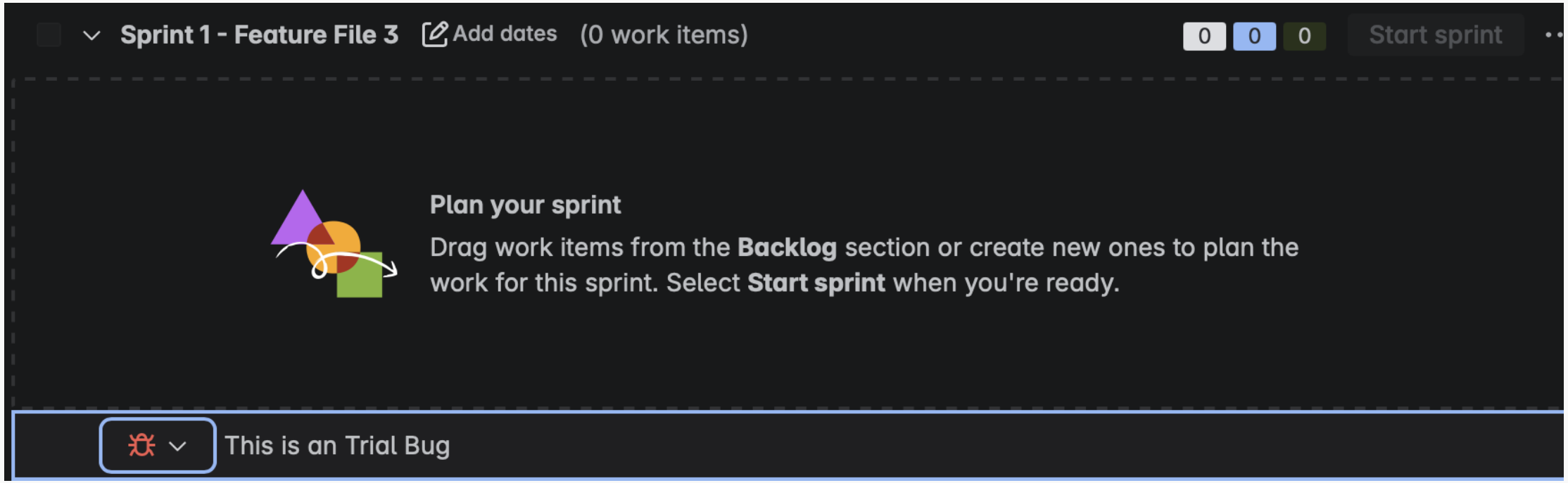
BACKLOGS

In your sprint 1 Click on Create and Add a Task



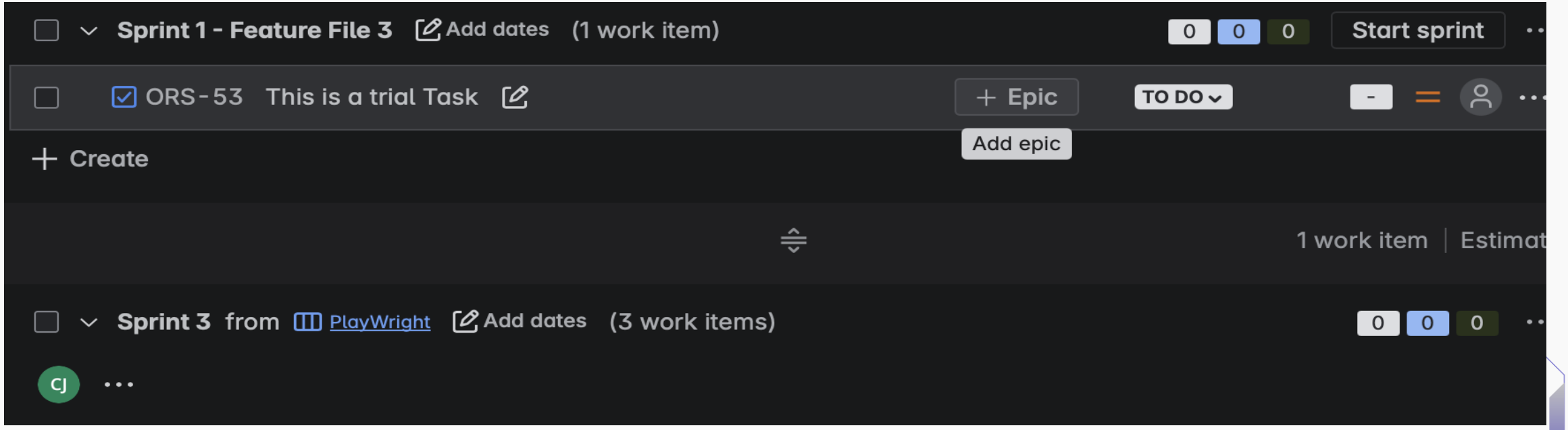
BACKLOGS

If it's a bug then click on the dropdown and change as bug



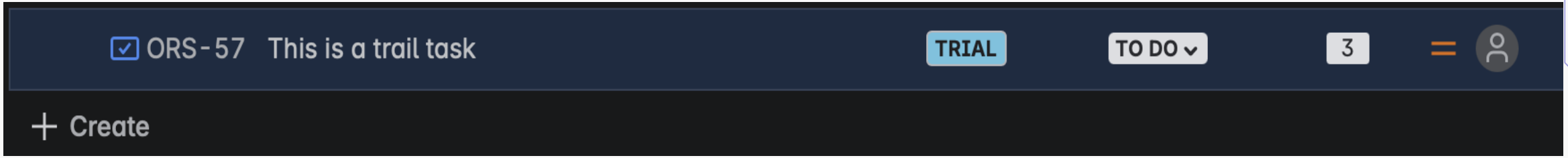
BACKLOGS

Hover on the Task and Click on Epic to add your epic



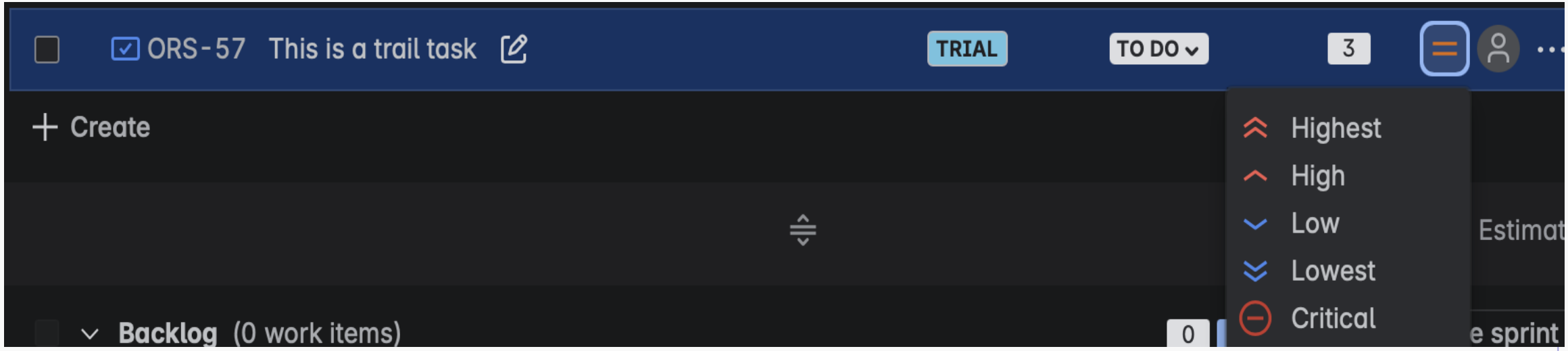
BACKLOGS

Click on the story points to add them



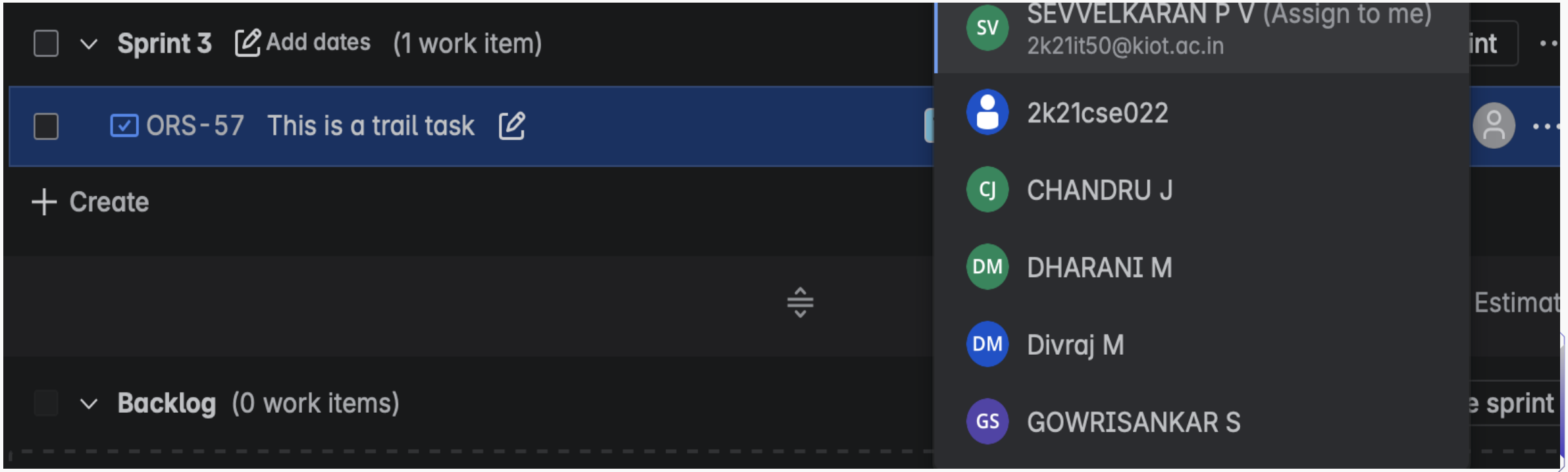
BACKLOGS

Click on the Priority to add the priority your task needs



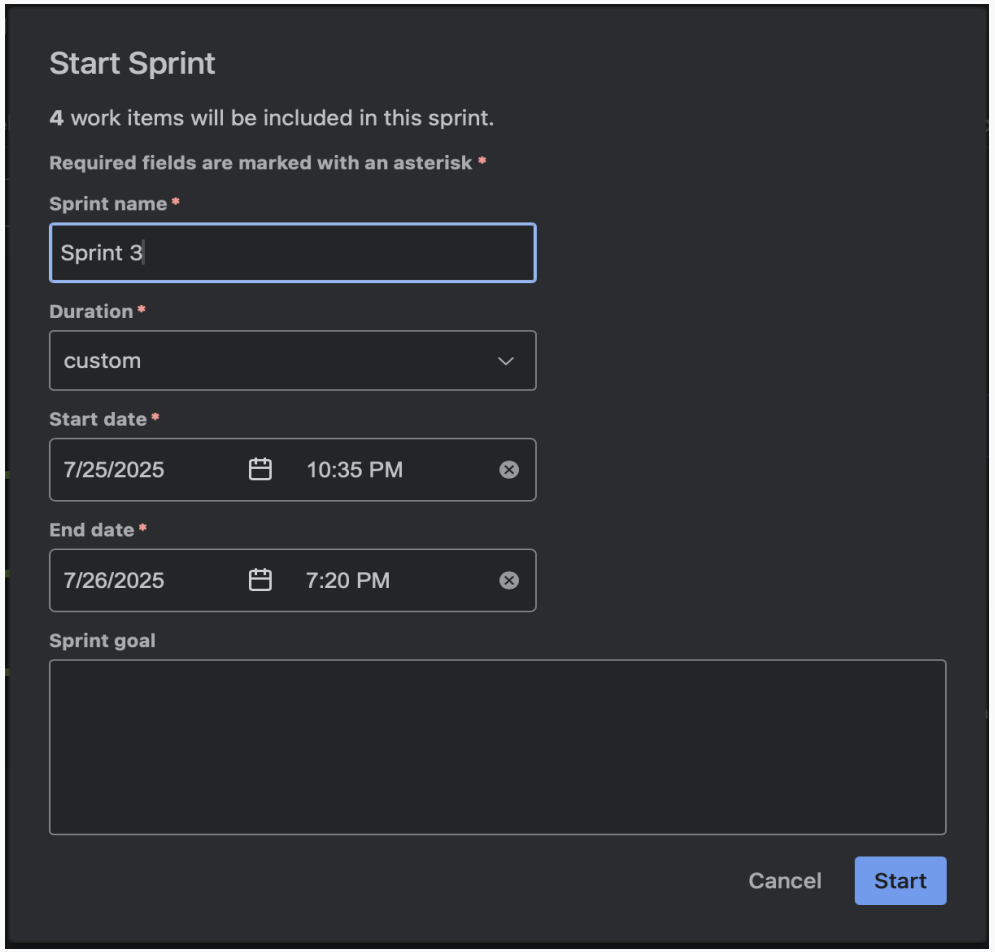
BACKLOGS

Assign the task to the respected individual in your team



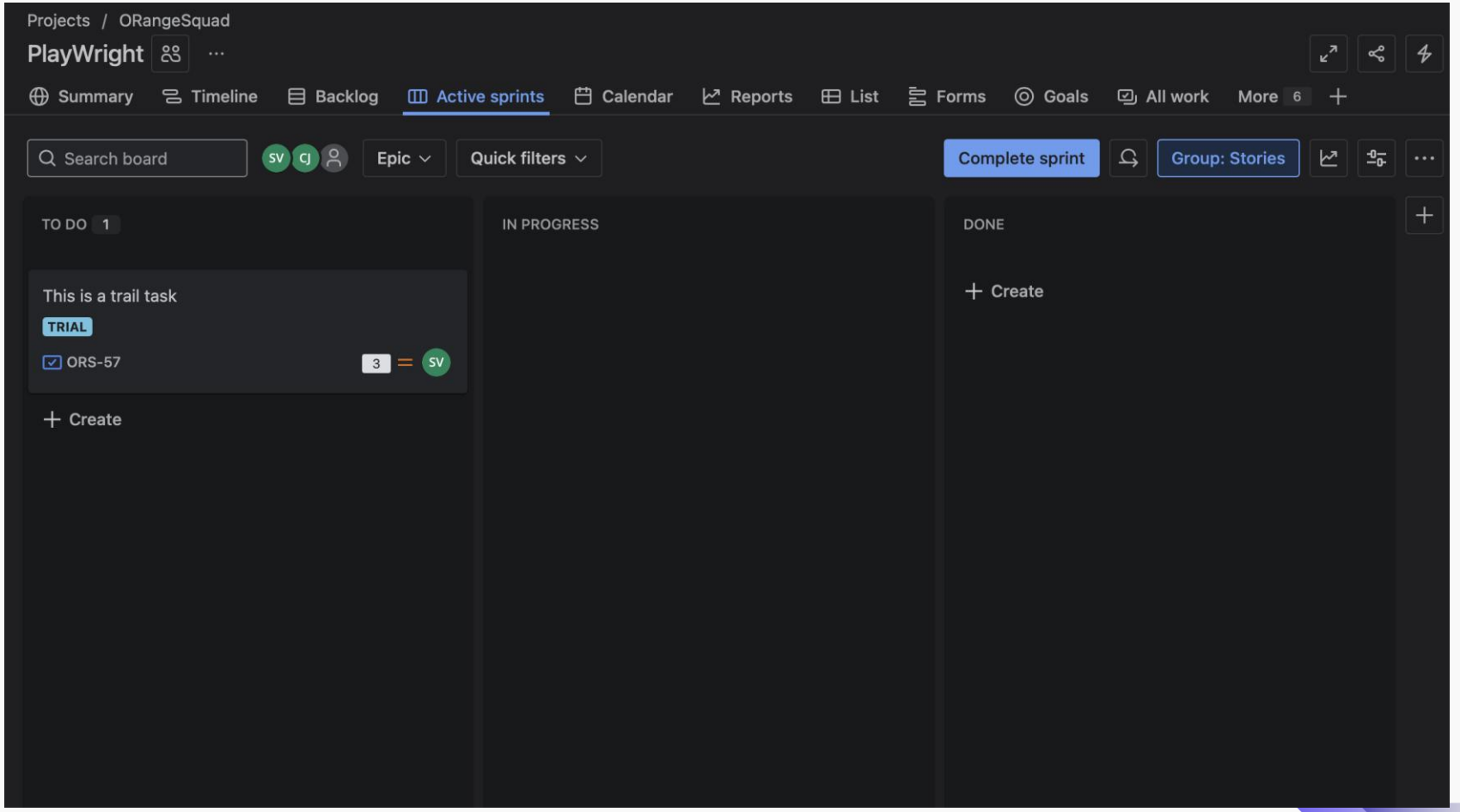
BACKLOGS

Once you set the tasks now click on start sprint



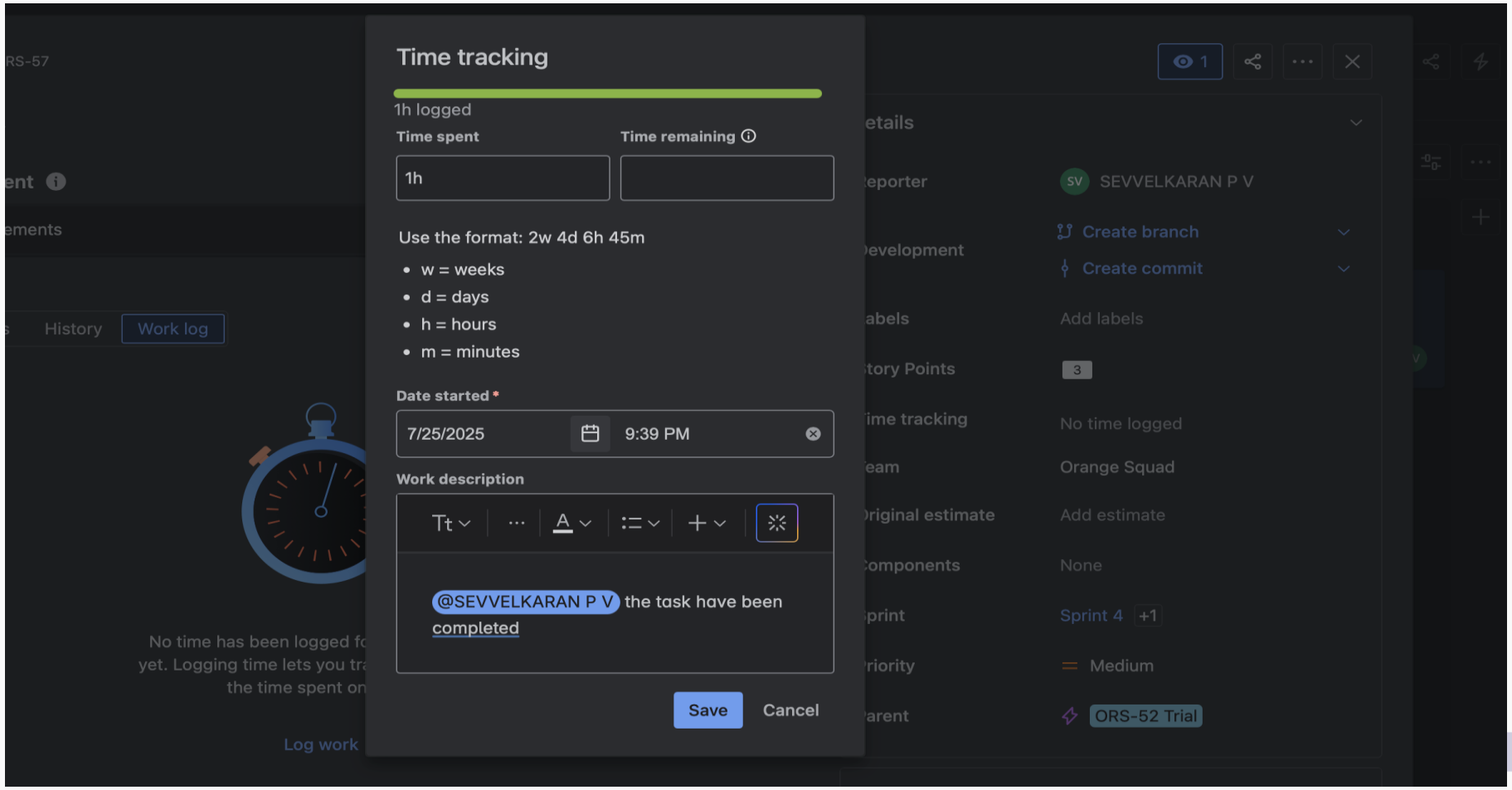
BACKLOGS

Once the sprint starts it will be reflected in Active sprints



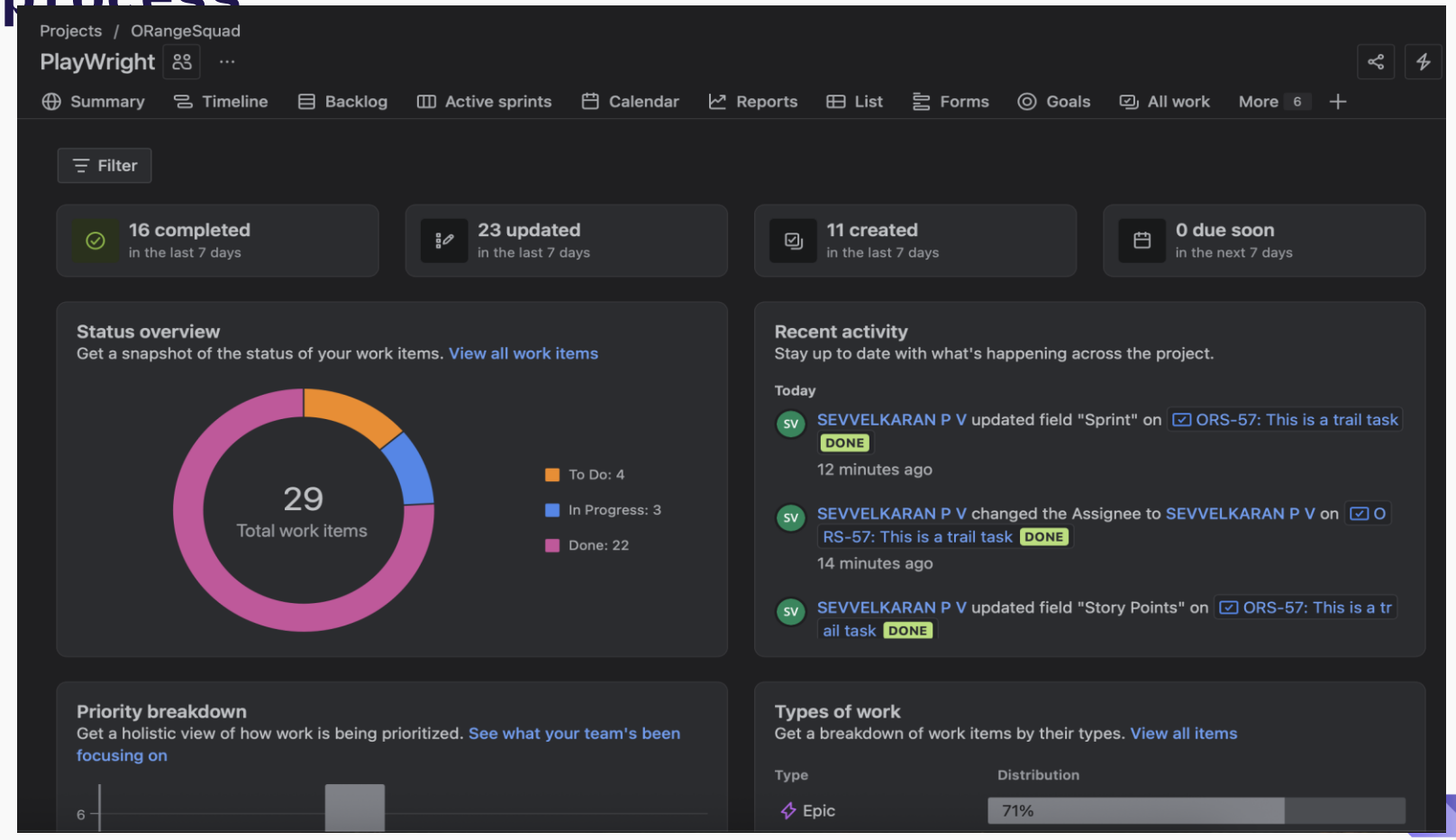
BACKLOGS

Once you are done for the day click on the task and add your log



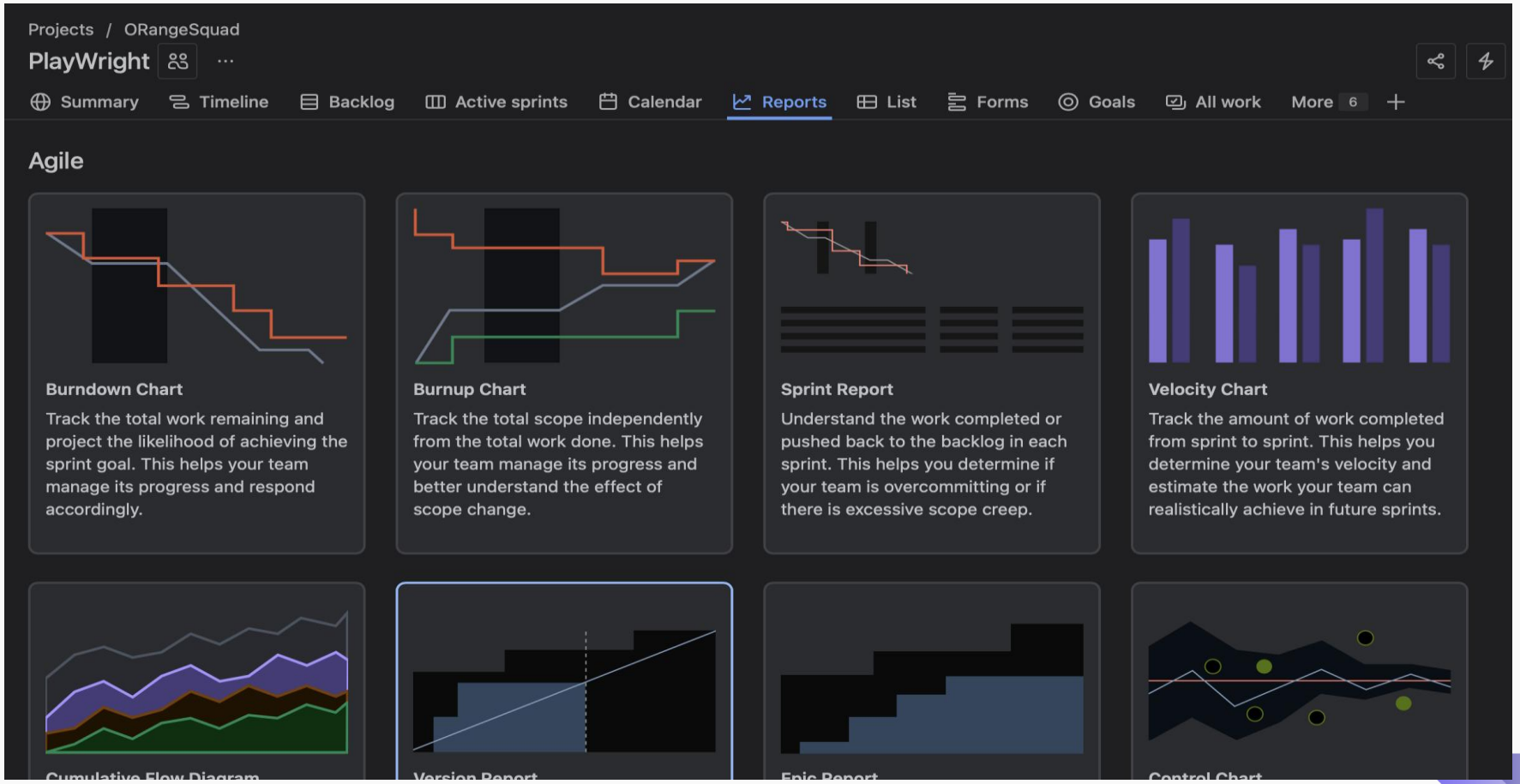
REPORTS

By Clicking on the Summary in dashboard you can check the over all process



REPORTS

Click the Reports from your dashboard

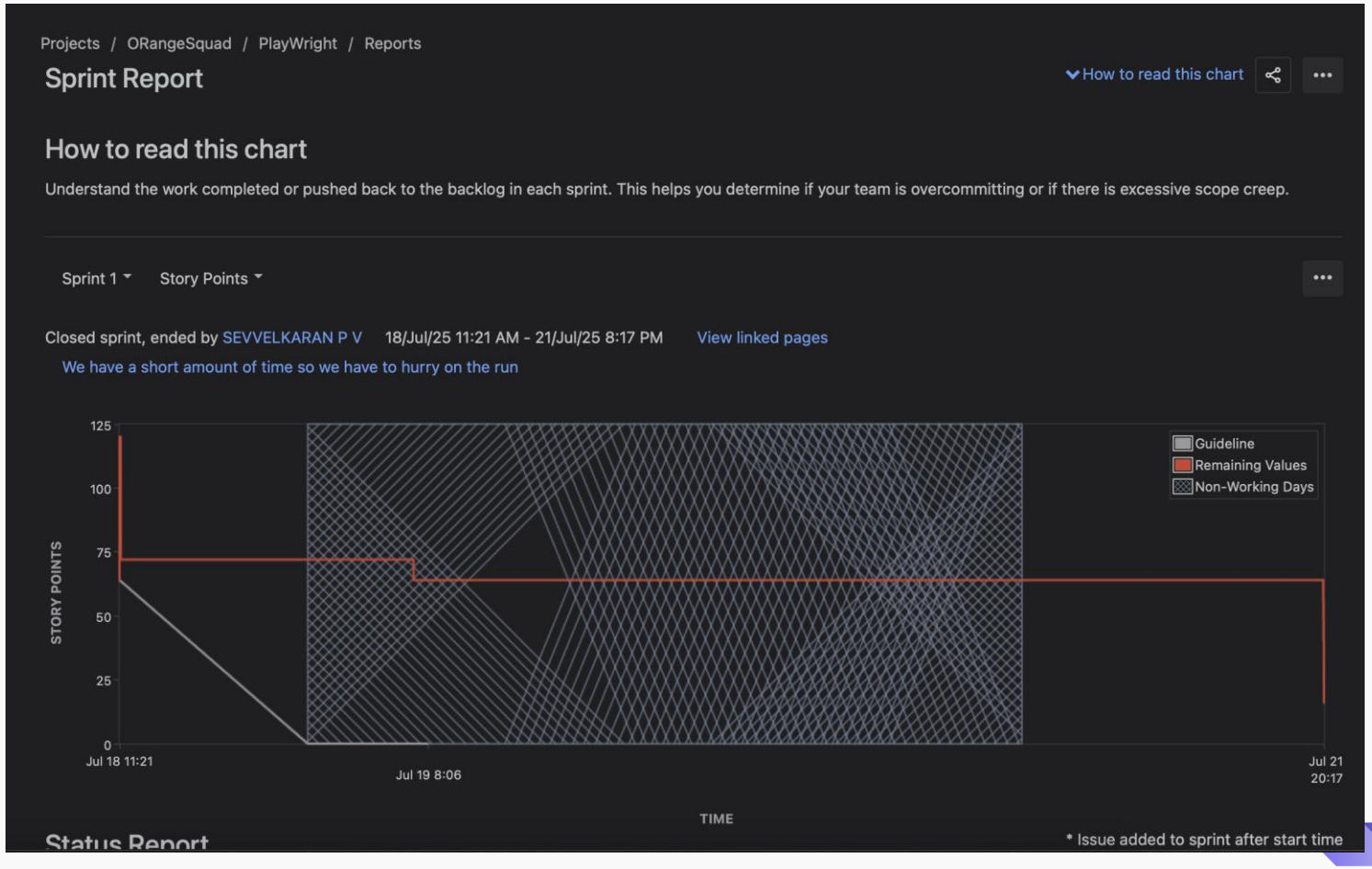


REPORTS

BurnUp Chart

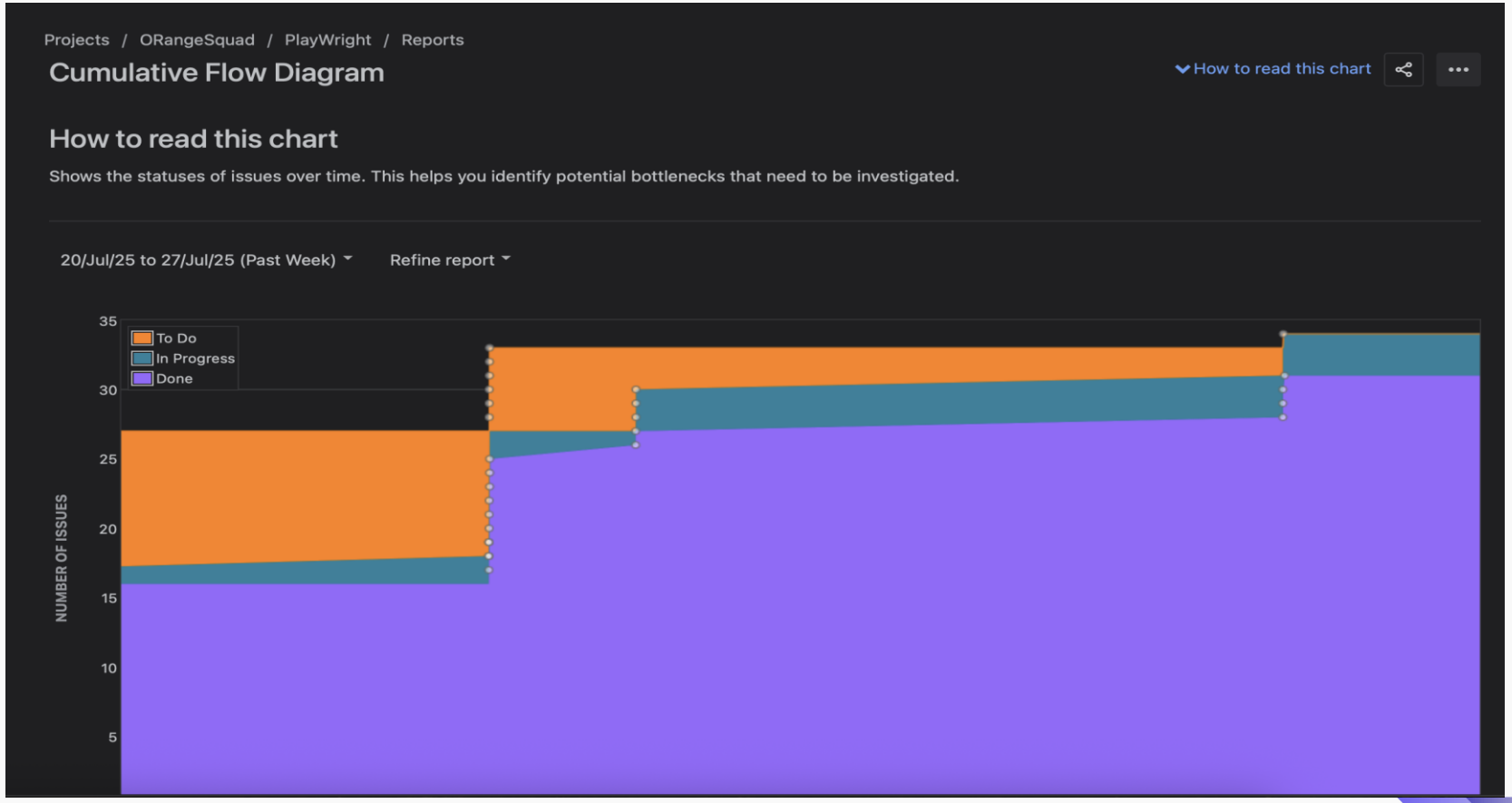
REPORTS

Sprint Report



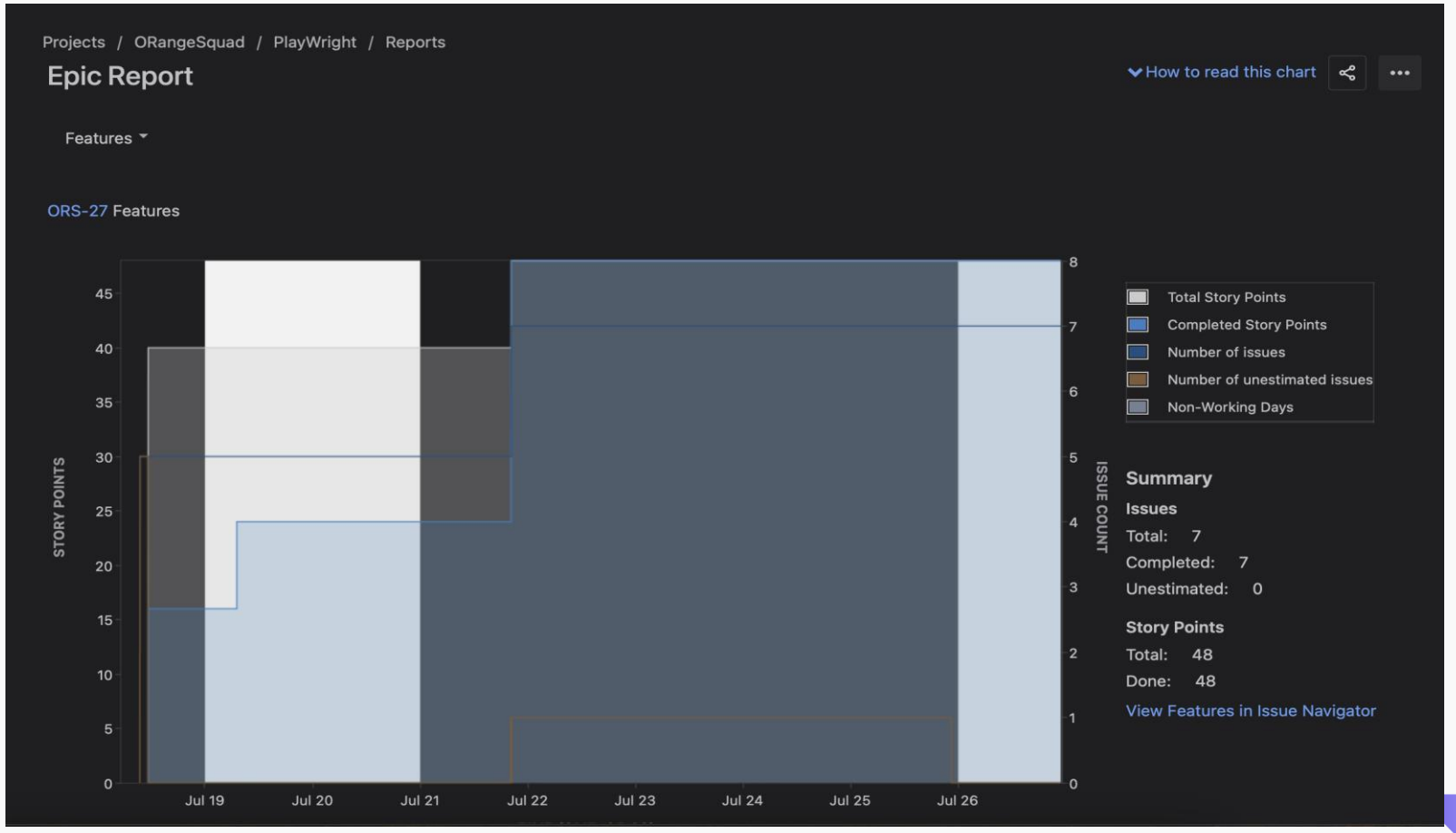
REPORTS

Cumulative Flow Diagram



REPORTS

Epic Report




THANK YOU
SmartCliff
Career Mobility Solutions