

Agile Project Management: Agile, Scrum, Kanban & XP

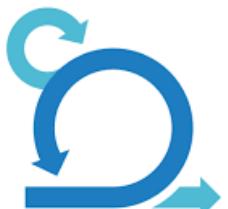
By GenMan Solutions



Project Management: Need for change



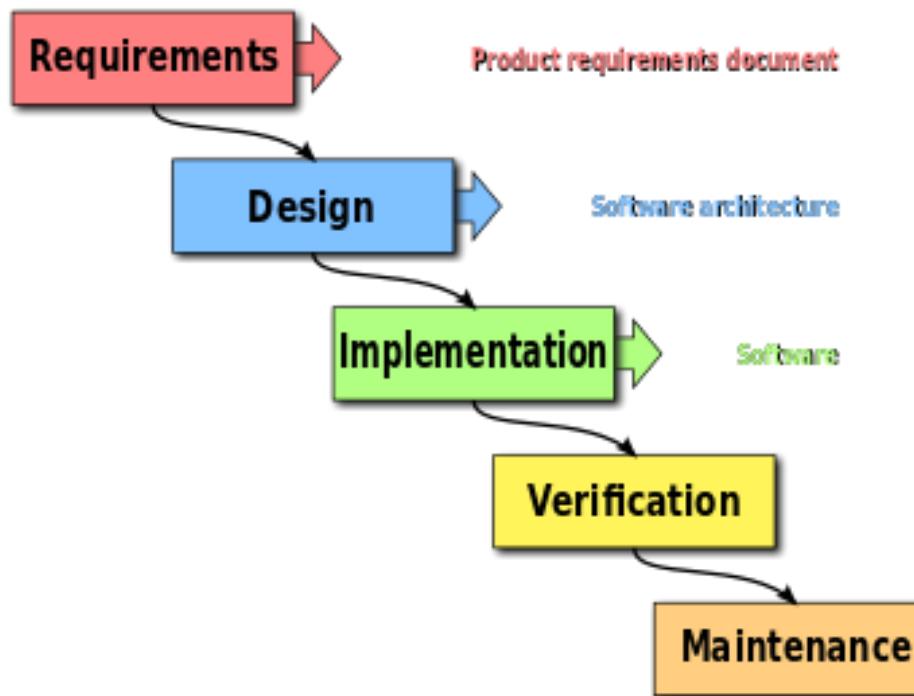
- Time bounded
- Requires definitive effort
- Needs to be well planned
- Must be completed within the assigned budget



Project Management: Need for change



Processes can't be left behind



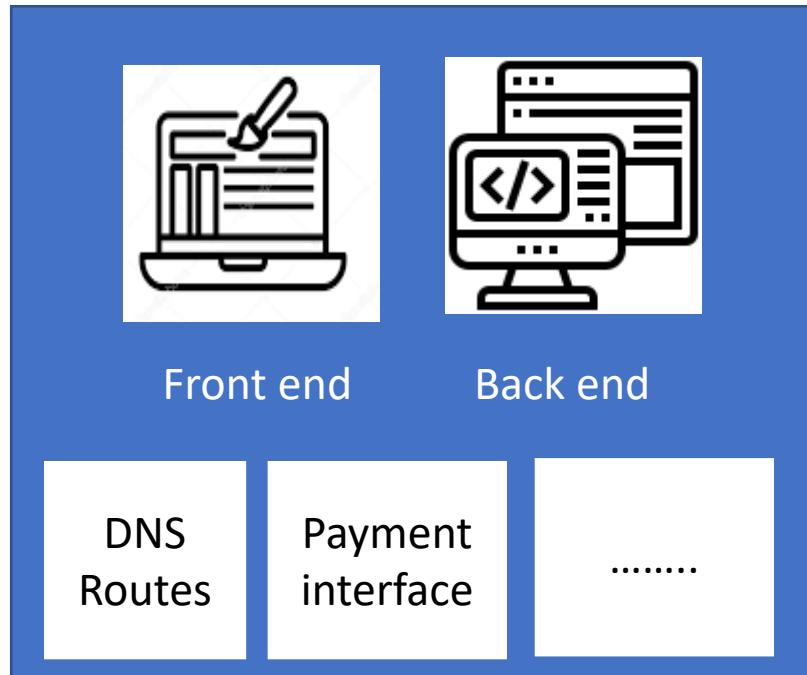
Waterfall Model



Project Management: Need for change

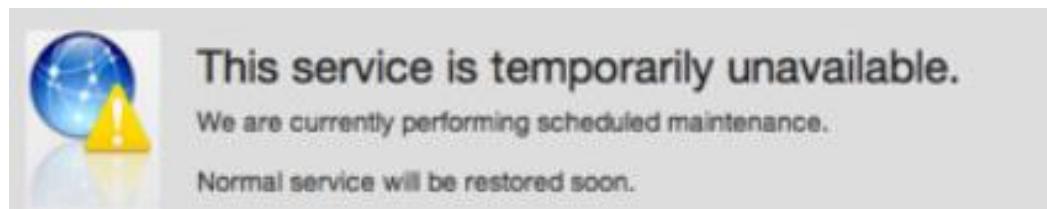


Flaws and weaknesses in historical approaches to project management and the requirements in the current period



Example: Any Mobile Application

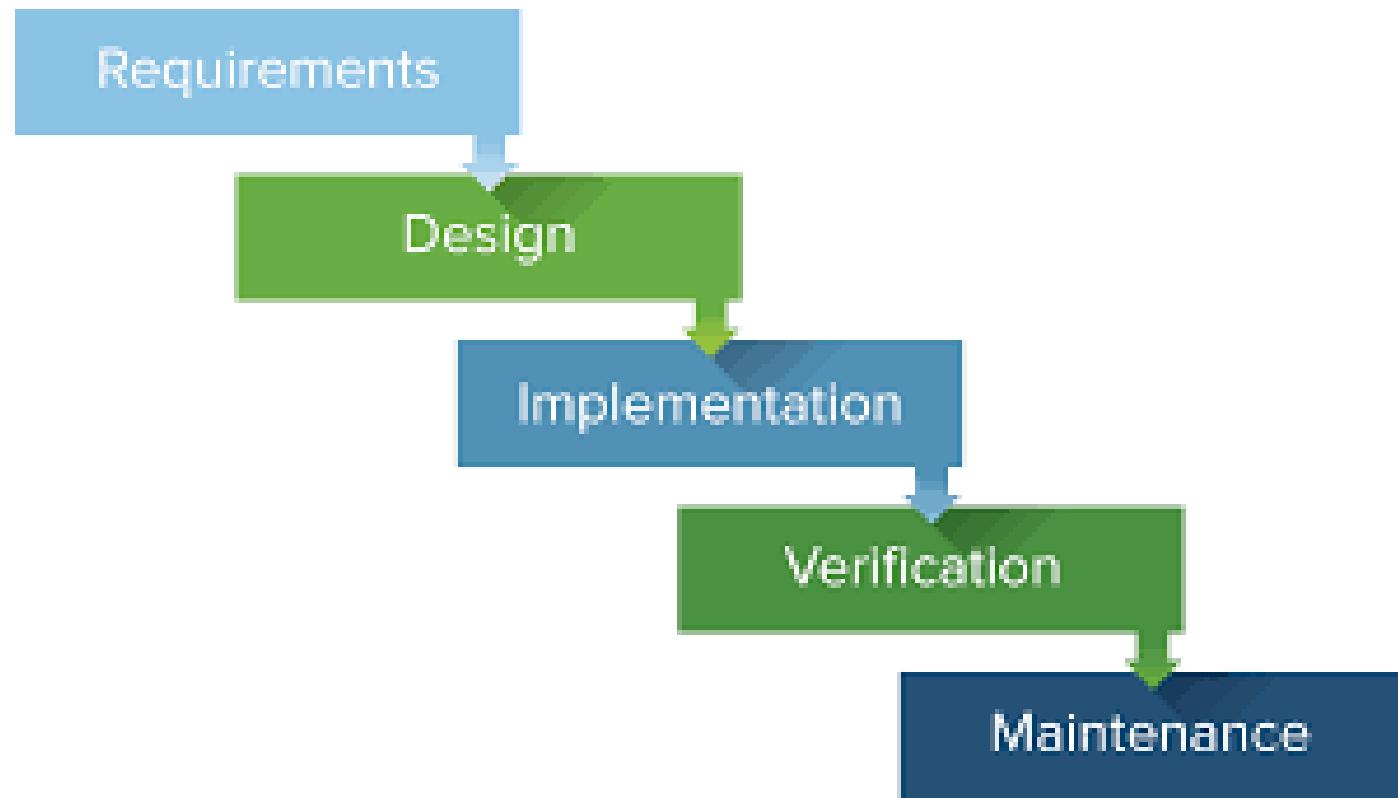
Time required to push changes



Project Management: Need for change



Waterfall Model



Project Management: Need for change



What is Agile?



What is Agile?

- Iterative approach to Software delivery
- Delivering faster in less time and less money
- Ability to create and quickly respond to change
- Philosophy to rapidly deploy an application
- Fully loaded tool-kit
- Just a mindset or way of thinking



Manifesto for Agile Software Development

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

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

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

Source: <https://agilemanifesto.org/>

Agile Manifesto & Principles



Agile Manifesto & Principles

Agile Manifesto

Agile Principles

Agile Alliance:
www.agilealliance.org



Manifesto for Agile Software Development

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

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

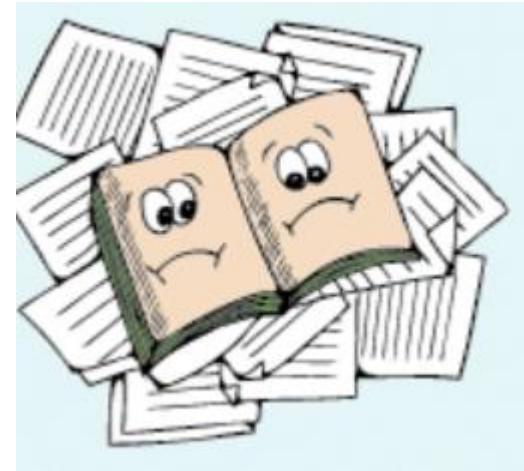
That is, while there is value in the items on the right, we value the items on the left more.

Source: <https://agilemanifesto.org/>



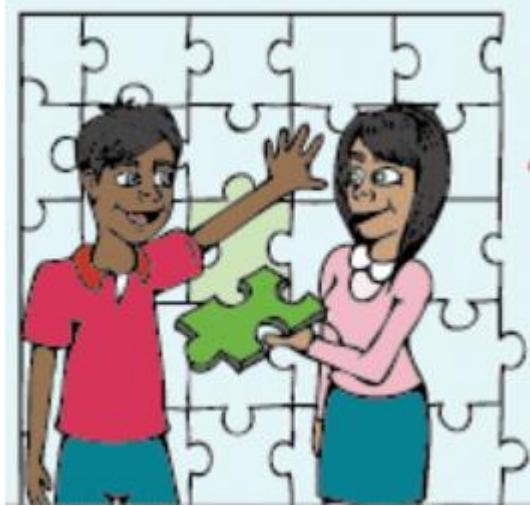
Individuals and interactions over processes and tools





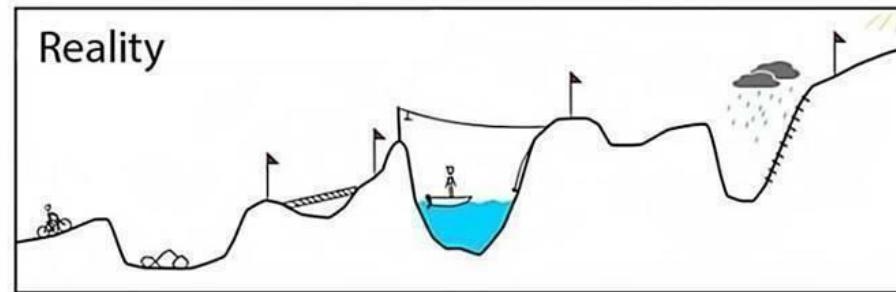
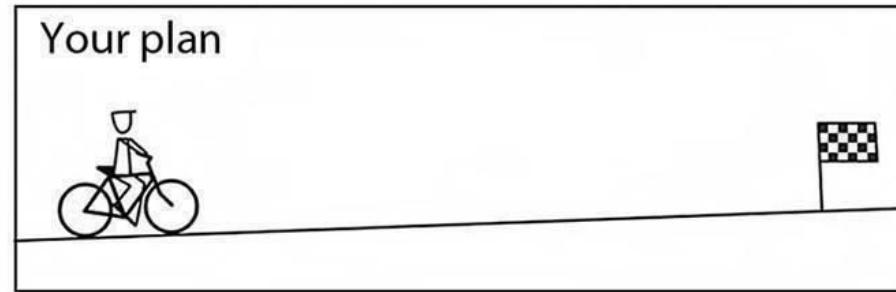
Working software over comprehensive documentation





Customer collaboration over contract negotiation





Responding to change over following a plan



Agile Principles



Customer Satisfaction



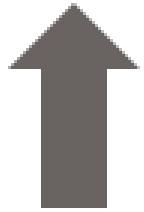
Welcome Change



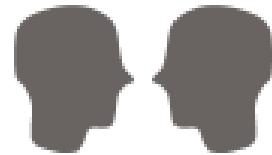
Deliver Frequently



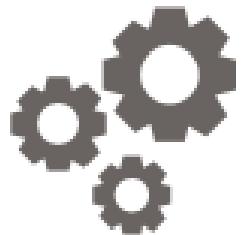
Working Together



Motivated Team



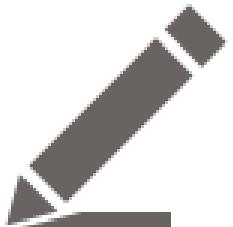
Face-to-Face



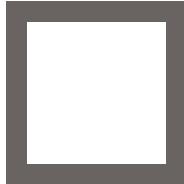
Working Software



Constant Pace



Good Design



Simplicity

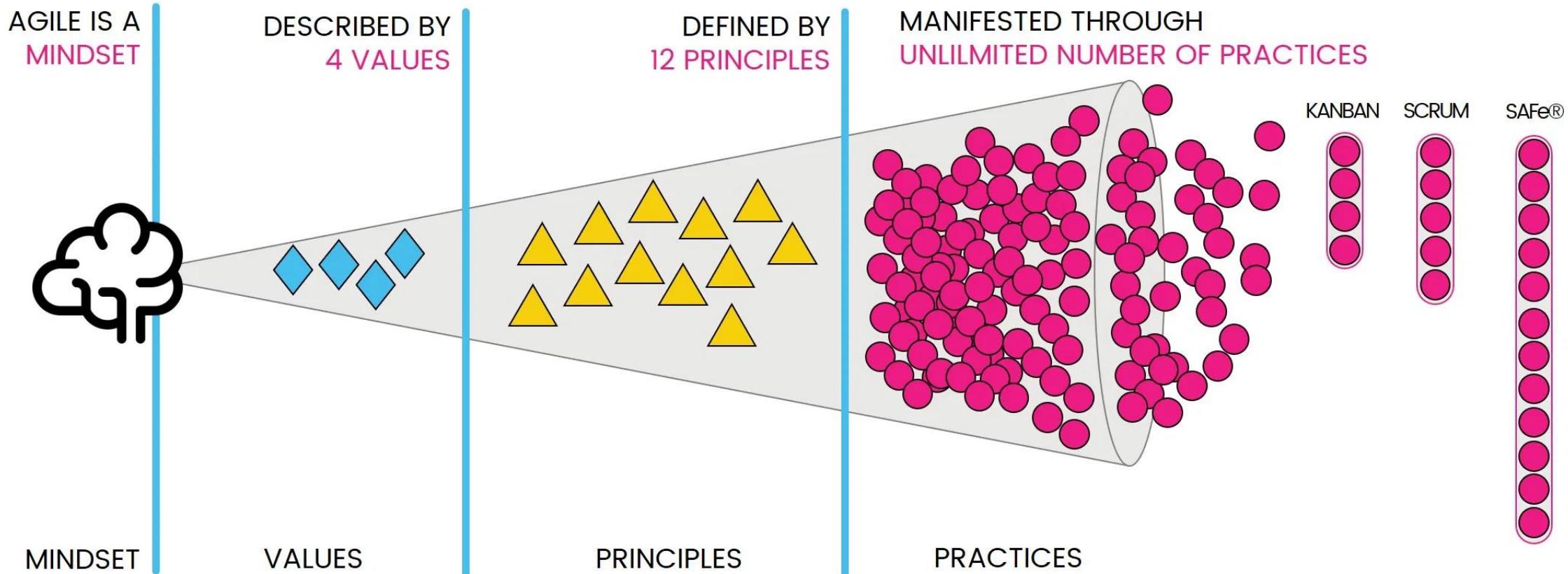


Self Organization



Reflect and Adjust

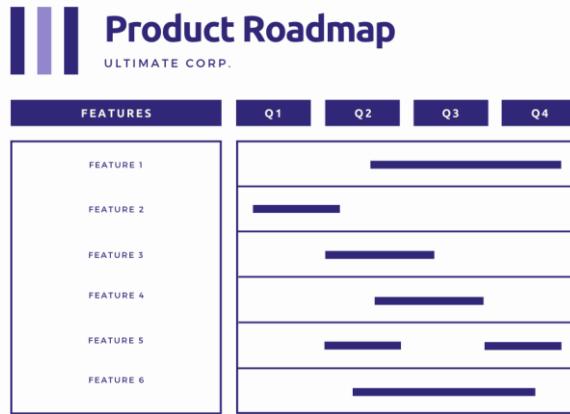
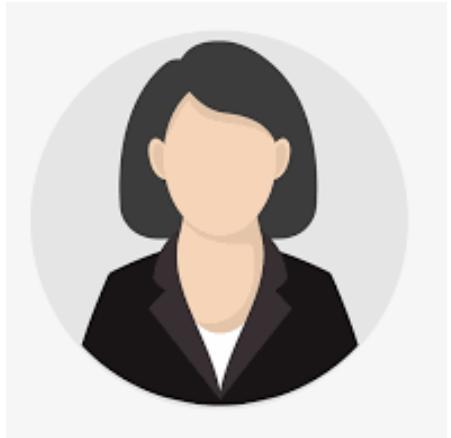




Adapted from Ahmed Sidky's Agile Mindset



Agile Principles



Customer Collaboration over
contract negotiation ?

Violating our principle that says the
developers should be working with the
business owners every day?



The Agile test

1. Does what we're doing at this moment support the early and continuous delivery of valuable software?
2. Does our process welcome and take advantage of change?
3. Does our process lead to and support the delivery of working functionality? Are the developers and the product owner working together daily?
4. Are customers and business stakeholders working closely with the project team?
5. Does our environment give the development team the support it needs to get the job done?
6. Are we communicating face to face more than through phone and email?
7. Are we measuring progress by the amount of working functionality produced?
8. Can we maintain this pace indefinitely?
9. Do we support technical excellence and good design that allows for future changes?
10. Are we maximizing the amount of work not done — namely, doing as little as necessary to fulfill the goal of the project?
11. Is this development team self-organizing and self-managing? Does it have the freedom to succeed?
12. Are we reflecting at regular intervals and adjusting our behavior accordingly?



What Agile is not?



Agile is not...

- A specific way of doing software development
- A framework, standard or a process
- An end-goal by itself
- An excuse to stop reducing documentation or an opportunity to eliminate planning
- One size fits all
- Scrum—Scrum follows Agile principles
- Kanban—Kanban applies Agile principles
- Limited to software developments



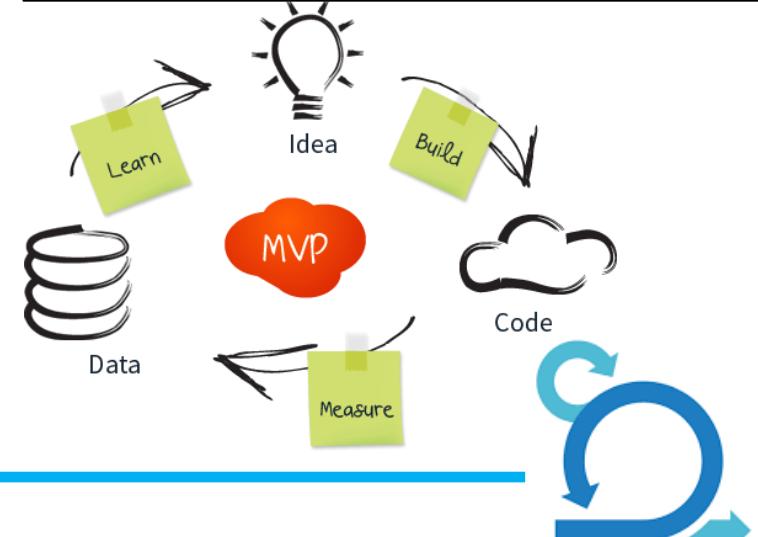
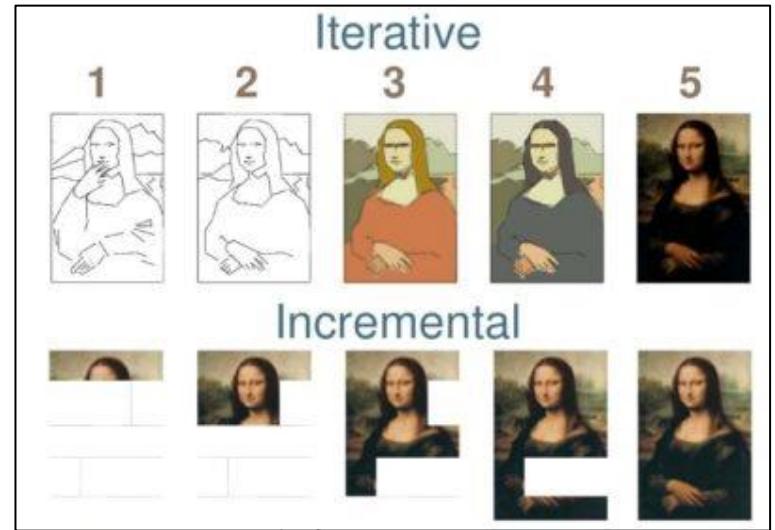
Agile Overview Summary

- Refers to any process that aligns with the concepts of Agile Manifesto
- Based on incremental delivery and iterative approach
- Encourages constant feedback from the end users
- Deliver value faster and foster the ability to better respond to market trends



Agile Overview Summary

- Refers to any process that aligns with the concepts of Agile Manifesto
- Based on incremental delivery and iterative approach
- Encourages constant feedback from the end users
- Deliver value faster and foster the ability to better respond to market trends



Agile Overview Summary

- Refers to any process that aligns with the concepts of Agile Manifesto
- Based on incremental, iterative approach
- Encourages constant feedback from the end users
- Deliver value faster and foster the ability to better respond to market trends



Agile Overview Summary

- Refers to any process that aligns with the concepts of Agile Manifesto
- Based on incremental, iterative approach
- Encourages constant feedback from the end users
- Deliver value faster and foster the ability to better respond to market trends



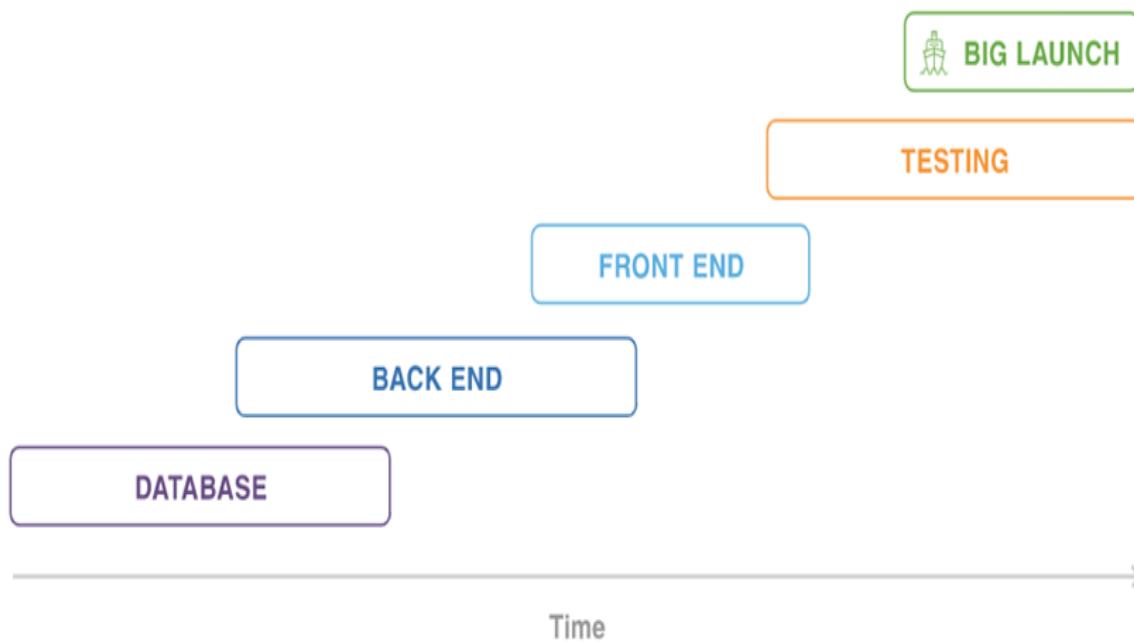
Differences and Similarities: Waterfall vs Agile



Differences and Similarities: Waterfall vs Agile



Waterfall



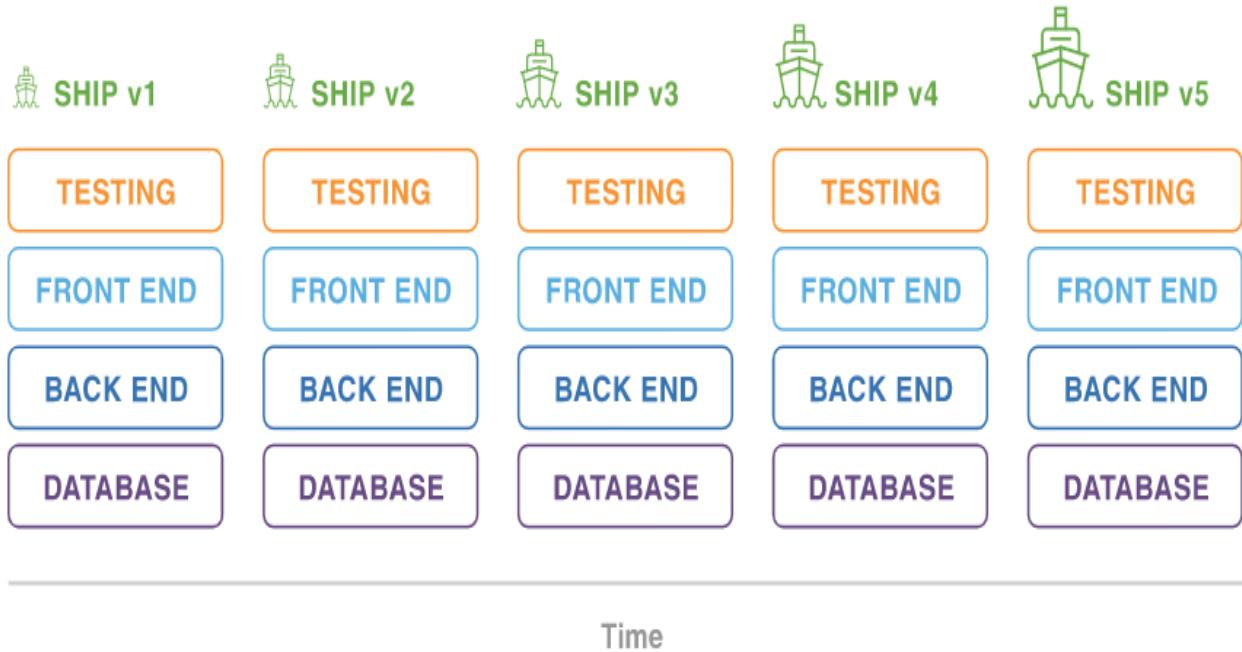
- Built in phases
- Painful to revisit previous phases
- High dependency on critical paths
- Customer can't interact with product until it's fully complete



Differences and Similarities: Waterfall vs Agile



Agile Project Management: Build, deliver, learn, and adjust



- Follows iterative approach
- Regular feedback intervals
- Iterations resolve blocking issue & let you interact with the product
- Quickly adapt new requirements/changes
- Shared skillset among the team



Differences and Similarities: Waterfall vs Agile



	Waterfall	Agile
Sequential	X	
Flexible		X
Accommodates change		X
Defined requirements	X	
Deliver quality products	X	X
Continually evolving		X
Rigid process	X	



Which one to choose: Waterfall/Agile



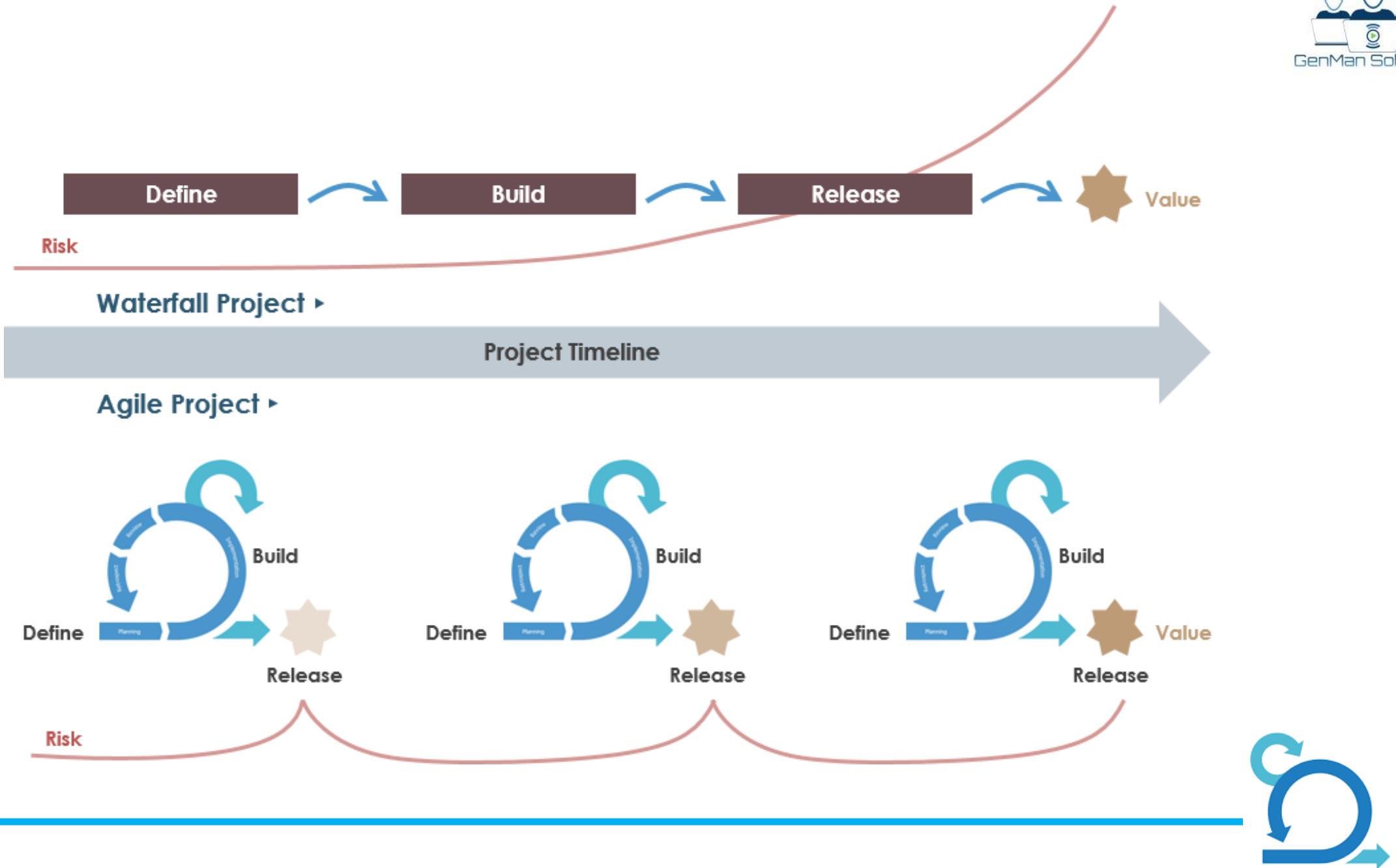
Use Waterfall if.....

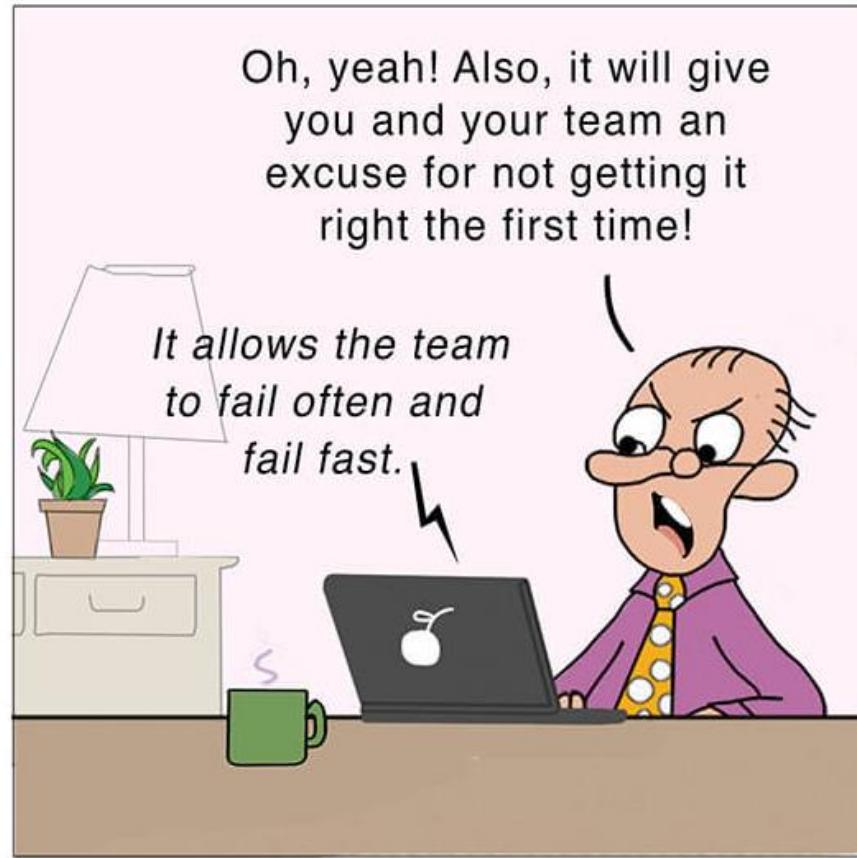
- You **don't expect changes in scope** and you're working with **fixed-price contracts**
- The project is very simple or **you've done it many times before**
- Requirements are **very well known and fixed**
- Customers know exactly what they want in advance
- You're working with **orderly and predictable projects**

Use Agile if.....

- The final product isn't clearly defined
- The clients/stakeholders **need the ability to modify the scope**
- You **anticipate any kind of changes** during the project
- **Rapid deployment** is the goal



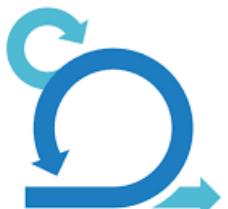




Failing fast is fine, but succeeding early is better. Disciplined Agile makes navigating a wealth of knowledge simple so you can identify solutions sooner. We call this guided continuous improvement.



Advantages & Disadvantages of Agile



Advantages of Agile

- Change is embraced
- End-goal can be unknown
- Faster, high-quality delivery
- Strong team interaction
- Customers are heard
- Continuous improvement



Disadvantages of Agile

- Planning can be less concrete
- Team must be knowledgeable
- Time commitment from developers is required
- Documentation can be neglected



Introduction to Key Agile Concepts

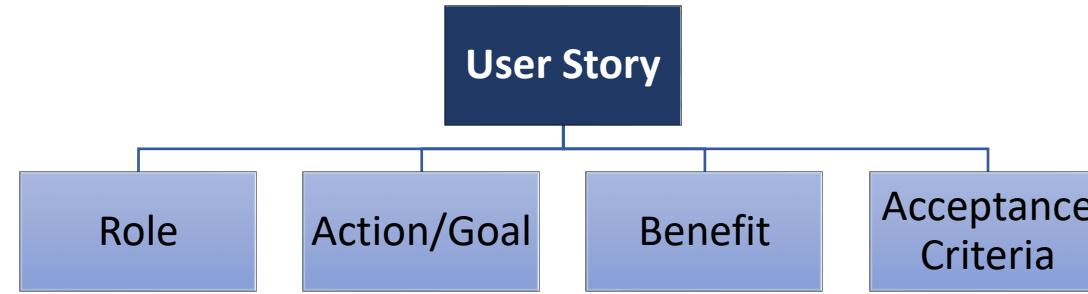


Introduction to Key Agile Concepts



User Story

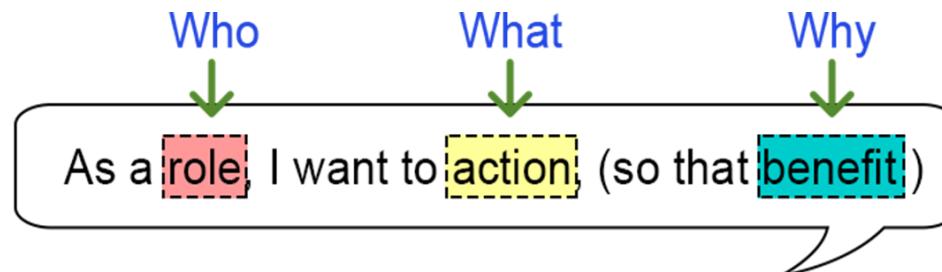
- Work divided into functional increments called “user stories”
- Short description of a feature from the perspective of the person who desires the new capability
- User story should be potentially shippable
- Size of a user story is measured in story points



Introduction to Key Agile Concepts



User Story



Role
The user should be an actual human who interacts with the system

- Be as specific as possible
- The development team is NOT a user

Action
The behavior of the system should be written as an action

- Usually unique for each User Story
- Describes intent & not the feature
- The "system" is implied and does not get written in the story
- Active voice, not passive voice ("I can be notified")

Benefit
The benefit should be a real-world result that is non-functional or external to the system

- Many stories may share the same benefit statement
- The benefit may be for other users or customers, not just for the user in the story.



Introduction to Key Agile Concepts



User Story

Acceptance Criteria

- Should be testable
- Should be clear and concise
- Everyone must understand
 - Should provide user perspective

Who is Responsible for Writing Acceptance Criteria?

When Should User Story Acceptance Criteria Be Written?

How Should You Format User Story Acceptance Criteria?



Introduction to Key Agile Concepts



User Story Examples

- As a [customer], I want [shopping cart feature] so that [I can easily purchase items online].
- As a [manager], I want to [generate a report] so that [I can understand which departments need more resources].
- As a [customer], I want to [receive an SMS when the item is arrived] so that [I can go pick it up right away].

As a [manager], I want to be able to [understand my colleagues progress], so I can [better report our success and failures].

A good user story should be: **INVEST**

- “I” ndependent (of all others)
- “N” egotiable (not a specific contract for features)
- “V” aluable
- “E” stimable (to a good approximation)
- “S” mall (so as to fit within an iteration)
- “T” estable (in principle, even if there isn’t a test for it yet)



Introduction to Key Agile Concepts



Example of a good User Story

As an [online visitor], I want to [add products in my shopping cart] so that [I can purchase multiple products at one go]

Acceptance Criteria [Abstract]

- Products can be added to the cart
- Products can be removed from the cart.
- Shopping cart will be empty initially
- Shopping cart will be empty after purchase
- Products can be added with multiple quantities in the cart
- Shopping cart will show the total product breakdown quantity and cost with grand total

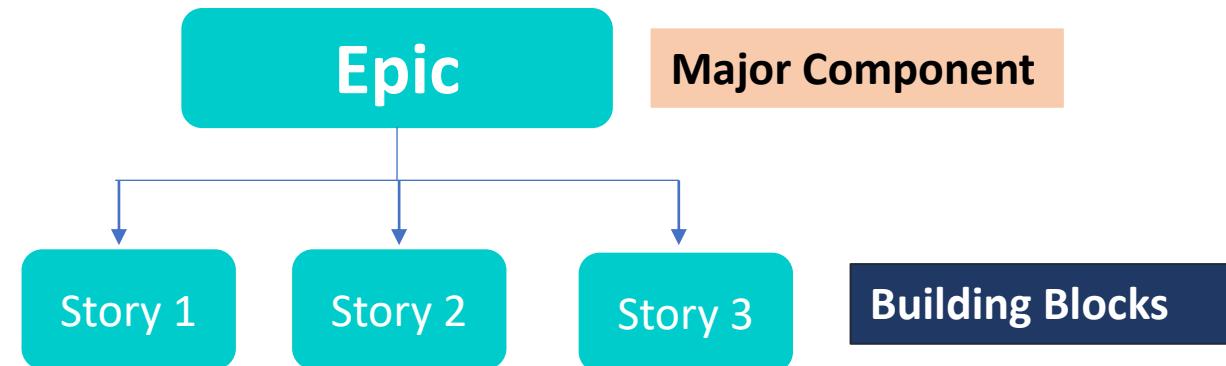


Introduction to Key Agile Concepts



Epic

- Series of user stories with a broader strategic objective
- A large user story that cannot be delivered as defined within a single iteration or is large enough that it can be split into smaller user tasks/user stories
- No standard format to represent epics



Introduction to Key Agile Concepts



Epic Example: Selecting Marketing Campaigns

1: As a VP Marketing, I want to review the performance of historical promotional campaigns so that I can identify and repeat profitable campaigns.

1a: As a VP Marketing, I want to select the timeframe to use when reviewing the performance of past promotional campaigns, so that ...

1b: As a VP Marketing, I can select which type of campaigns (direct mail, TV, email, radio, etc.) to include when reviewing the performance of past so that ...

1b1: As a VP Marketing, I want to see information on direct mailings when reviewing historical campaigns so that...

1b2: As a VP Marketing, I want to see information on TV ads when reviewing historical campaigns so that...

1b3: As a VP Marketing, I want to see information on email ads when reviewing historical campaigns so that...

and so on for each type of ad campaign.

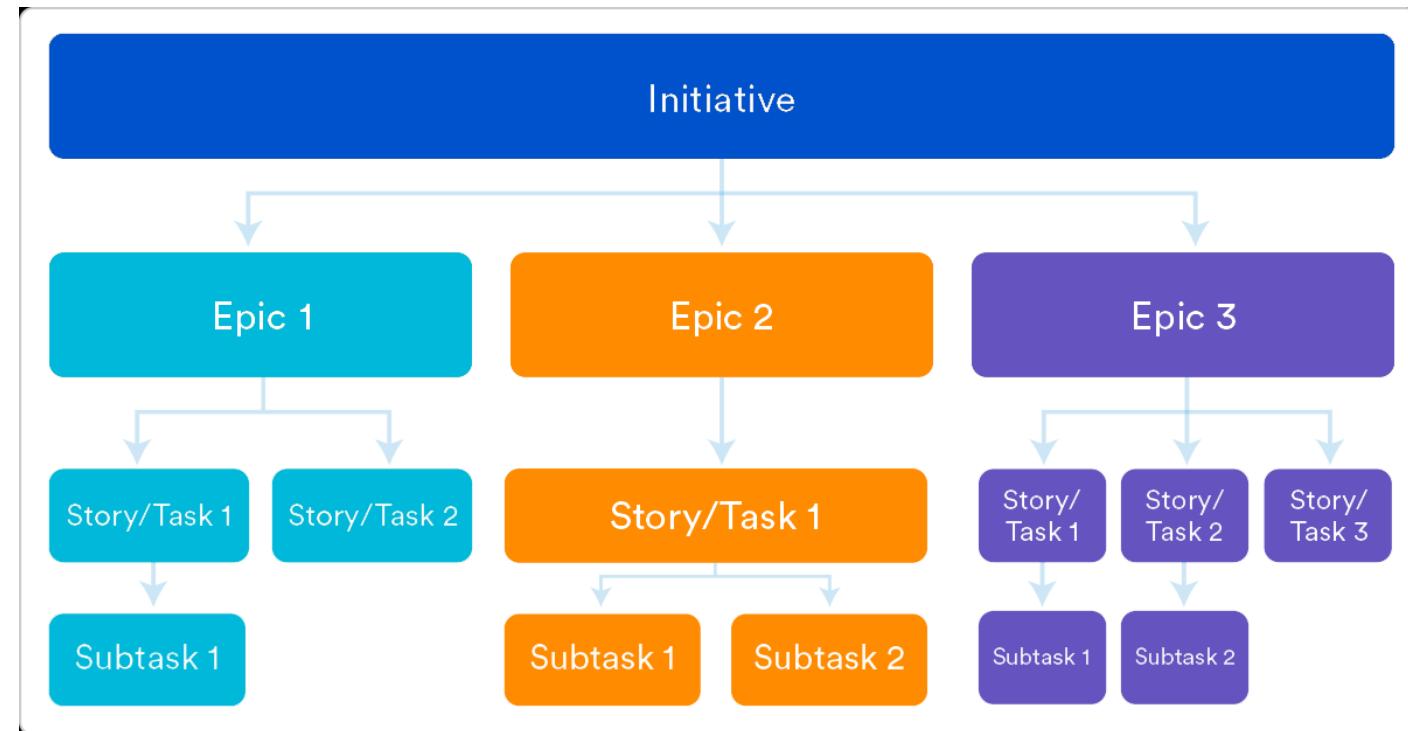


Introduction to Key Agile Concepts



Initiatives

- Initiatives are collections of epics that drive toward a common goal
- The product roadmap is expressed and visualized as a set of initiatives plotted along a timeline
- Completion of epics will lead to the completion of the initiative



Introduction to Key Agile Concepts



Themes

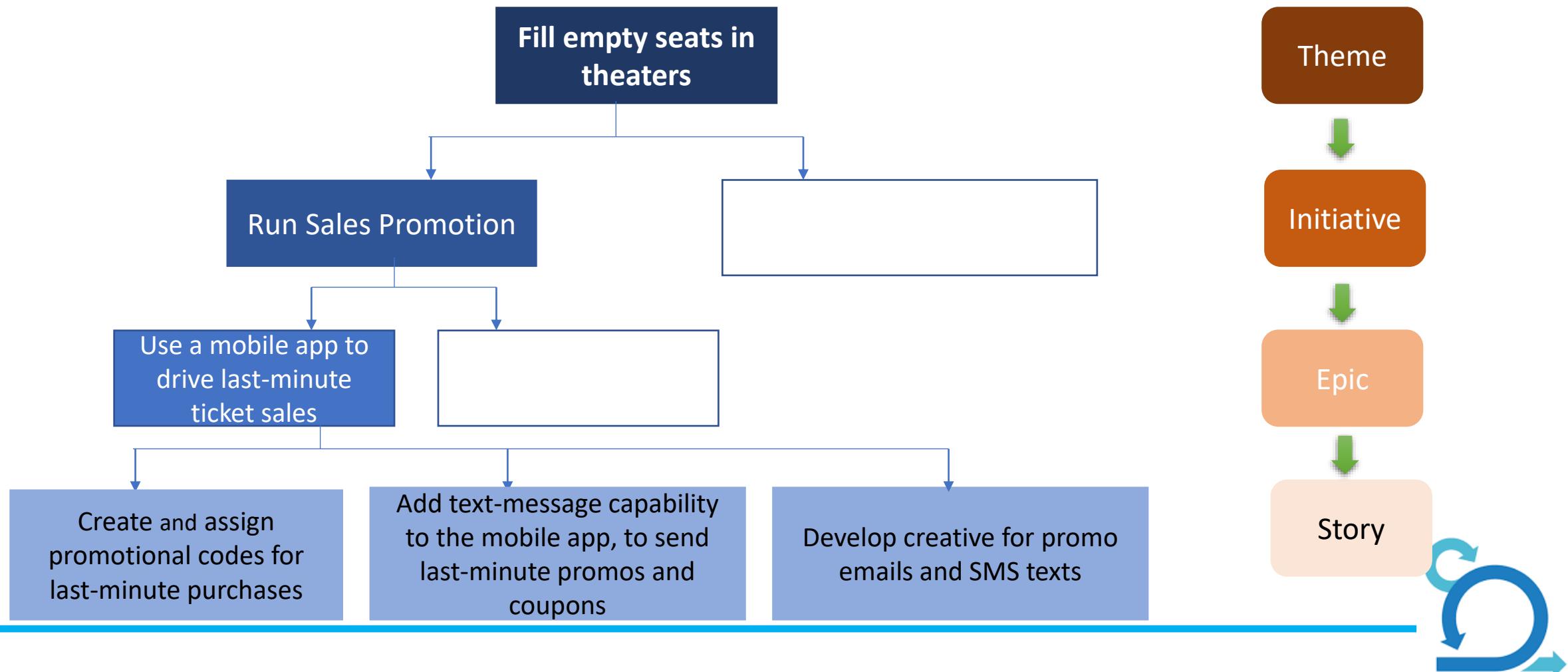
- Themes are large focus areas that span the organization
- A theme is an organization goal that drive the creation of epics and initiatives



Introduction to Key Agile Concepts



Theme, Initiatives, Epic & Story



Introduction to Key Agile Concepts



Benefits of Using the Theme-Initiative-Epic-Story Development Framework

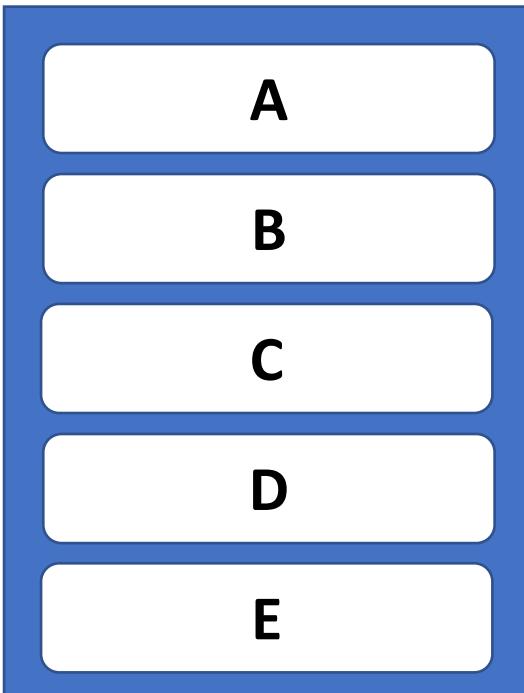
- It allows for more strategically sound decisions
- It improves performance monitoring and timeline estimates
- It keeps the team focused on key goals



Introduction to Key Agile Concepts



Product Backlog



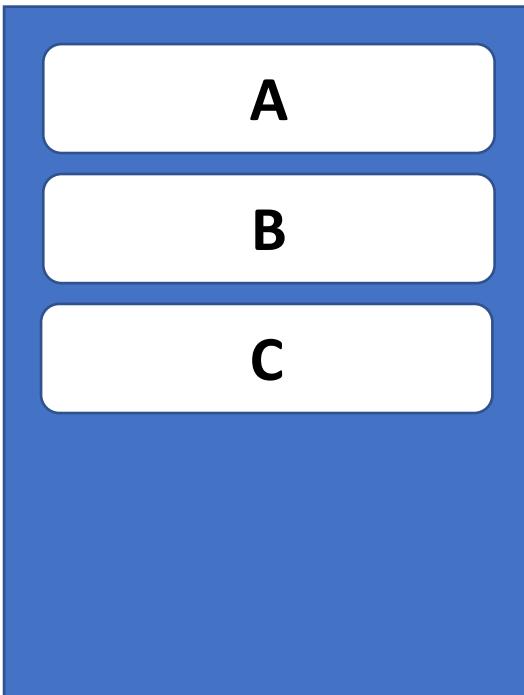
- Lists & Prioritizes the task-level details to execute a plan- To do list!
- Includes user stories, new features, changes to existing functionalities, bug fixes, and other tasks like infrastructure changes etc.
- Single source of requirements that defines the product
- Contains short description of functionality desired in the product
- Can be represented in physical form or in electronic form
- Owned & maintained by the product owner
- User Stories is the most common format
- Dynamic in nature: addition, deletion and reordering of requirements possible



Introduction to Key Agile Concepts



Product Backlog



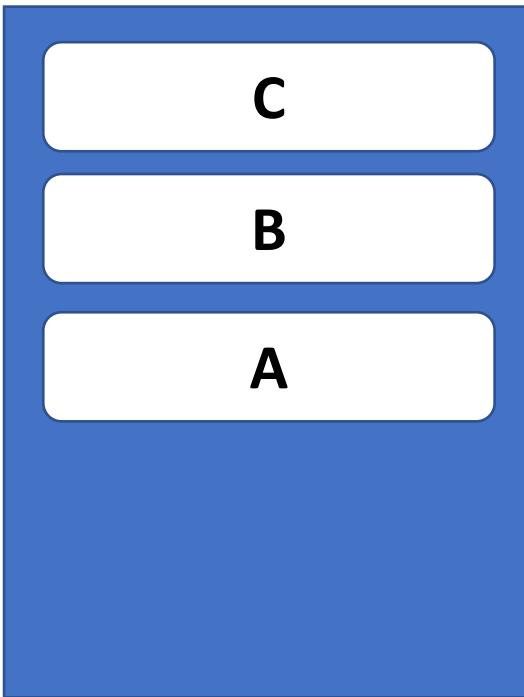
- Lists & Prioritizes the task-level details to execute a plan- To do list!
- Includes user stories, new features, changes to existing functionalities, bug fixes, and other tasks like infrastructure changes etc.
- Single source of requirements that defines the product
- Contains short description of functionality desired in the product
- Can be represented in physical form or in electronic form
- Owned & maintained by the product owner
- User Stories is the most common format
- Dynamic in nature: addition, deletion and reordering of requirements possible



Introduction to Key Agile Concepts



Product Backlog



- Lists & Prioritizes the task-level details to execute a plan- To do list!
- Includes user stories, new features, changes to existing functionalities, bug fixes, and other tasks like infrastructure changes etc.
- Single source of requirements that defines the product
- Contains short description of functionality desired in the product
- Can be represented in physical form or in electronic form
- Owned & maintained by the product owner
- User Stories is the most common format
- Dynamic in nature: addition, deletion and reordering of requirements possible



Introduction to Key Agile Concepts



Difference Between a Product Backlog and Product Roadmap



Tactical Backlog

A screenshot of a Jira interface showing a backlog of issues for a team. The backlog is organized into sections such as "Sprint 4", "Tech Debt", "Product", "UI", "Server", and "Recently Updated". Each section contains a list of issues with their descriptions, due dates, and assignees.

From Product Vision to Backlog



Introduction to Key Agile Concepts



Difference Between a Product Backlog and Product Roadmap

	Product Roadmap	Product Backlog
Content	High-level: themes and epics or outcomes and goals	Task-level: user stories and defects
Audience	Executive team (and other stakeholders)	Primary for product and development teams
Intent	Convey a strategy	Conveys tactical steps in execution of plan
Time Frame	Varies, typically ~3 Months	1 or 2 sprints

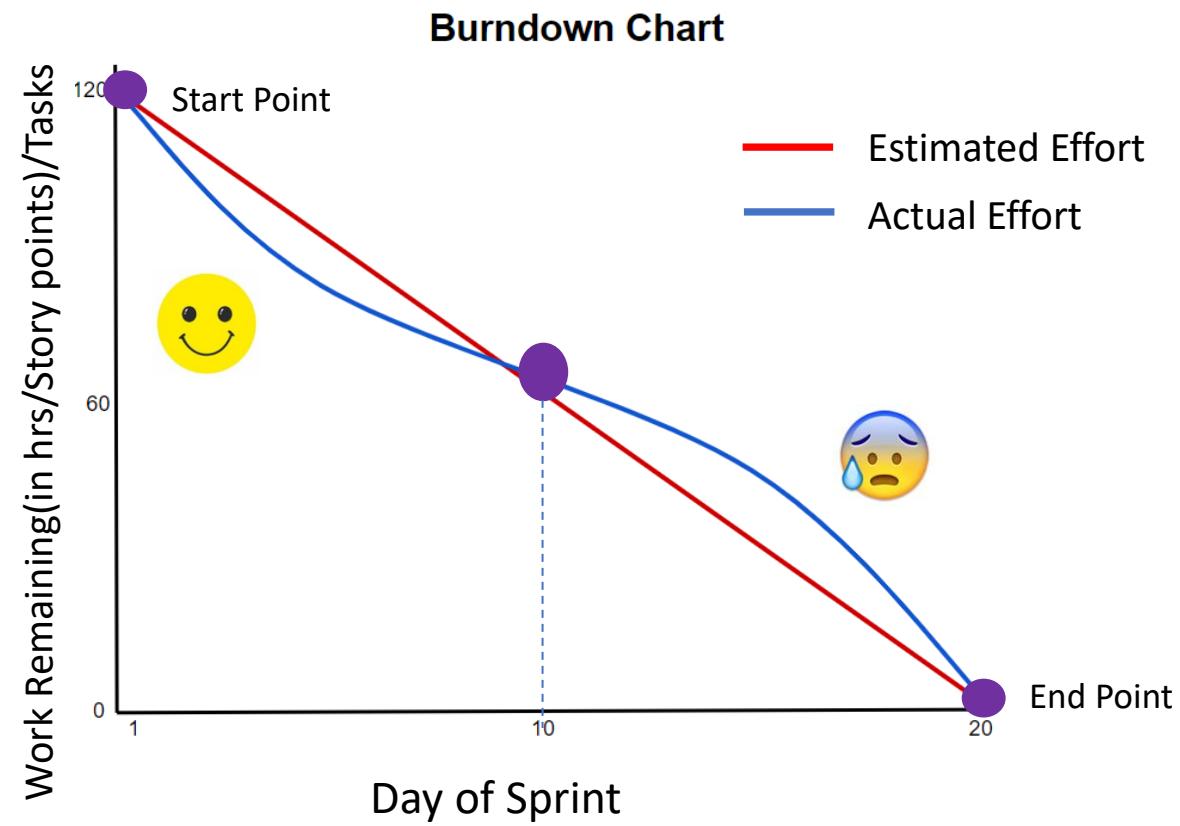


Introduction to Key Agile Concepts



Burn down Chart

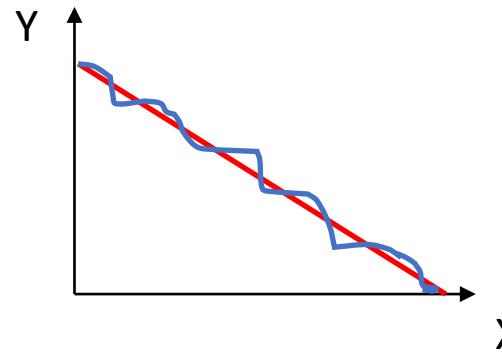
A burndown chart is a visual display of work completed and remaining in a project, sprint, or iteration



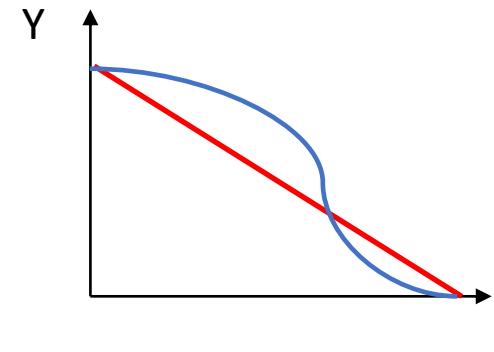
Introduction to Key Agile Concepts



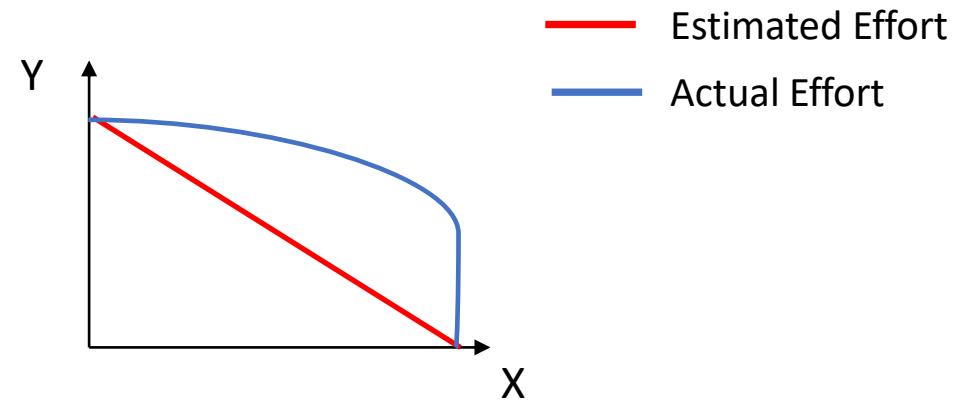
Burn down Chart



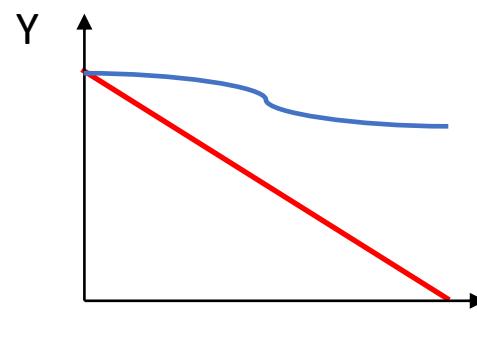
1. Ideal Scenario



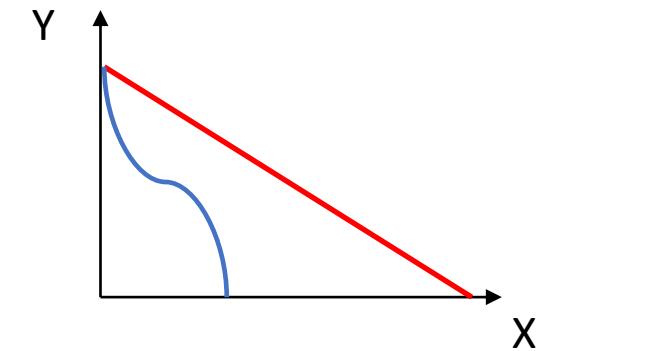
2. Still, a good scenario



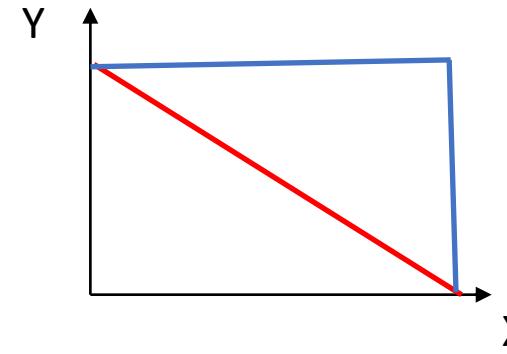
3. Decent scenario



4. Missed the deadline



5. There's something wrong



6. Team forgot about it



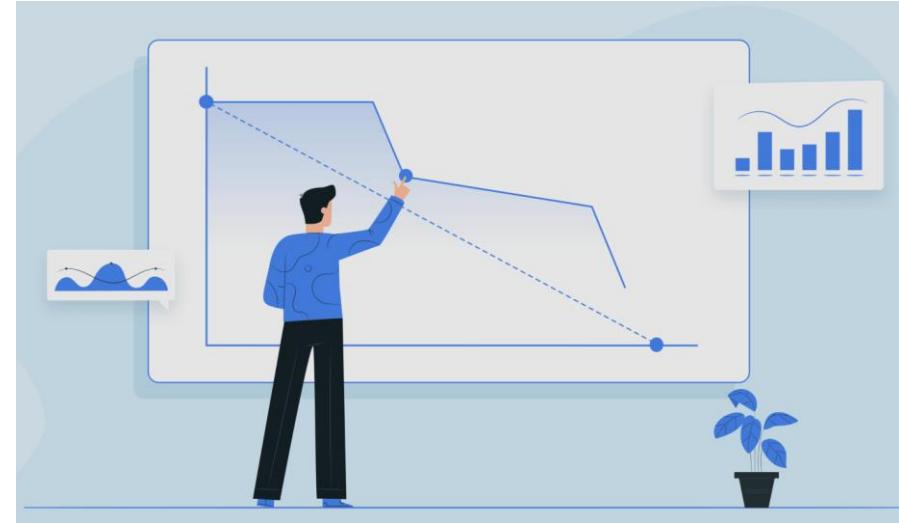
Introduction to Key Agile Concepts



Burn down Chart

Benefits

- Simple & easy way to track team's progress
- Timely prevention of issues
- Helps keep the team motivated



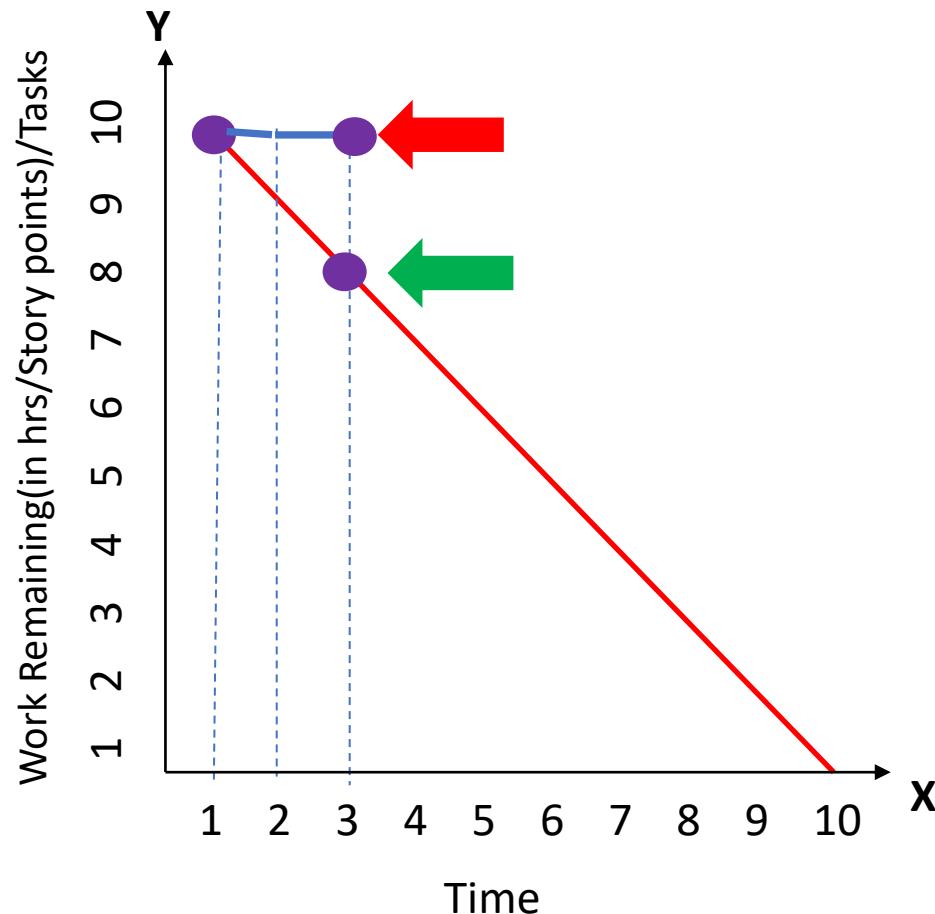
Introduction to Key Agile Concepts



Burn down Chart

Creating Burn down Chart

Mike, a project manager is asked to create a burndown chart for a project by his leadership to know how things progressing based on what was planned.



Project has 10 tasks to complete in 10 days



Introduction to Key Agile Concepts



Minimum Viable Product (MVP)

It is the version of a new product that allows a team to collect the maximum amount of validated learning about customers with the least amount of effort.

- Agile Alliance

Why MVP

- Release a product to the market as quickly as possible
- Test an idea with real users before committing a large budget to the product's full development
- Learn what resonates with the company's target market and what doesn't



Introduction to Key Agile Concepts

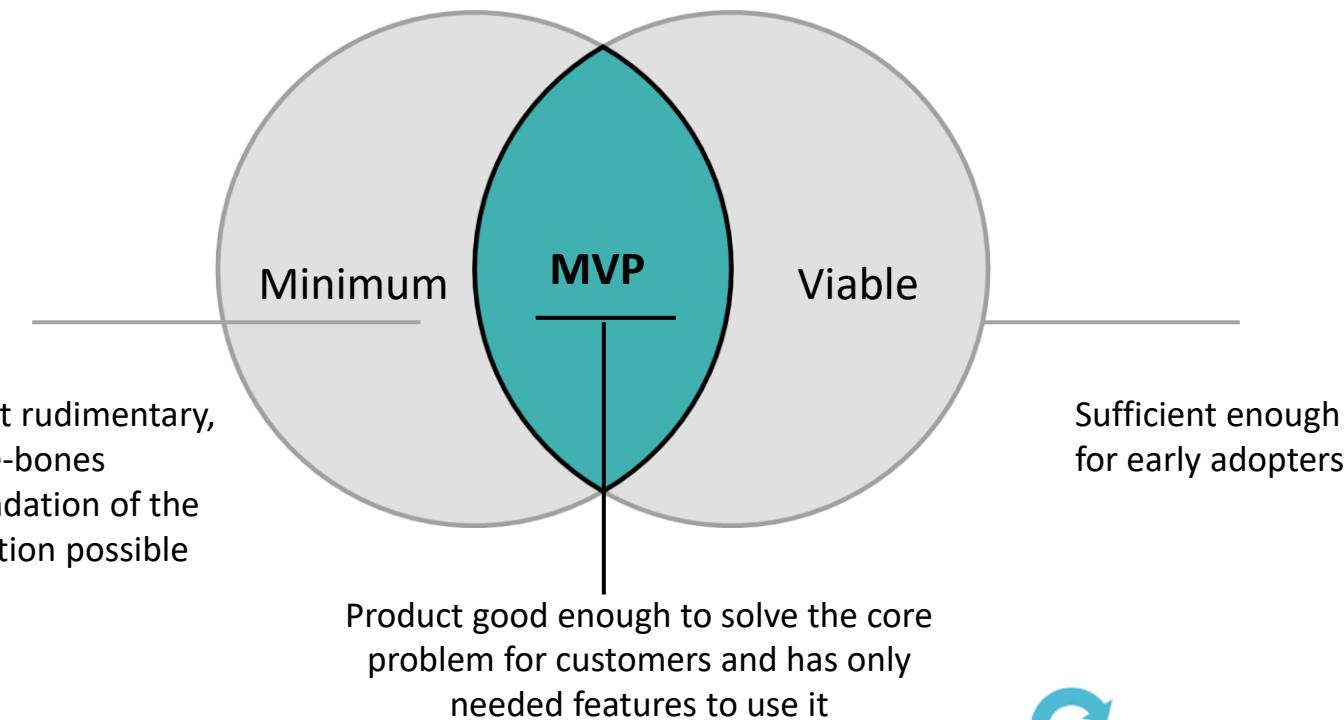


Minimum Viable Product (MVP)

It is the version of a new product that allows a team to collect the maximum amount of validated learning about customers with the least amount of effort.

- Agile Alliance

Minimum + Viable Product



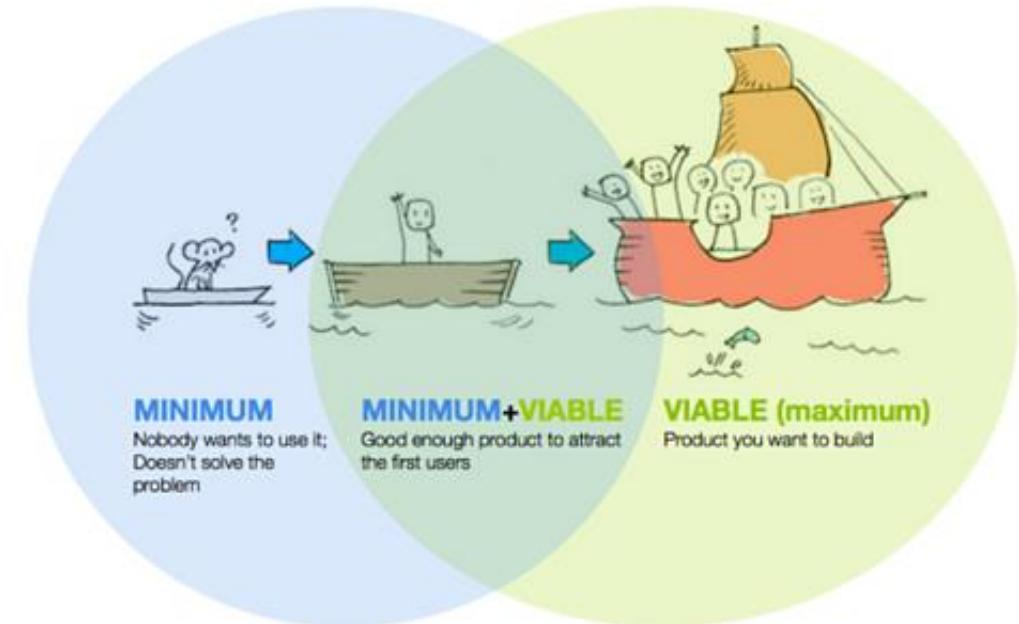
Introduction to Key Agile Concepts



Minimum Viable Product (MVP)

Why MVP?

- Release a product to the market as quickly as possible
- Test an idea with real users before committing a large budget to the product's full development
- Learn what resonates with the company's target market and what doesn't



Introduction to Key Agile Concepts



Minimum Viable Product (MVP)

Use case:

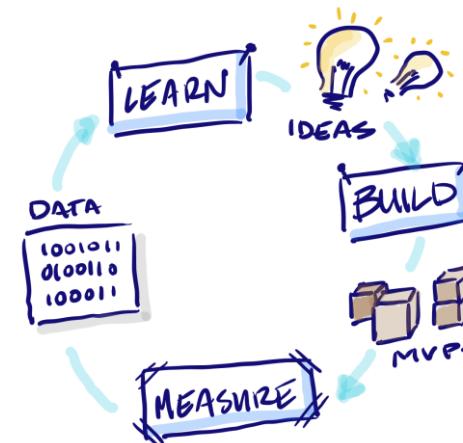
Your target audience needs a specific means of transport, but they are not sure if they want to buy a premium product right away.



What do you do then?

Create the most basic version of the Product

Gather feedback on your product



Introduction to Key Agile Concepts



Minimum Viable Product (MVP)

— HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT —



1



2



3



4

1

— ALSO HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT —



1



2



3



4

2

— HOW TO BUILD A MINIMUM VIABLE PRODUCT —



1



2



3



4

3

image source: <http://www.expressiveproductdesign.com/>



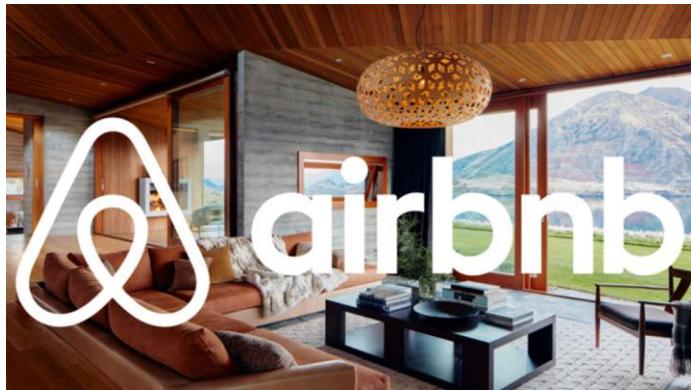
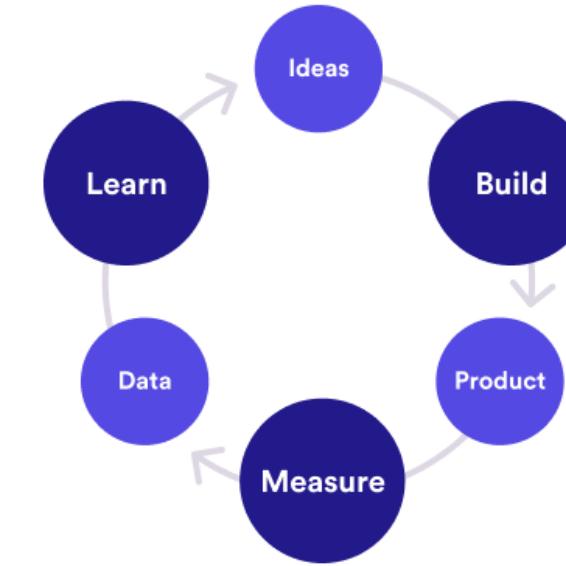
Introduction to Key Agile Concepts



Minimum Viable Product (MVP)

Important points to be noted:

- Make sure your planned MVP aligns with your business objectives
- Identify specific problems you want to solve for your users
- Product must be viable
- MVP is based on iterative process of building



Introduction to Key Agile Concepts



Velocity

At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity.

- Agile Alliance

Velocity = Units of work completed in a given timeframe



Unit of work can be hours
or user stories or story
points

Typically measured in
iterations or sprints, or
weeks



Introduction to Key Agile Concepts



Velocity

Sprint 1

User Stories	Story Points	Status
A	3	Complete
B	5	Incomplete
C	8	Complete

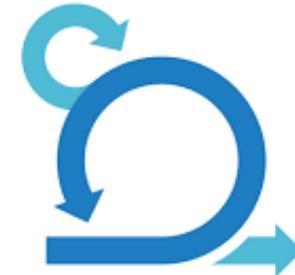
Velocity = 3 + 8 = 11 Story Points/ Sprint

Sprint 2



Velocity = 13 Story Points/ Sprint

Sprint 3



Velocity = 6 Story Points/ Sprint

Average velocity: $(11+13+6)/3= 10$ Story Points/Sprint

For the next sprint, the product manager should pick up User Stories equivalent or not more than 10 story points



Introduction to Key Agile Concepts



Velocity

Total Story points against remaining User Story (A) : 60

Average Velocity (B) : 10 Story Points/ Sprint

Forecast for the remaining effort for the Project: A/B = 60/10
= 6 Sprints with each sprint of 10 story points



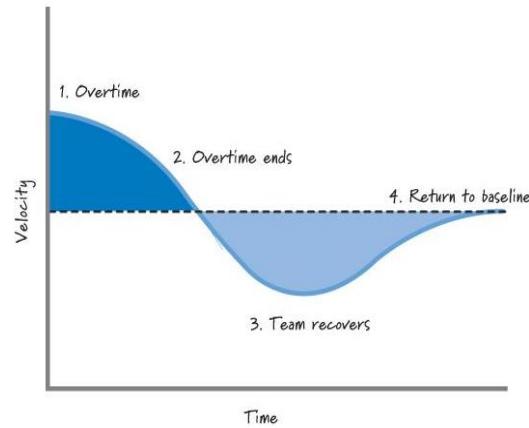
Introduction to Key Agile Concepts



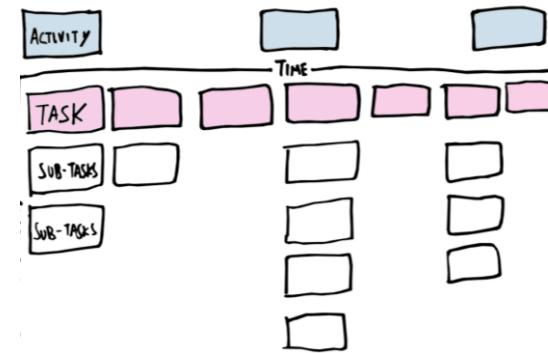
Velocity



Failing to bring a story to completion



Velocity see-sawing



Decomposition of User Stories



Accurate Estimation



Introduction to Key Agile Concepts



Estimation

38+19 is about 60

For reasonable guess, you estimate basis
what you know or see



Introduction to Key Agile Concepts



Agile Estimation

Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration



Introduction to Key Agile Concepts



Agile Estimation

Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration



Introduction to Key Agile Concepts



Agile Estimation

Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration



Introduction to Key Agile Concepts



Agile Estimation

Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration



Introduction to Key Agile Concepts



Agile Estimation

Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration

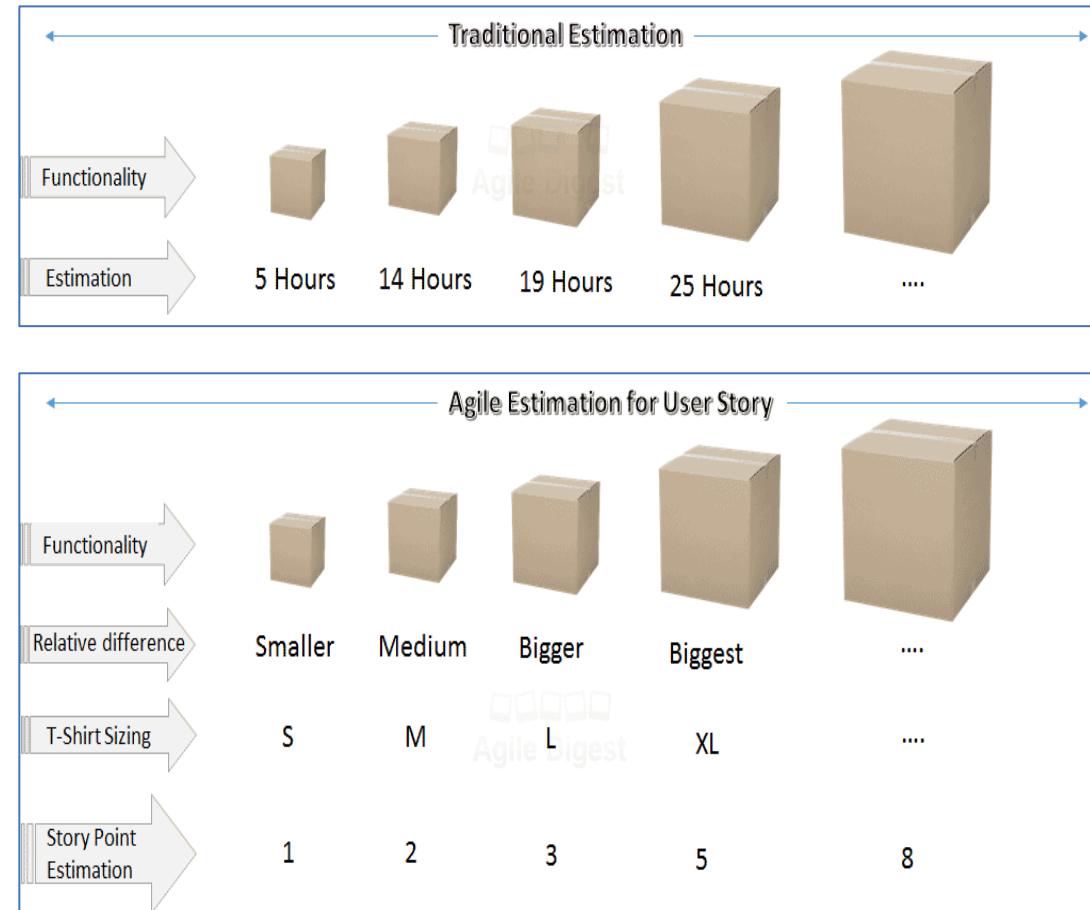


Introduction to Key Agile Concepts



Agile Estimation

Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration



Introduction to Key Agile Concepts



Agile Estimation



Story Points

3 Story Points: Study Room

5 Story Points: Bedroom 1
and Bedroom 2

8 Story Points: Kitchen



Introduction to Key Agile Concepts



Agile Estimation

Influencing Factors of Story Point :

- Business Value
- Complexity
- Risks
- Dependencies
- Amount of work



Image Source: Dilbert.com

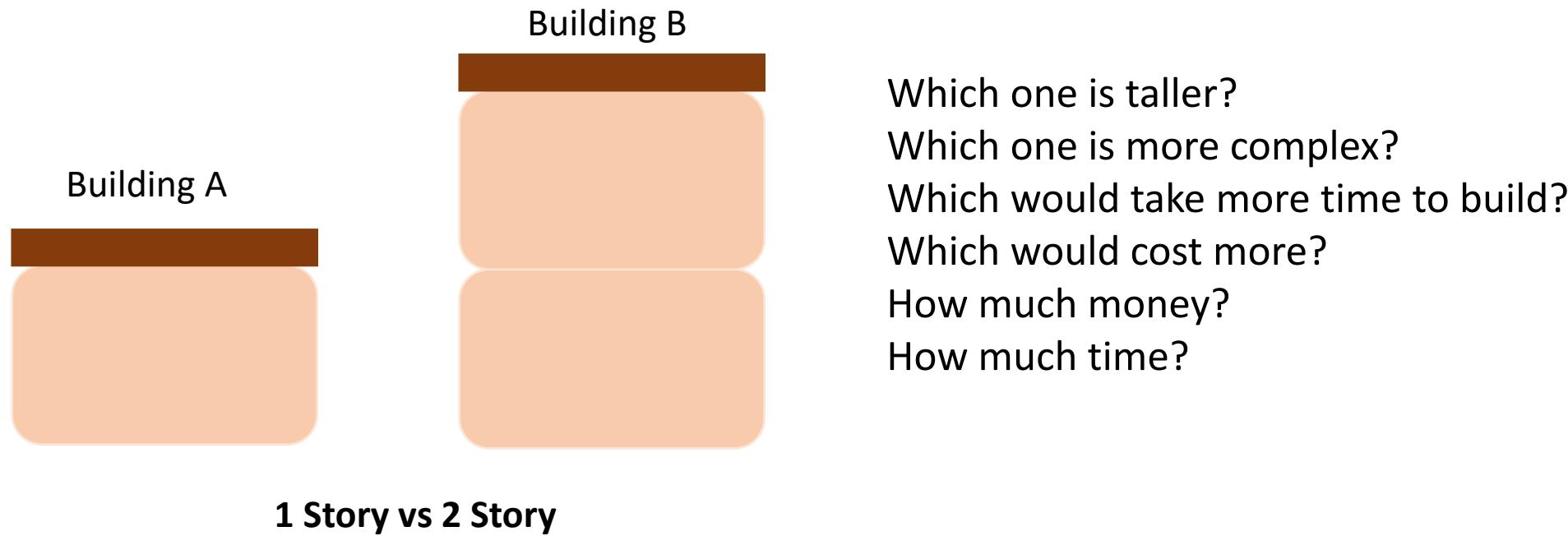


Introduction to Key Agile Concepts



Agile Estimation: But why relative estimation?

Illustration 1



Introduction to Key Agile Concepts



Agile Estimation: But why relative estimation?

Illustration 2



100 Story vs 101 Story

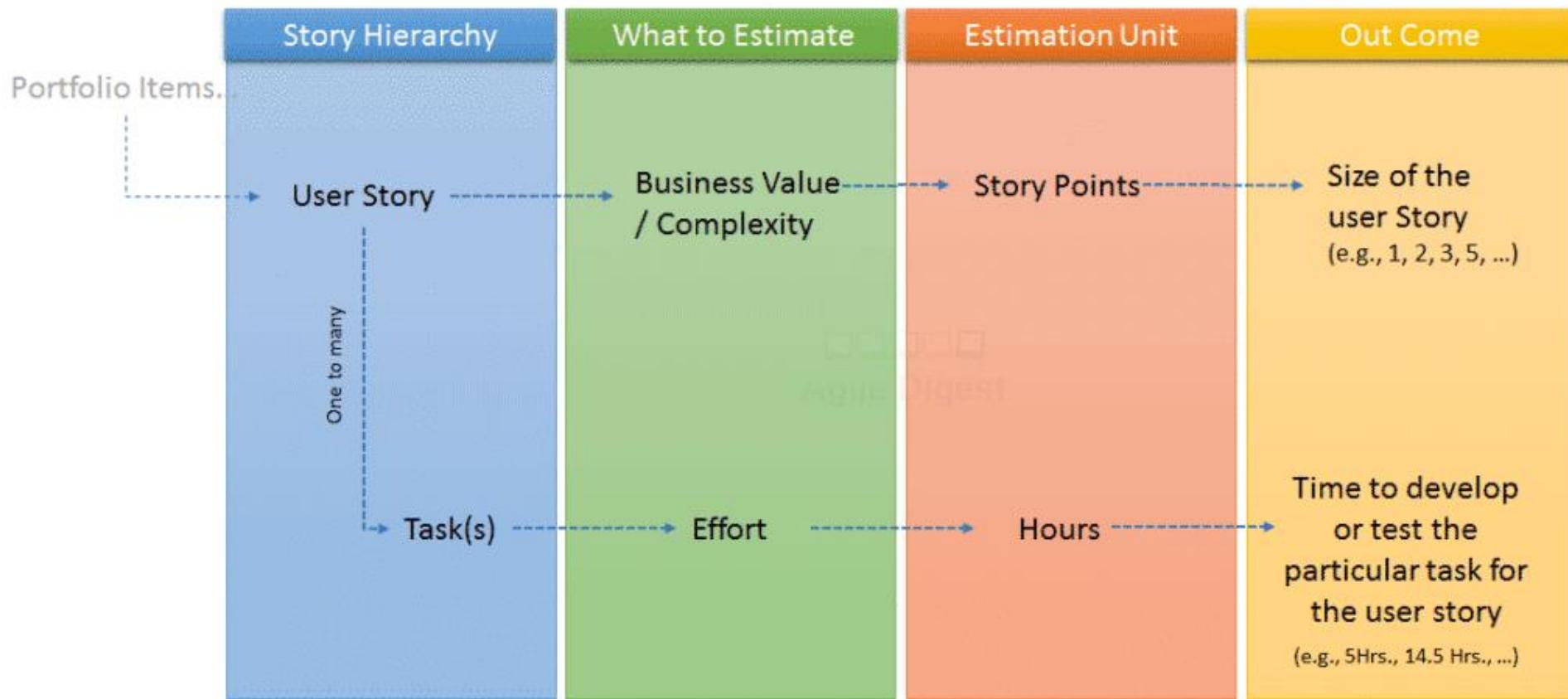
What about these
Which building is taller?
Which one is more complex?
Which would take more time to build?
Which would cost more?
How much money?
How much time?



Introduction to Key Agile Concepts



Estimation



Introduction to Key Agile Concepts



Agile Estimation

Agile Estimation is a team sport



Introduction to Key Agile Concepts



Agile Estimation

Agile Estimation is a team sport

Estimation methods include:

- T-shirt sizes (XS, S, M, L, XL) or the
- Fibonacci sequence (1, 2, 3, 5, 8, 13, 21, 34, etc.)



Introduction to Key Agile Concepts



Agile Estimation

T-shirt sizes (XS, S, M, L, XL)



Small
1pt



Medium
2-3pt



Large
5pt



Extra Large
8 Points



Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

1,2,3 is equivalent to 10,20,30

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89... and so on.

$$0+1=1$$

$$1+1=2$$

$$1+2=3$$



Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

0, 1, 1, 2, 3, 5, 8, 13, 21.....

Infinite

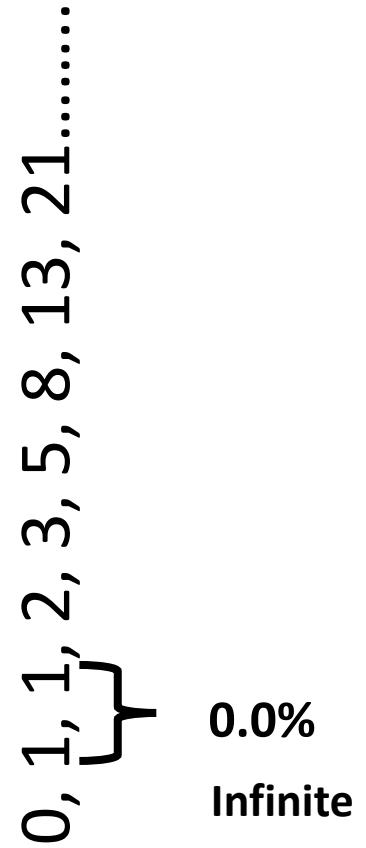


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

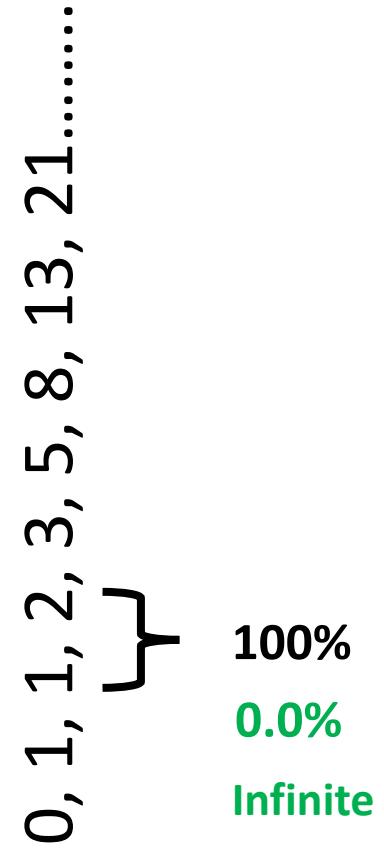


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

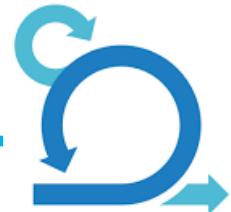
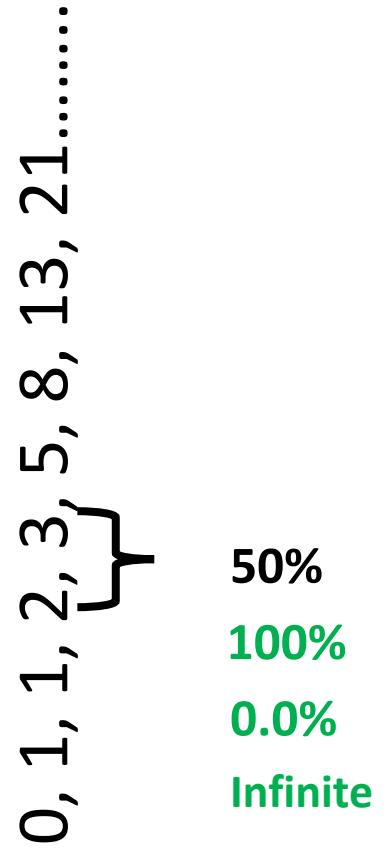


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

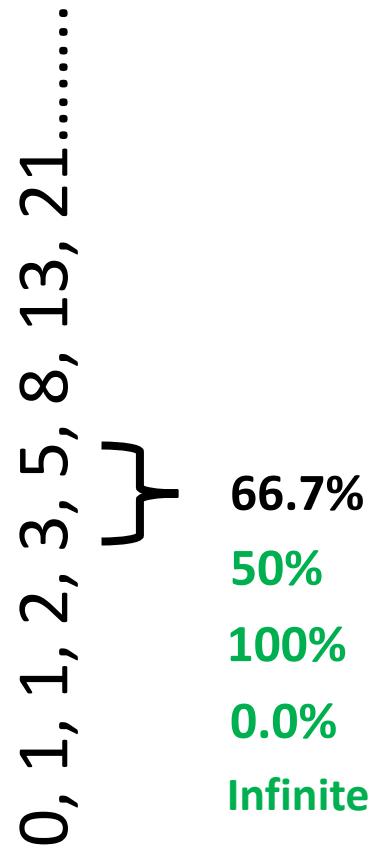


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

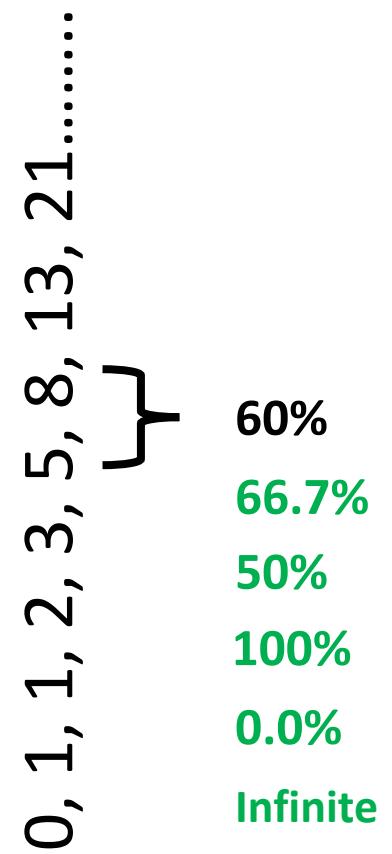


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

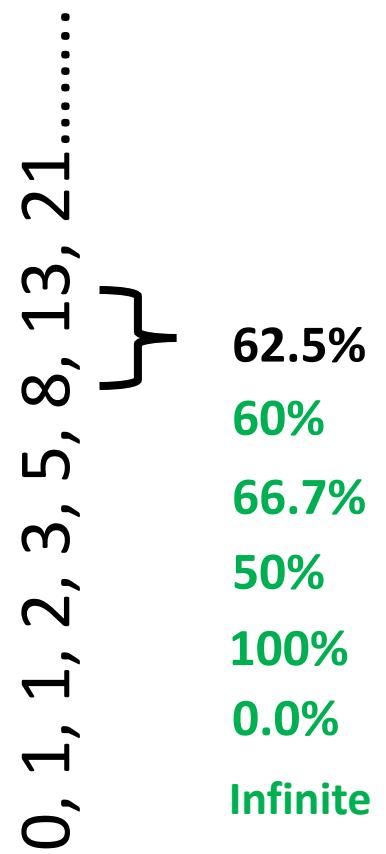


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

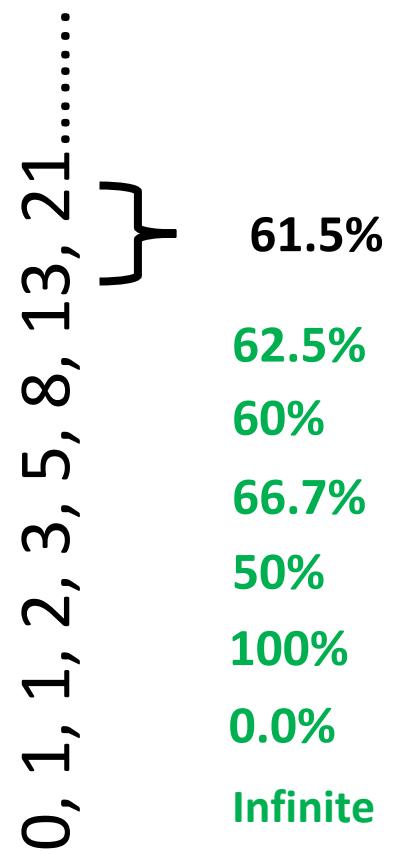


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence



Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence

In Agile Estimation, slightly modified version of Fibonacci estimation is used

1, 2, 3, 5, 8, 13, 21, 34, 55, 89... and so on.

Why Use the Fibonacci Sequence for Agile Estimation?

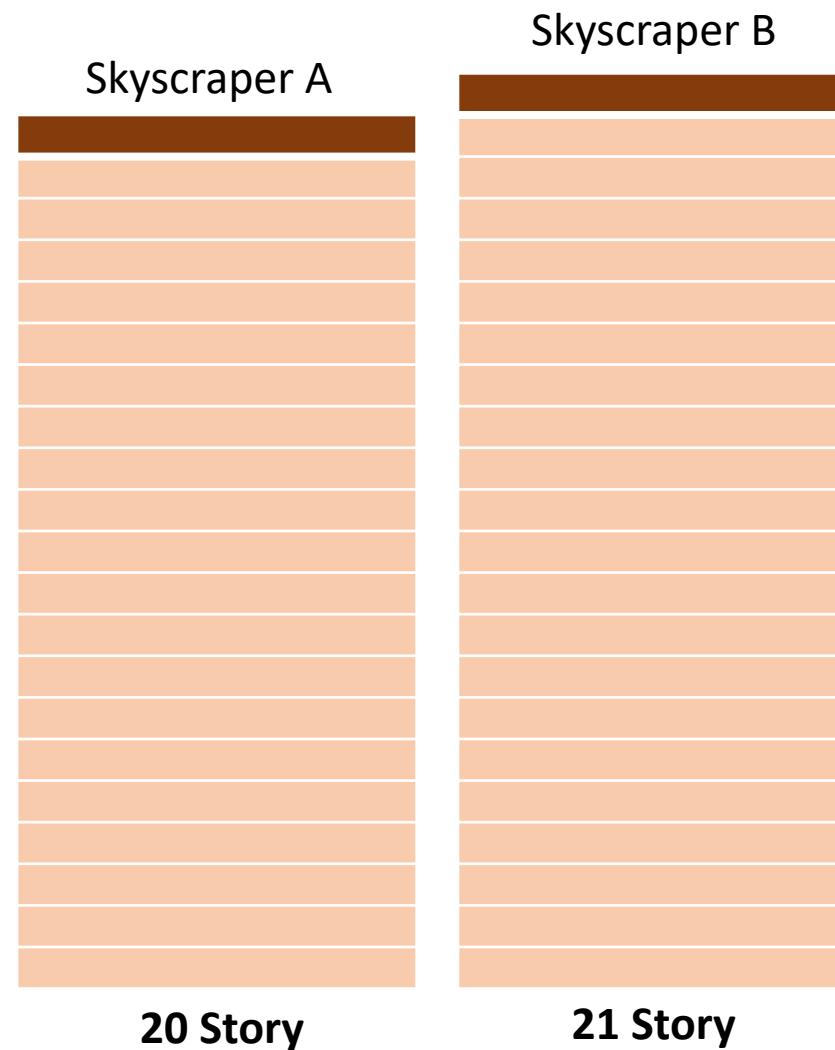
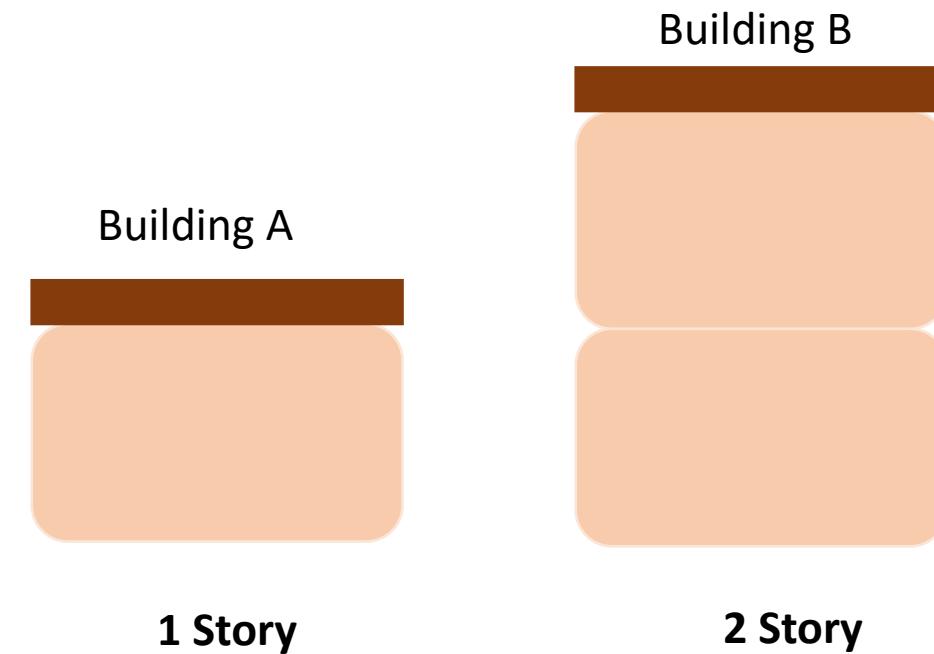


Introduction to Key Agile Concepts



Agile Estimation

Fibonacci sequence



Story Points are Relative



1 – QUICK TO DELIVER AND MINIMAL COMPLEXITY. AN HOUR

Example: add field to a form

2 – QUICK TO DELIVER AND SOME COMPLEXITY. MULTIPLE HOURS

Example: Add parameter to form, validation, storage

3 – MODERATE TIME TO DELIVER, MODERATE COMPLEXITY, POSSIBLE UNKNOWNS

Example: Migrate somewhat complex static CSS into a CSS pre-processor

5 – LONGER TIME TO DELIVER, HIGH COMPLEXITY, LIKELY UNKNOWNS

Example: Integrate with third-party API for pushing/pulling data, and link to user profiles in platform

8 – LONG TIME TO DELIVER, HIGH COMPLEXITY, CRITICAL UNKNOWNS

Example: Overhaul the layout/HTML/CSS/JS of a web application

13 – LONG TIME TO DELIVERY, HIGH COMPLEXITY, MANY CRITICAL UNKNOWNS

Example: Migrate application from an outdated data store to new DB technology and ORM

21 – YOU'RE DOING THIS WRONG. 😊



Introduction to Key Agile Concepts



Agile Estimation Techniques for user story

- Delphi
- Wide Band Delphi
- Complexity Bucket
- Estimation Poker



Introduction to Key Agile Concepts



Agile Estimation Techniques for user story

- Delphi
- Wide Band Delphi
- Complexity Bucket
- **Estimation Poker**



Estimation/Planning Poker



Scrum Master can help coordinate the estimation



Product Owner explaining all the aspects of the story requirement, dependencies, acceptance criteria and business value

Developers, testers, mutually discussing

- Amount of work
- Associated Risks
- Technical Changes
- Dependencies
- Complexities
- And the value



Estimation/Planning Poker



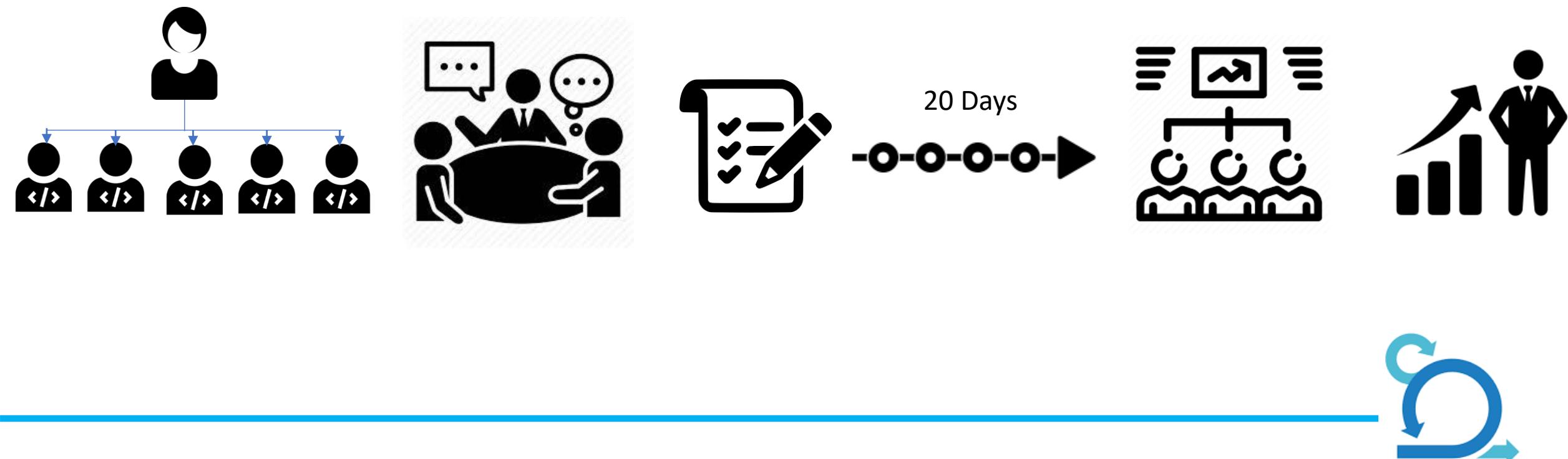
1, 2, 3, 5, 8, 13, 21, ...



Scrum

Scrum is a process framework used to manage product development and other knowledge work.

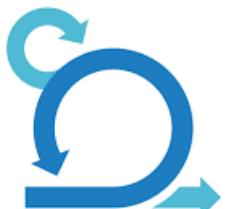
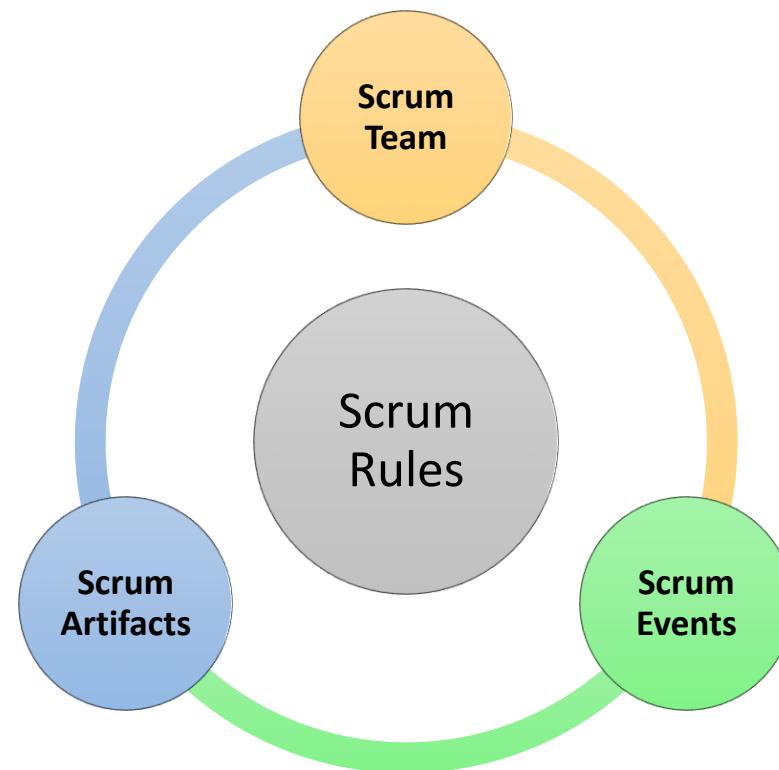
- Agile Alliance



Scrum

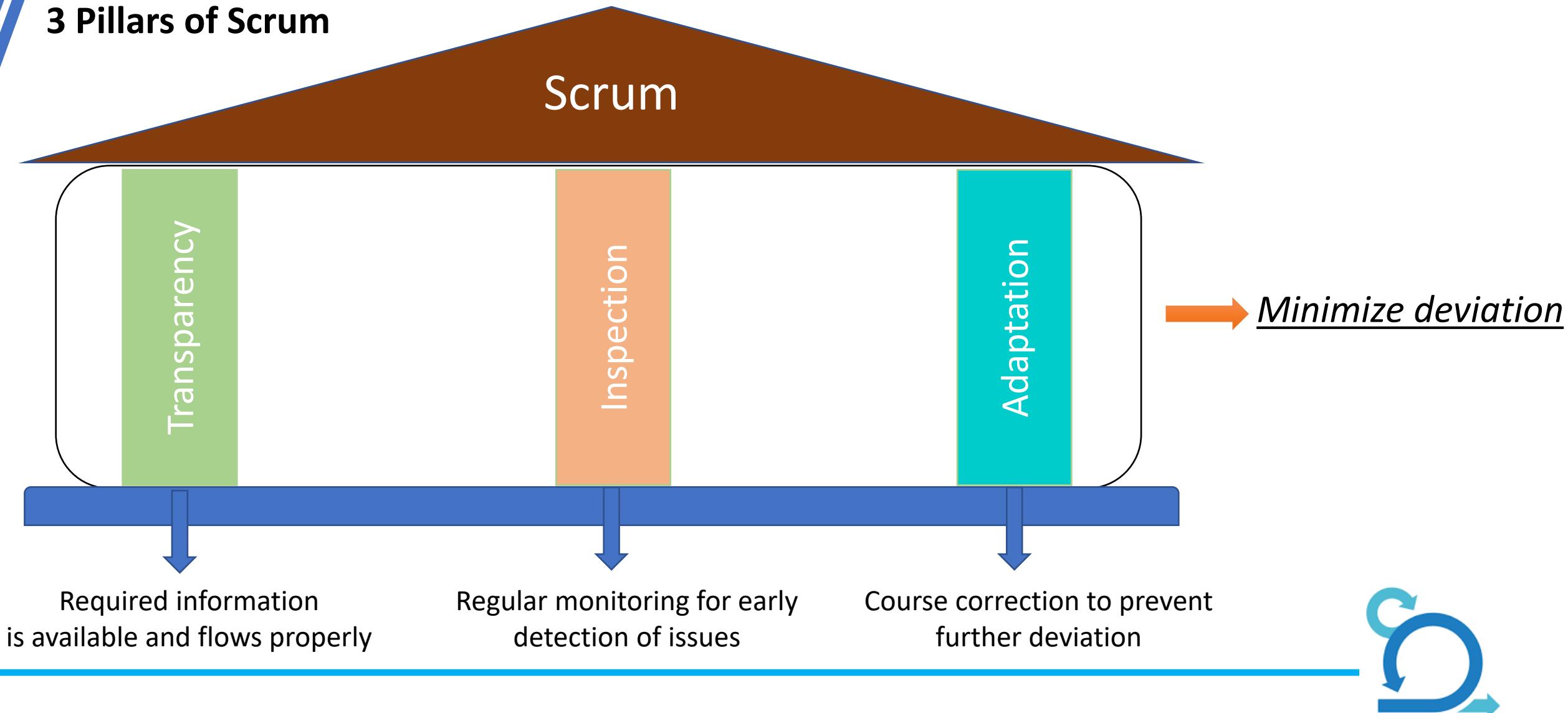
Scrum is a set of rules for structuring the team, processes and techniques for making the development process agile..

4 components of Scrum Framework



Scrum

3 Pillars of Scrum



Scrum

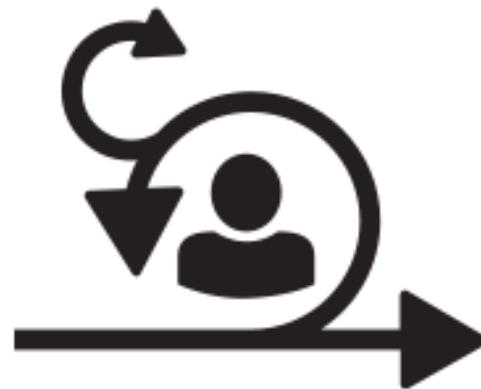
Scrum Team



Product Owner



Development Team



Scrum Master



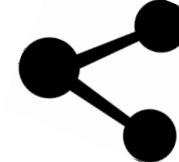
Scrum



Product Owner

Responsibility

- Maximize the total value of work done by the development team
- Product backlog management



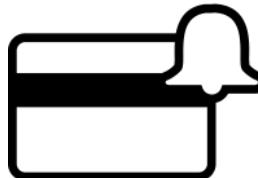
Development Team



Scrum



Product Owner
amazon



Task 1



Task 2



Task 3

.....



Scrum

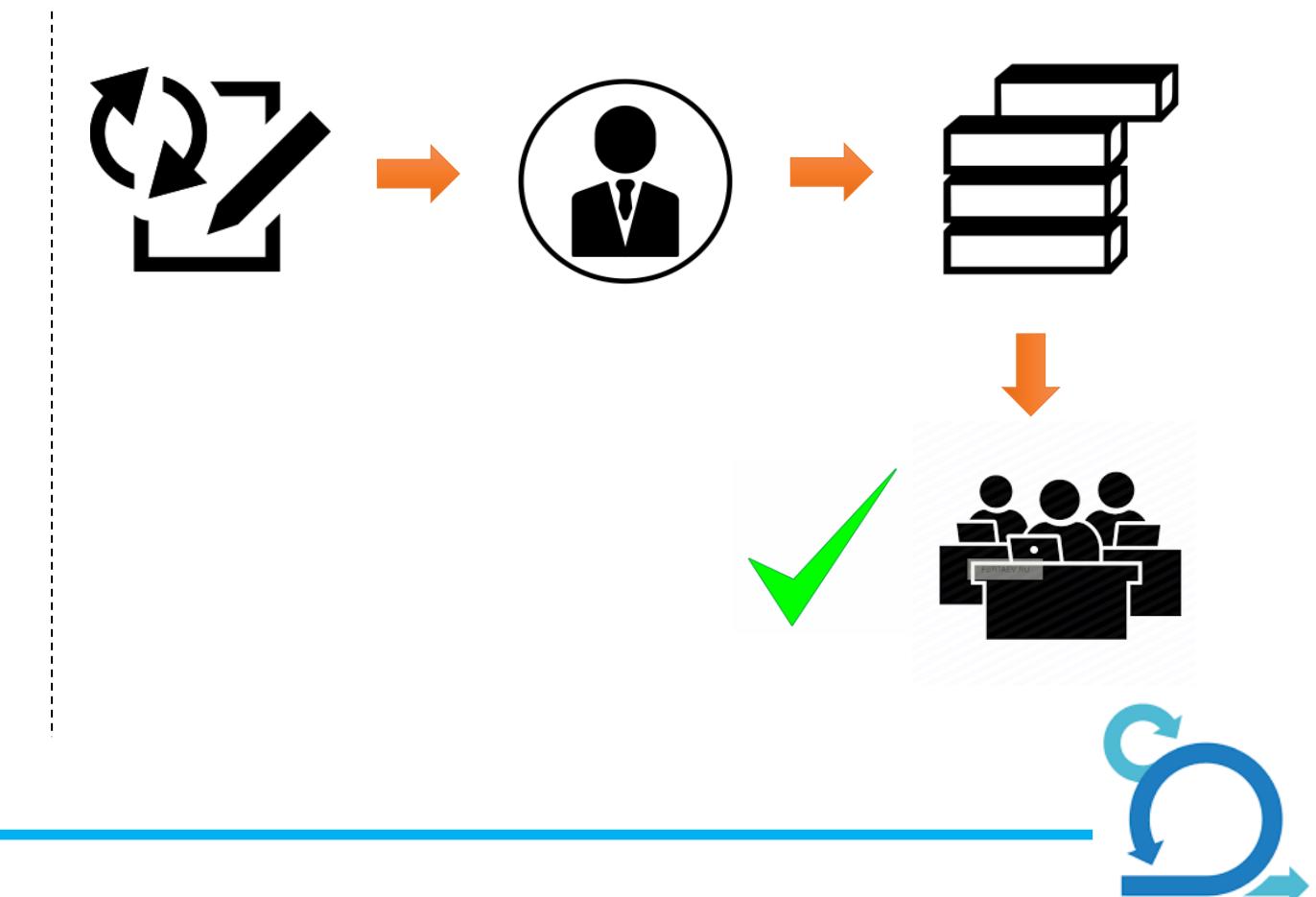
Organization also provides authority to a Product Owner..



Product Owner



Product Owner



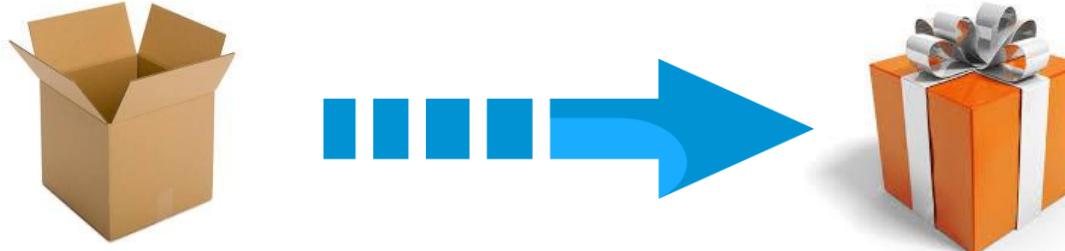
Scrum



Development Team

Responsibility

- Deliver a potentially releasable product at the end of each Sprint



Authority

- How the work would be done is left to the development team



Scrum



© All rights reserved

Development Team

Characteristics of Scrum Development Team

- It is cross-functional, no external dependency



- It is self-organizing



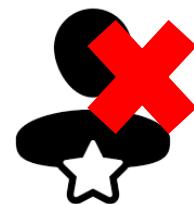
Scrum



Development Team

Characteristics of Scrum Development Team

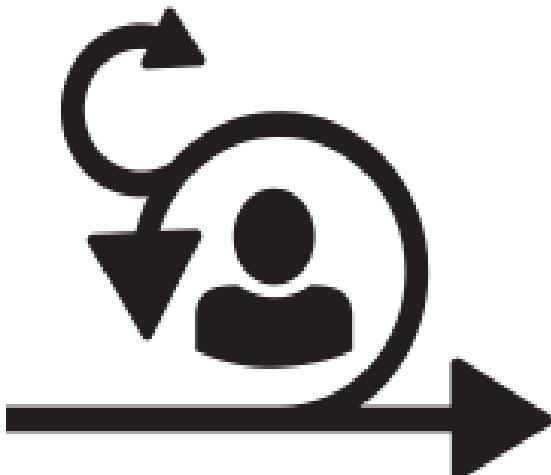
- **No titles and sub teams** within the development team



- Team size: A team of **3 to 9 team members** is good



Scrum



Scrum Master

Responsibility

- Ensure proper practical implementation of all concepts
- All Scrum related things such as maintaining the product backlog, helping the development team create high value products etc.
- Facilitating Scrum Events
- Ensure Scrum Artifacts are properly maintained
- Single point of contact to understand Scrum



Scrum

Scrum Events

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective



Scrum

Features of Scrum Events

- **Regular:** Events can be monthly/weekly/daily



Scrum

Features of Scrum Events

- **Time boxed:** limit on the duration of these events



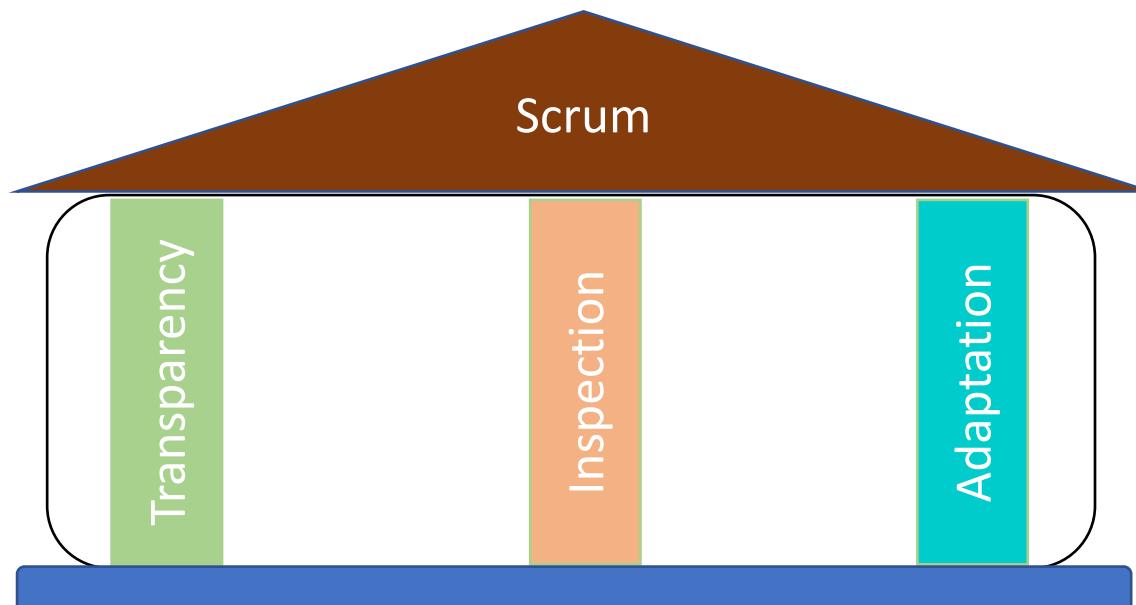
Daily Scrum is a 15- minutes meeting



Scrum

Features of Scrum Events

- Based on three pillars of transparency, inspection & adaptation



Scrum

Sprint



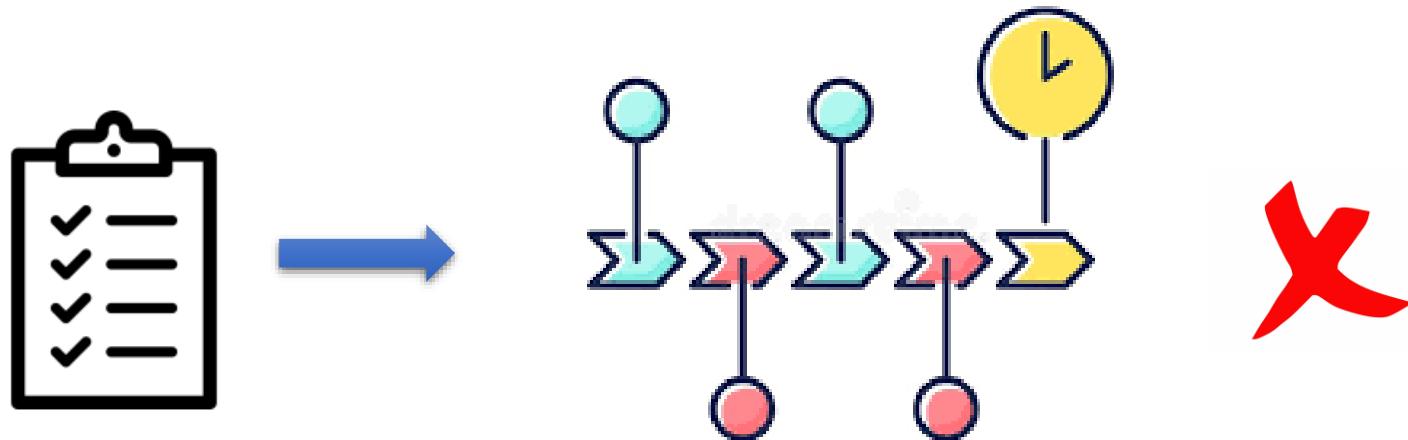
Product Owner

amazon



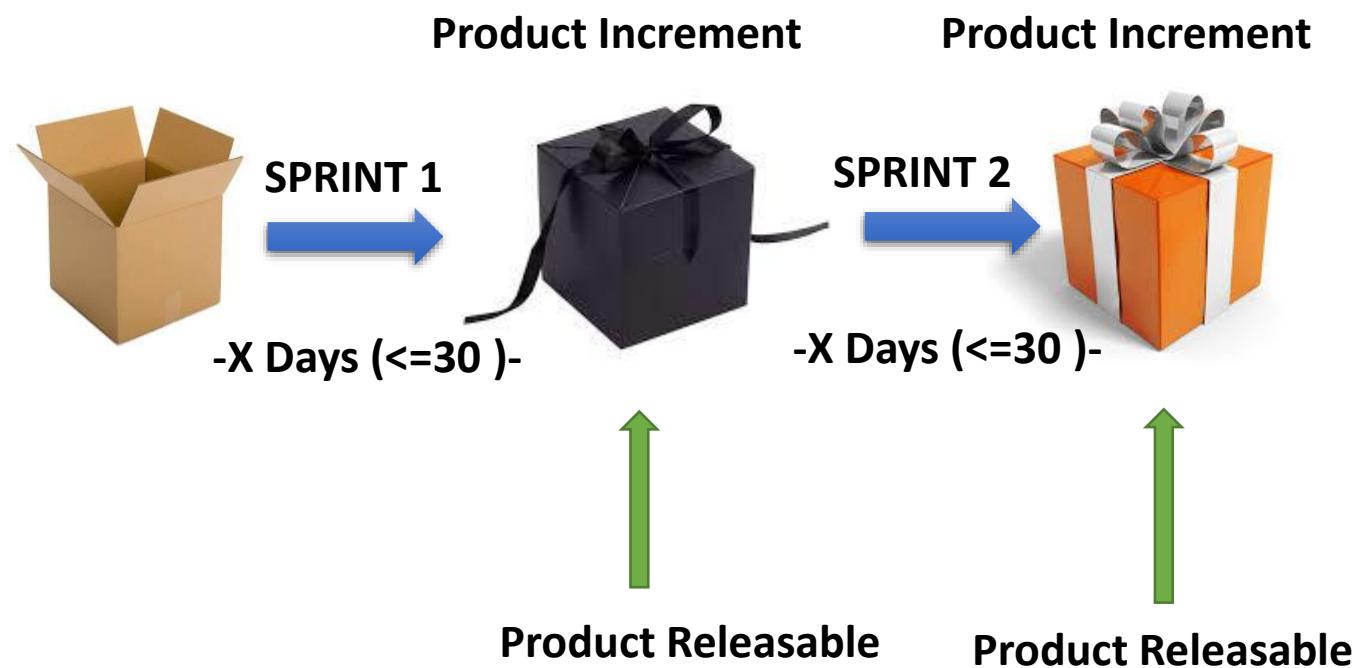
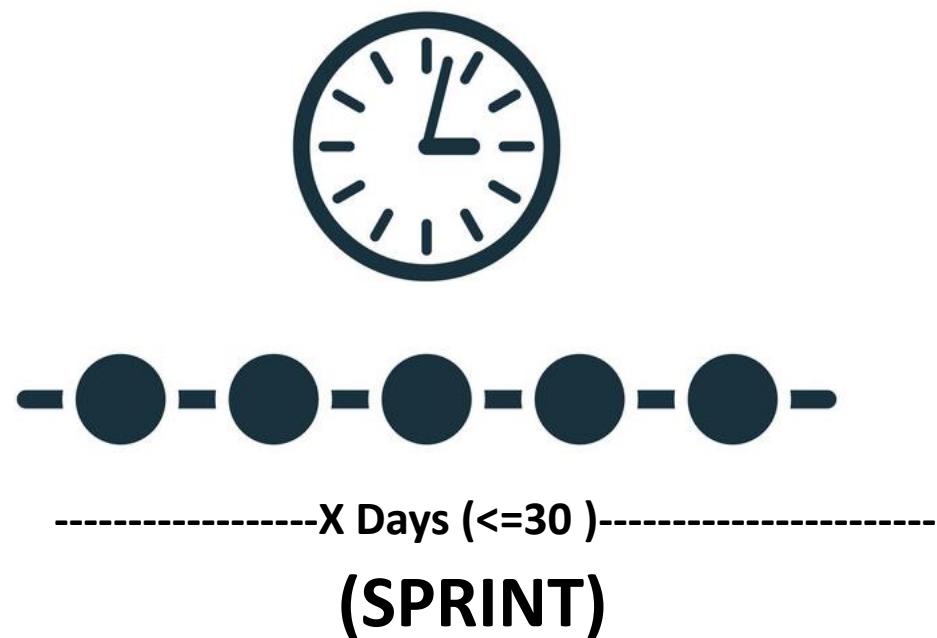
Scrum

Sprint



Scrum

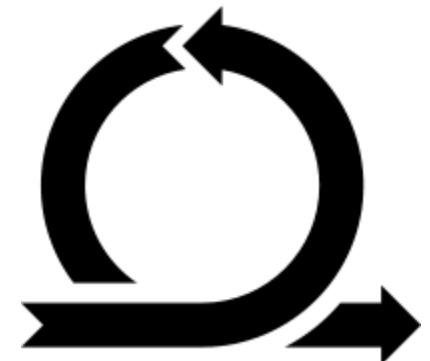
Sprint



Scrum

Sprint Planning

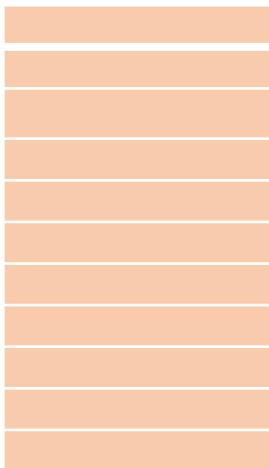
- Sprint planning happens once in each Sprint
- Entire Scrum Team: Product Owner, Scrum Master & Development Team must be a part of the event
- Time box for the event is 8 hours (upper limit)
- Two agendas
 - What is to be done in this Sprint
 - Establish how it will be done



Scrum

Sprint Planning

Product Backlog

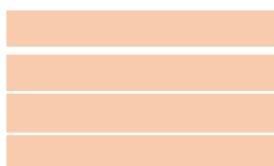


Product Owner



Product Owner discusses the product backlog with the development team

Sprint Backlog



What is to be done?

- **Team Velocity:** How much work development team was able to do in the previous few sprints
- **Estimation Poker**

How it will be done?

- Plan is not fixed, it evolves as more clarity increases
- Development team will come up with the plan



Scrum

Daily Scrum



- Event is primarily for the **Development Team**
 - **Scrum Master** facilitates the event
 - **Product Owner** can attend the event (not necessary)
-
- Daily Scrum is implementation of two pillars of scrum: Regular inspection & early adaptation

Daily Scrum is a 15-minutes event
done everyday

Same place, same time



Scrum

Daily Scrum Meeting



15-minutes everyday
Same place, same time

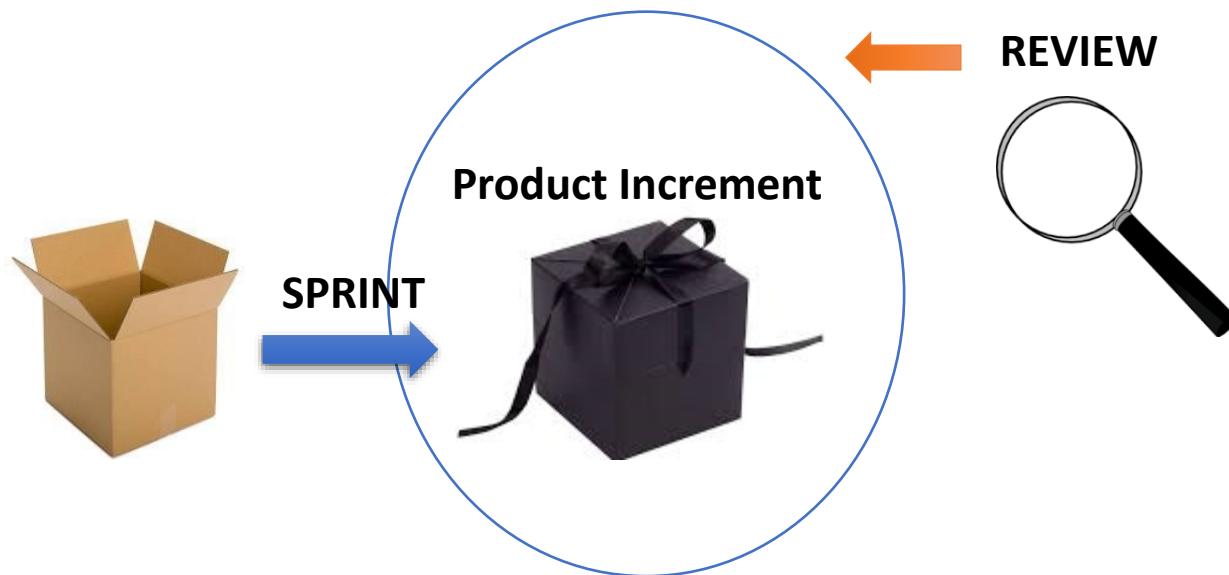
3 important questions in Daily Scrum

- What did I do yesterday?
- What will I do today?
- Any difficulties or impediments stopping me from the Sprint goal?



Scrum

Sprint Review



Attendees

- **Entire Scrum Team:** Product Owner, Scrum Master, the Development Team & other stakeholders

Objective

- Help the scrum team work more effectively
- Increase transparency within the team and between other stakeholders and Scrum team

Time box for the event is 4 hours or less



Scrum

Sprint Review



What happens in Sprint Review?

- 1) Product Owner** invites all attendees
- 2) Product Owner** highlights completed & not-completed items
- 3) Development team** shares Sprint experience, highlight challenges
Development team demonstrates completed item
Stakeholders may ask questions on the demonstrated item
- 4) Product Owner** opens the backlog & initiate discussion on what to do next.
Priorities, budgets, timelines & capabilities etc. are discussed.
- 5) At the end of Sprint Review, Product Owner** should have updated backlog



Scrum

Sprint Retrospective

Scrum Team does this event

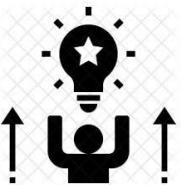
Agenda

Inspect the performance of the team

What went well?



What went wrong?



At the end of Scrum Retrospective, Scrum Team should know:

- Area of improvements
- Plan of working on them

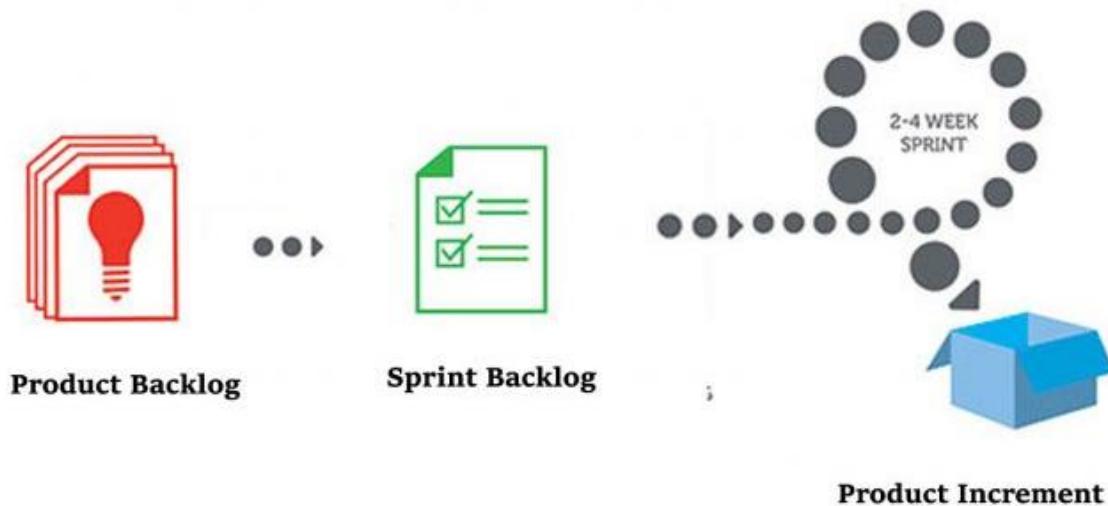
Inspection &
Adaptation of Scrum
Team



Scrum

Scrum Artifacts

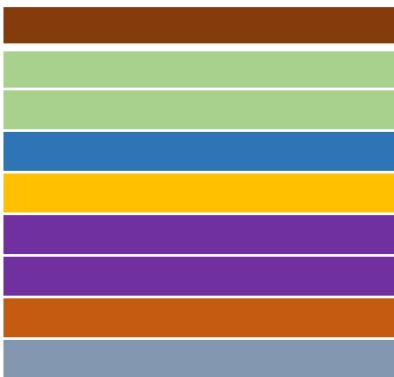
- Product Backlog
- Sprint Backlog
- Increment



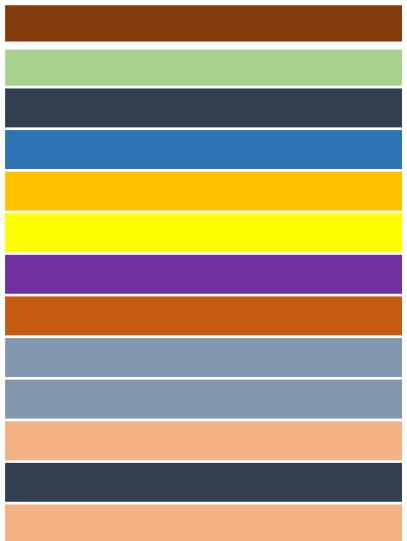
Scrum

Product Backlog

At the initial stage of starting the process



At a later stage when requirements changes



Ever Evolving

Never Fixed

Add details to new requirements



Product Owner Development Team

Product Backlog Refinement



Scrum

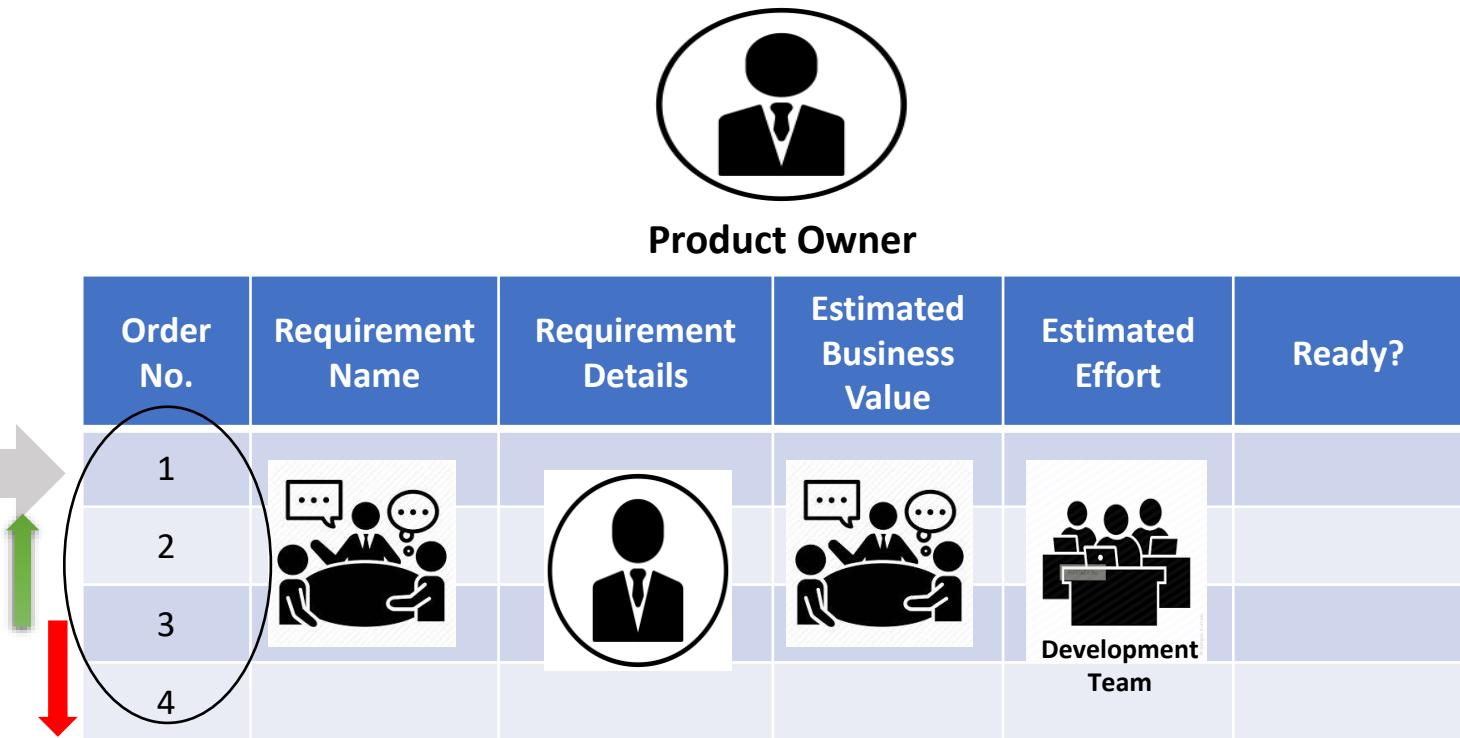
Product Backlog



Business Team



Raw requirements



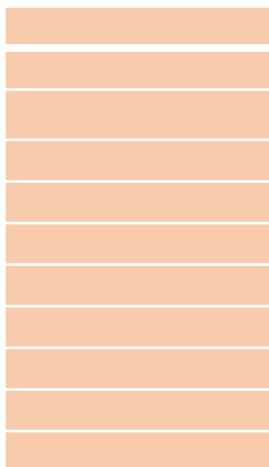
Backlog Refinement \leq 10% of the capacity of the development team



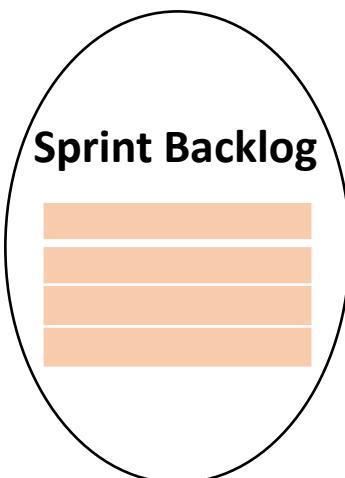
Scrum

Sprint Backlog

Product Backlog



Product Owner discusses the product backlog with the development team

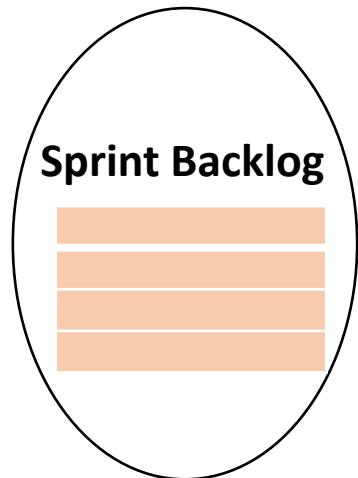


Sprint Planning Event
Sprint Review Event



Scrum

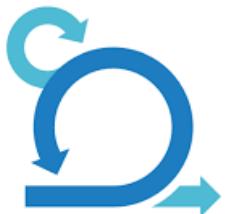
Sprint Backlog



- Daily monitoring of Sprint Backlog is done in the Daily Scrum
- Owned by the development team
- Any deviation from the goal should be highlighted to the Product Owner & Stakeholders immediately



Scrum

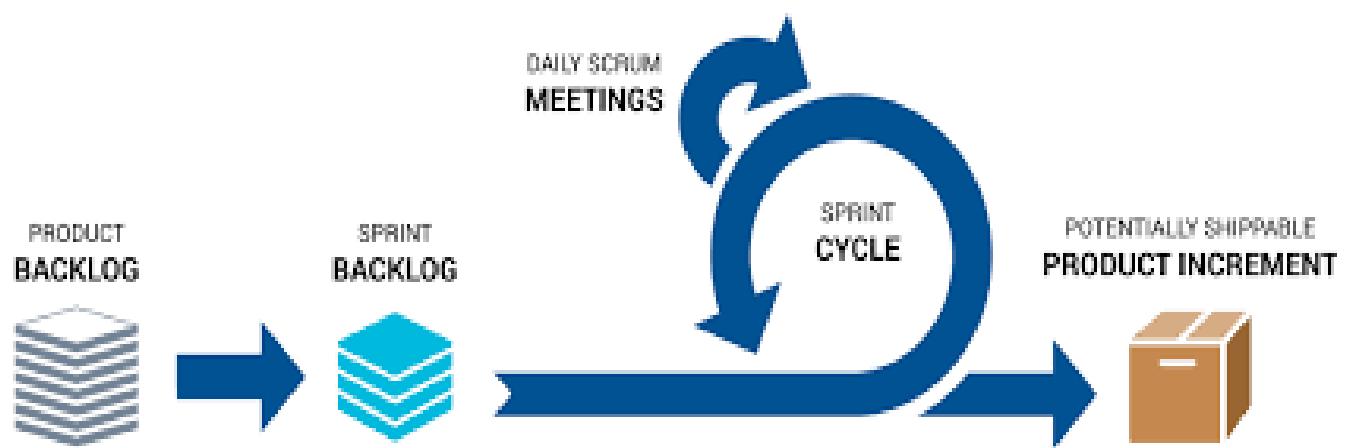


Scrum

Increment

Increment is the **final releasable product**

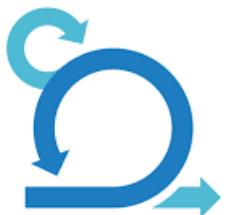
- It should be **usable**
- It should be **inspectable**
- It should **include increments of previous sprint**



Definition if “Done”



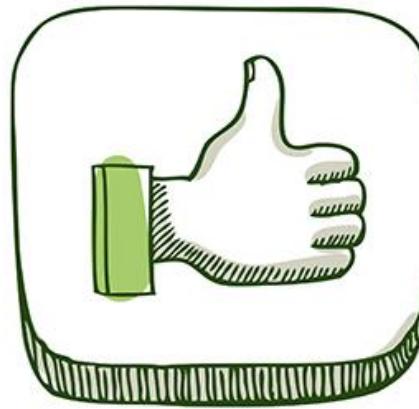
Advantages & Disadvantages of Scrum



Advantages of Scrum



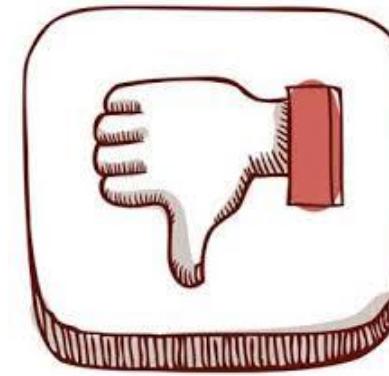
- More transparency and project visibility
- Increased team accountability
- Easy to accommodate changes
- Increased cost savings
- Faster Delivery
- Customers are heard



Disadvantages of Scrum



- Risk of scope creep
- Team requires experience and commitment
- The wrong Scrum Master can ruin everything
- Poorly defined tasks can lead to inaccuracies



Introduction to Kanban

Flexible & easy to implement method for process improvement



Scrum: Has Strict Rules



Extreme Programming: Has Strict Practices



Introduction to Kanban

Kan 看 ban 板

- Kan = Card
- Ban = Signal

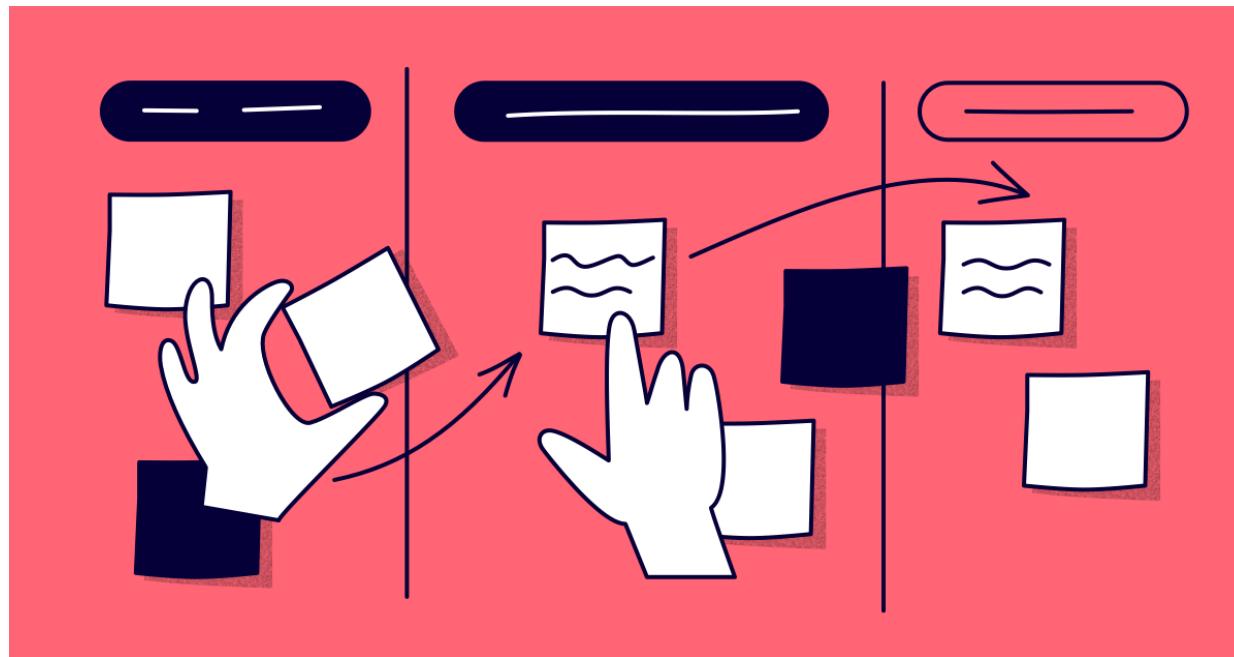


TOYOTA

Identified bottlenecks in the car making process using Kanban



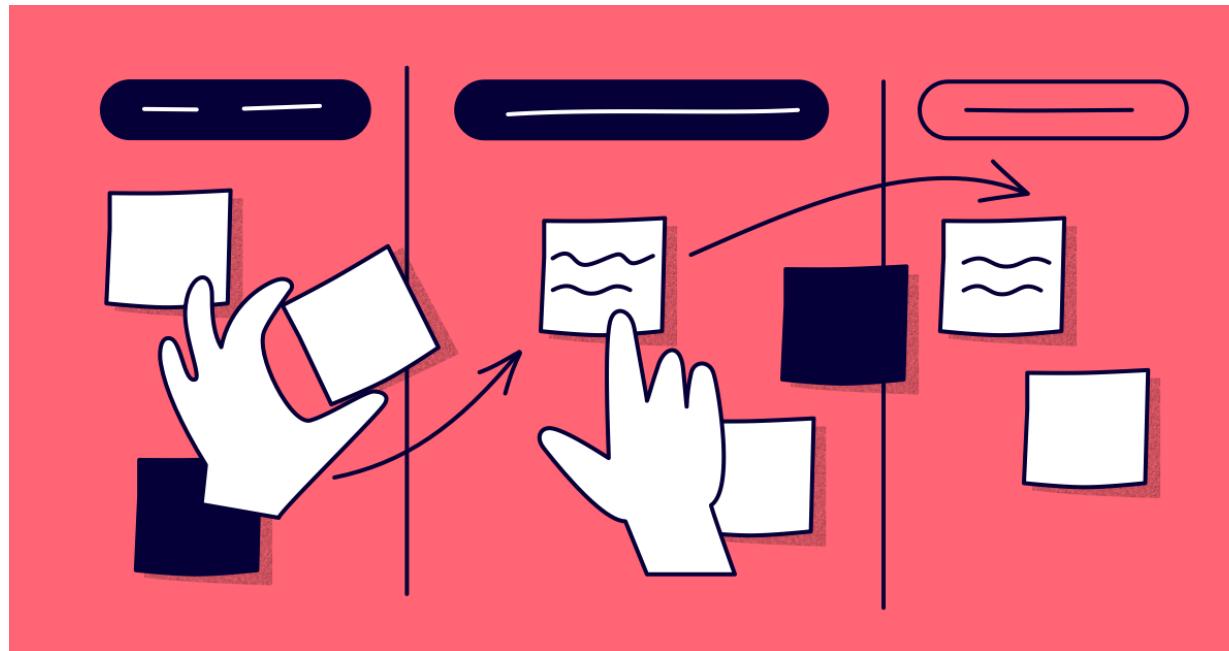
Introduction to Kanban



Visualizing the workflow
to
achieve process improvement



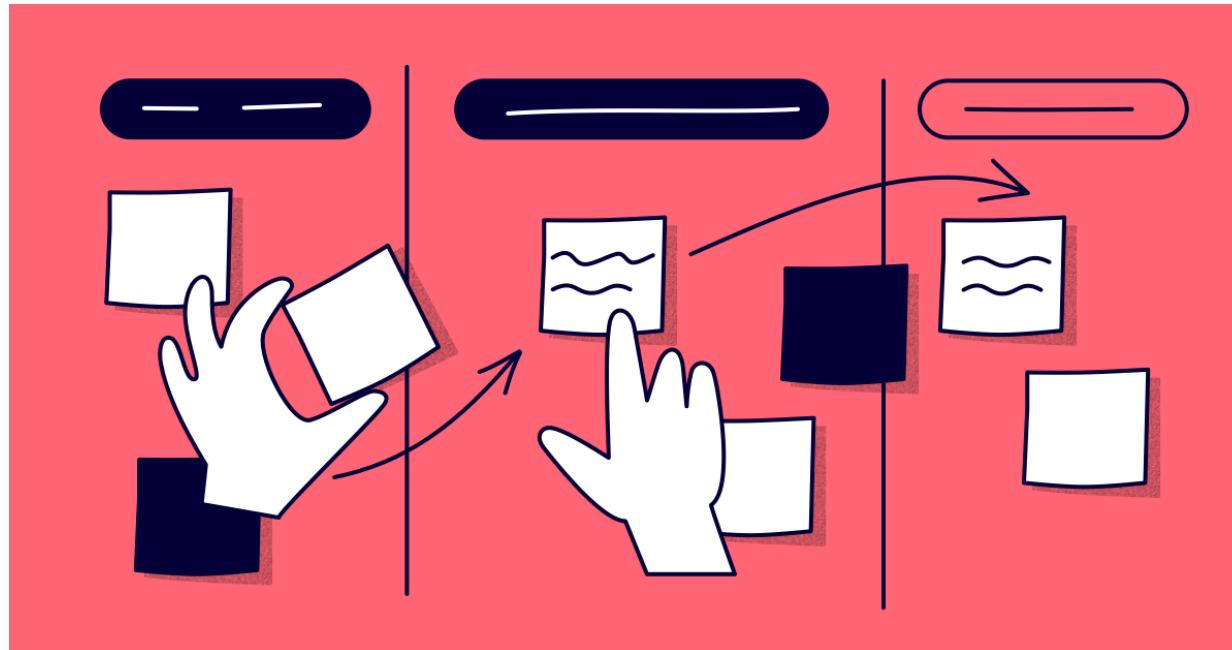
Introduction to Kanban



Visualizing the workflow
to
achieve process improvement



Introduction to Kanban

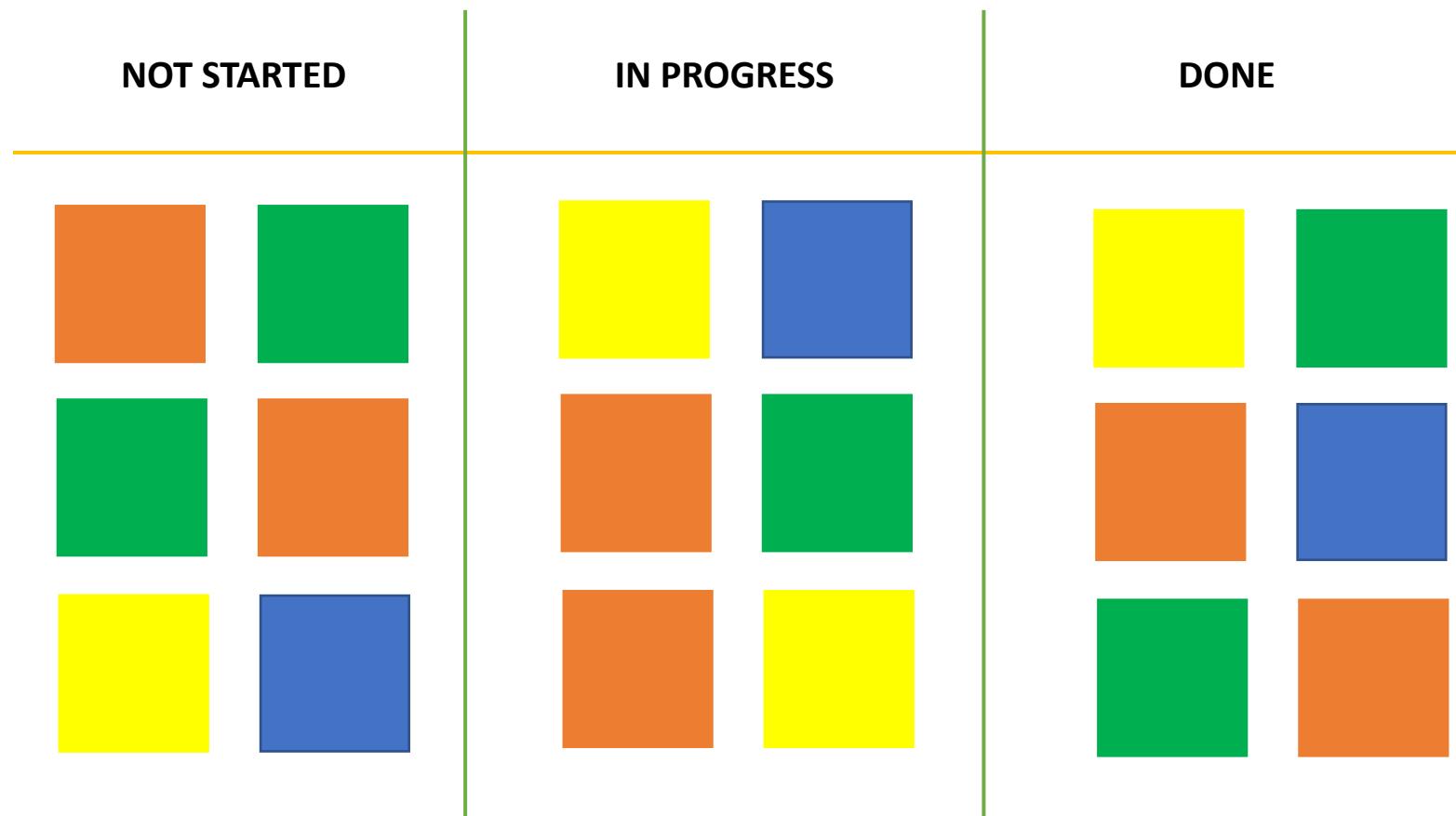


Visualizing the workflow
to
achieve process improvement

Limiting progress
Minimizing waste
Implementing feedback loops



Kanban Board



Kanban Board



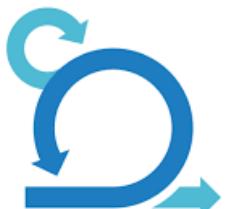
Physical Kanban Board

Team Kanban Board

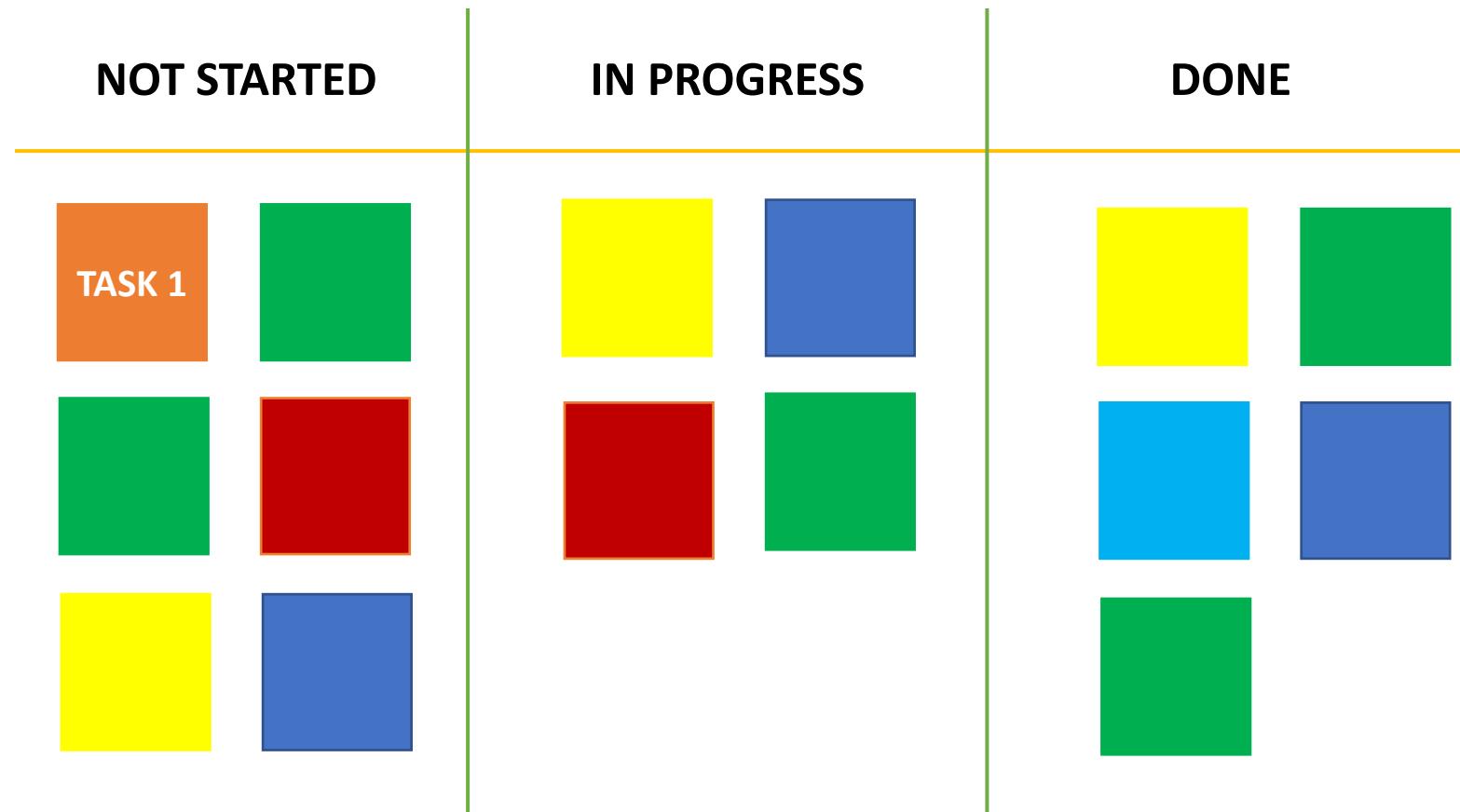
QUICK FILTERS: Critical partners Only my partners Recently updated

1 To do	4 In progress	3 Code review Max 2	1 Done
<ul style="list-style-type: none"> + TIS-28 ↑ Research options to travel to Pluto 	<ul style="list-style-type: none"> + TIS-25 ↑ Engage Jupiter Express for travel + TIS-25 ↑ Add Deimos Tours as a travel partner + TIS-20 ↑ Engage Saturn Lines for group tours + TIS-24 ↑ Sign Contract for SunSpot Tours 	<ul style="list-style-type: none"> + TIS-27 ↑ Engage Saturn Resort as PTP + TIS-27 ↑ Engage Speedy SpaceCraft + TIS-26 ↑ Reach out to the Red Titan Hotel 	<ul style="list-style-type: none"> + TIS-23 ↑ Engage JetShuttle SpaceWays for travel

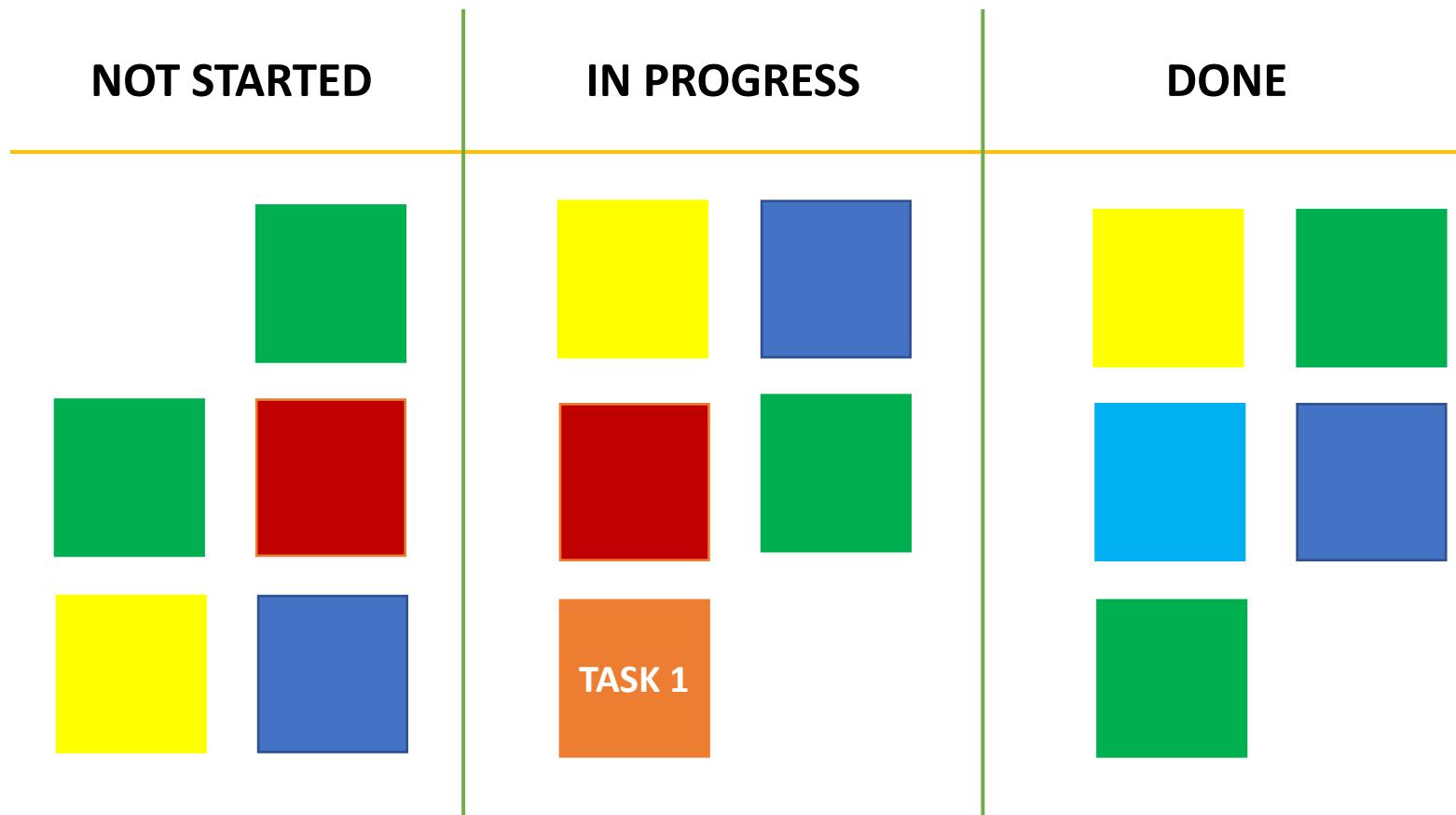
Software Based Board



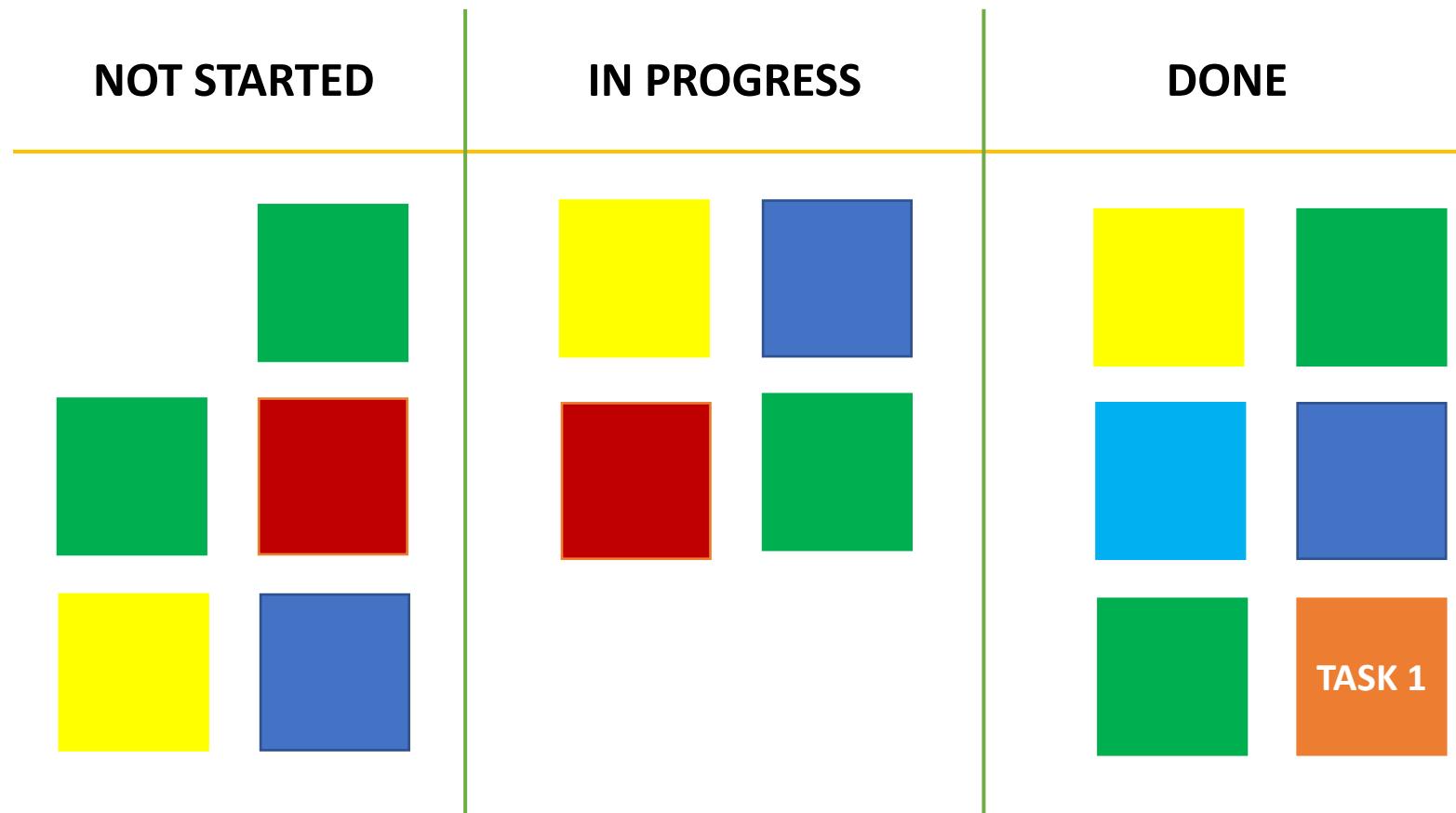
Kanban Board



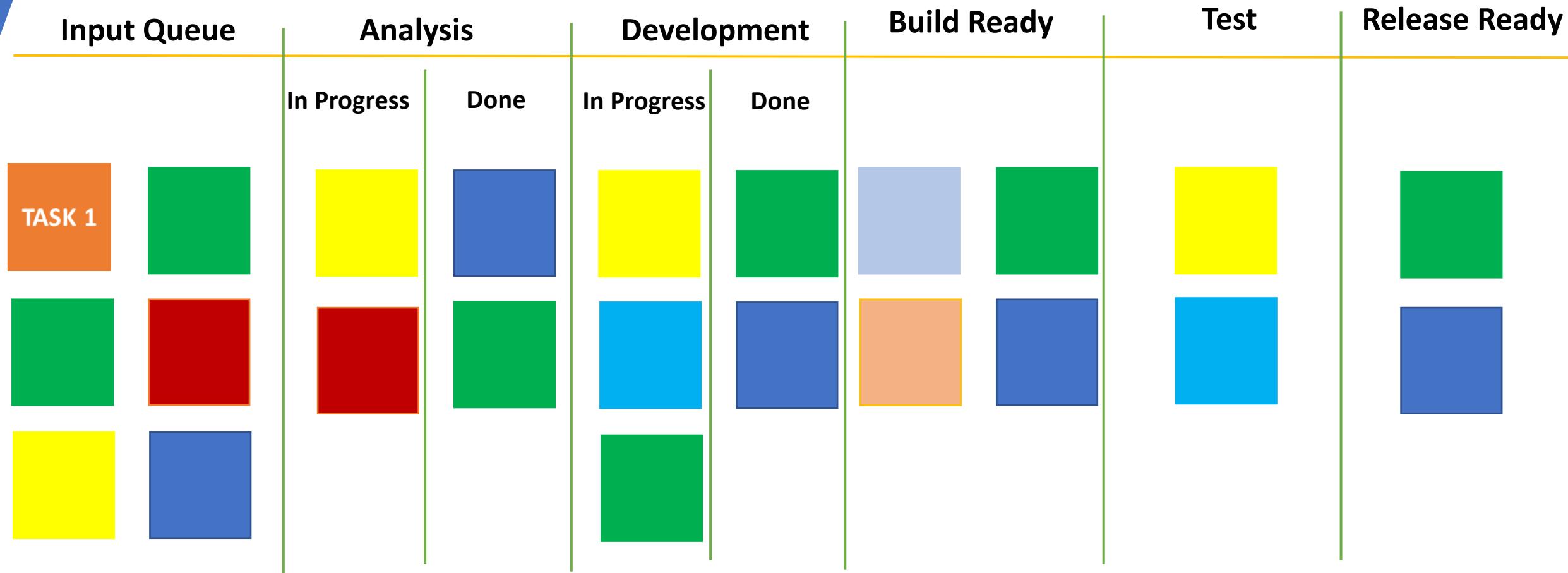
Kanban Board



Kanban Board



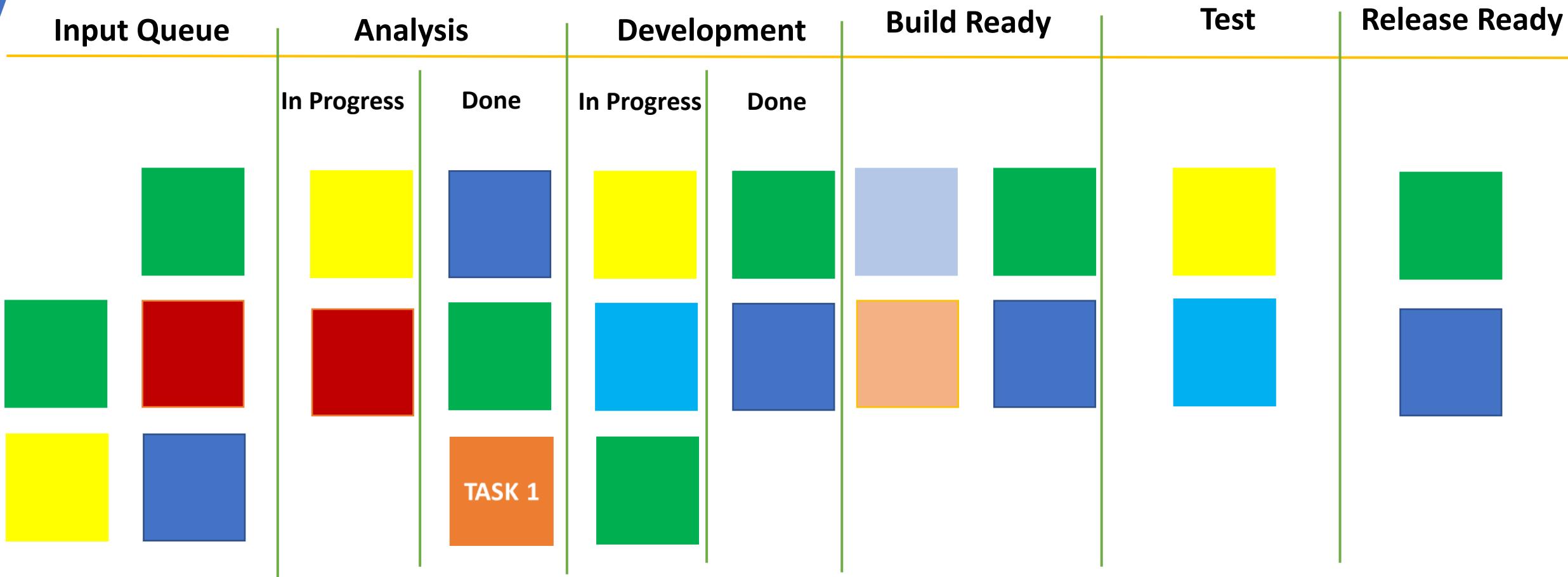
Kanban Board



Board Reference: David Anderson's "Kanban"



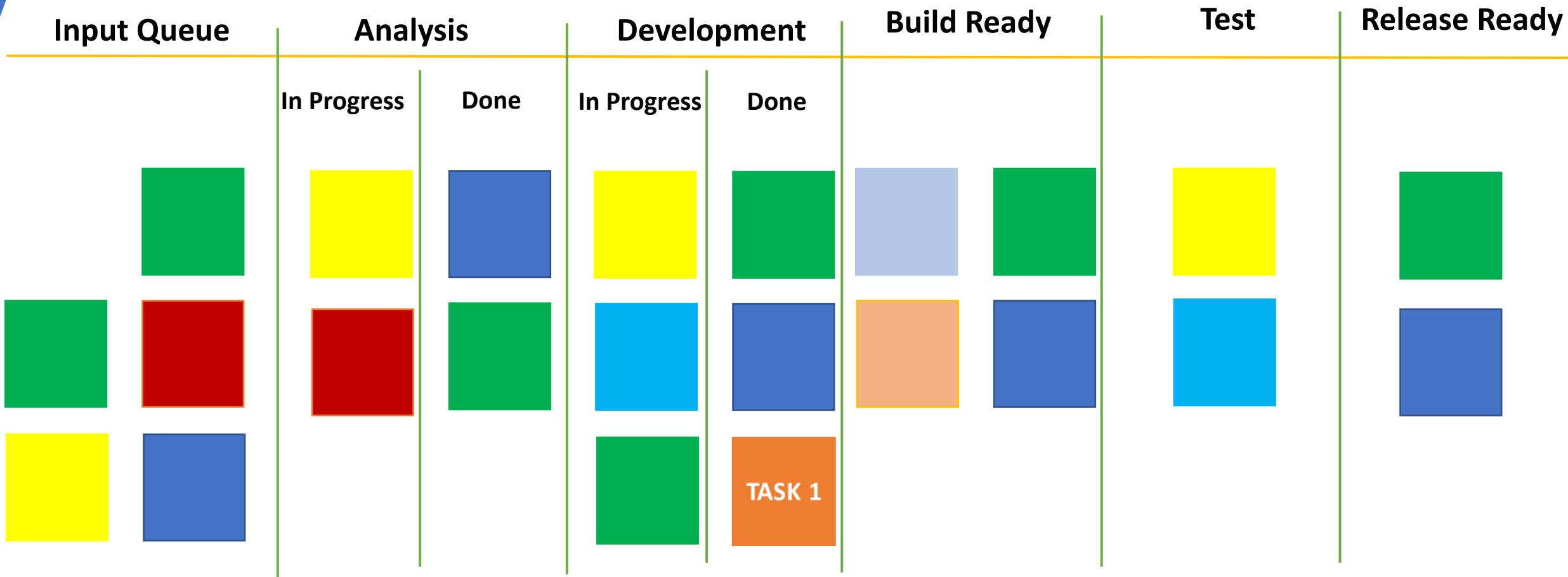
Kanban Board



Board Reference: David Anderson's "Kanban"



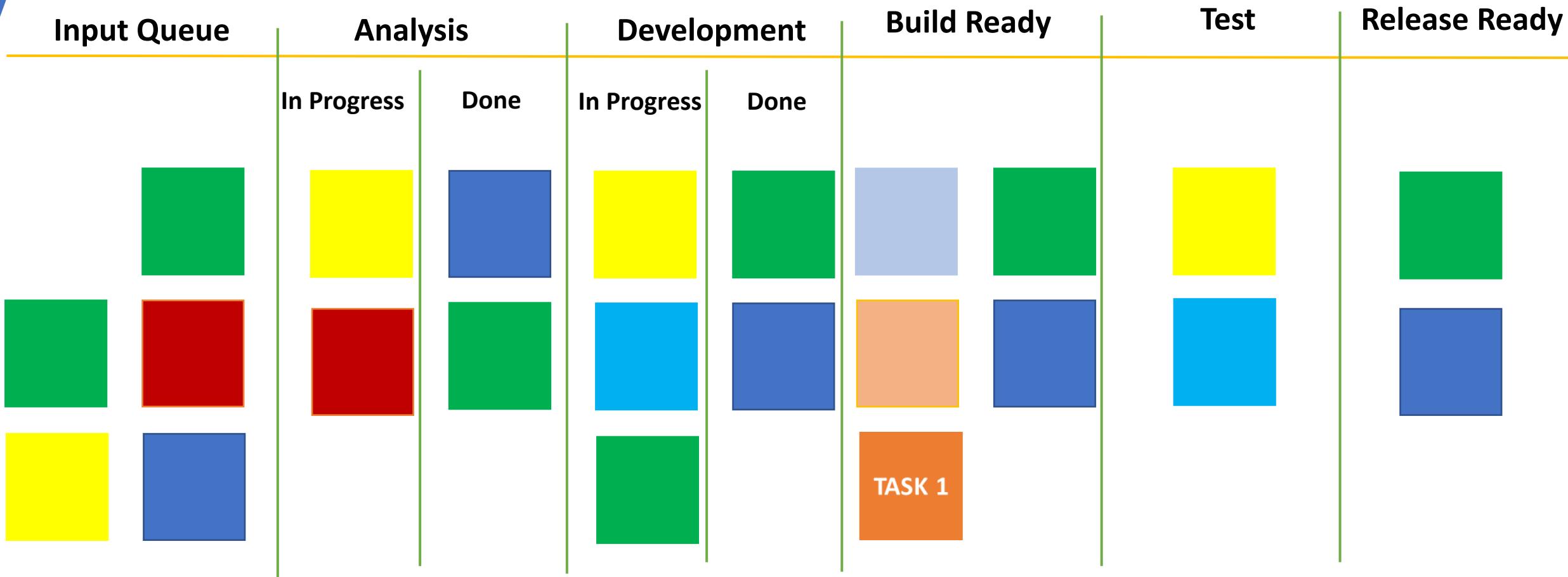
Kanban Board



Board Reference: David Anderson's "Kanban"



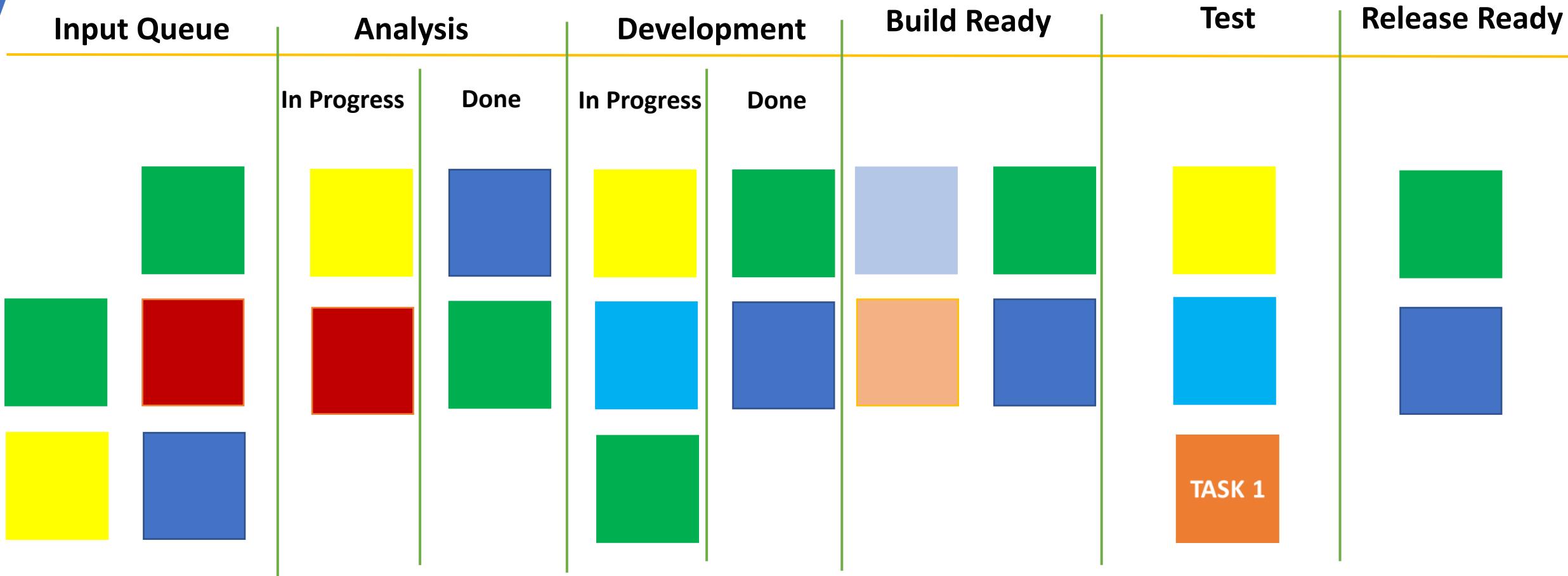
Kanban Board



Board Reference: David Anderson's "Kanban"



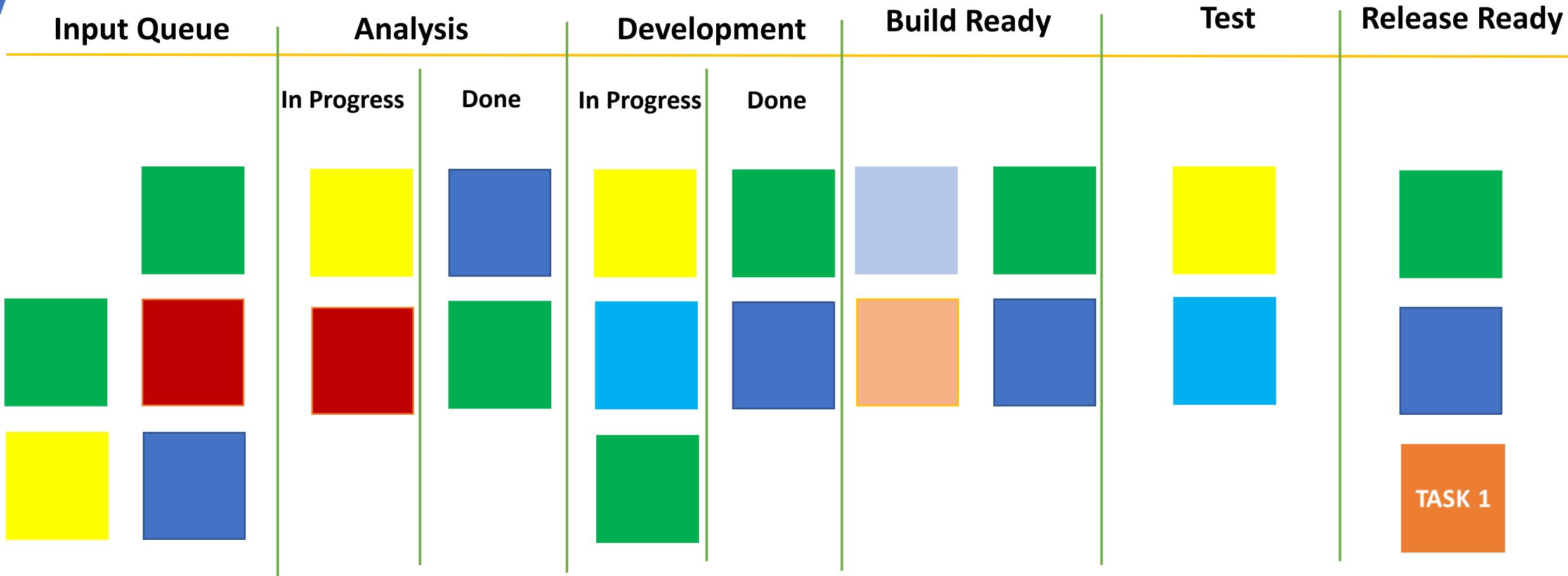
Kanban Board



Board Reference: David Anderson's "Kanban"



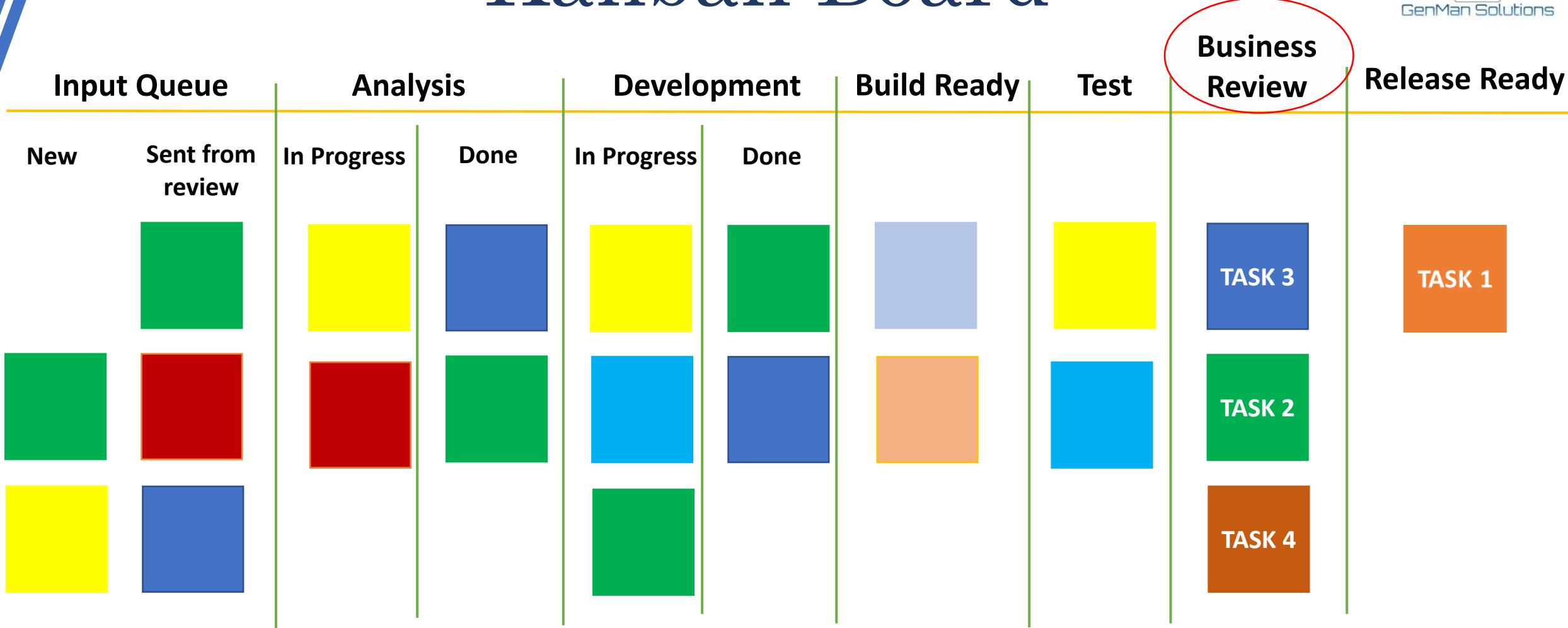
Kanban Board



Board Reference: David Anderson's "Kanban"



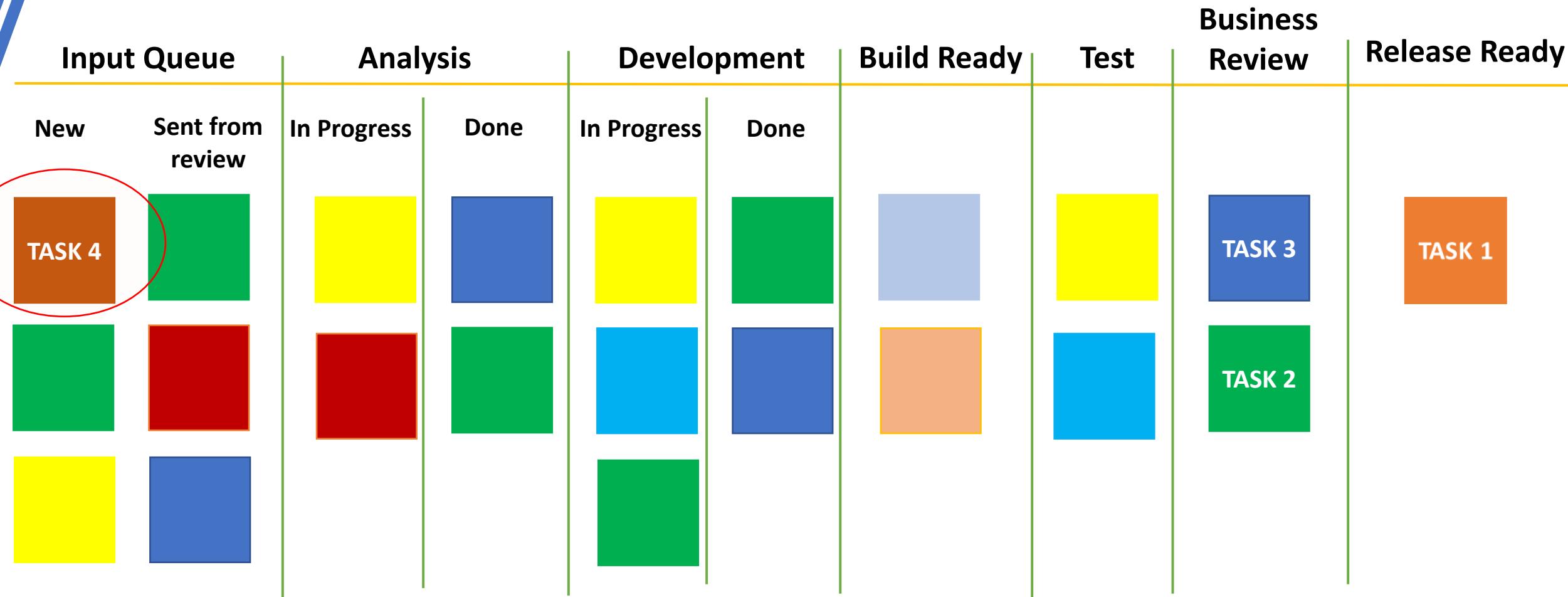
Kanban Board



Board Reference: David Anderson's "Kanban"



Kanban Board



Board Reference: David Anderson's "Kanban"



Kanban Board

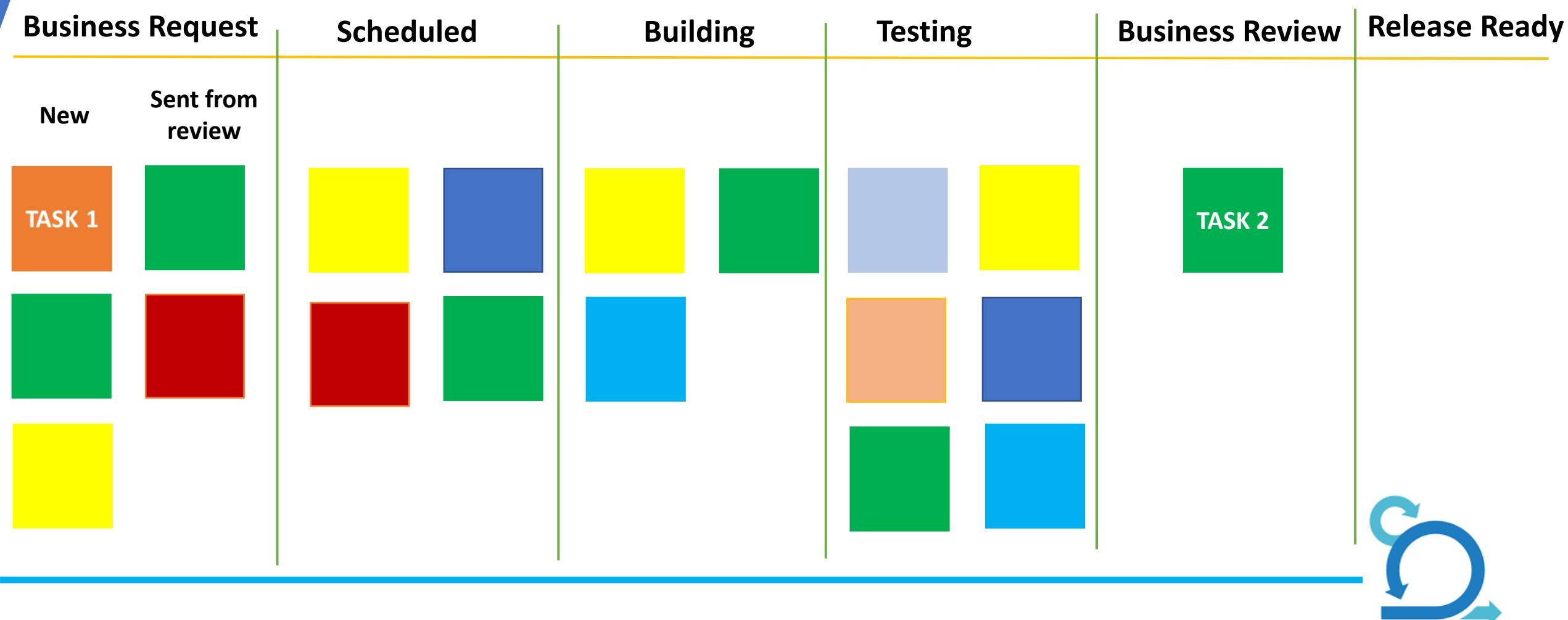


Board Reference: David Anderson's "Kanban"



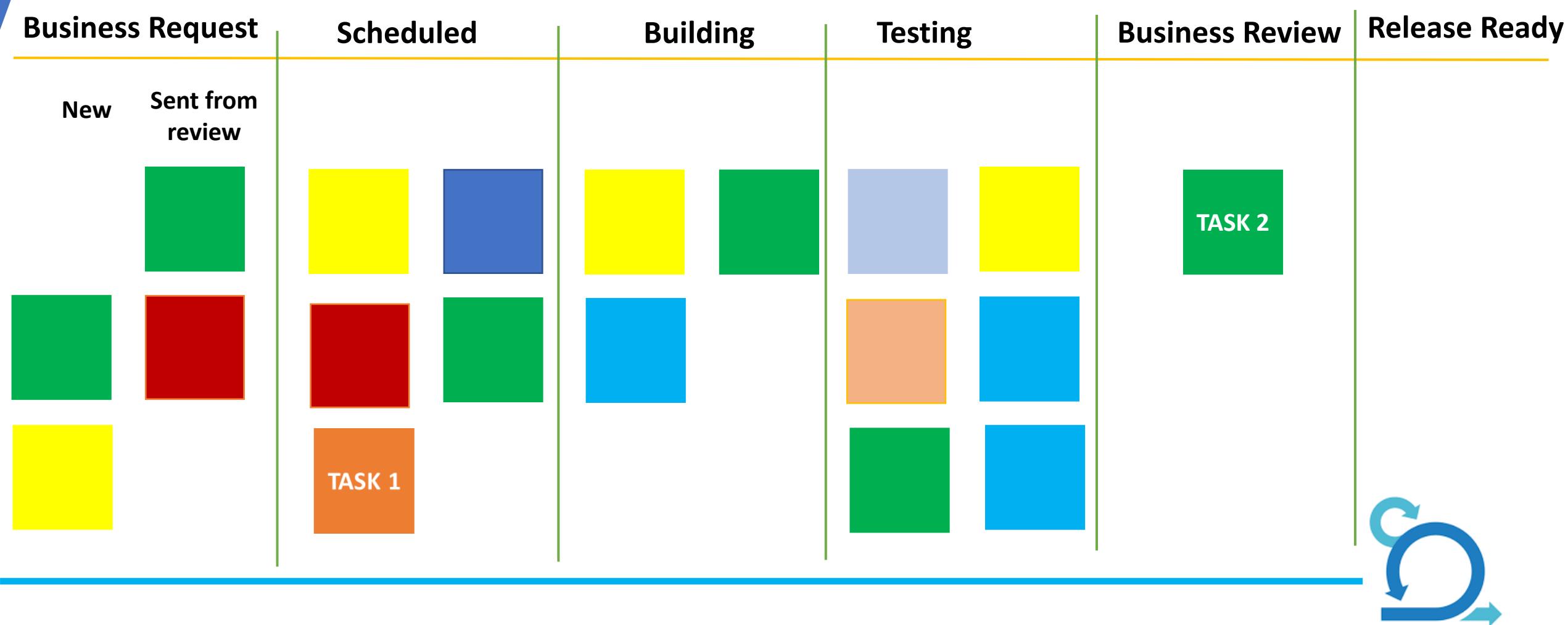
Kanban Board

Finding inefficiencies/issues in the process



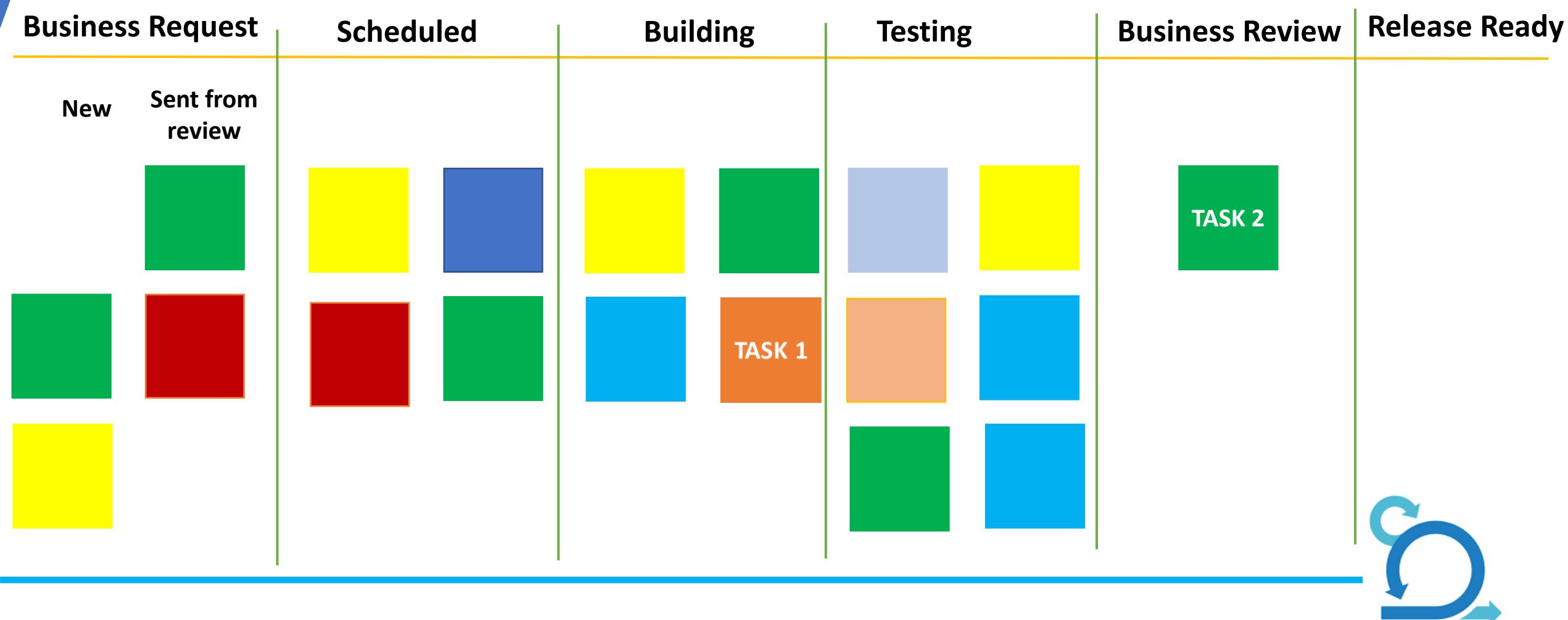
Kanban Board

Finding inefficiencies/issues in the process



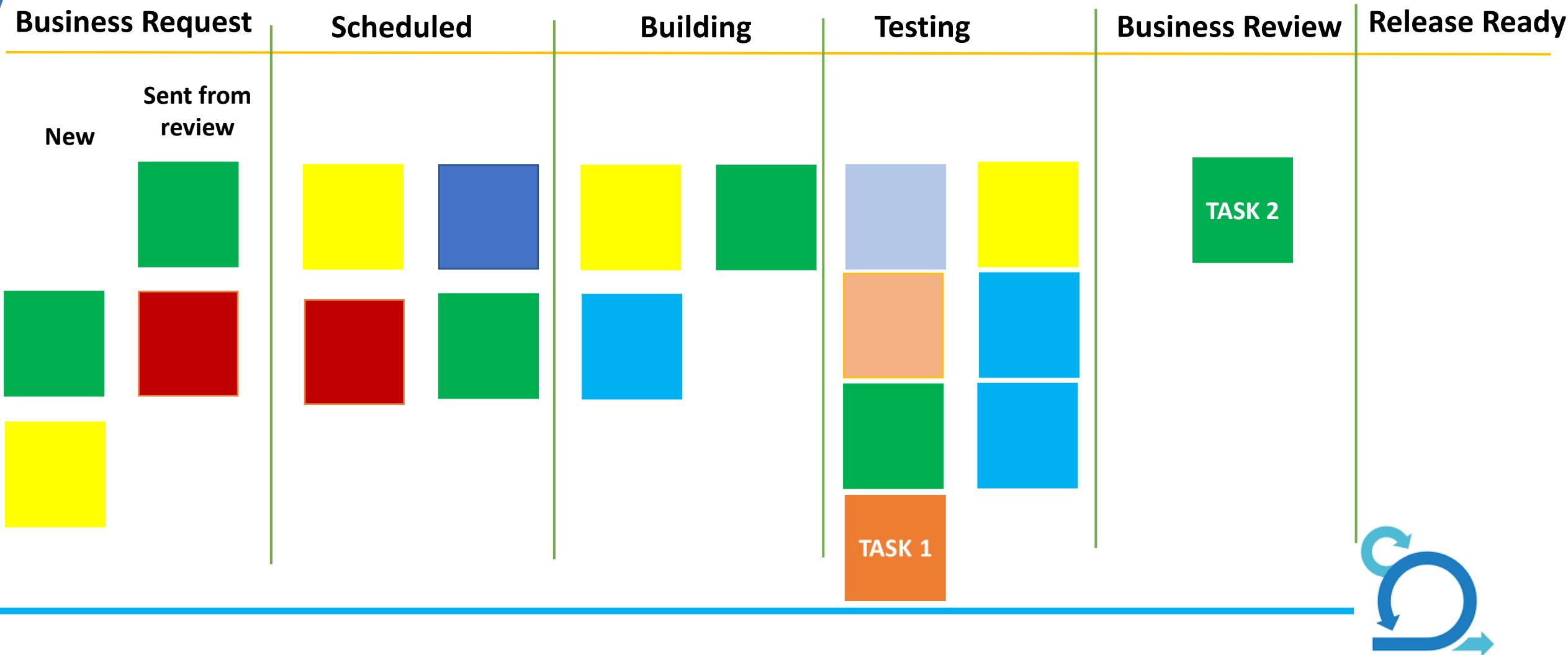
Kanban Board

Finding inefficiencies/issues in the process



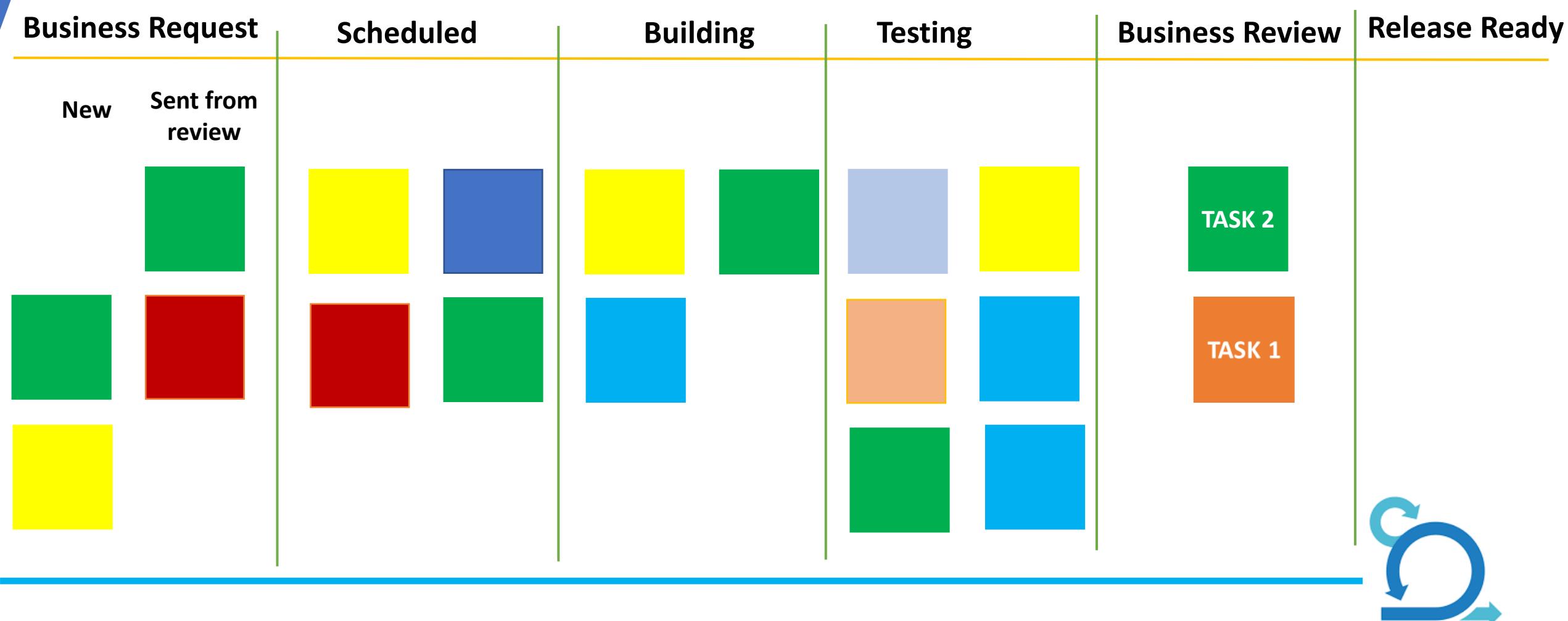
Kanban Board

Finding inefficiencies/issues in the process



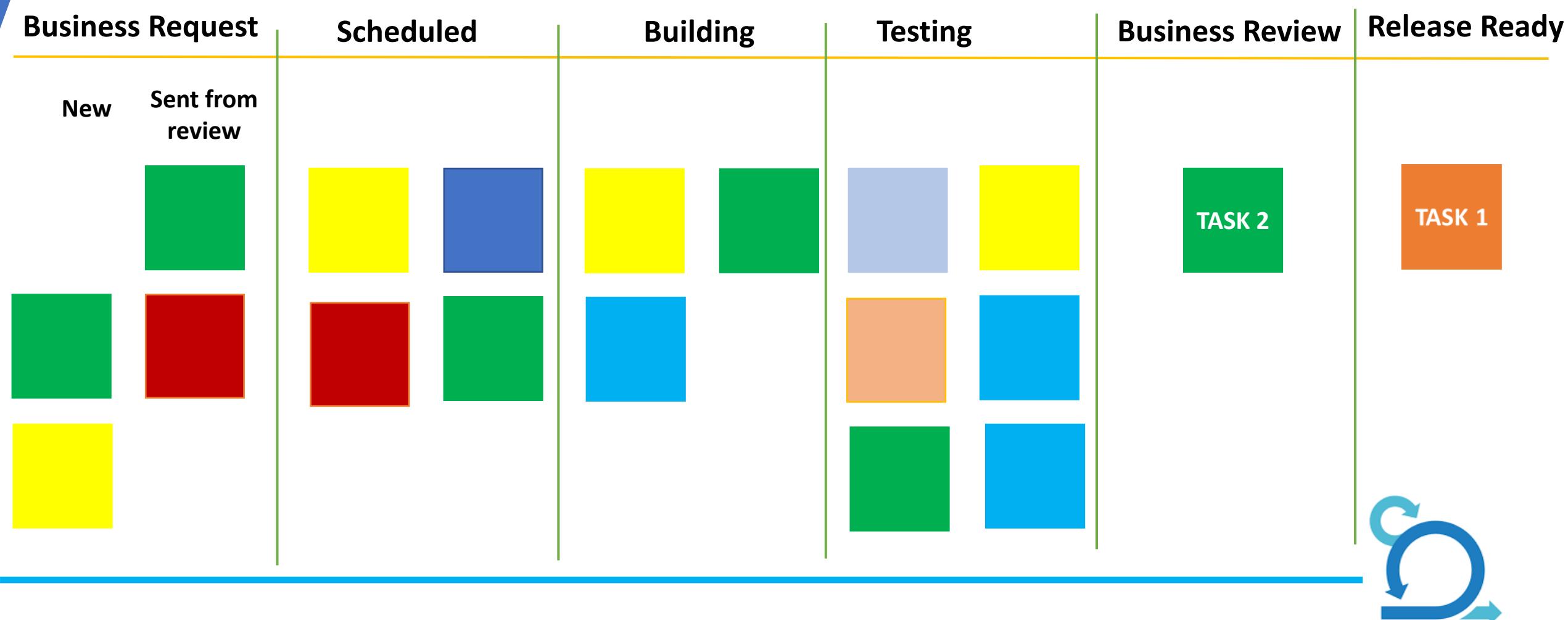
Kanban Board

Finding inefficiencies/issues in the process



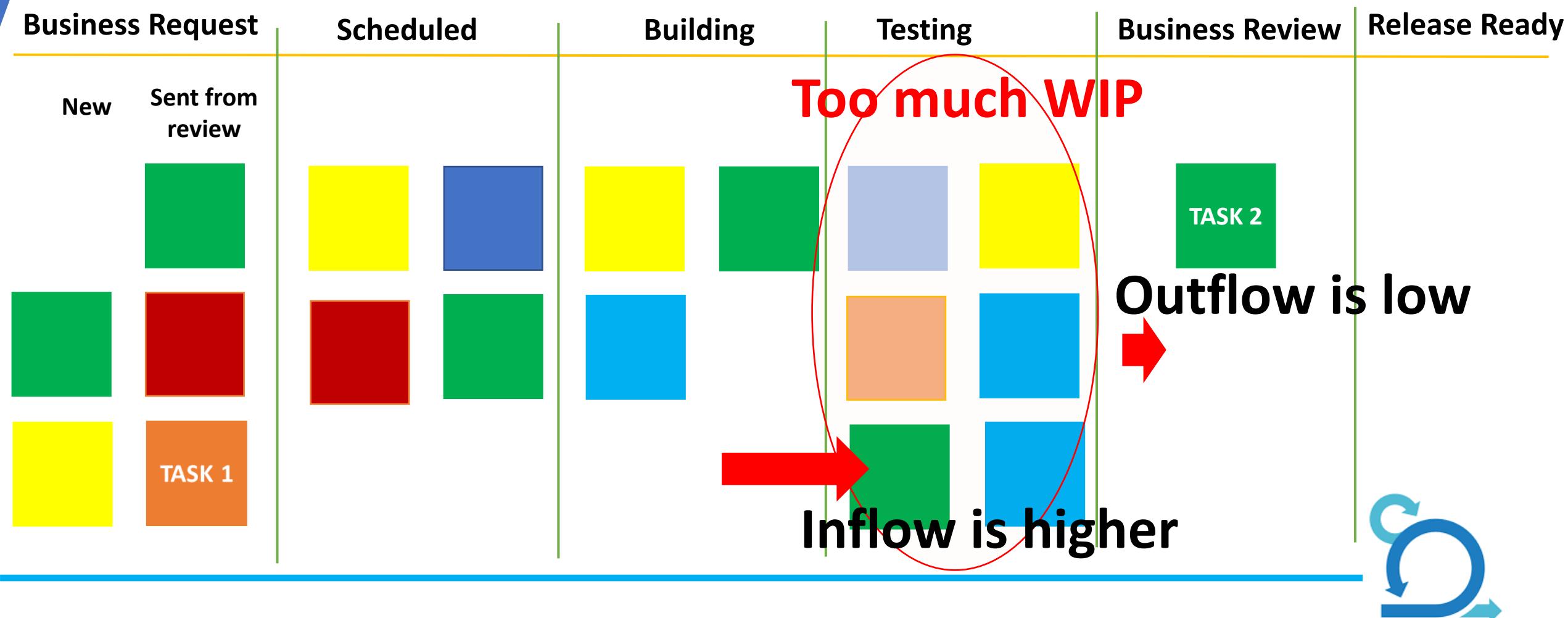
Kanban Board

Finding inefficiencies/issues in the process



Kanban Board

Finding inefficiencies/issues in the process



Kanban Board

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage



Kanban Board

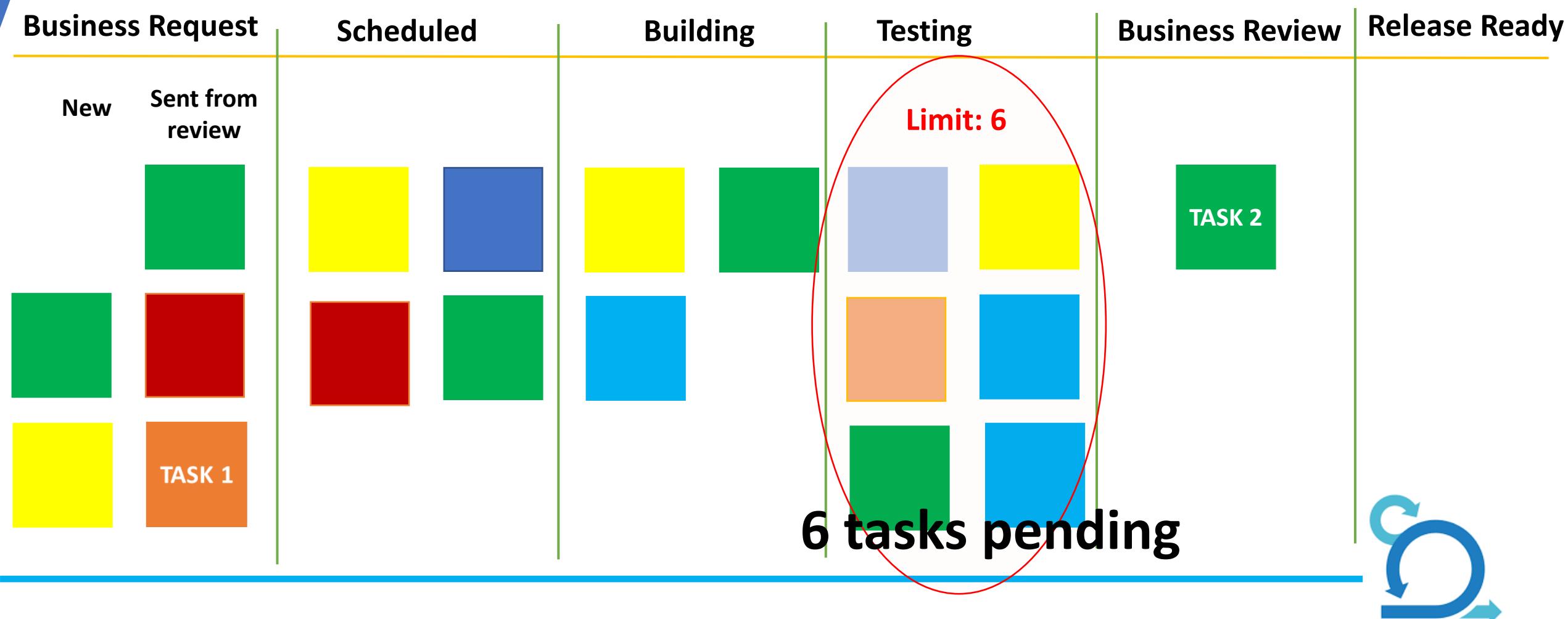
Limiting the WIP

- A **limit is set on the maximum number of tasks** that can be present in a column or that step at a time
- No further inflow into the step until the pending tasks in that step are cleared



Kanban Board

Finding inefficiencies/issues in the process



Kanban Board

Limiting the WIP

- A **limit is set on the maximum number of tasks** that can be present in a column or that step at a time
- No further inflow into the step until the pending tasks in that step are cleared
- WIP limits can be set on all the steps
- How do we decide what limit to set?



Kanban Board

Finding inefficiencies/issues in the process

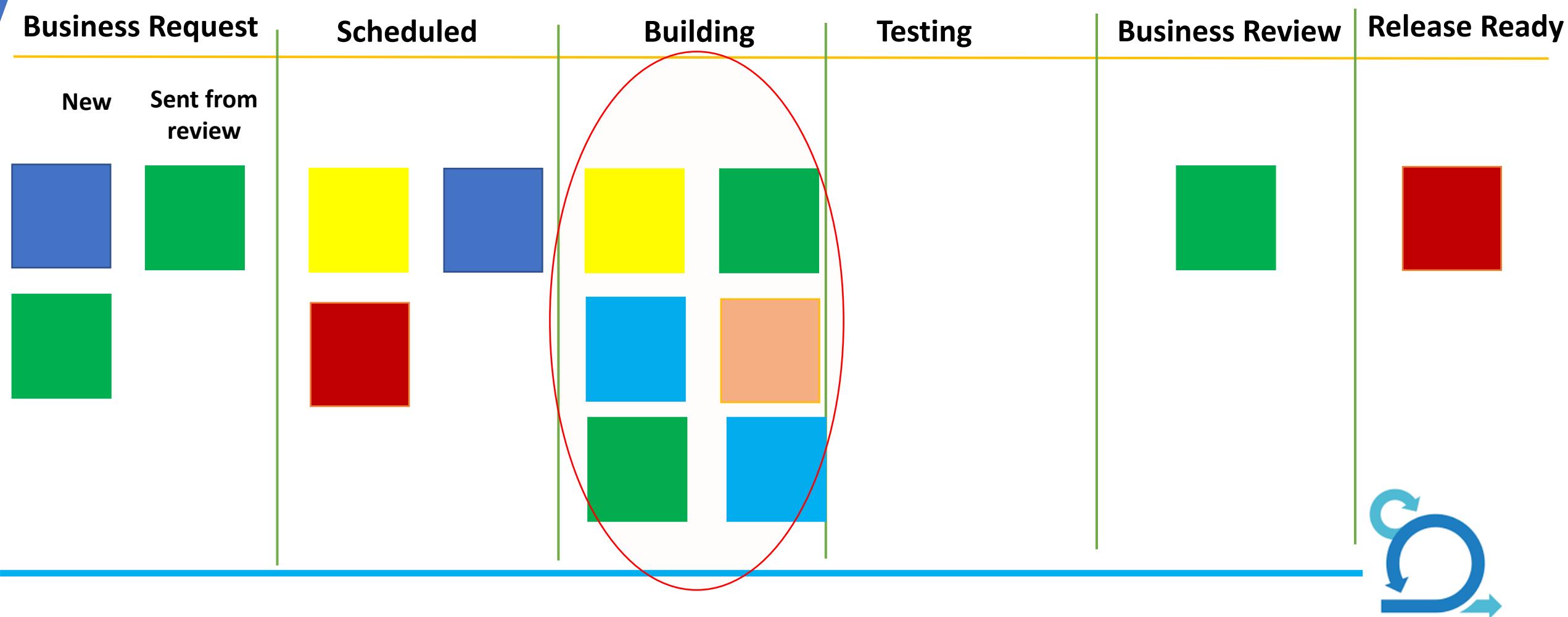
Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage
2	Under utilization of resources Pending tasks not moving to the next step	



Kanban Board



Finding inefficiencies/issues in the process



Kanban Board

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage
2	Under utilization of resources Pending tasks not moving to the next step	Reallocate the resources somewhere else where they can be utilized optimally

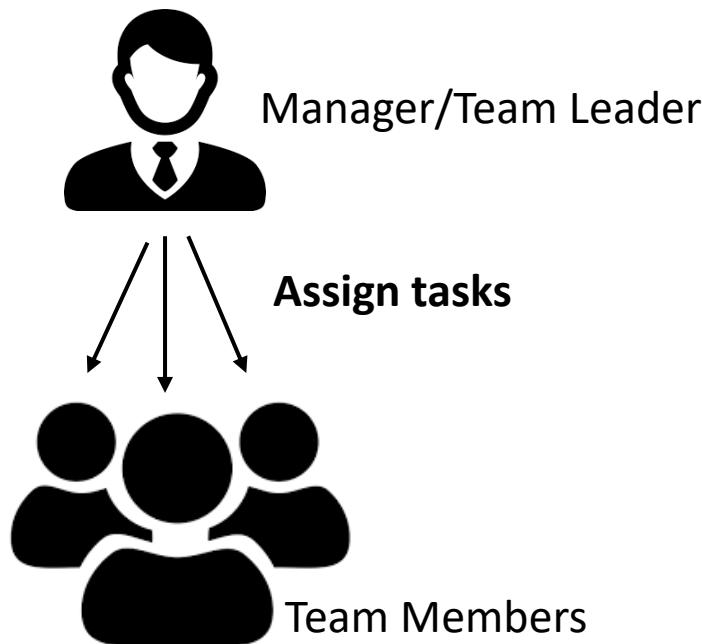
Managing the Flow



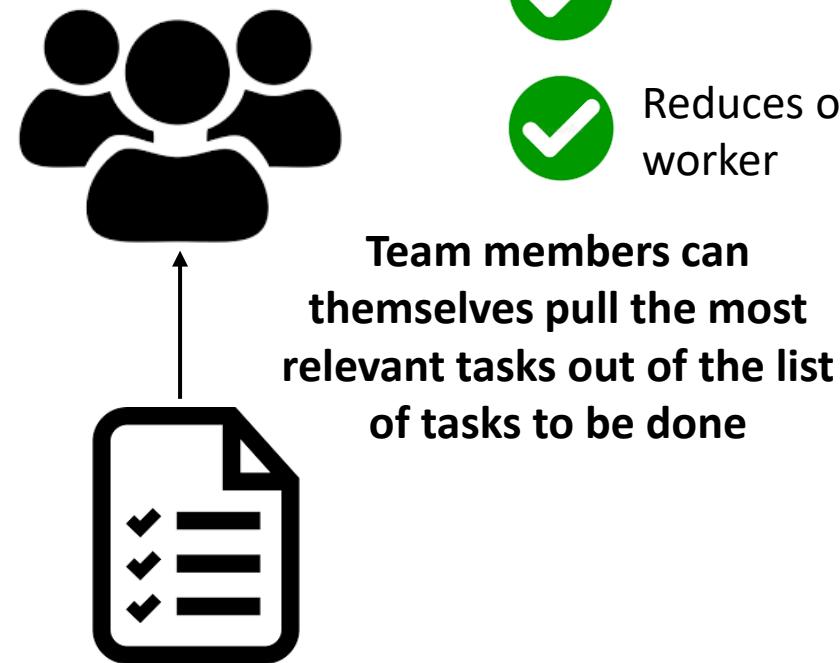
Kanban

Kanban is also known as a **Pull based system**

Push based system



Pull based system

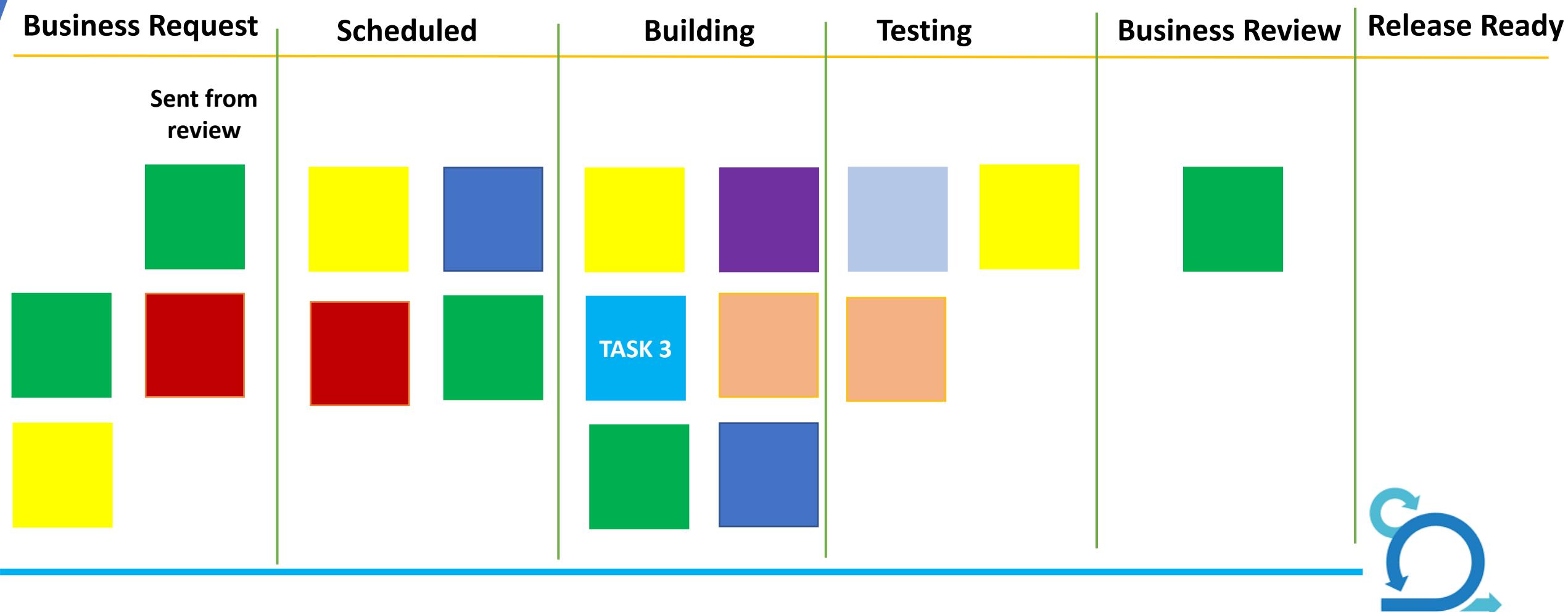


-  Optimizes the work flow
-  Reduces lead time
-  Reduces over load on worker



Kanban Board

Finding inefficiencies/issues in the process



Kanban

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage
2	Under utilization of resources Pending tasks not moving to the next step	Reallocate the resources somewhere else where they can be utilized optimally
3	Task is taking longer time than it should be	



Kanban Board



Finding inefficiencies/issues in the process

Issue: Task is taking longer time than it should be

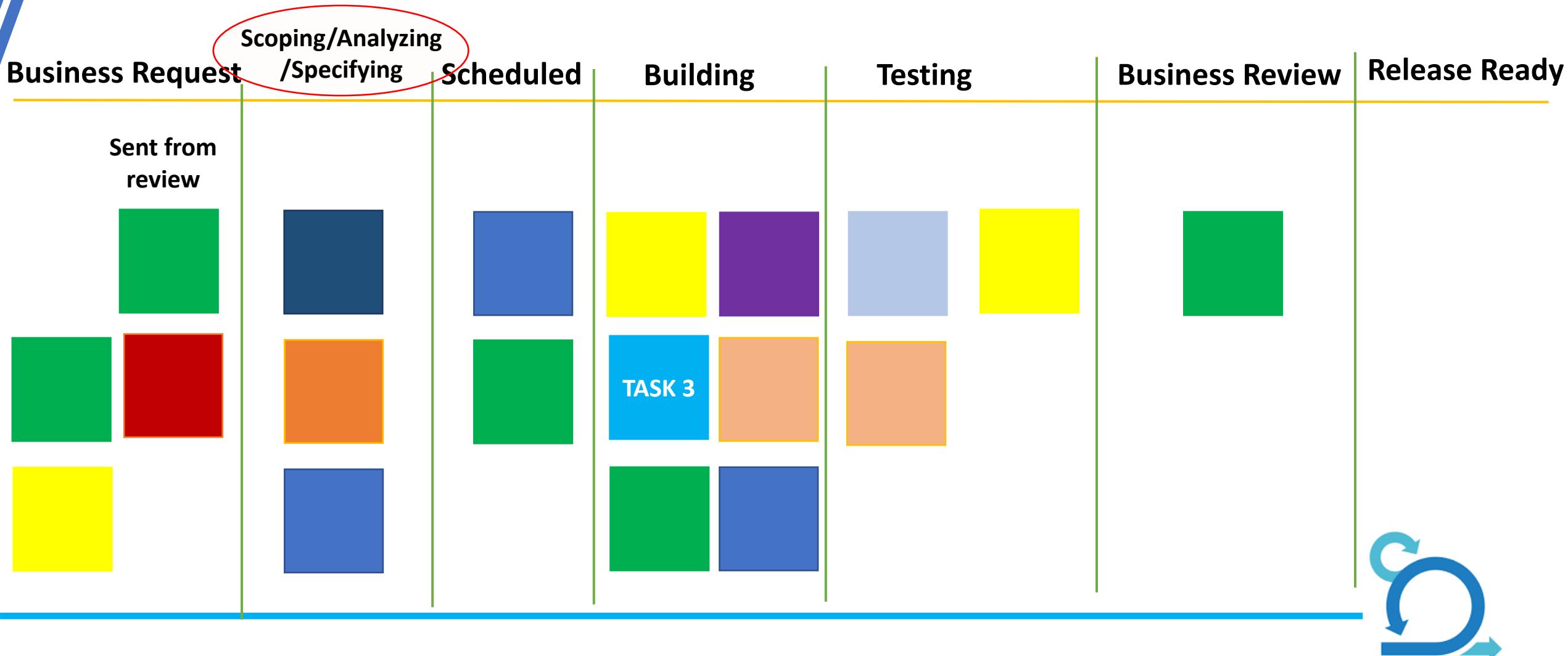
Reason	Suggestions
Unequal sized tasks	Break down the task into smaller sized tasks



Kanban Board



Finding inefficiencies/issues in the process



Kanban Board



Finding inefficiencies/issues in the process

Issue: Task is taking longer time than it should be

Reason	Suggestions
Unequal sized tasks	Break down the task into smaller sized tasks
Task is stuck due to some bug or issue	Keep a regular track of the board



Kanban

Finding inefficiencies/issues in the process

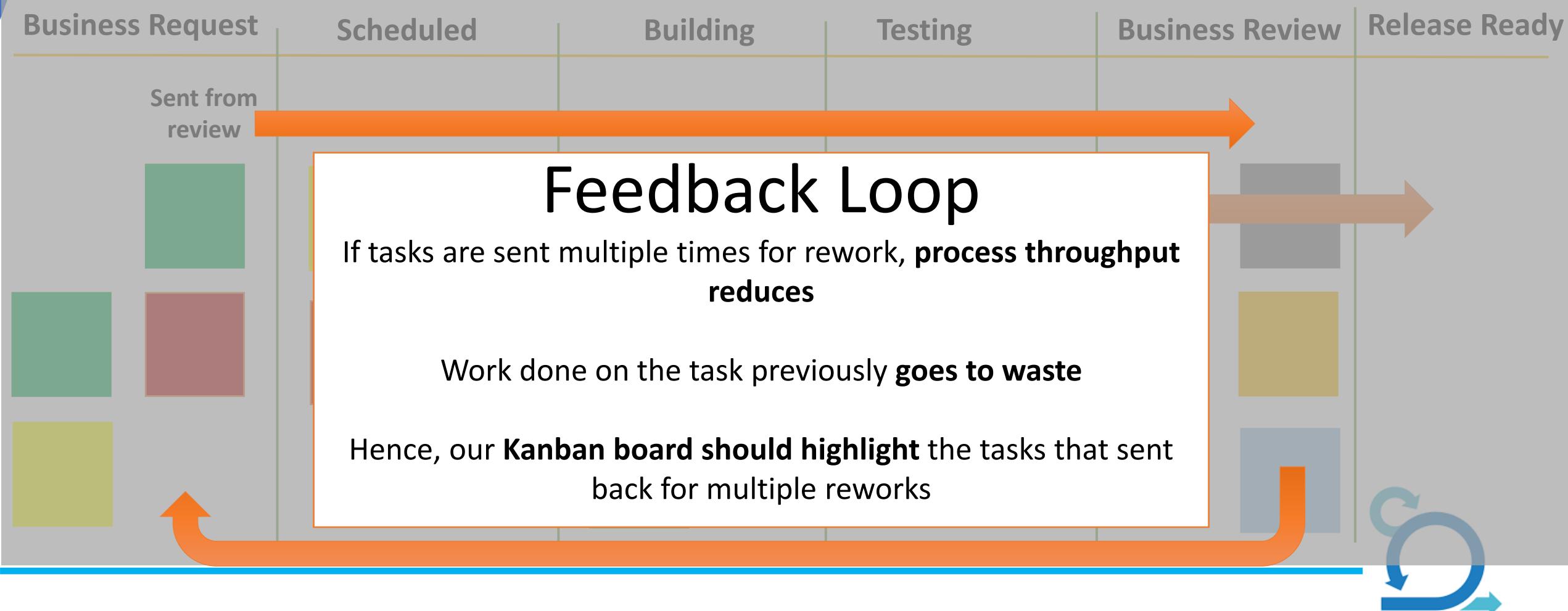
Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage
2	Under utilization of resources Pending tasks not moving to the next step	Reallocate the resources somewhere else where they can be utilized optimally
3	Task is taking longer time than it should be	Keep a regular track of the board Break down the task into smaller sized tasks



Kanban Board

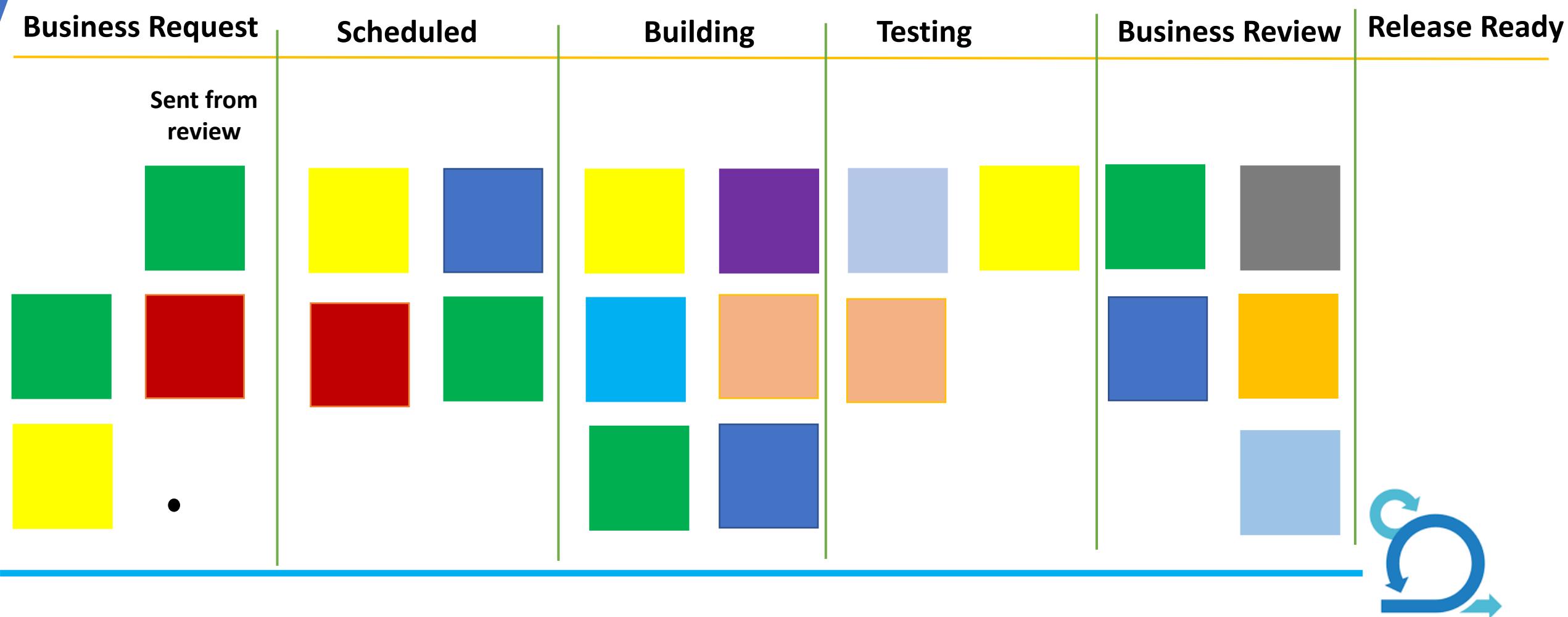


Finding inefficiencies/issues in the process



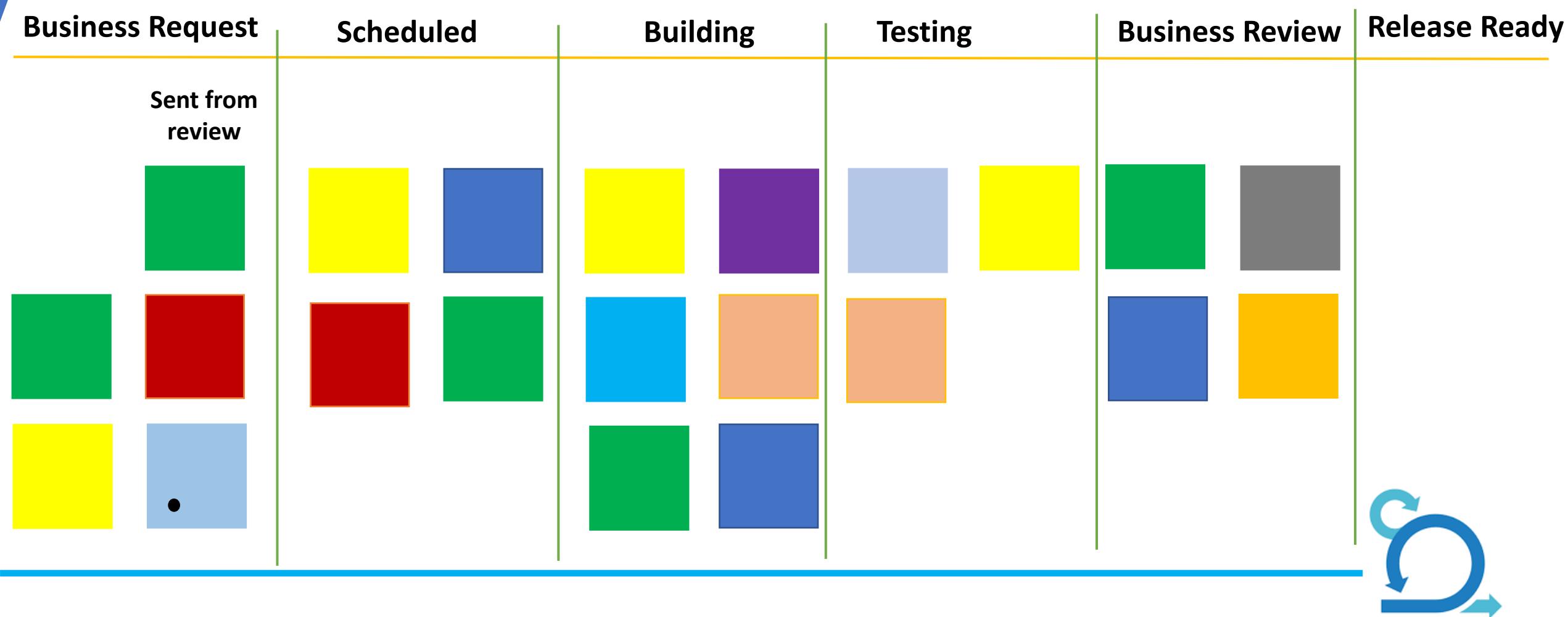
Kanban Board

Finding inefficiencies/issues in the process



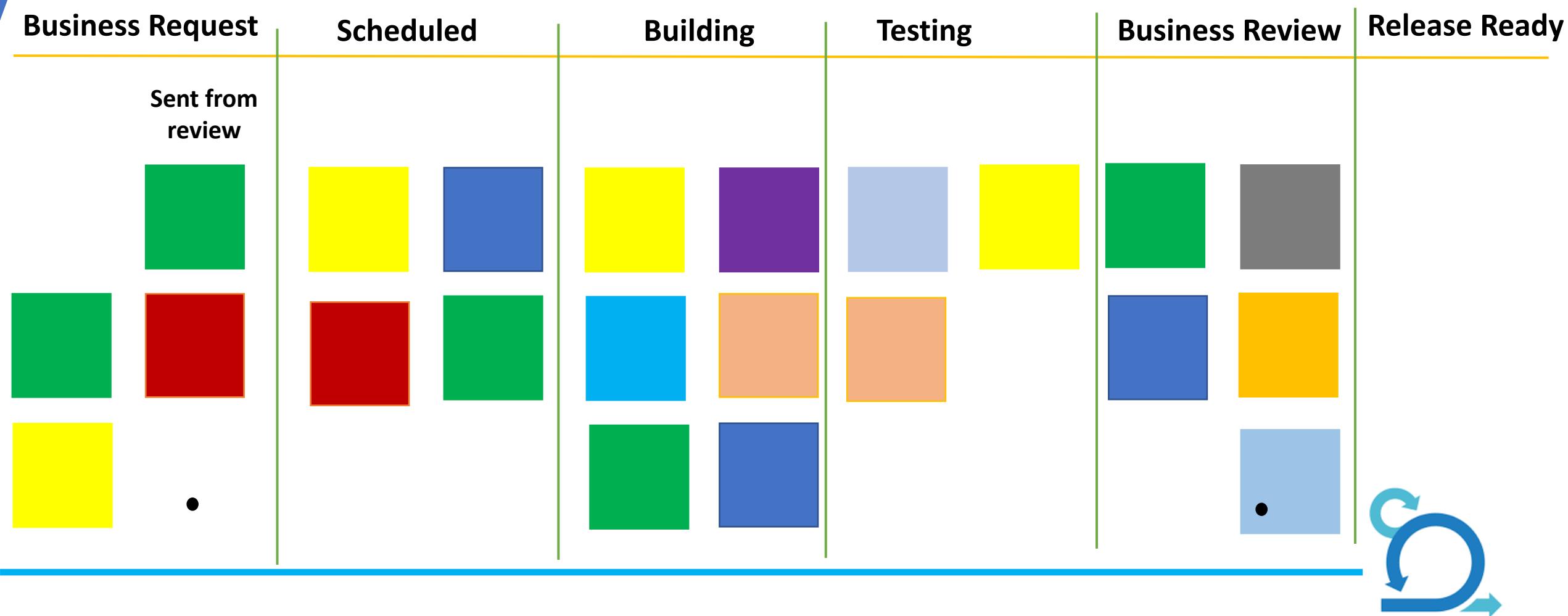
Kanban Board

Finding inefficiencies/issues in the process



Kanban Board

Finding inefficiencies/issues in the process



Kanban

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage
2	Under utilization of resources Pending tasks not moving to the next step	Reallocate the resources somewhere else where they can be utilized optimally
3	Task is taking longer time than it should be	Keep a regular track of the board Break down the task into smaller sized tasks
4	Multiple feedback loops	Make a mark on the task for the number of rounds that task is going for rework



Kanban

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
1	Too much work in progress(WIP) on a single stage Limiting the WIP	More team members need to work on that stage (for e.g. testing stage in this scenario) instead of any of the previous stage
2	Under utilization of resources Pending tasks not moving to the next step	Reallocate the resources somewhere else where they can be utilized optimally
3	Task is taking longer time than it should be	Keep a regular track of the board Break down the task into smaller sized tasks
4	Multiple feedback loops	Make a mark on the task for the number of rounds that task is going for rework Find optimum WIP limit for your review step



Finding inefficiencies/issues in the process



Kanban

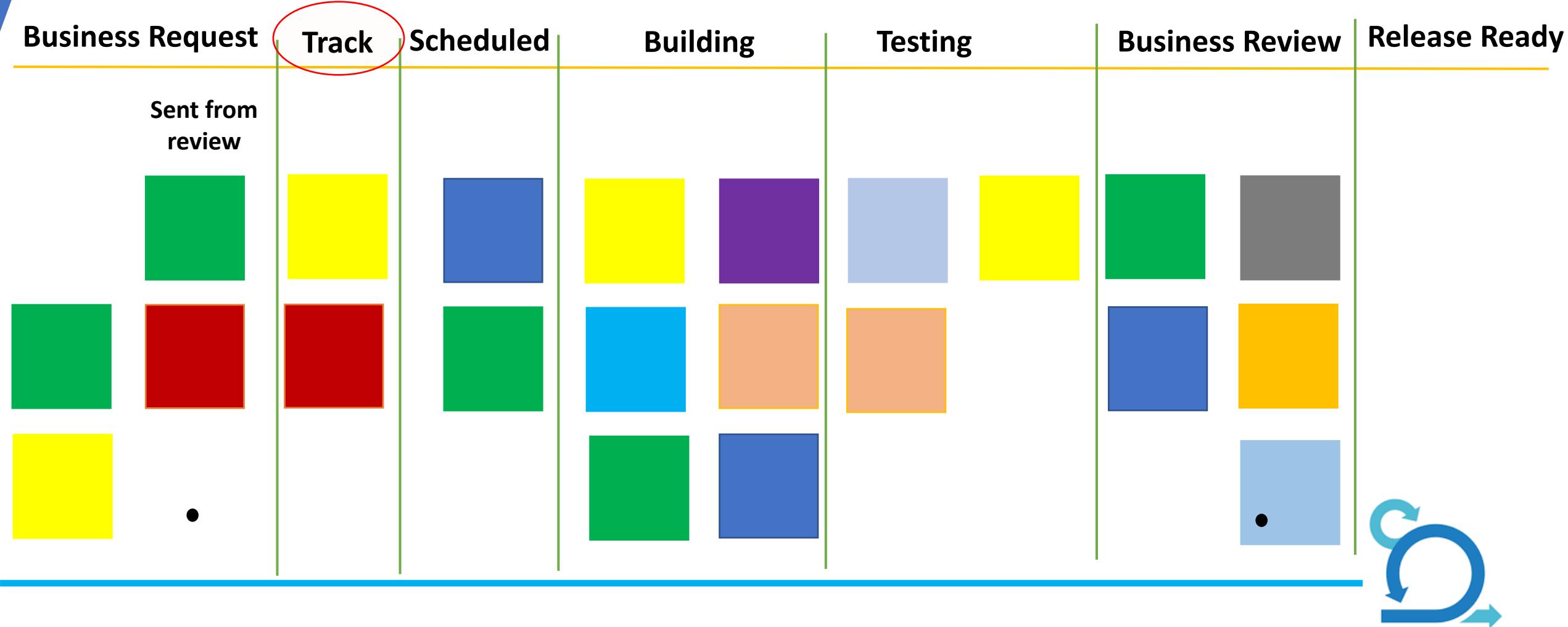
Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
4	Having a lot of external dependencies	



Kanban Board

Finding inefficiencies/issues in the process



Kanban

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
4	Having a lot of external dependencies	Develop a practice of regular follow ups on tasks listed in “Tracking” column No need of a WIP limit on this column



Kanban

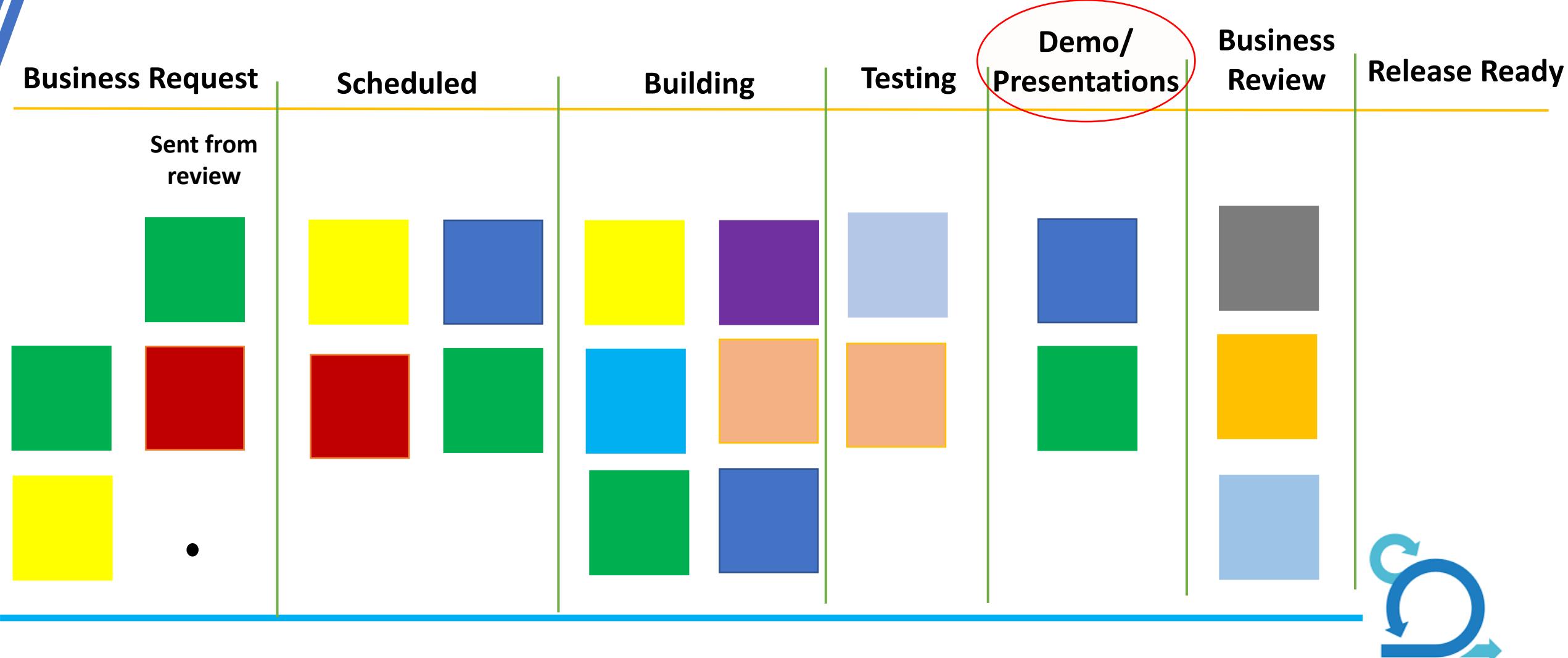
Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
4	Having a lot of external dependencies	Develop a practice of regular follow ups on tasks listed in “Tracking” column No need of a WIP limit on this column
5	Team required to work on non-product features	



Kanban Board

Finding inefficiencies/issues in the process



Kanban

Finding inefficiencies/issues in the process

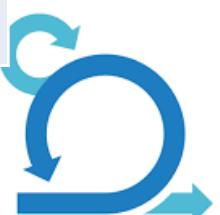
Sl. no.	Situation	Suggestion
4	Having a lot of external dependencies	Develop a practice of regular follow ups on tasks listed in “Tracking” column No need of a WIP limit on this column
5	Team required to work on non-product features	Create a task for such features and keep a track on the same
6	Automate/Upgrade tasks	Add such improvement tasks also on the board



Kanban

Finding inefficiencies/issues in the process

Sl. no.	Situation	Suggestion
4	Having a lot of external dependencies	Develop a practice of regular follow ups on tasks listed in “Tracking” column No need of a WIP limit on this column
5	Team required to work on non-product features	Create a task for such features and keep a track on the same
6	Automate/Upgrade tasks	Add such improvement tasks also on the board
7	Assigning work to new person	Assign the new person to the slowest step



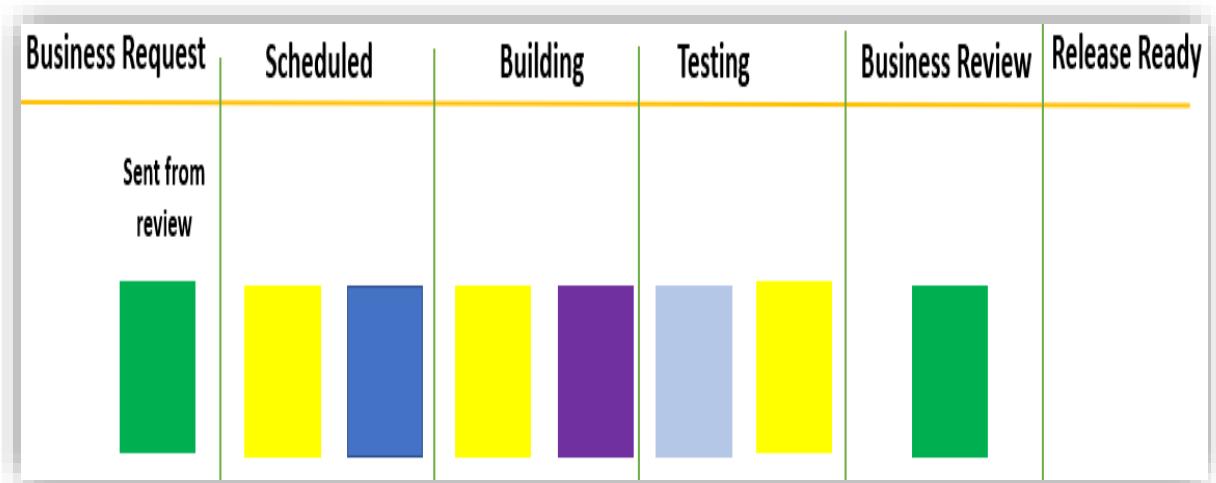
Defining “Done”



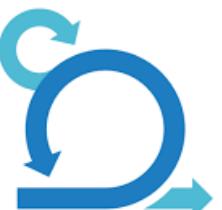
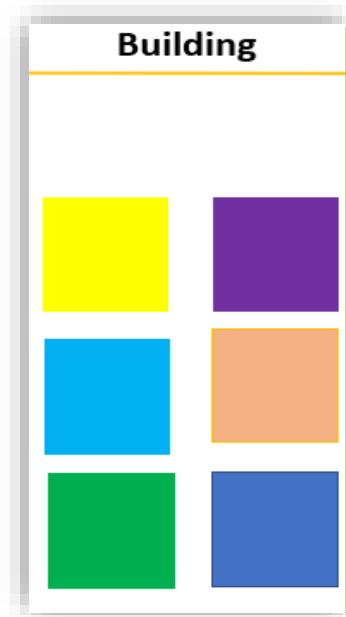
Kanban

Defining “Done”

Define done for each feature

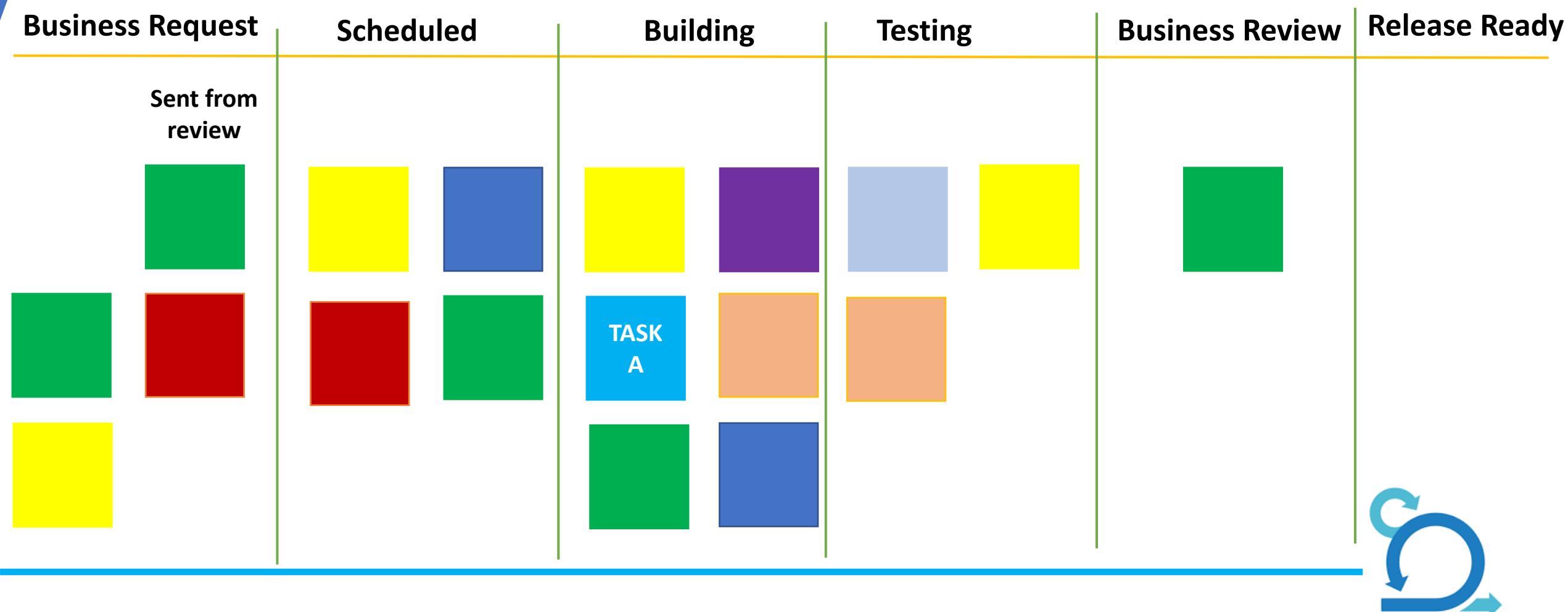


Define completion for each step

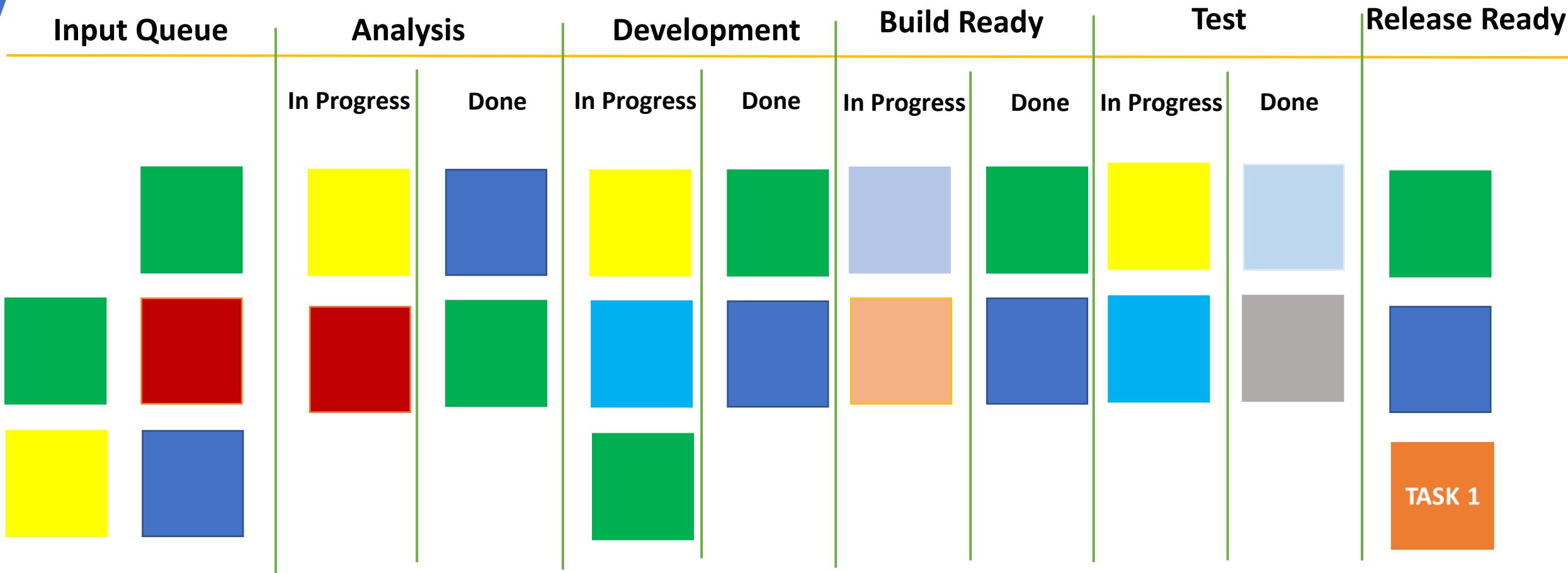


Kanban

Finding inefficiencies/issues in the process

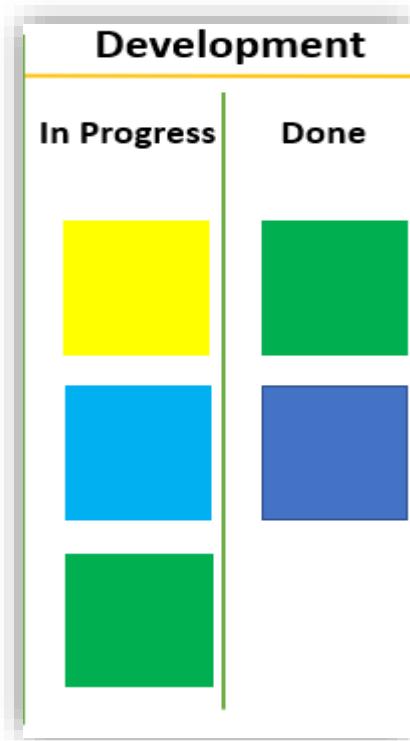


Kanban



Kanban

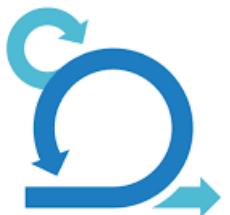
Define completion for each step



- If a step is complete on a task, it does **not necessarily mean that the task moves to the next step**
- **Task will simply sit in the done column of the previous step**
- Do not have separate WIP limits for sub columns “In progress” and “Done”
- **Write “Done” rules as notes on the Kanban board**
- **Always check if the rules for “Done” were met in the previous step or not**



The Daily Standup



The Daily Standup



Not a review meeting or demo of features built or discussion about the product etc.

Scrum has rules on:

- How long the meeting will be?
- Who will be running the meeting?
- What will be discussed in the meeting ?



Kanban is flexible with no such rules

- Any team member can run the meeting even a fresher
- No fixed duration of the meeting, depends on the team's experience
- Daily stand up is the time to get the board up-to-date, identify issues, highlight problems if any etc.



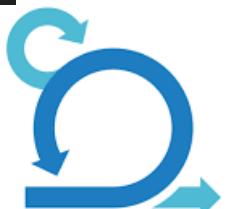
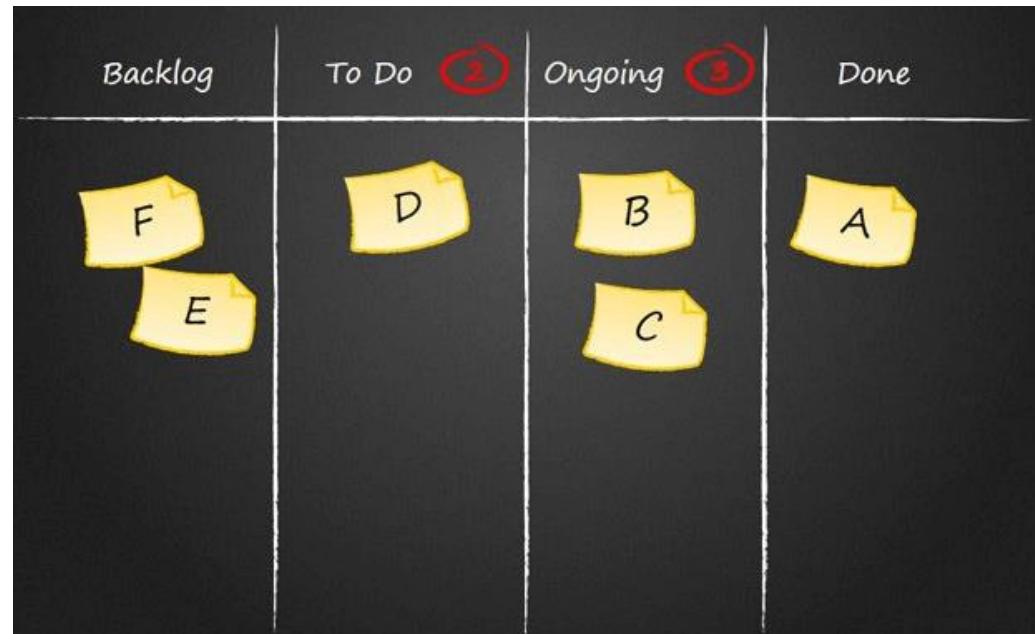
Specifying rules



Specifying rules

Ensure rules are explicitly mentioned and everyone understands them clearly

- Writing WIP limit on top of the column
- Define “Done”



Kanban Flow Metrics



“By when will the work be completed?”

or

“How much of the work will be completed in this release?”

Answer these
questions based
on metrics



**To predict output of a Process, Process needs to be
Predictable/Uniform**

Note down key
metrics

If process is
predictable



Kanban Flow Metrics

WIP: Work in Progress

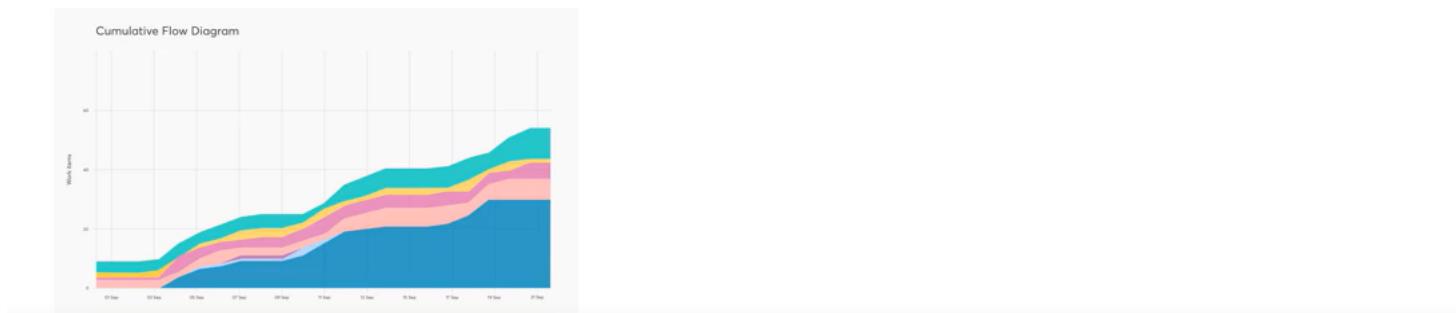
Throughput

Cycle Time

Little's Law

Cumulative Flow Diagram(CFD)

For visualization of these metrics



It can be measured even if Kanban is not implemented

Kanban Flow Metrics: Work in progress

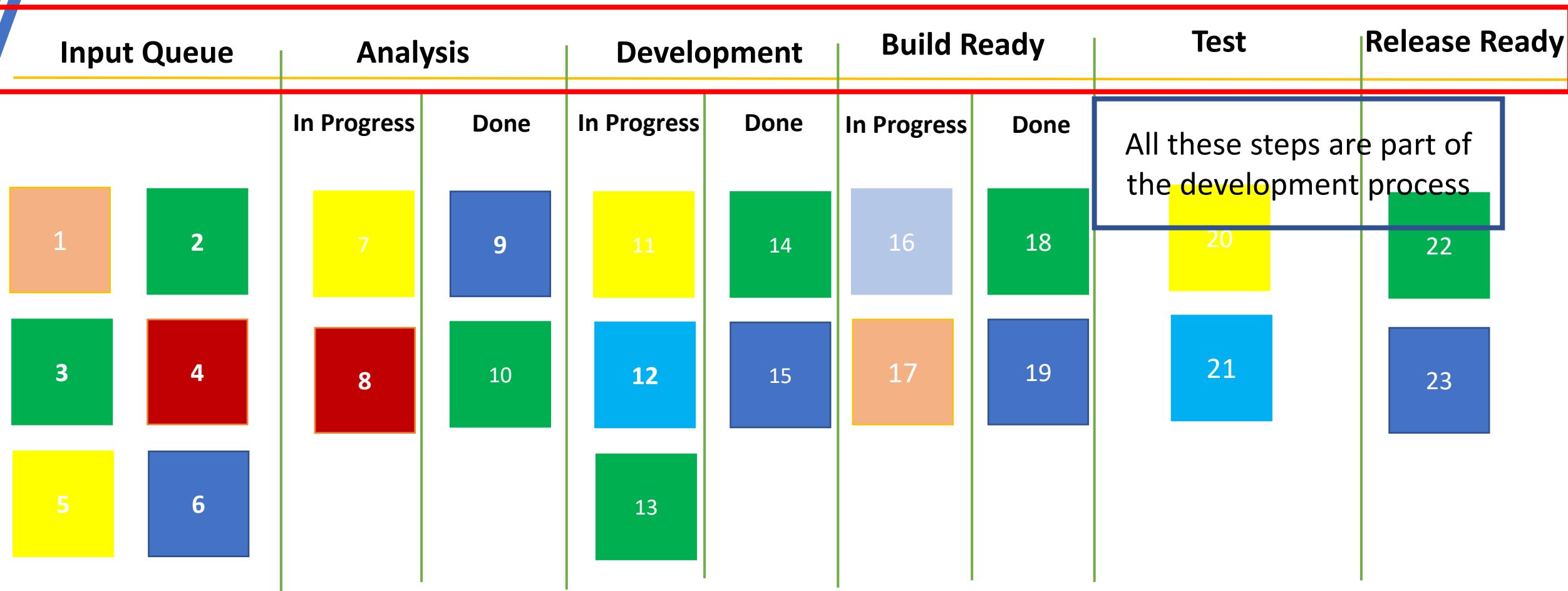


Kanban board clearly shows started/ completed items

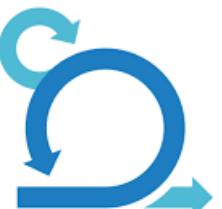
Number of items on which the team is currently working



Work in Progress



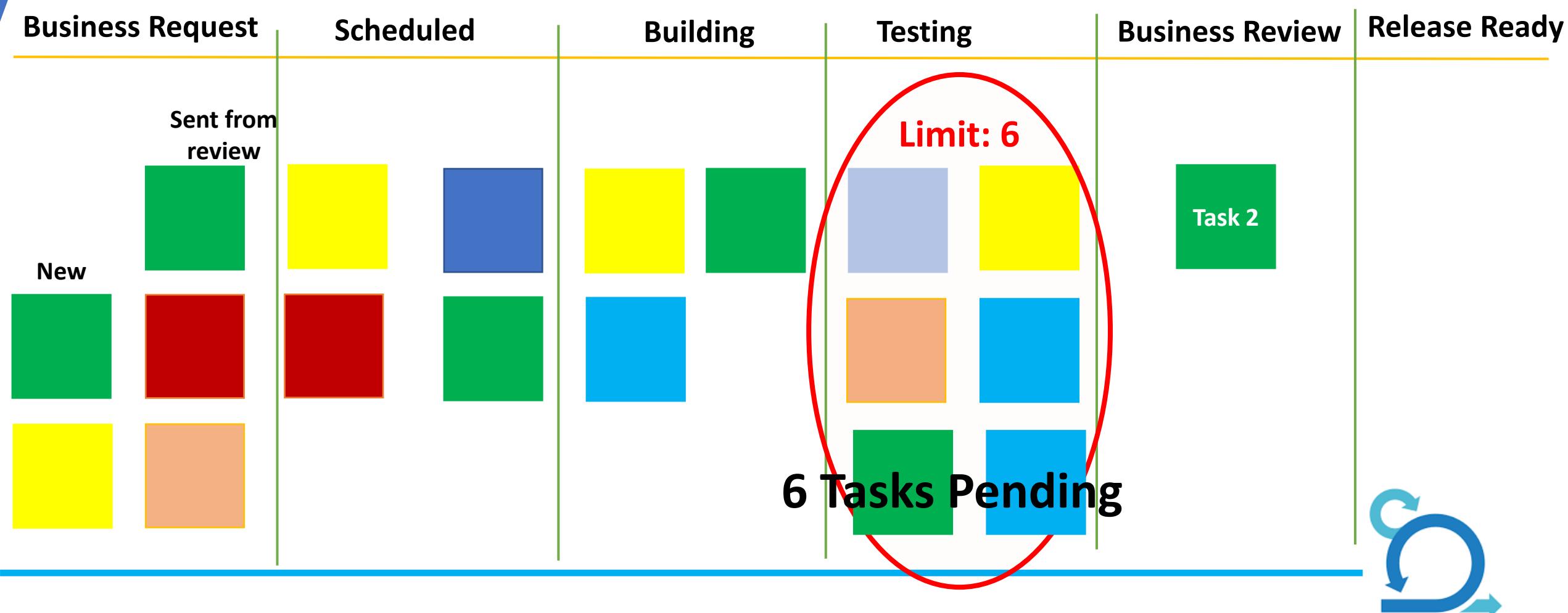
WIP = 23



WIP Limit



Finding inefficiencies/issues in the process



Cycle Time

Cycle Time: Total time a task takes to complete / Amount of time for which a work item is in work in progress state



Day 5
Input Queue

Release Ready

Day 20



Cycle Time= 15 Days



Cycle Time

Why Cycle Time?

- To answer Customer's questions like "When will the task be done?"
- To estimate development cost for the task
- To calculate Flow Efficiency



Flow Efficiency = Total time team worked on the item/Total time item was in WIP



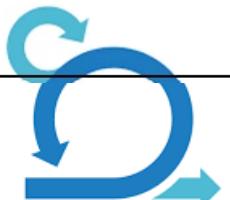
Cycle Time

Flow Efficiency = Total time team worked on the item/Total time item was in WIP

- **Total time team worked on the item = Number of days the task was WIP – Number of days no one worked on the task**
 - Number of days no one worked on the task = **3 days**
 - 2 days (No. of days item stayed in done column of development and not picked up by testing team)
 - + - 1 day (No. of days item stayed in done column of testing and not picked up for Business Review)
- **Total time item was in WIP or Cycle time = 15 days**

Total time for which team actively worked on the item = 15 days – 3 days = 12 days

Flow Efficiency = 12/15 = 0.8 = 80%



Throughput

Amount of work items completed in a unit time

Amount of work items = No. of tasks implemented on the Kanban Board
or No. of stories or no. of features implemented

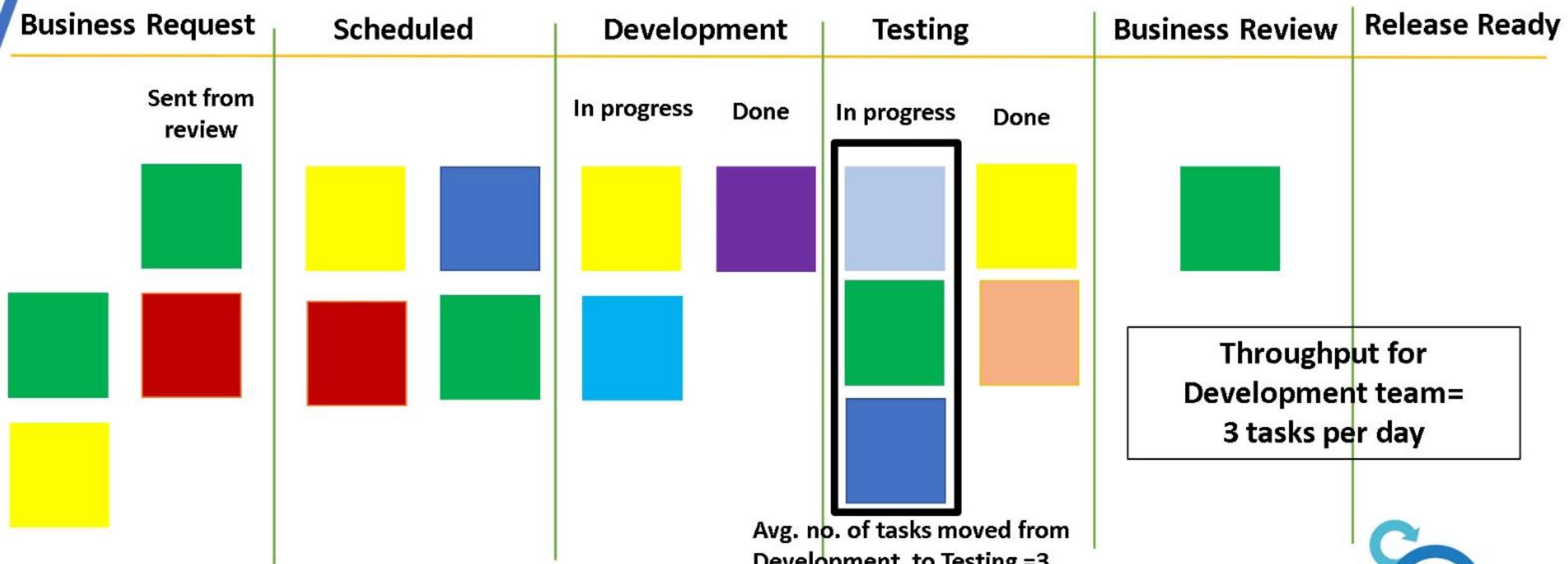
Unit time can be measured in days or months or sprints

Example: 10 tasks per week or 2 feature per month



Throughput

Throughput for each step of the process



Throughput

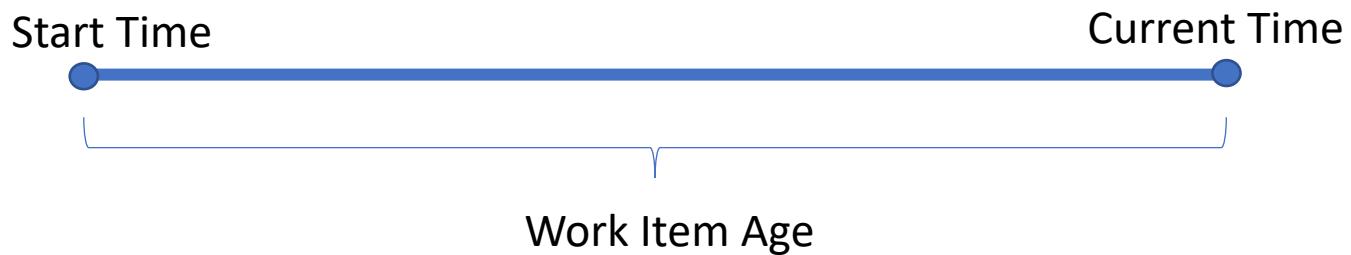
Throughput is not the same as arrival rate

Velocity and Throughput are same, only units may be different



Work Item Age

Elapsed time between when a work item "started" and the current time



Provides Transparency to the flow



Agile Team tracks the start date of each item to calculate Work Item Age



Work Item Age

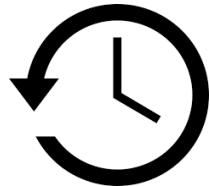
Work Item Age



Leading Indicator

Relevant for non-finished item

Cycle Time



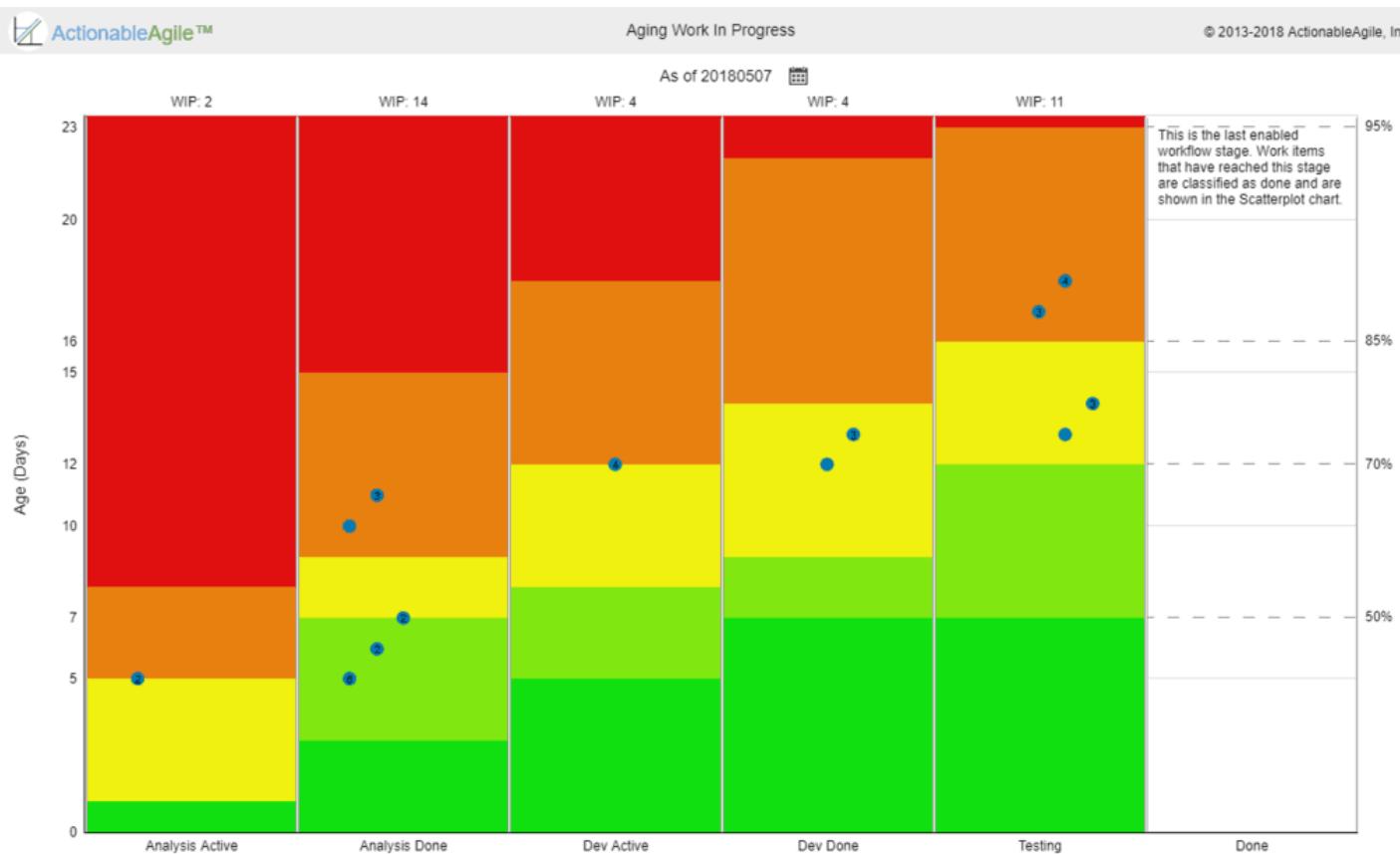
Lagging Indicator

Relevant for finished items



Work Item Age

Ageing Work in Progress Chart



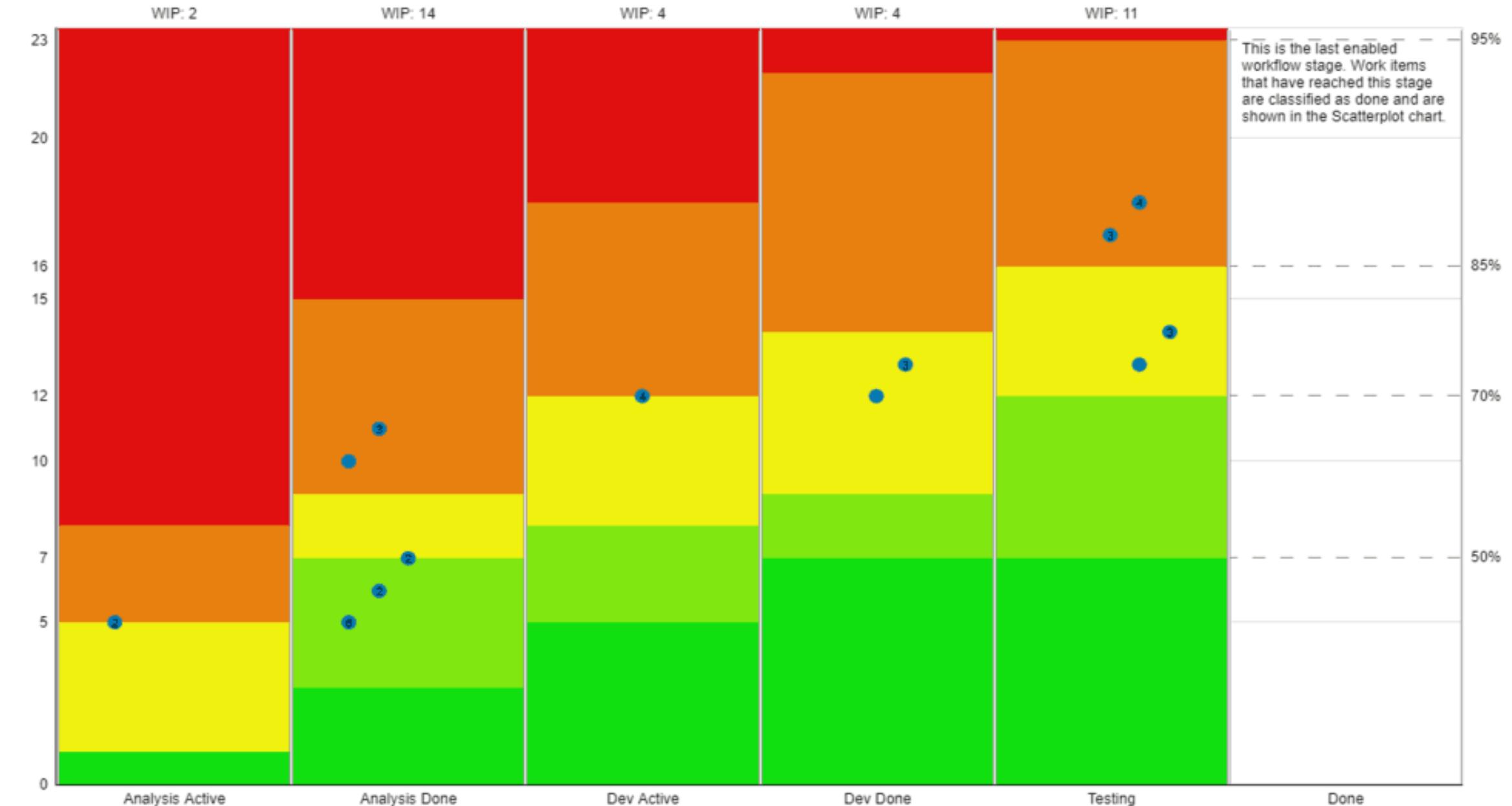
Applications

- Help identify items that are struggling and require attention
- Calculate the amount of time for which item has been under WIP on Kanban Board
- Predict flow risks based on historical cycle times



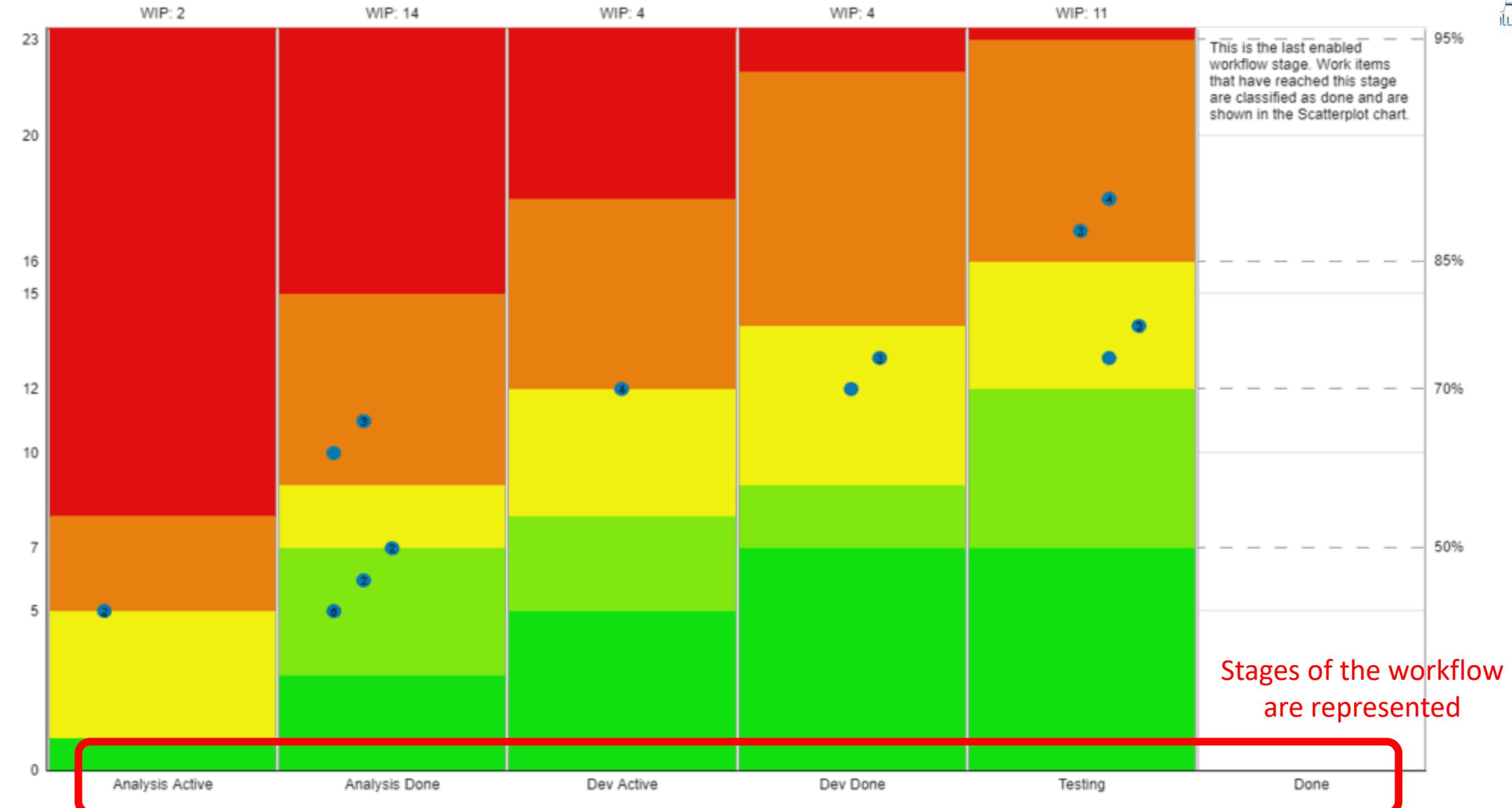


As of 20180507





As of 20180507





As of 20180507

WIP: 2

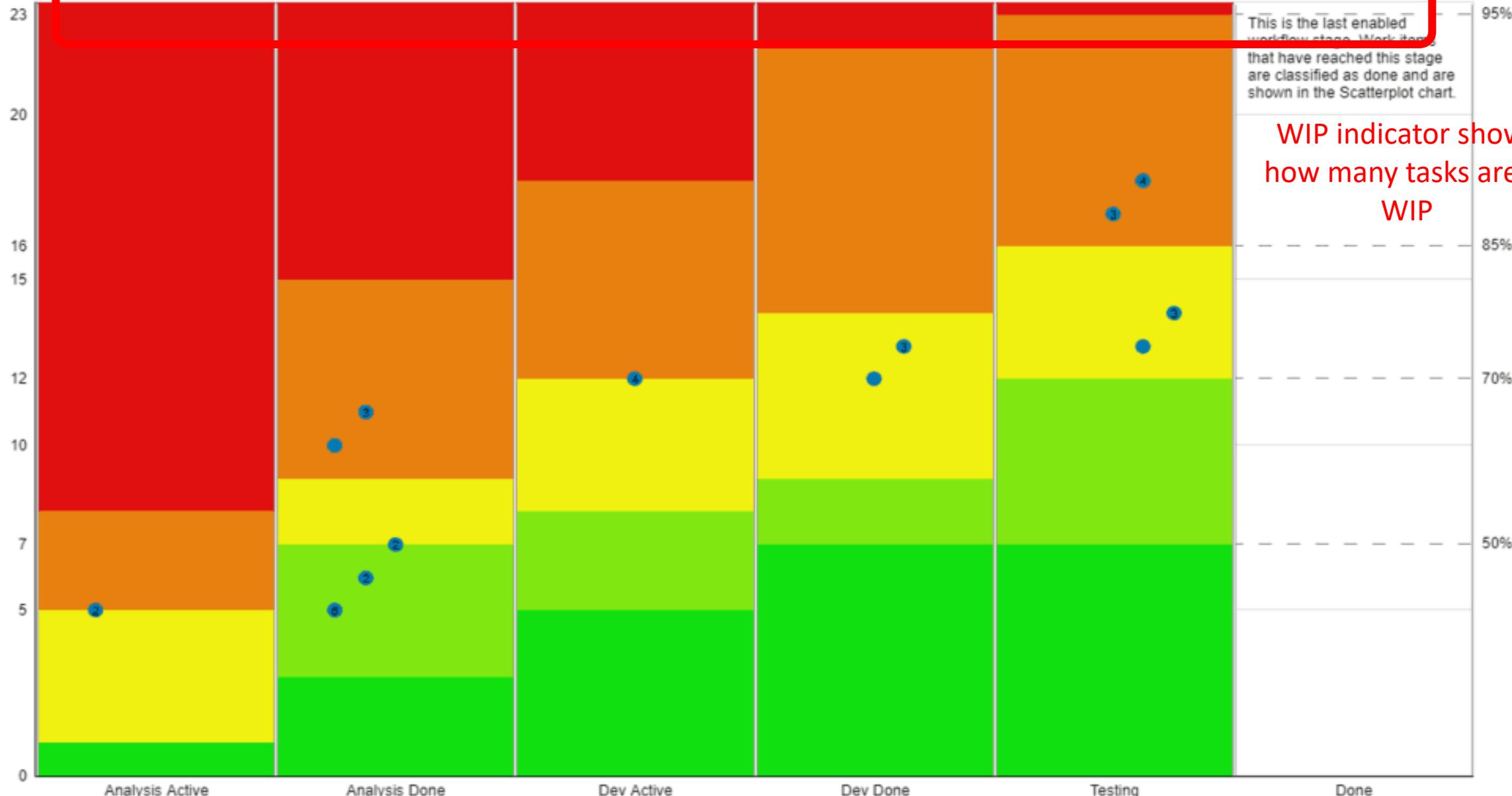
WIP: 14

WIP: 4

WIP: 4

WIP: 11

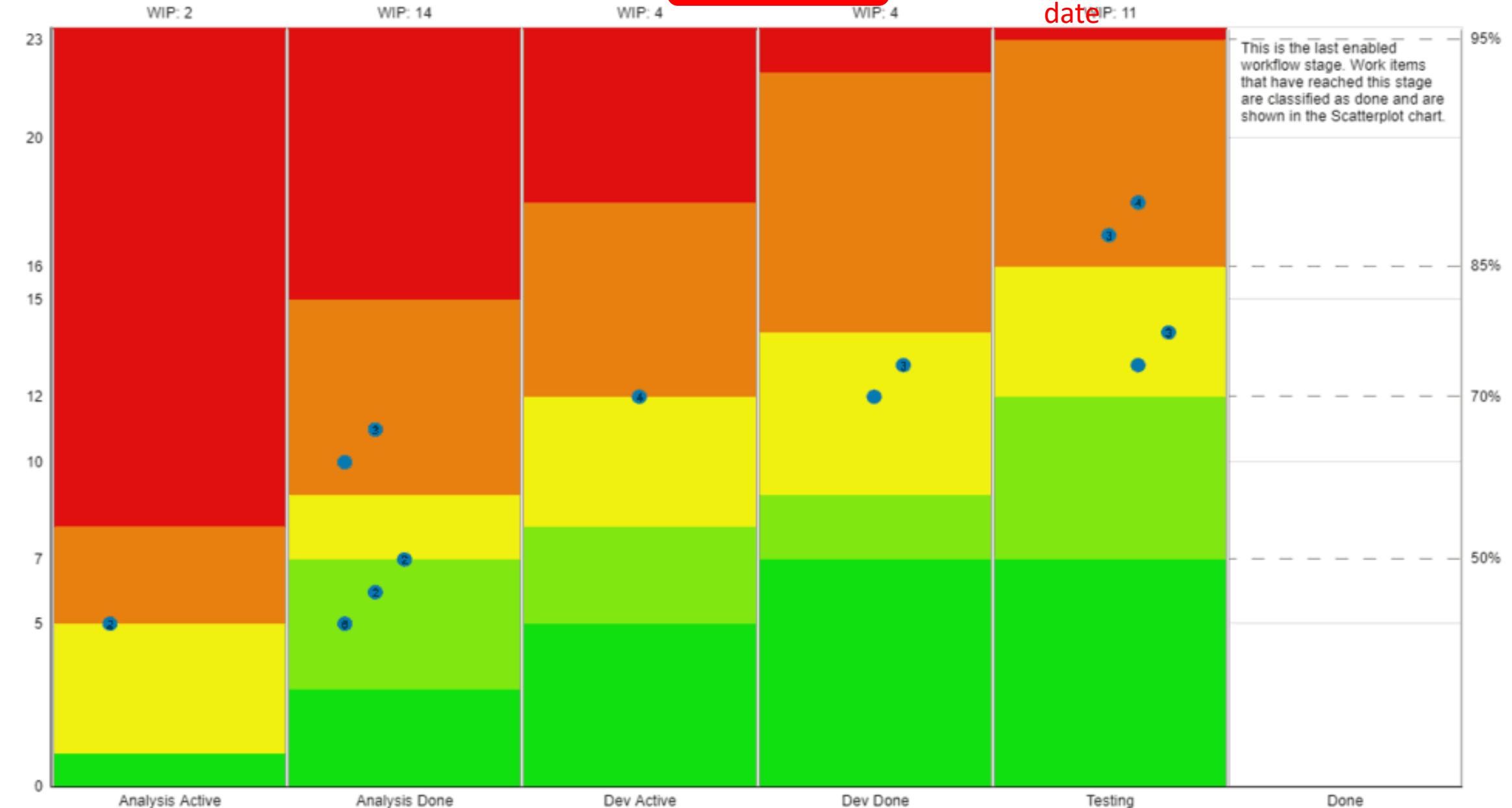
Age (Days)





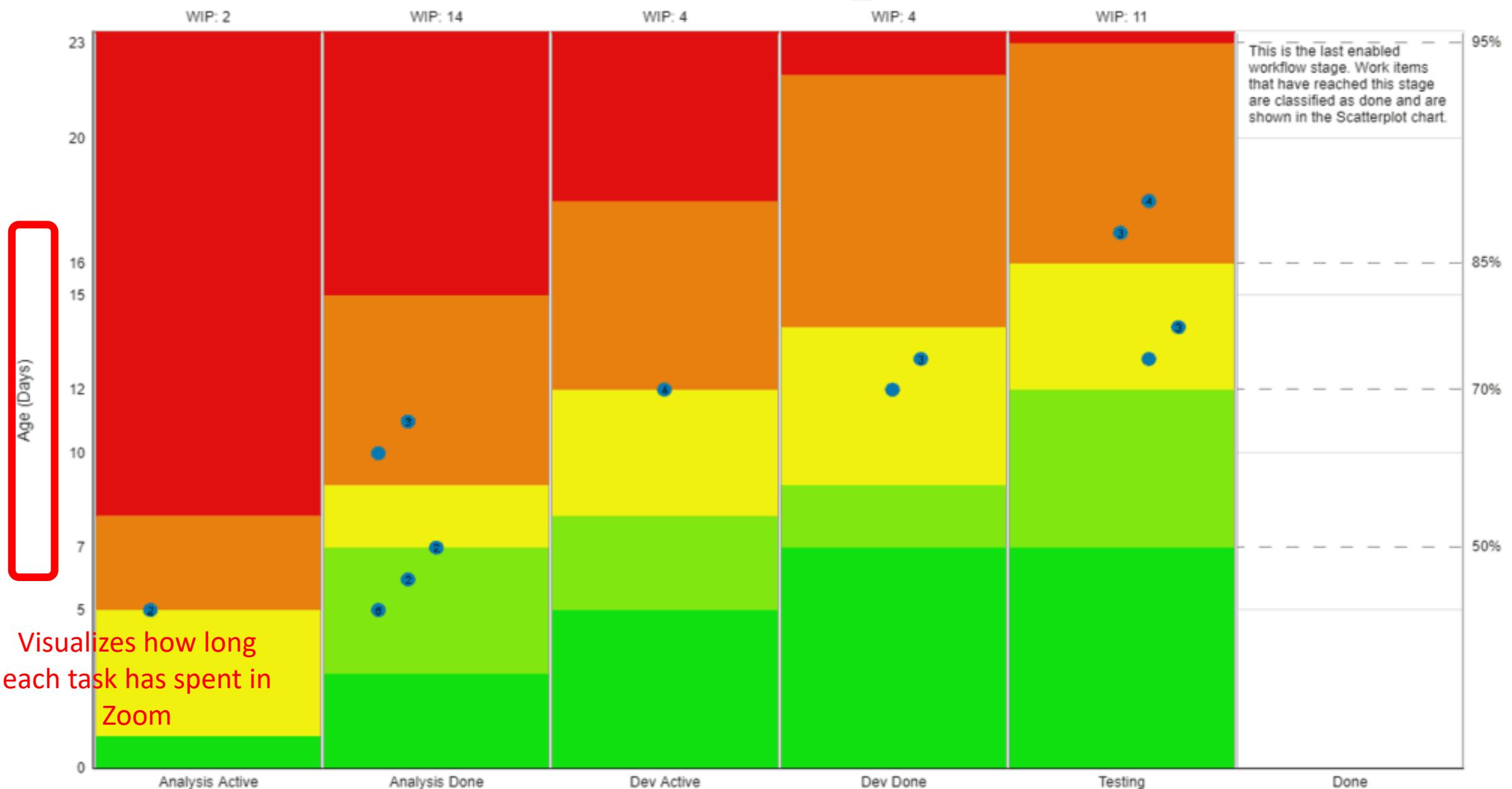
As of 20180507

Calendar item
indicating the current
date





As of 20180507





As of 20180507

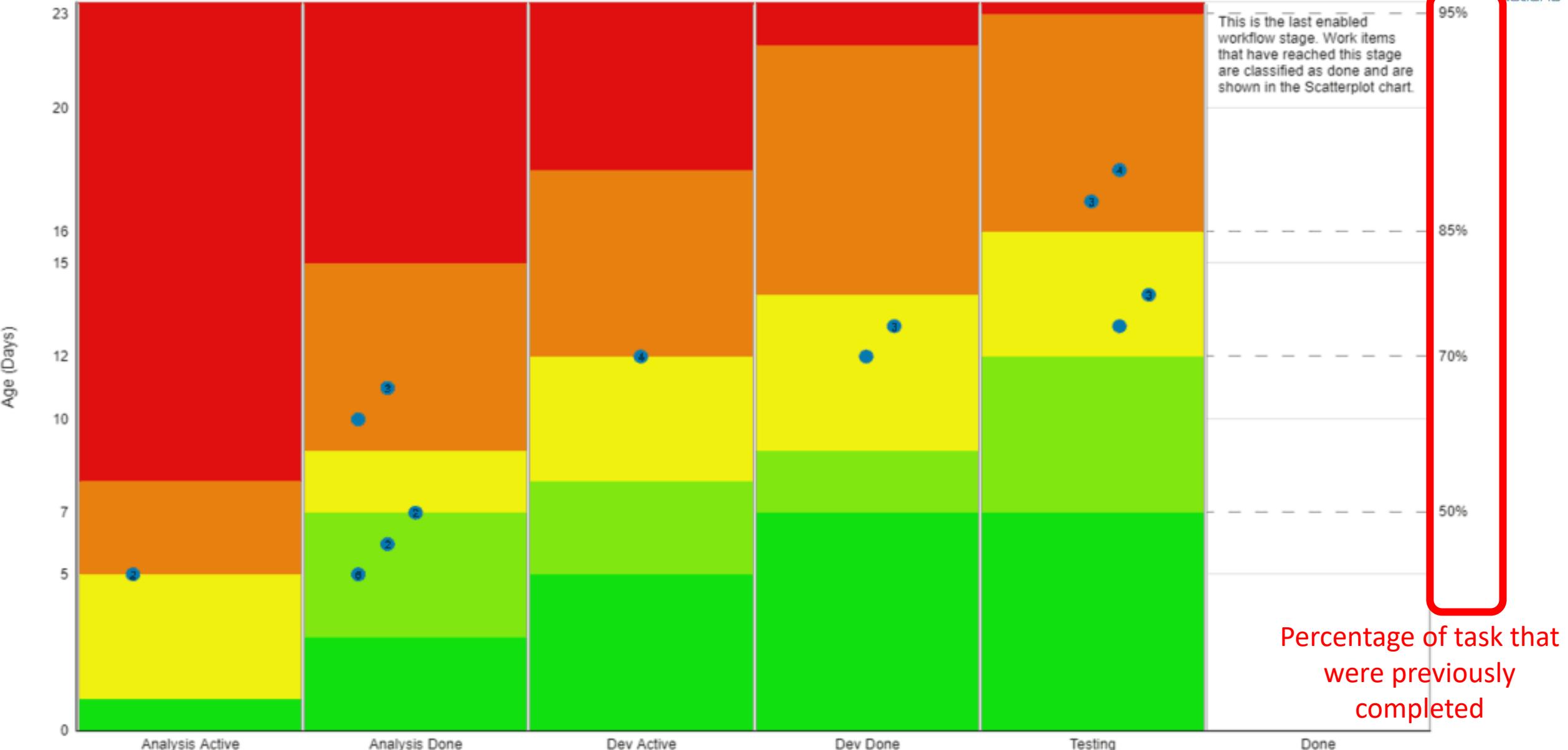
WIP: 2

WIP: 14

WIP: 4

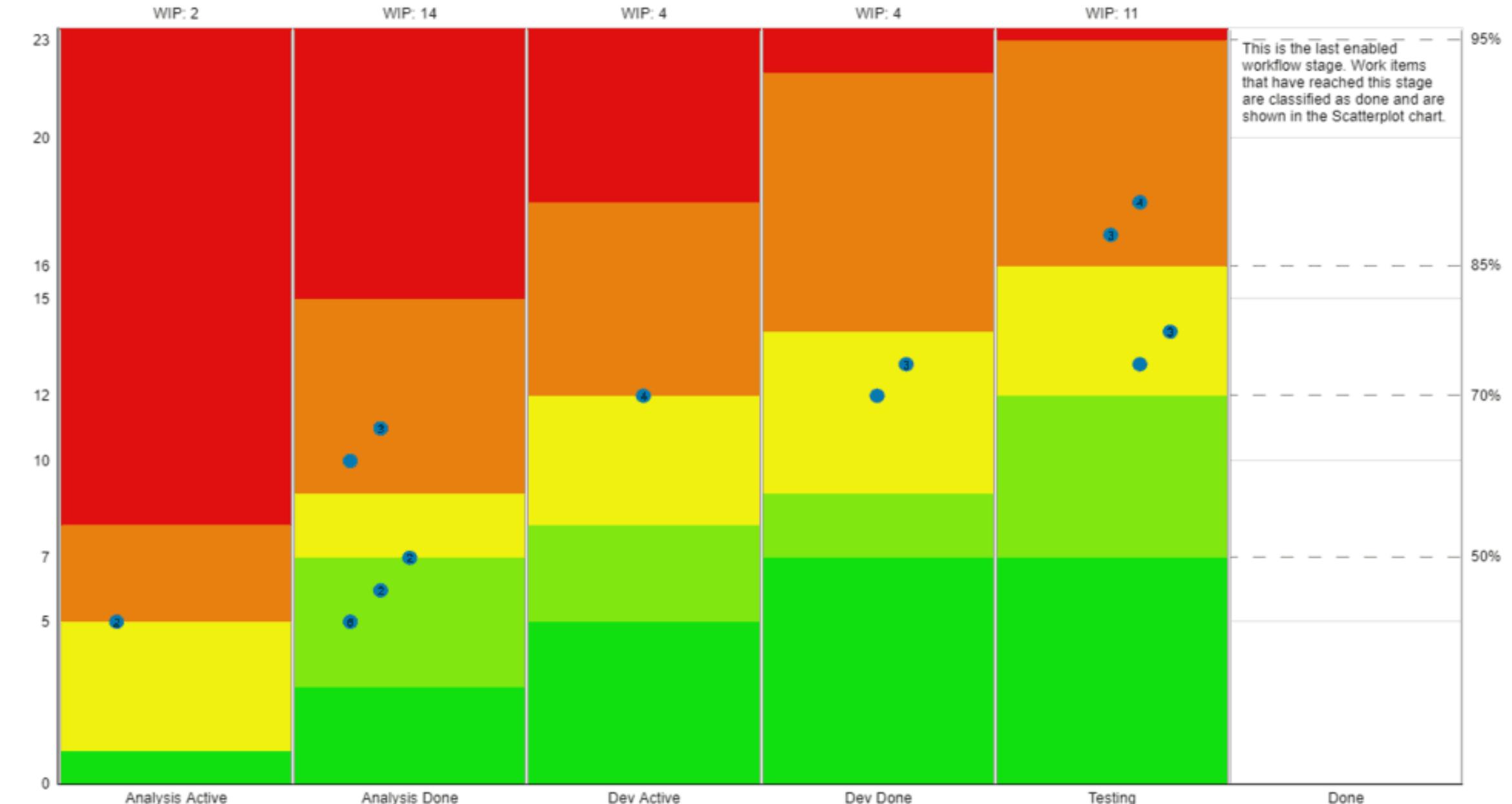
WIP: 4

WIP: 11





As of 20180507



Leading & Lagging Indicators

Measure & Improve



(Actionable)

Leading

Influence
future
performance

Analyse
past
performance

Lagging

(Non-Actionable)



Leading & Lagging Indicators



Lagging Indicator



Conversion rate

(Ability to track progress)

Website Traffic

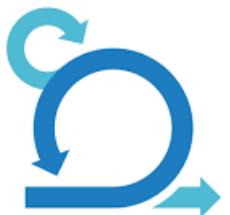


Leading Indicator

(Ability to adjust your course)



Little's Law



Little's Law

Links the three metrics **Throughput**, **Work in progress** & **Cycle Time**



$$L = \lambda W \rightarrow \text{Avg. Waiting time}$$

Avg. Length of queue
Or
Number of items
waiting

Avg. Rate of Arrival

John Little



Little's Law



$$L = \lambda W$$

Avg. Length of queue
Or
Number of items waiting

6 people will be in the queue

Avg. Rate of Arrival

2 people enter every minute

Avg. Waiting time

Each person has to wait 3 minutes in the shop to get a burger

$$6 = 2 \times 3$$



Little's Law

$$L = \lambda W$$

Avg. Length of queue
Or
Number of items waiting

Avg. Rate of Arrival

Avg. Waiting time

$$\text{Avg. WIP} = \text{Avg. Throughput} * \text{Avg. Cycle Time}$$

Assumption: Stable flow i.e. no. of items being picked by development team is equal to avg. number of items completed



Little's Law



$$\text{Avg. WIP} = \text{Avg. Throughput} * \text{Avg. Cycle Time}$$

OR

$$\text{Avg. Cycle Time} = \text{Avg. Throughput} * \text{Avg. WIP}$$

OR

$$\text{Avg. Throughput} = \text{Avg. WIP} * \text{Avg. Cycle Time}$$



Little's Law

$$\text{Avg. WIP} = \text{Avg. Throughput} \times \text{Avg. Cycle Time}$$

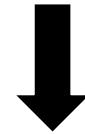
WIP: Unit is No. of work items

Cycle time: Unit is Days

Throughput: Unit is No. of work items per day

$$\text{No. of work items} = \cancel{\text{Days}} \times \underline{\text{No. of work items}}$$

~~Day~~



$$\text{No. of work items} = \text{No. of work items}$$



Little's Law

$$\text{Avg. WIP} = \text{Avg. Throughput} \times \text{Avg. Cycle Time}$$

For 30 days sprint, try to collect data for previous 10–12 sprints

→ Avg. WIP
 Avg. Throughput
 Avg. Cycle Time

Uses of Little's Law

- Checking Process Predictability
- Building Project management intuition



Process can be considered as stationary & predictable



Process has higher randomness



Little's Law

How not to use Little's Law

$$\boxed{\text{Avg. WIP}} = \boxed{\text{Avg. Throughput}} \times \boxed{\text{Avg. Cycle Time}}$$

10 **5** **✗**

A diagram showing the formula for Little's Law: Avg. WIP equals Avg. Throughput times Avg. Cycle Time. The numbers 10 and 5 are placed under the respective terms, while a large red 'X' is placed under the term 'Avg. Cycle Time'.

Cycle time depends on a lot of factors other than WIP & Throughput

Don't use it to predict future values/forecasting

Don't use it to make Project completion commitments



Little's Law

Assumptions necessary for Little's law to work

Measuring units for three metrics in the formula should be consistent

Average arrival rate should be equal to average departure rate/Throughput

WIP should be same as the beginning & the end of the chosen interval

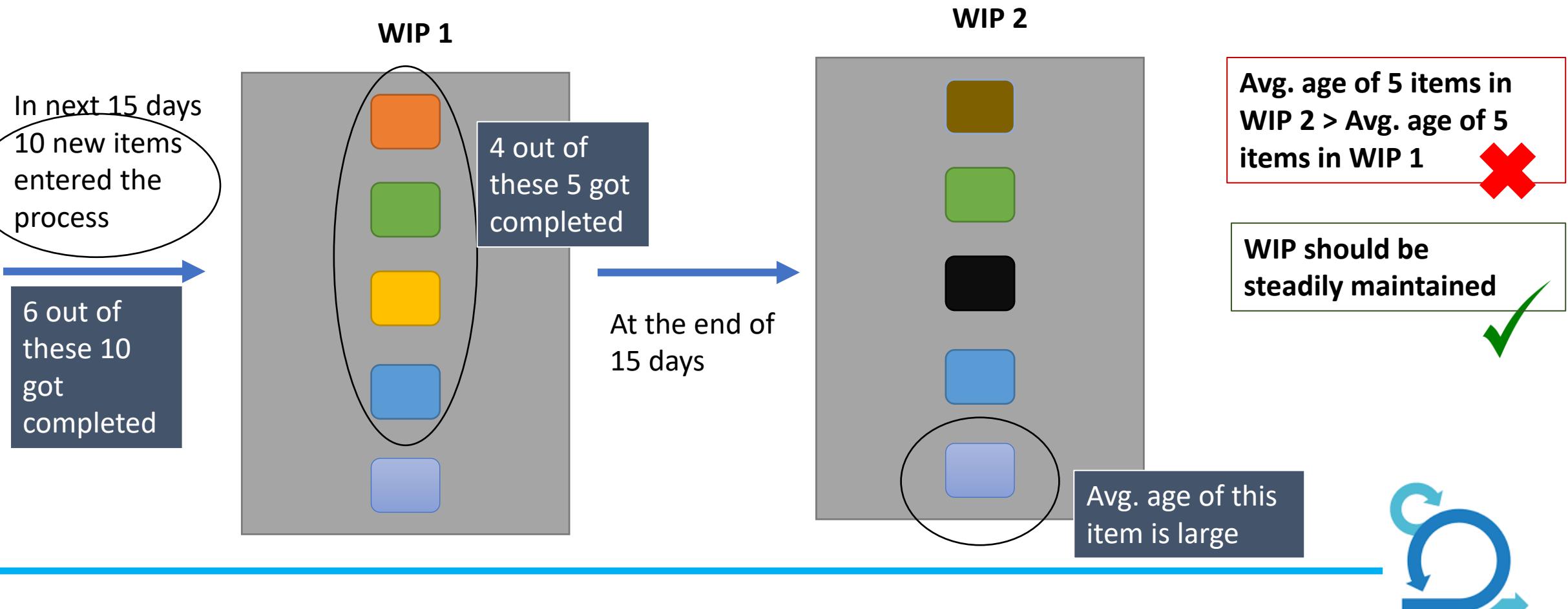
All work that enters the process will also exit the process

Average age of WIP should remain the same



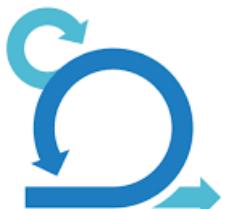
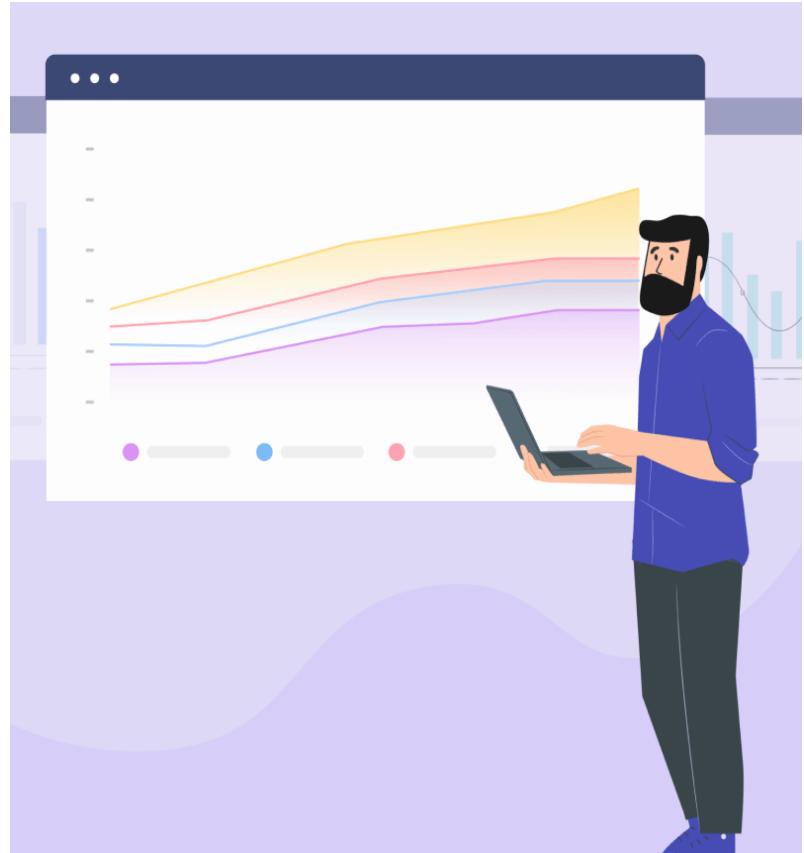
Little's Law

Assumptions necessary for Little's law to work

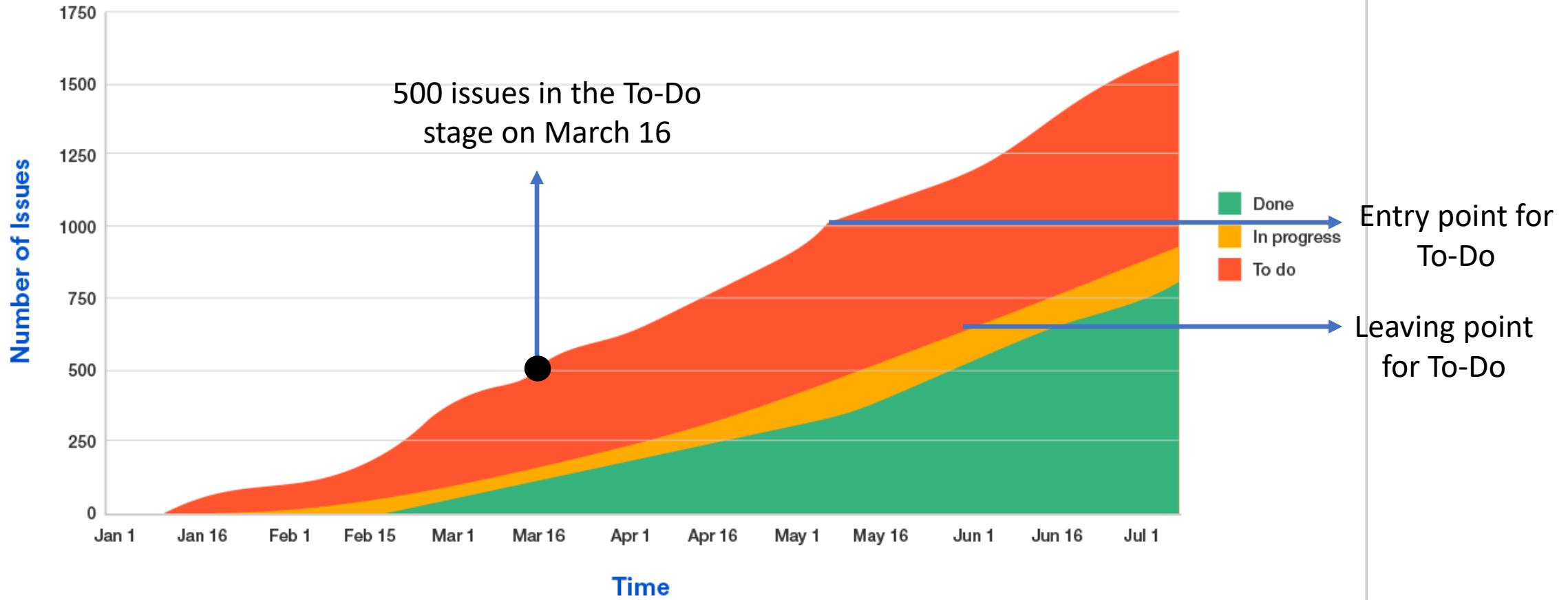


Cumulative Flow Diagram

- Tracks progress for Agile Teams
- Shows state of WIP & Project pace
- Help identify potential risks & bottlenecks to be on track

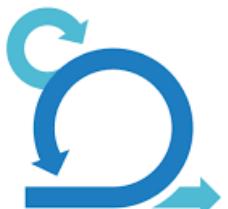
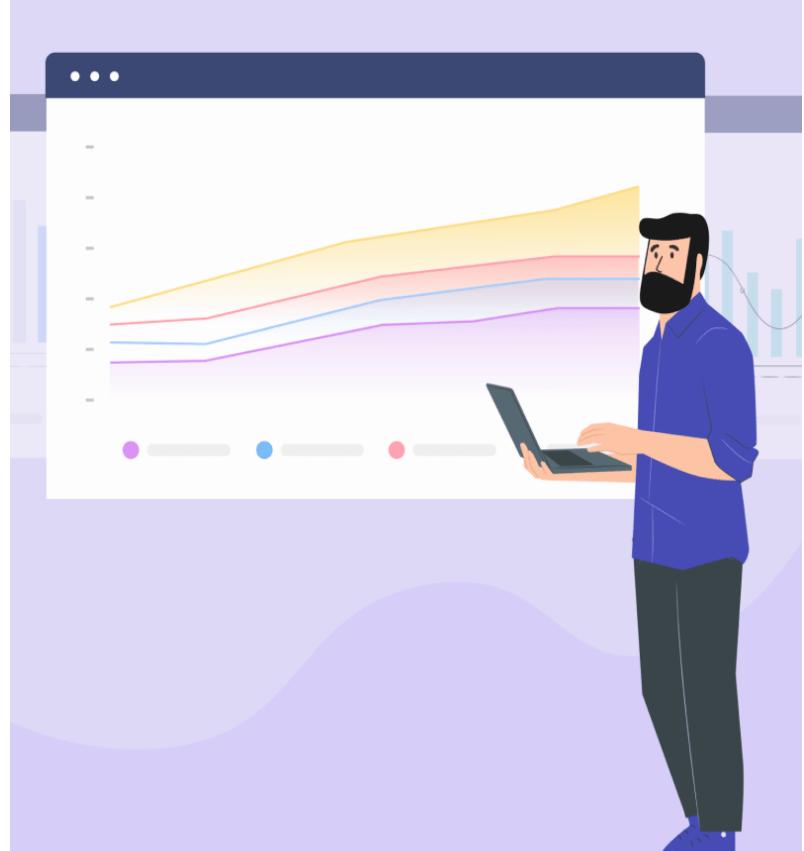


Cumulative Flow

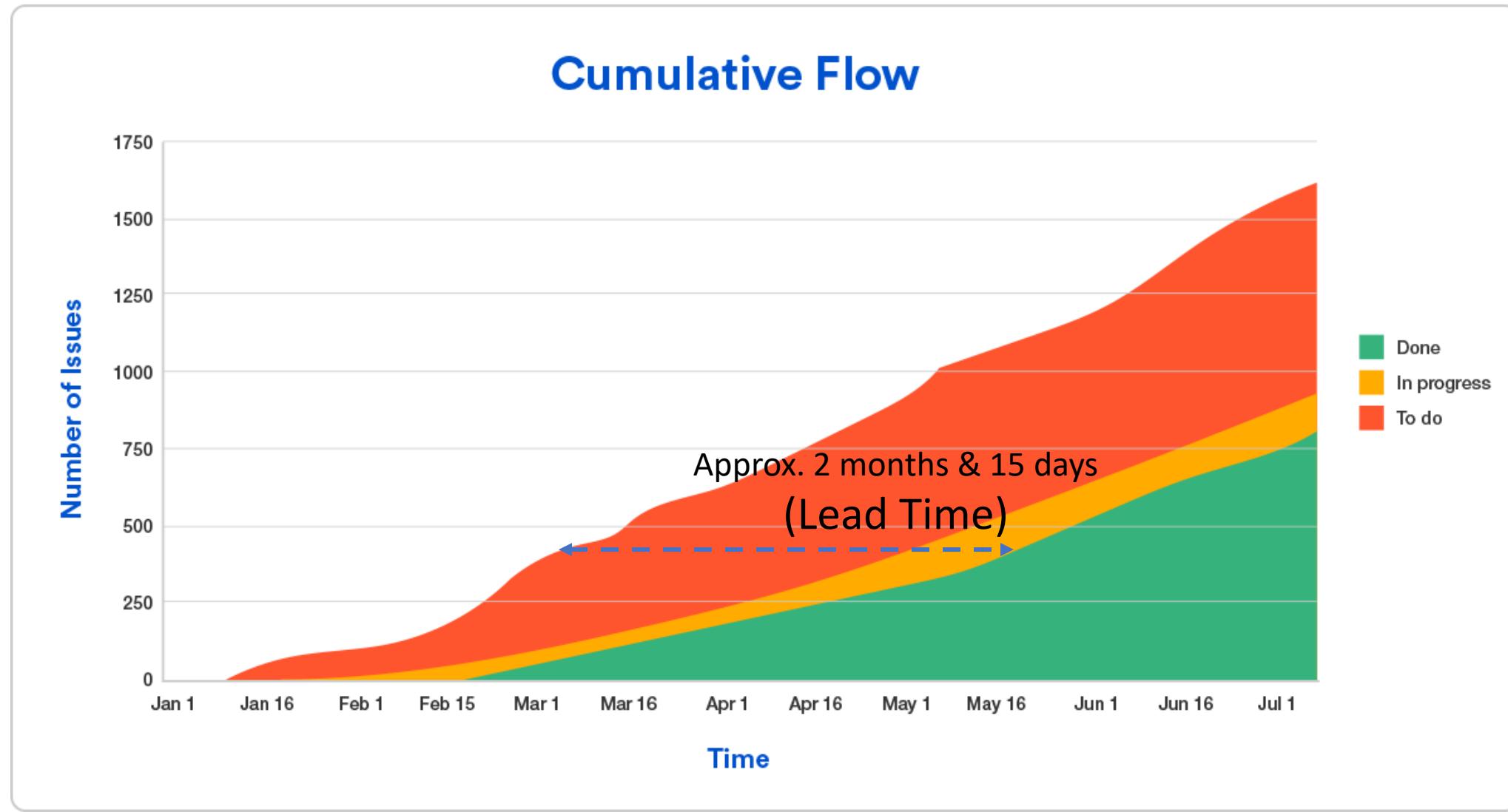


Cumulative Flow Diagram

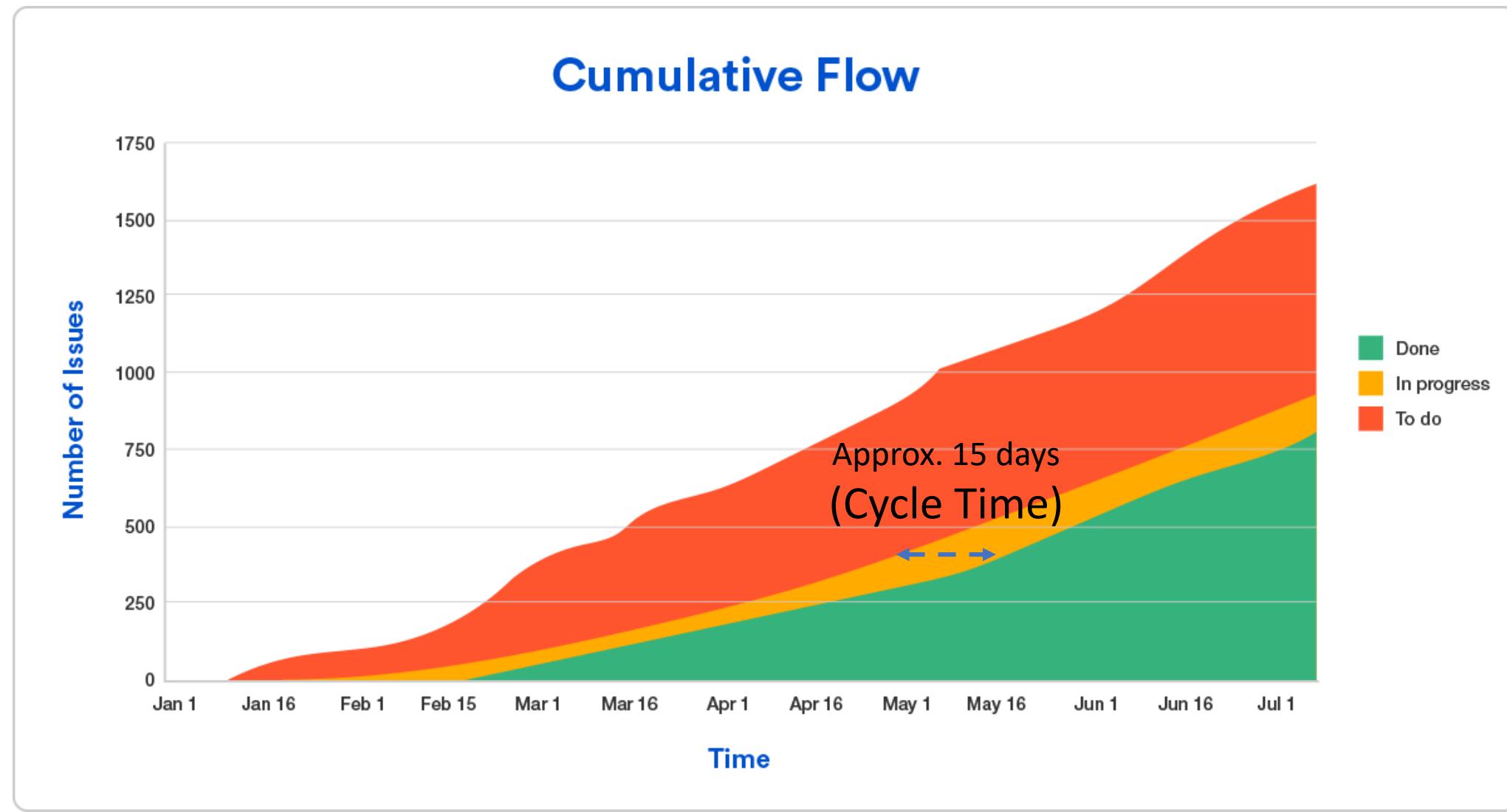
- Tracks progress for Agile Teams
- Shows state of WIP & Project pace
- Help identify potential risks & bottlenecks to be on track
- Help identify lead time, Cycle time, Throughput & Work in progress



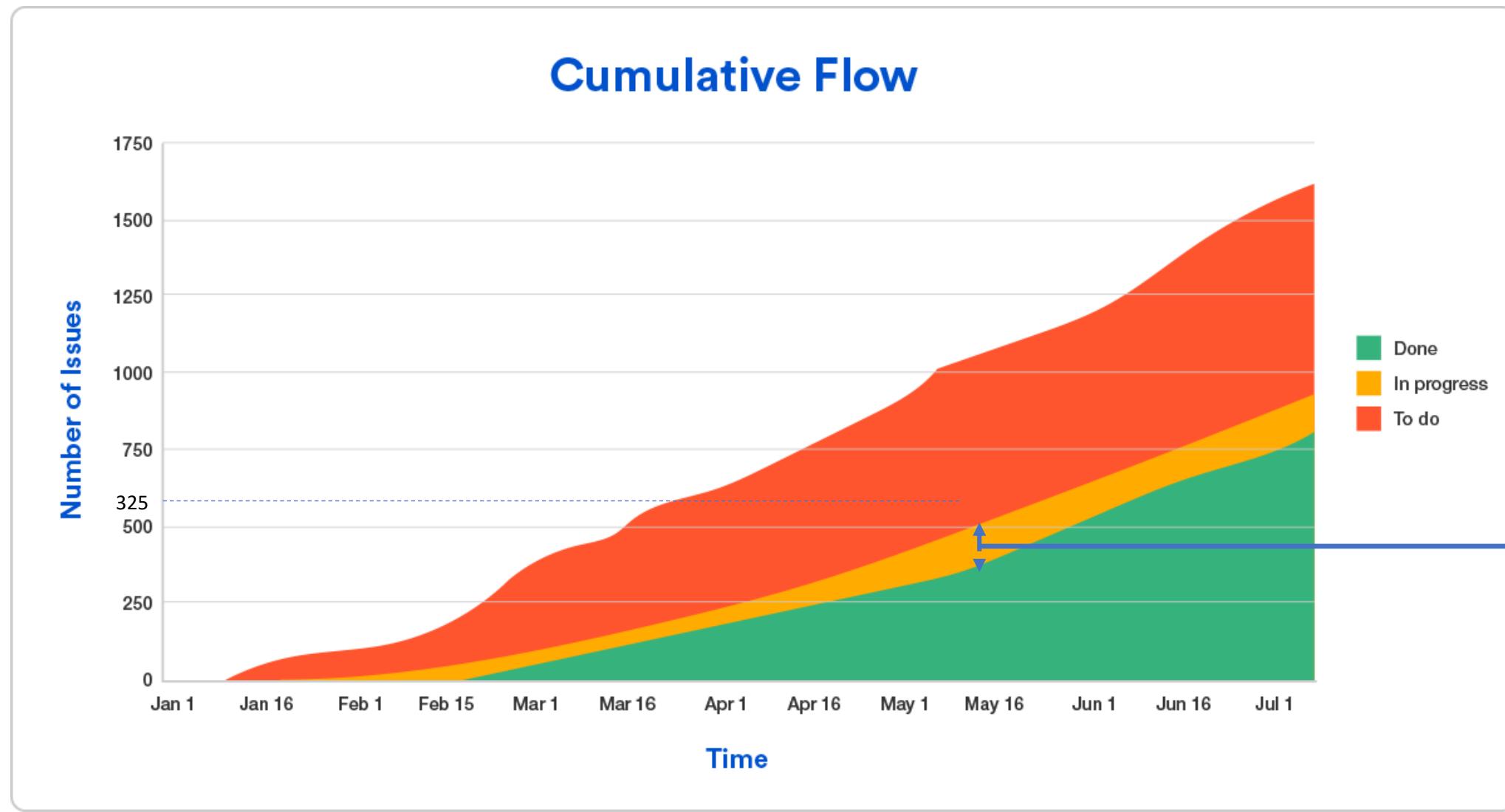
Lead Time: Period between a new task's appearance in your workflow and its final departure from the system



Cycle Time: Represents the amount of elapsed time that a work item spends as Work In Progress



WIP: Represents the amount of issues that have entered a given process but have not exited

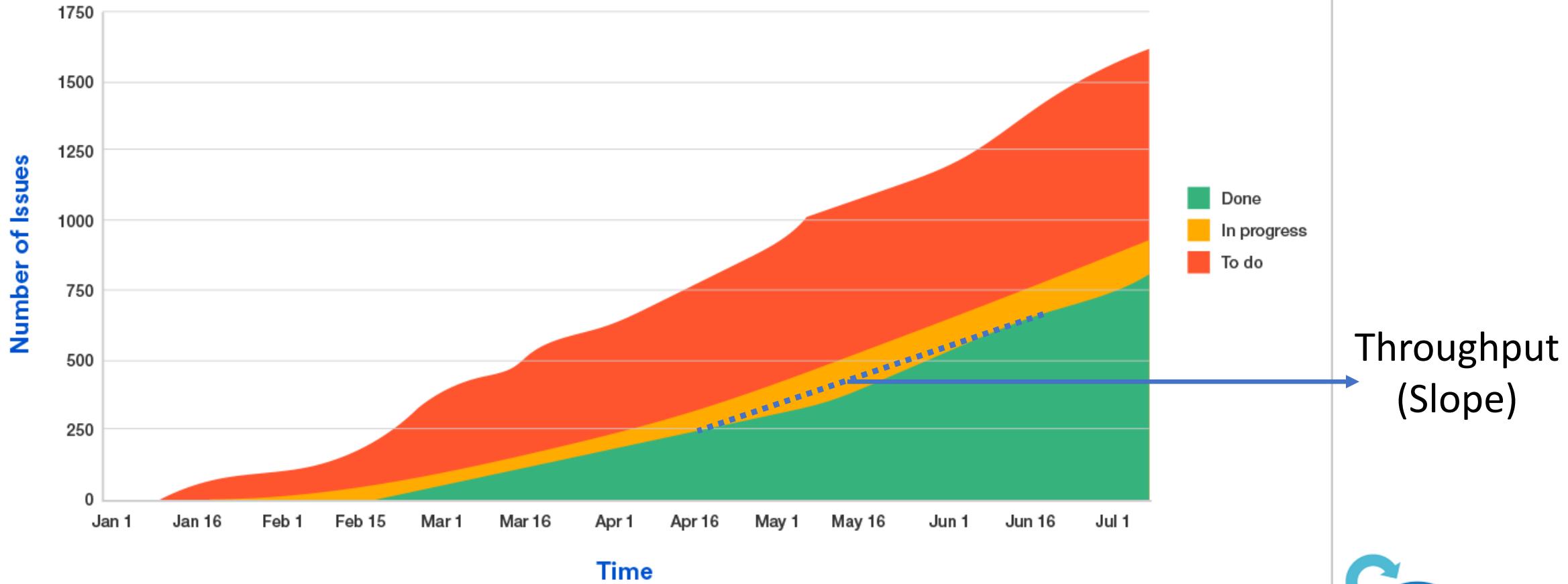


WIP
(Approx. 175)

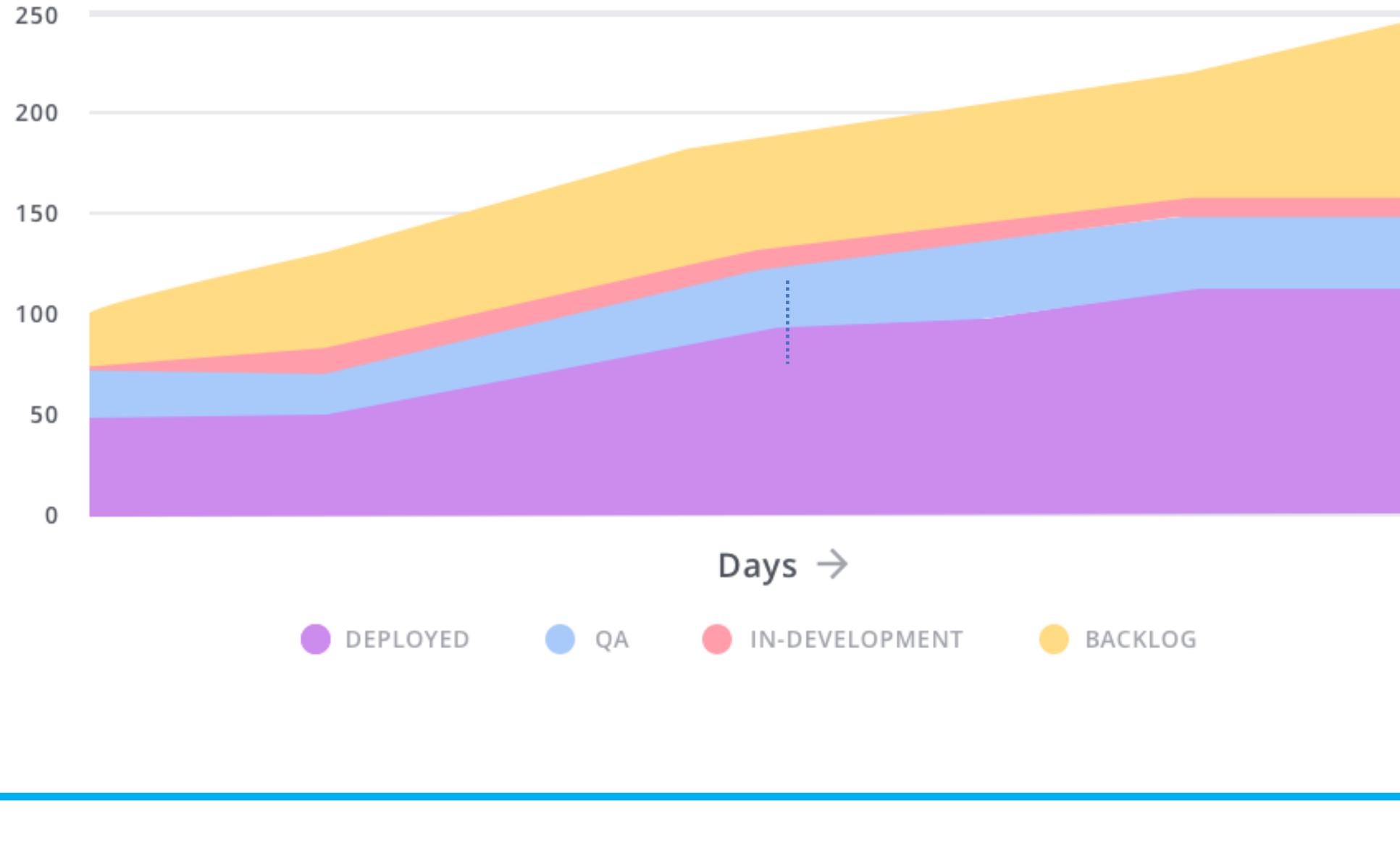
Throughput: Represents the amount of WIP (number of work items) completed per unit of time



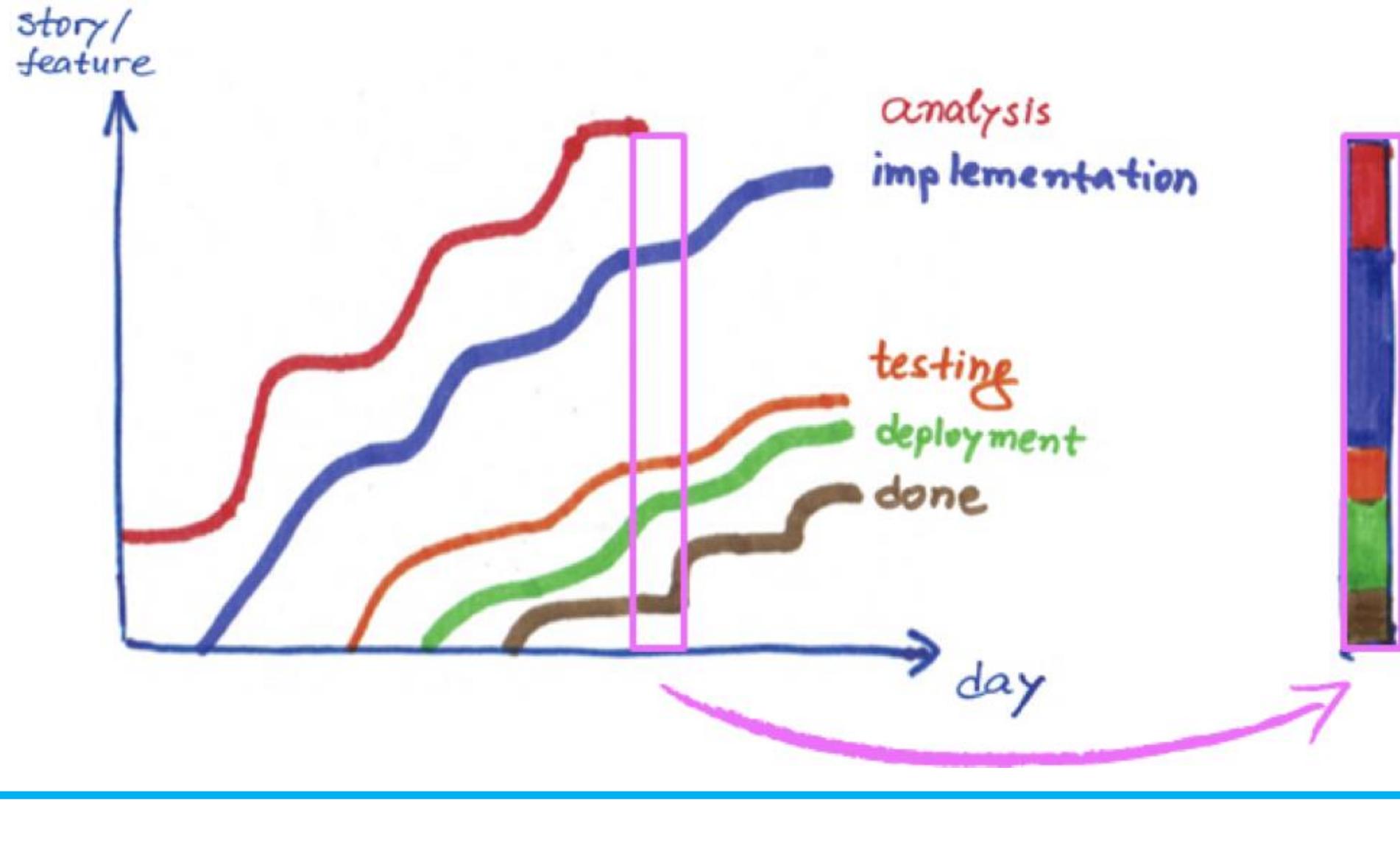
Cumulative Flow



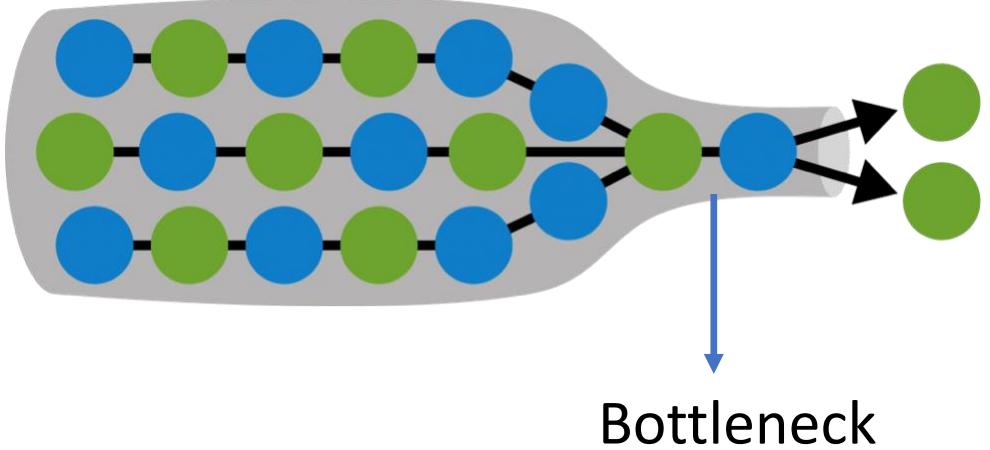
CUMULATIVE FLOW DIAGRAM



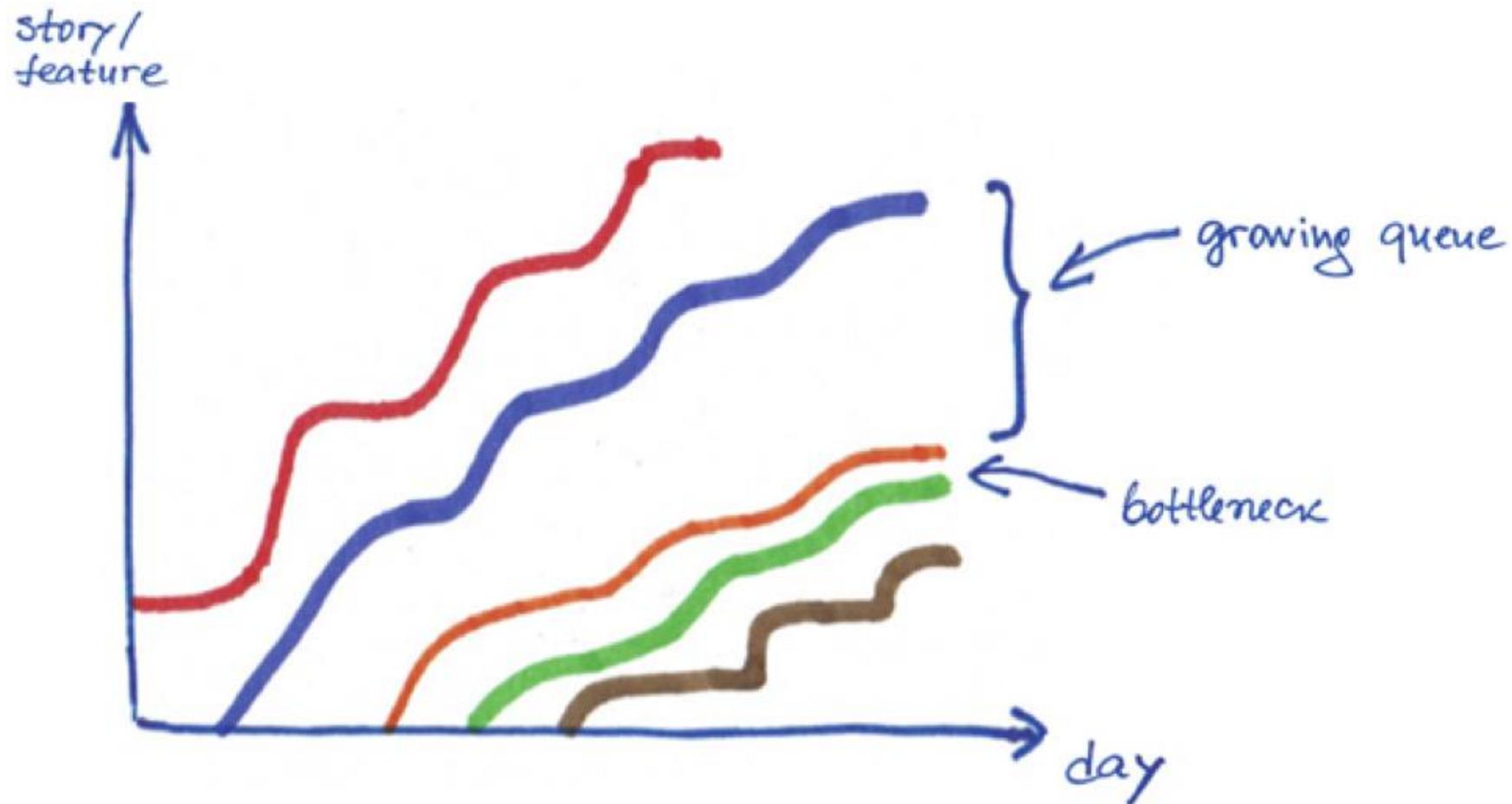
Bottlenecks



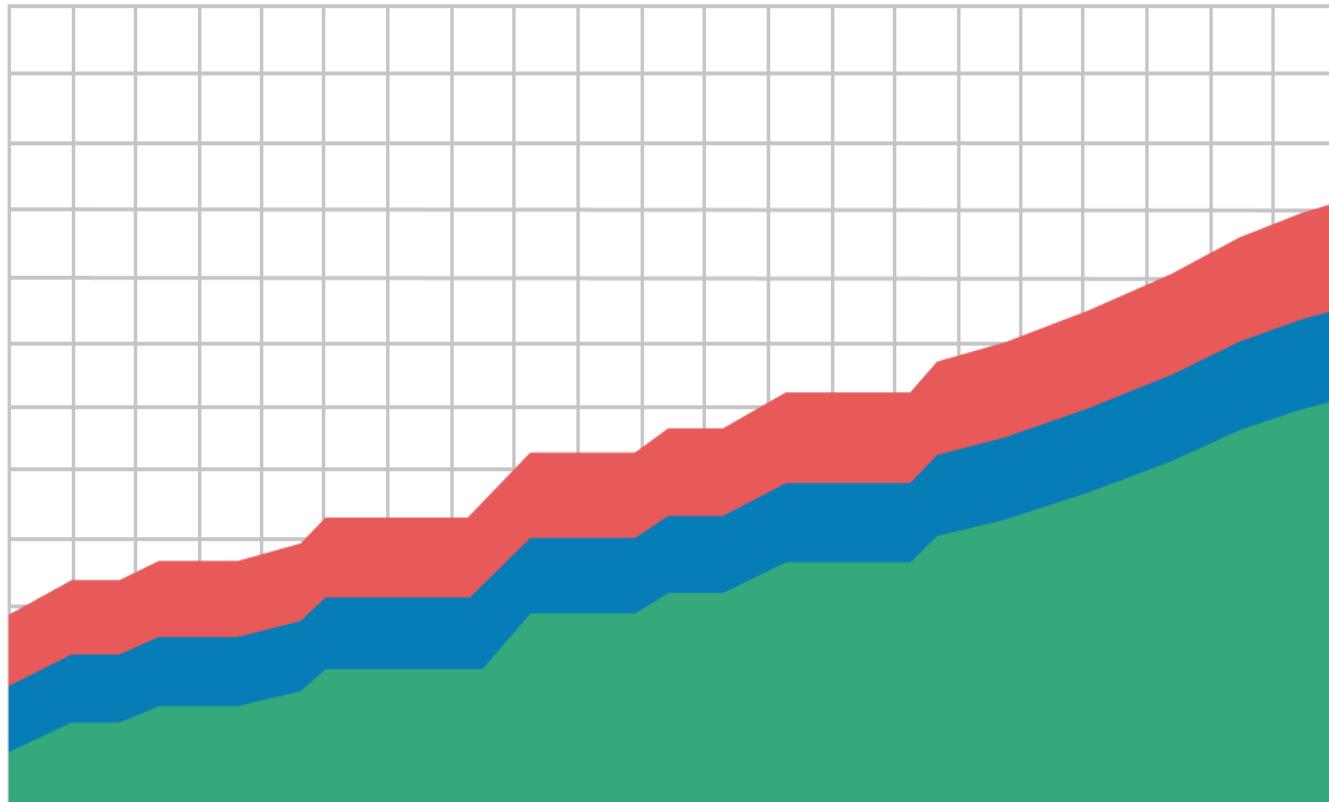
Bottlenecks: Lowest Throughput



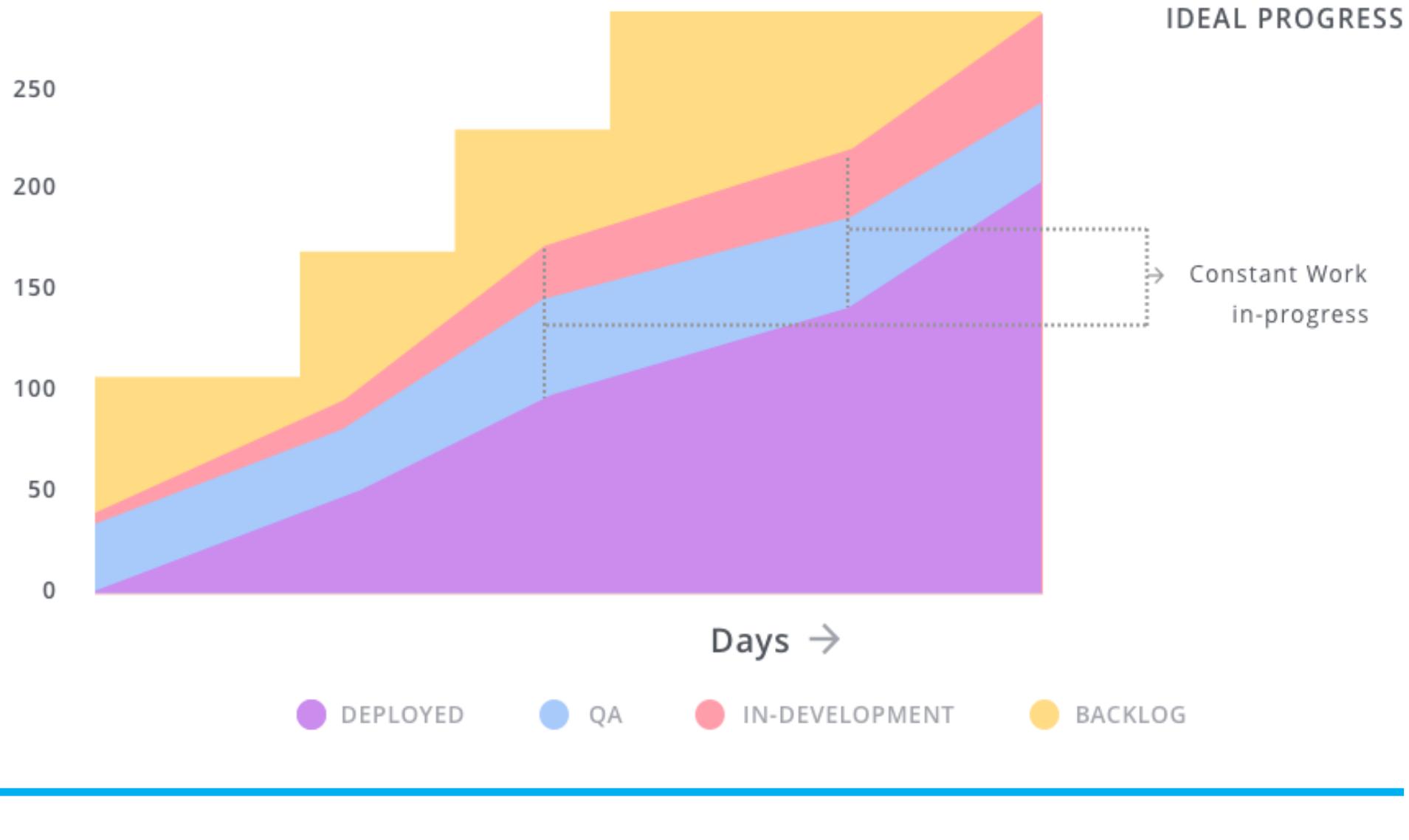
Bottlenecks: Lowest Throughput



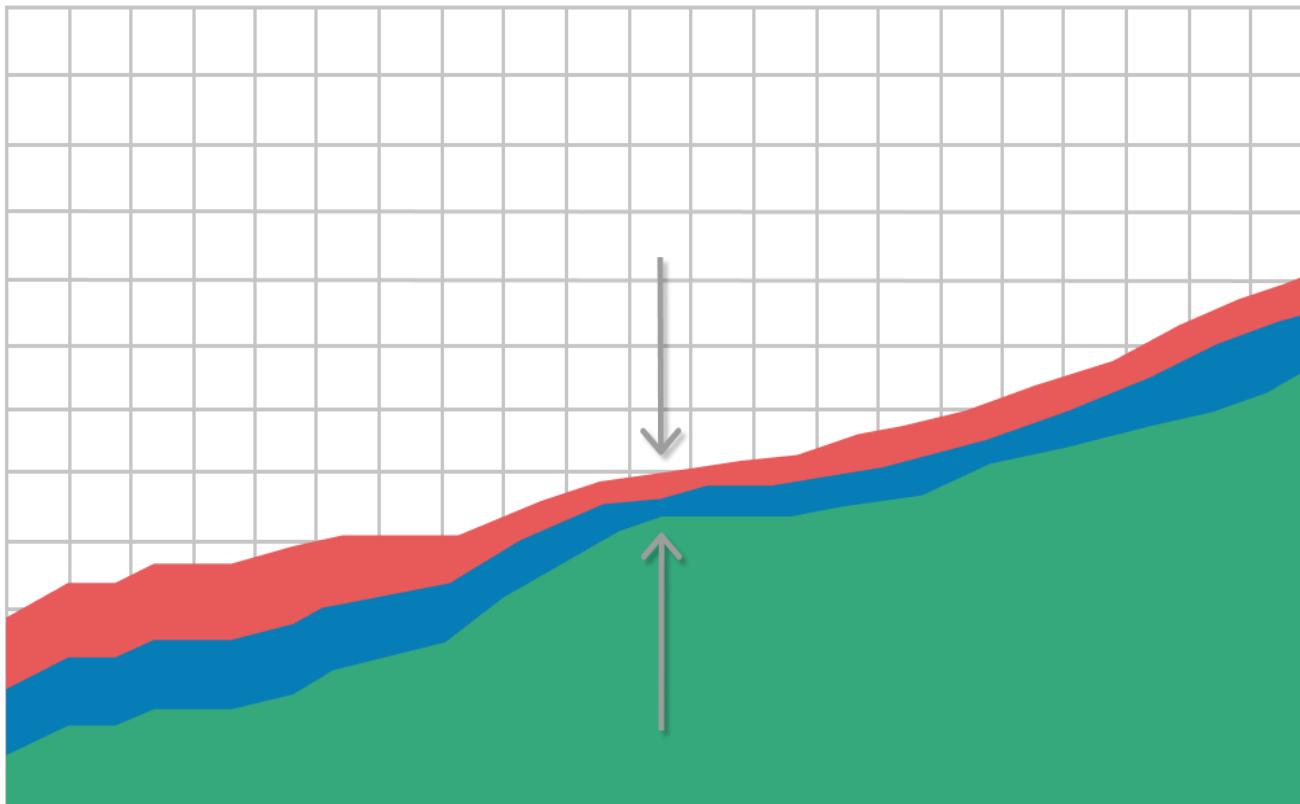
The Bands are Progressing in Parallel



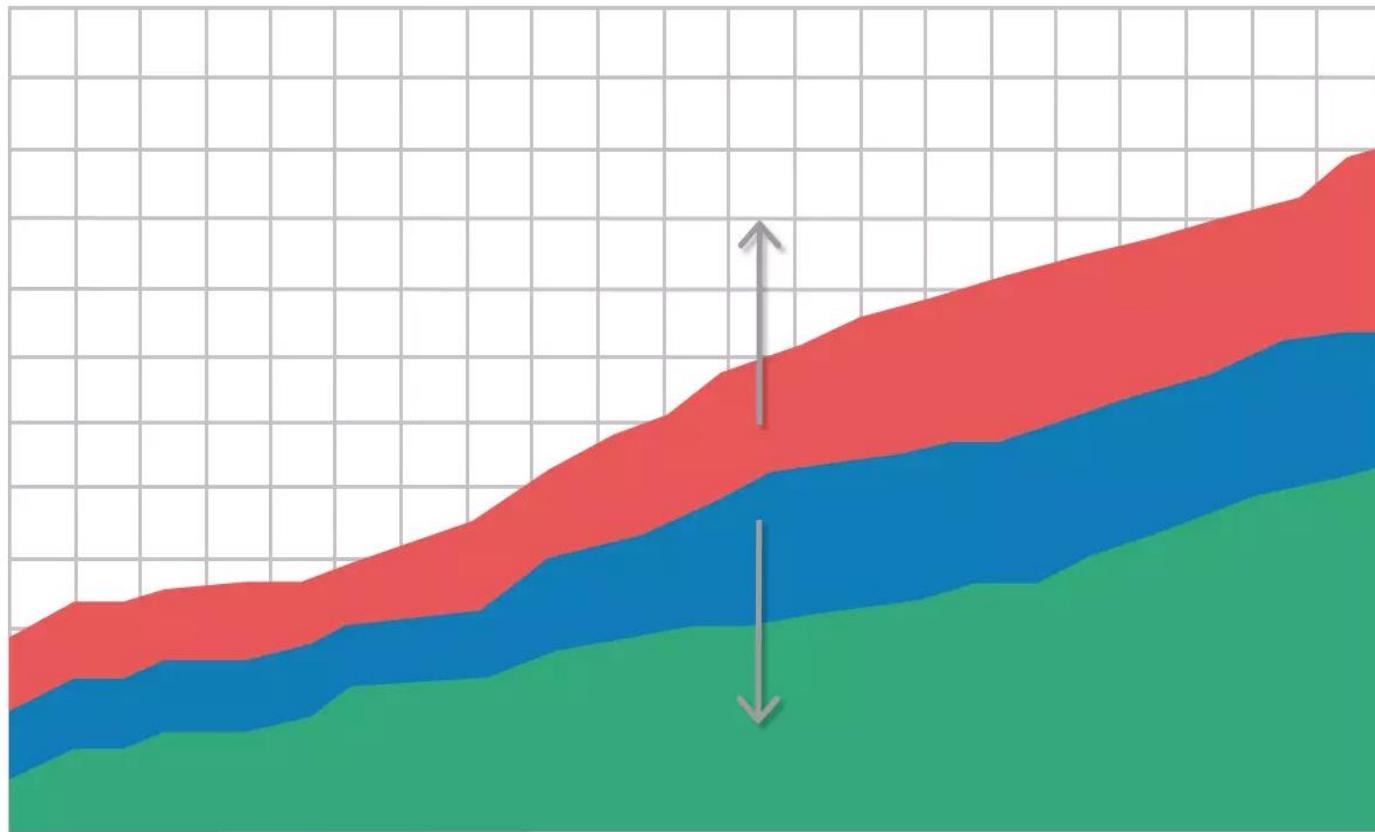
The Bands are Progressing in Parallel



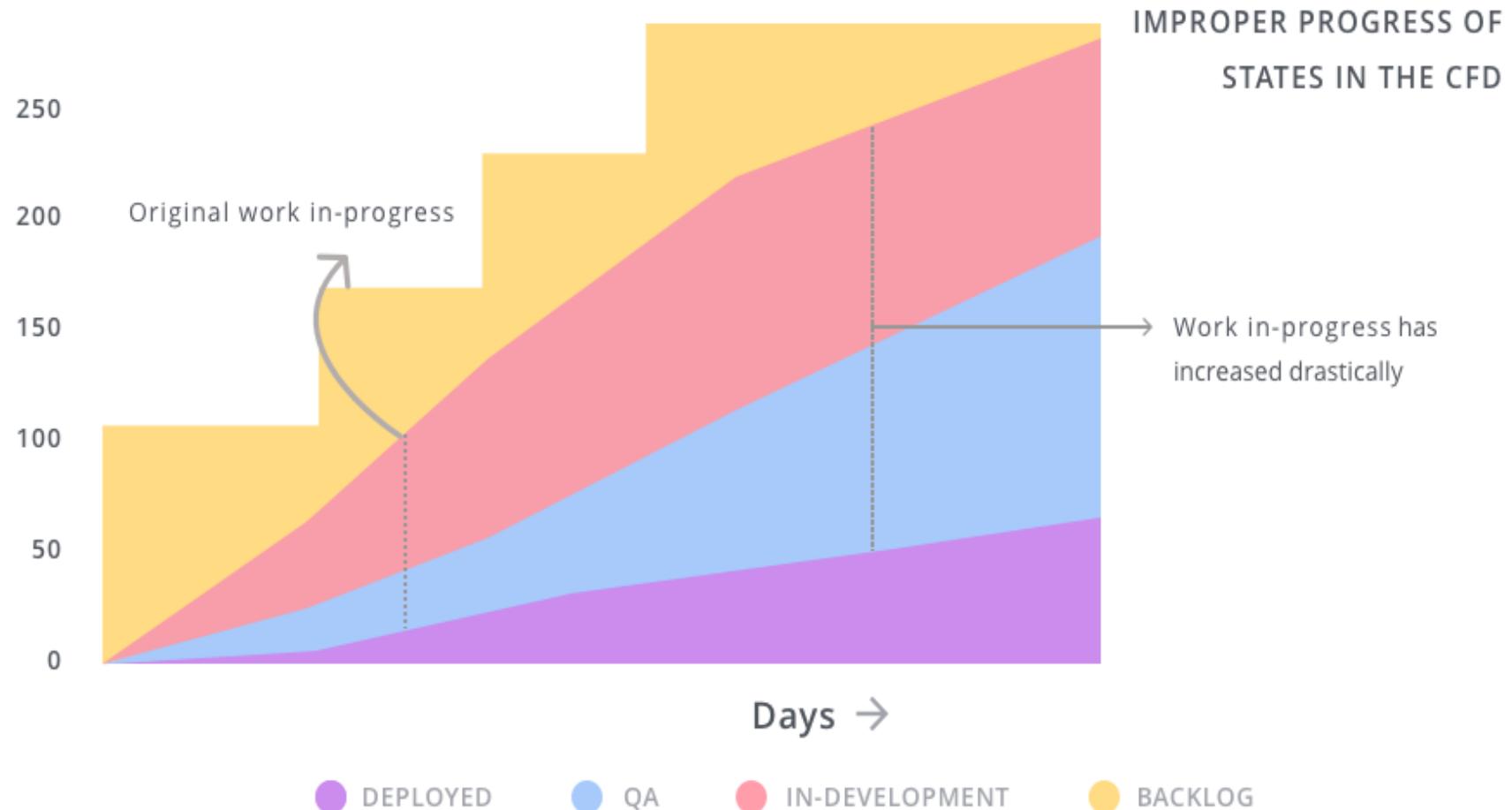
The Band is Rapidly Narrowing



The Band is Rapidly Widening



The Band is Rapidly Widening



Extreme Programming (XP)



Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

- Agile Alliance

Why is it called Extreme Programming?



Code Review

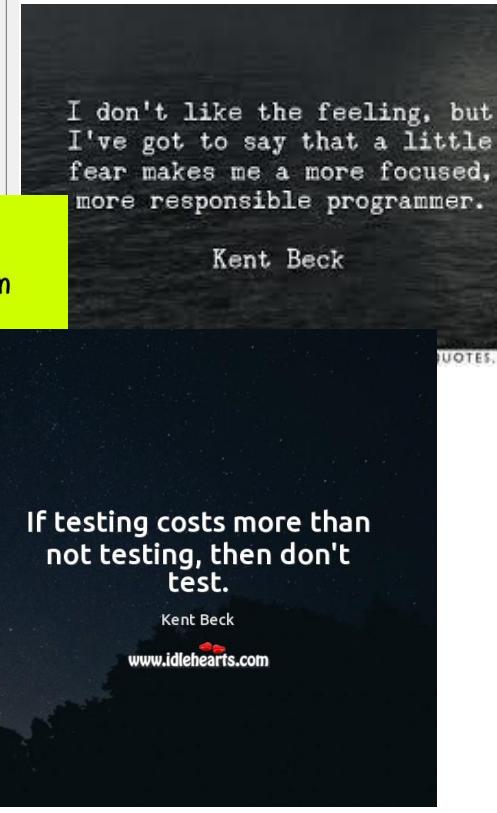
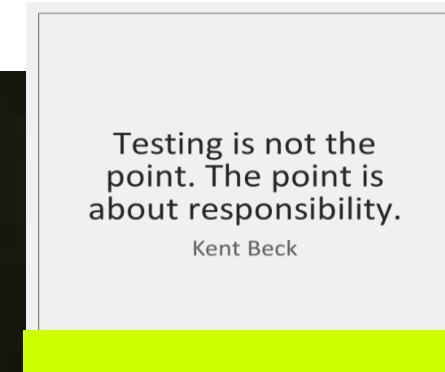
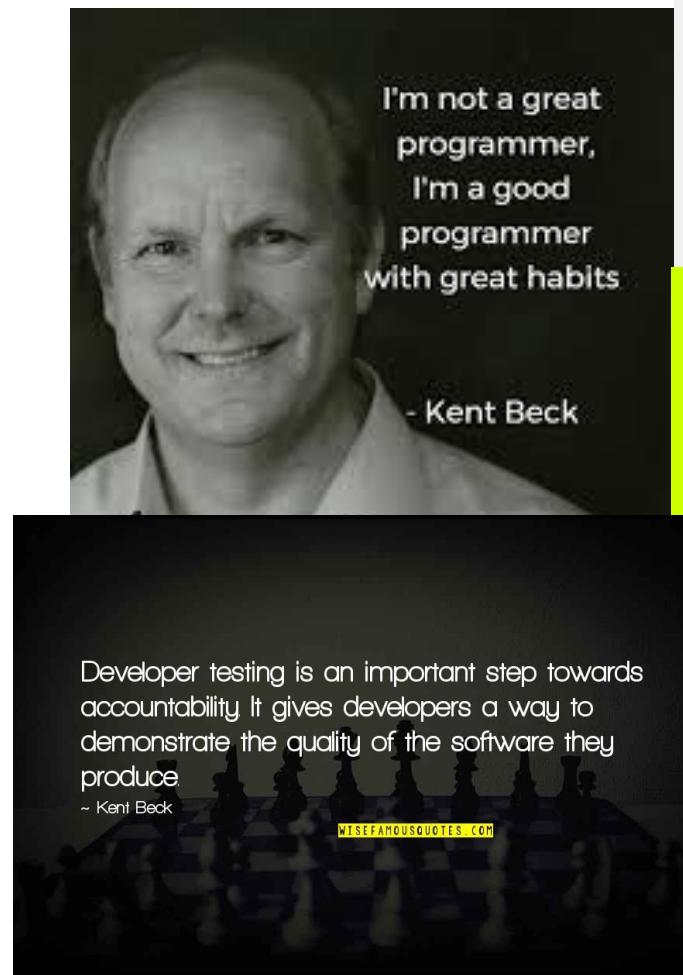


Extreme Programming (XP)

XP was introduced in 1996 by Kent Beck



FIAT CHRYSLER AUTOMOBILES



Extreme Programming (XP)



Extreme Programming is all about...

- Client satisfaction
- Bringing the social change
- Adaptiveness

It ran into controversy around...

- Pair programming
- Incremental design
- Scalability



Extreme Programming (XP)



Values behind XP practices and methods

- Simplicity
- Communication
- Courage
- Feedback
- Respect



Extreme Programming (XP)



Values behind XP practices and methods

- **Simplicity**
- Communication
- Courage
- Feedback
- Respect



Extreme Programming (XP)

Values behind XP practices and methods

- Simplicity
- **Communication**
- Courage
- Feedback
- Respect

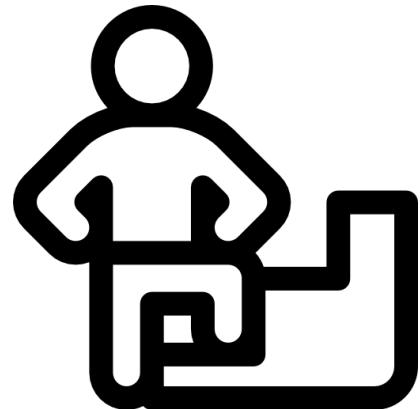


Extreme Programming (XP)



Values behind XP practices and methods

- Simplicity
- Communication
- **Courage**
- Feedback
- Respect



Extreme Programming (XP)



Values behind XP practices and methods

- Simplicity
- Communication
- Courage
- **Feedback**
- Respect



Extreme Programming (XP)



Values behind XP practices and methods

- Simplicity
- Communication
- Courage
- Feedback
- **Respect**



Extreme Programming (XP)

Practice Exercise

The development team is organized into pairs, with each pair working in front of a single workstation.

Which of the five aspects of extreme programming do you think that this improves?

- A. Communication
- B. Simplicity
- C. Feedback
- D. Respect
- E. Courage



Extreme Programming



Mentality of Sufficiency

How would you program if you had all the time in the world?

- Write Tests
- Restructure your code often
- Talk to fellow programmers and with the customer often



Extreme Programming Practices



Primary Practices

Part 1	Part 2
Sit Together	Stories
Whole Team	Weekly Cycle
Informative Workspace	Quarterly Cycle
Energized Work	Slack
Pair Programming	Ten-Minute Build
	Continuous Integration
	Test First Programming
	Incremental Design

Corollary Practices : Refer to the resources section



Extreme Programming Practices



Primary Practices

Sit Together

- Whole Team
- Informative Workspace
- Energized Work
- Pair Programming



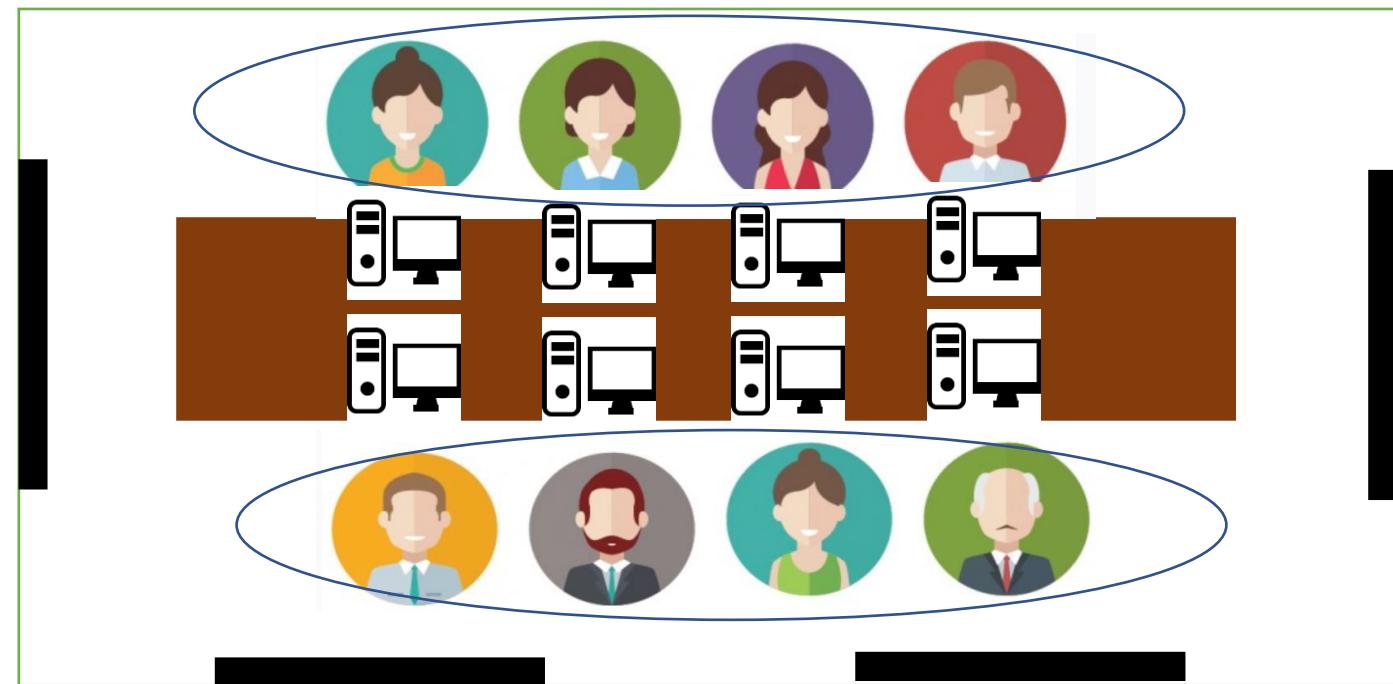
All members are accessible to
each other



Extreme Programming Practices



Primary Practices



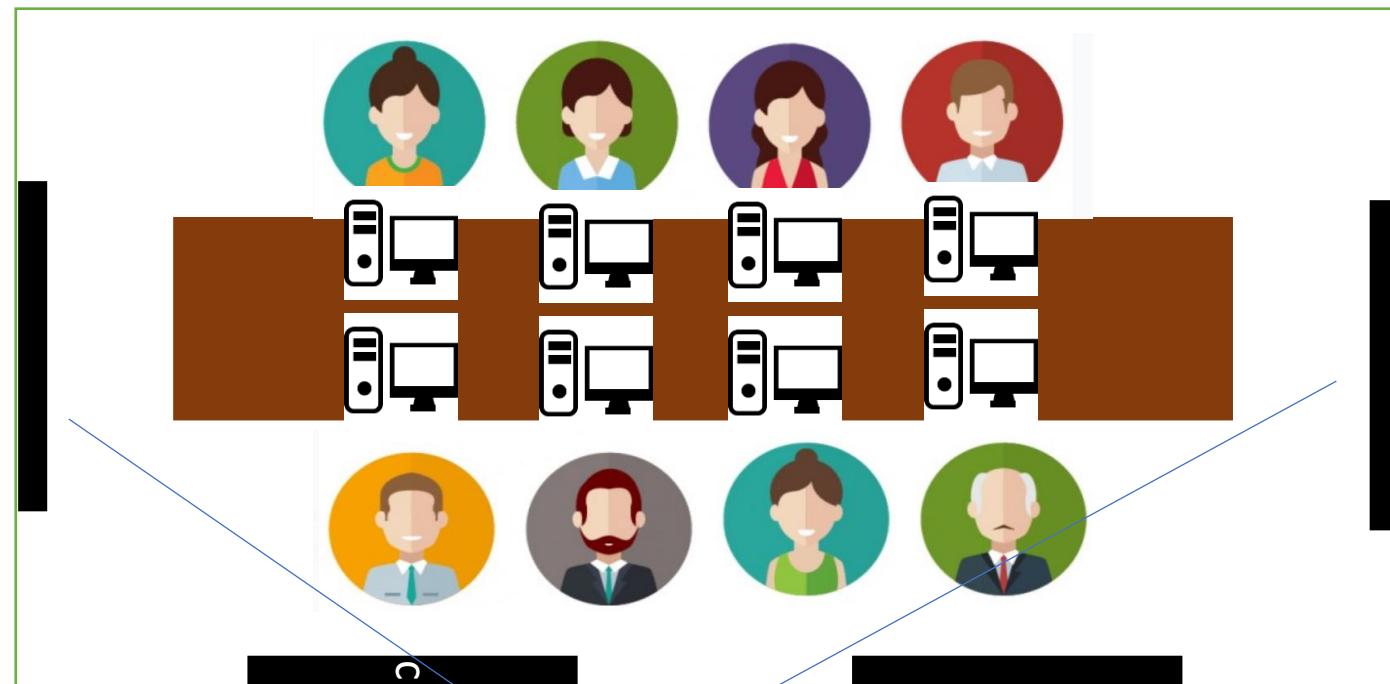
All the required skills are
available to plan, develop, build,
test & release



Extreme Programming Practices



Primary Practices



Visual & Information



Extreme Programming Practices



Primary Practices

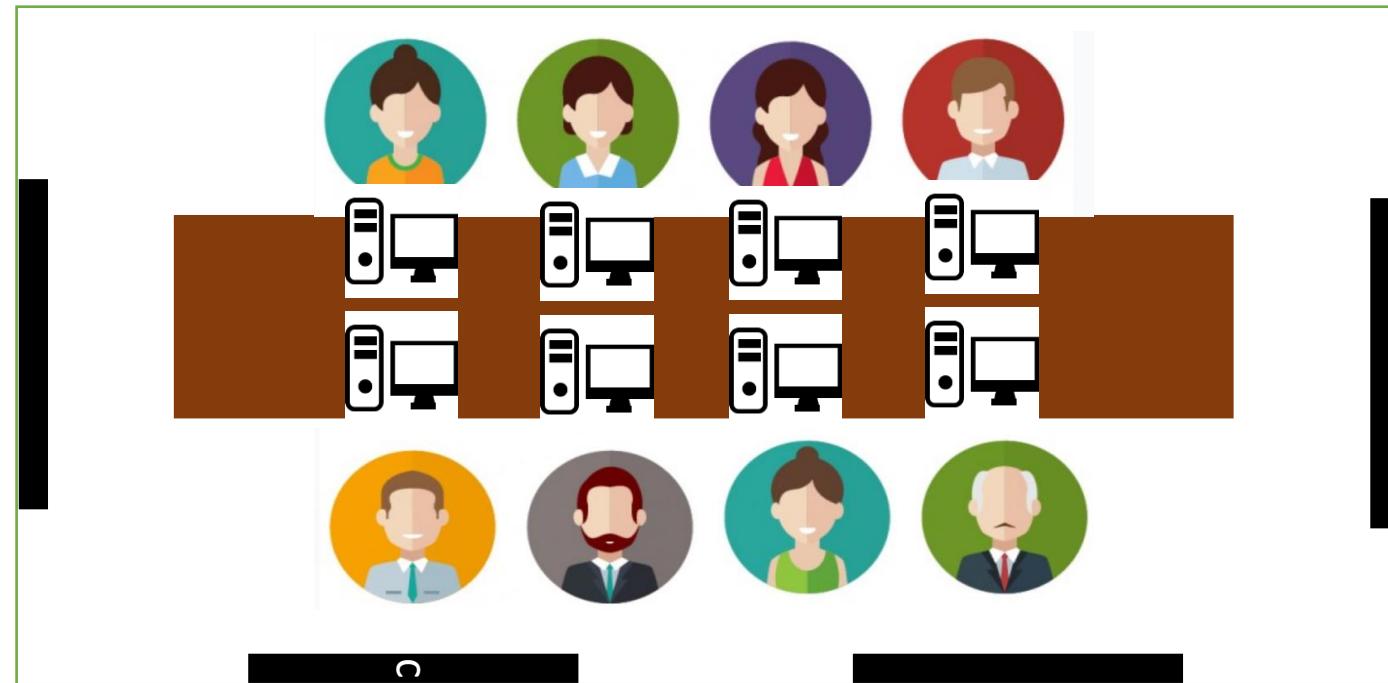
Sit Together

Whole Team

Informative Workspace

Energized Work

Pair Programming



40 Hours
Sustainable



Extreme Programming Practices



Primary Practices

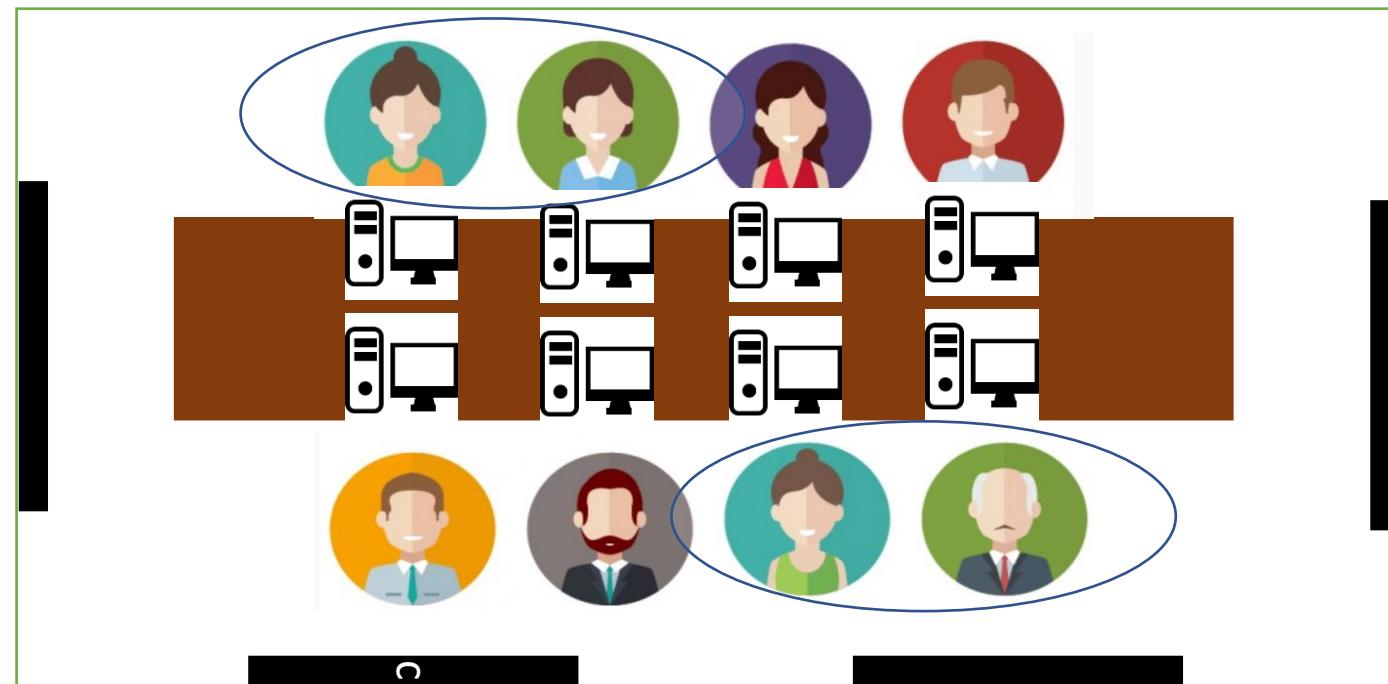
Sit Together

Whole Team

Informative Workspace

Energized Work

Pair Programming



Pair Programming



Extreme Programming Practices



Primary Practices

Stories

Weekly Cycle

Quarterly Cycle

Slack

Ten-Minute Build

Continuous Integration

Test First Programming

Incremental Design

Connextra A Connextra Story Card

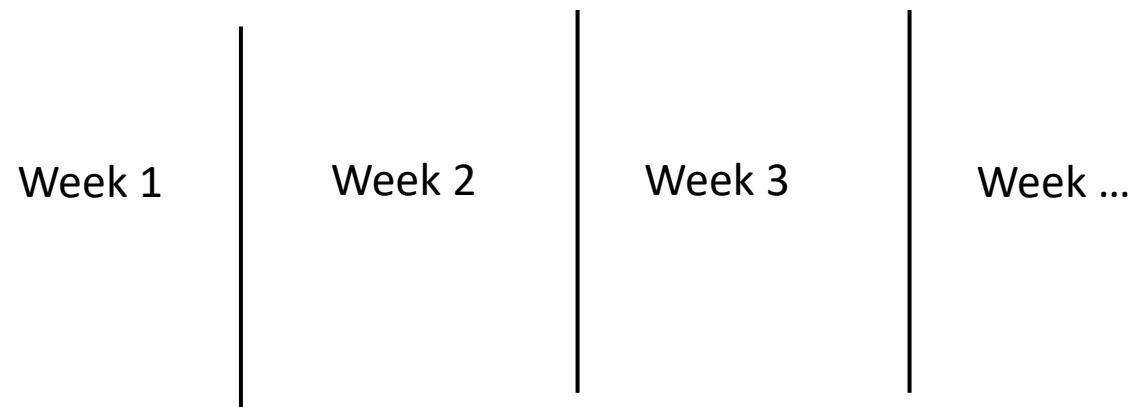
Perspective	Title	Reserved for priority
	<u>WRITING GOOD STORIES</u>	
Reason	As a Connextra employee - I want to know how to write good stories So that I can submit cards to the planning game that are clear and will be accepted in the next iteration.	
		Requirements
Author	Tim	Date
	8/Nov/01	Reserved for estimate



Extreme Programming Practices



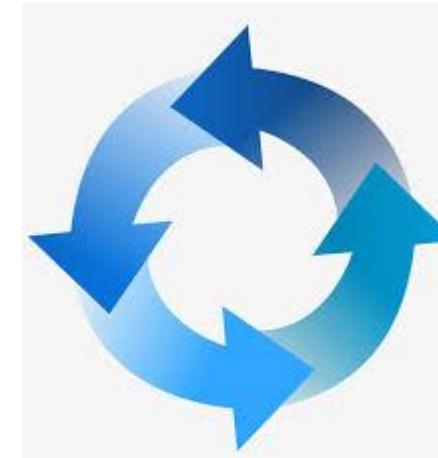
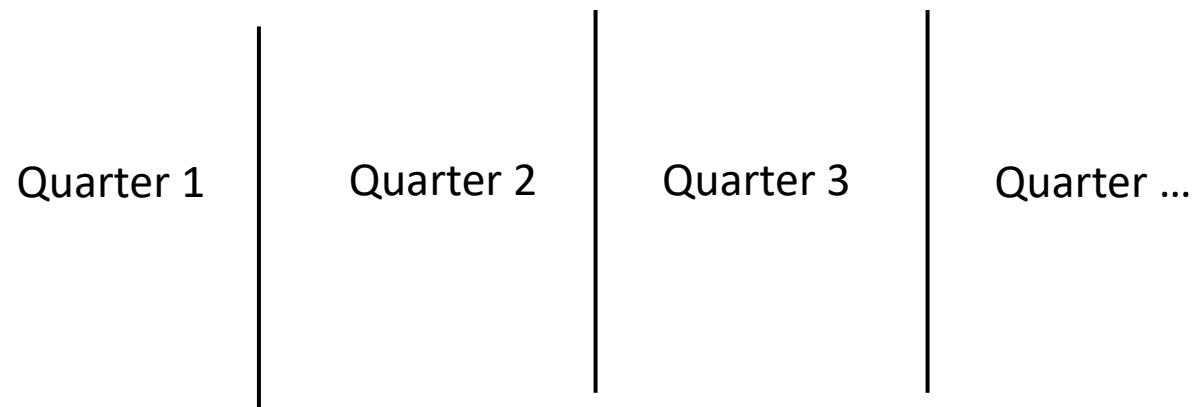
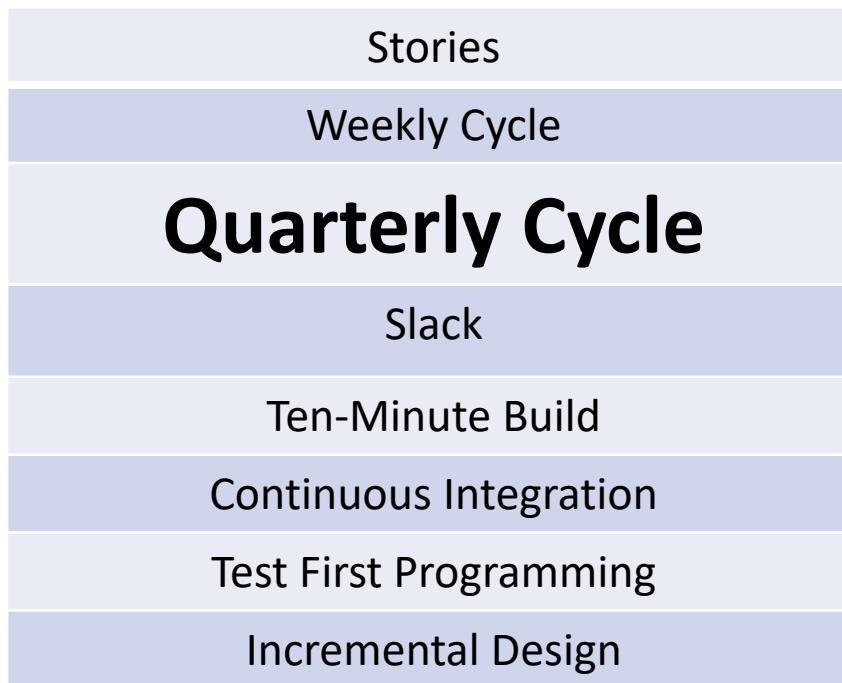
Primary Practices



Extreme Programming Practices



Primary Practices



Extreme Programming Practices



Primary Practices

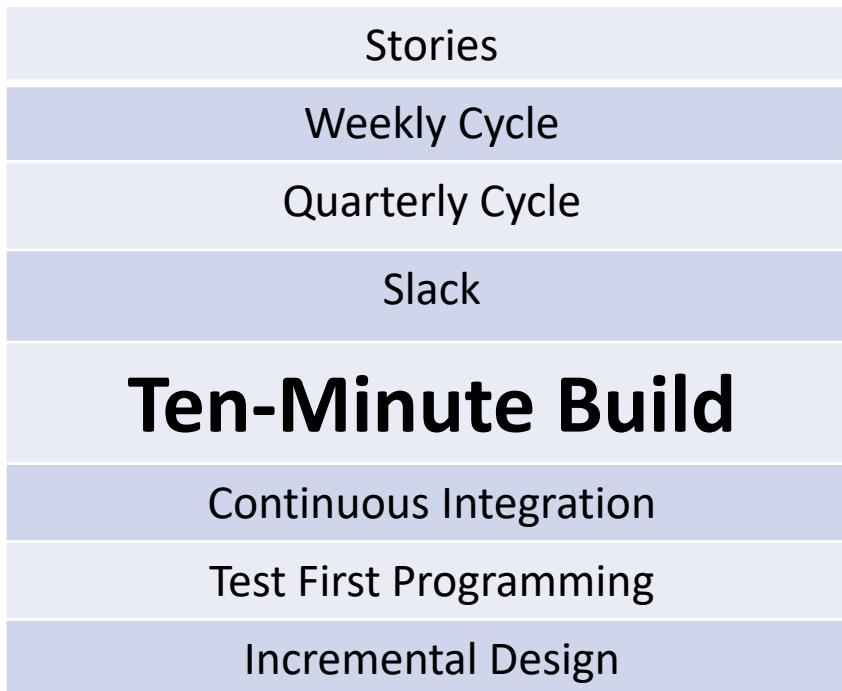
Stories
Weekly Cycle
Quarterly Cycle
Slack
Ten-Minute Build
Continuous Integration
Test First Programming
Incremental Design



Extreme Programming Practices



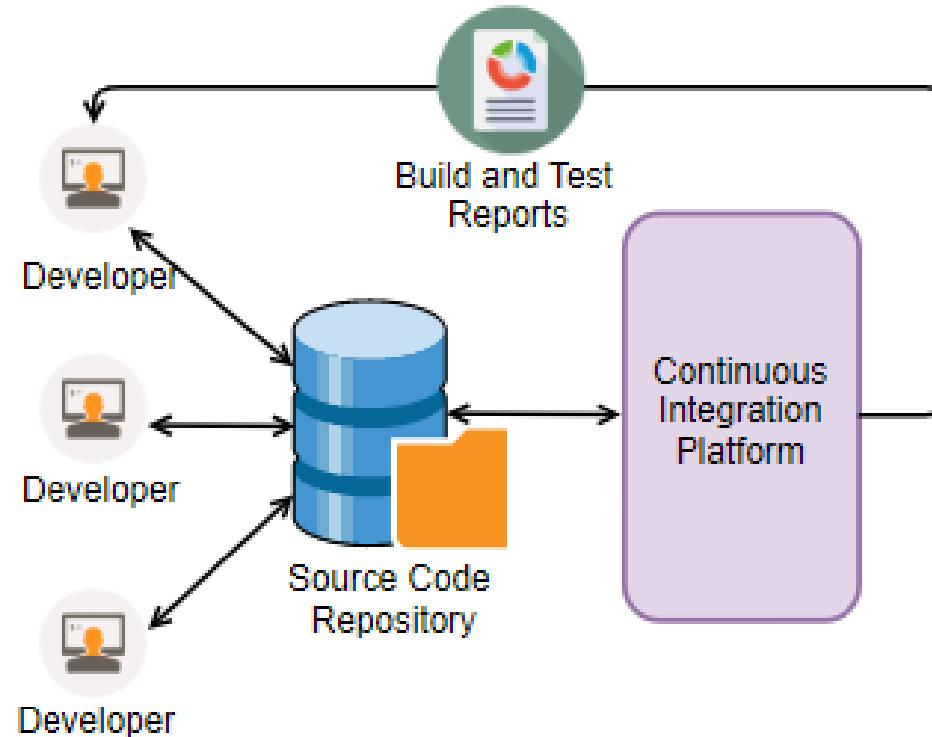
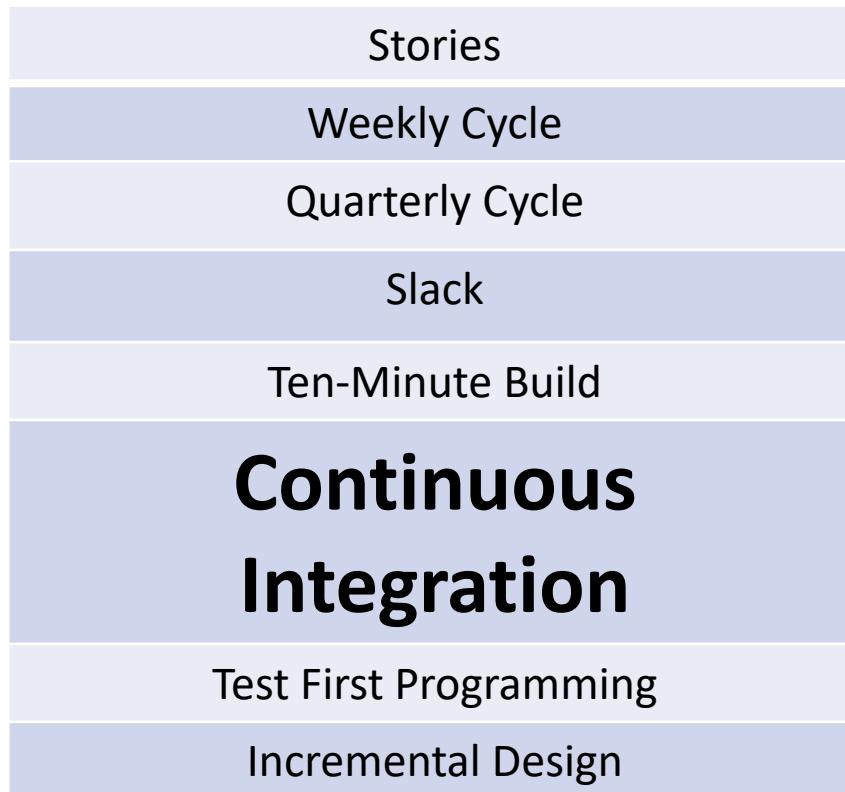
Primary Practices



Extreme Programming Practices



Primary Practices



Asynchronous Integration and Synchronous Integration



Extreme Programming Practices



Primary Practices

Stories

Weekly Cycle

Quarterly Cycle

Slack

Ten-Minute Build

Continuous Integration

Test First Programming

Incremental Design

develop code -> write tests -> run tests 

The practice of Test-First Programming follows the path of:

Write failing automated test -> Run failing test -> develop code to make test pass -> run test -> repeat



Extreme Programming Practices



Primary Practices

Stories

Weekly Cycle

Quarterly Cycle

Slack

Ten-Minute Build

Continuous Integration

Test First Programming

Incremental Design

