SDLC

Sunil Nagaraj , 21/10/21

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- Software Development

CONCEPT

Software Development Life Cycle

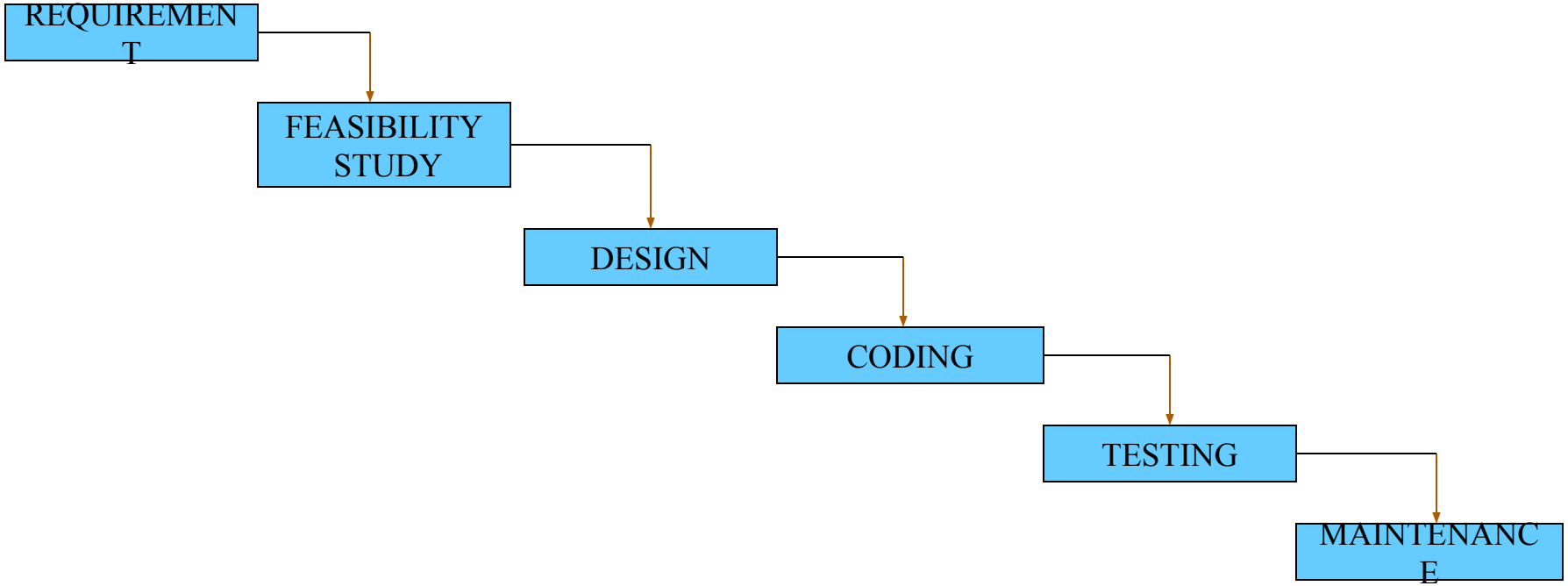
Software

- Software is defined as set of instructions to perform specific task

Software Development Life Cycle

- SDLC is a process of converting user requirements into fully developed software.
- SDLC is step by step procedure to convert user requirements into a software
- SDLC involves activities required to transfer user information in fully developed software
- SDLC starts with identification of requirements & ends with validation of developed software against its requirements

SDLC Phases



Requirement

- Requirements definition is the process by which the needs of the customer are translated into a clear, detailed specification of what the system must do.
- Requirements will be collected from the clients.
- Testing should begin in the requirements phase of the Software Development Life Cycle.

Feasibility Study

- Feasibility study is used to determine if the project should get the go-head.
- If the project is to proceed, the feasibility study will produce a project plan and budget estimates for the future stages of development.
- In this phase, the development team visits the customer and studies their system.

Design Phase

- During design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests works.

- Types Of Design
  - High Level Design [HLD]
  - Low Level Design [LLD]

High Level Design

- High level design on SDLC, software analysis will convert the requirements into a usable product in terms of functional architecture and data base design.
- It designs the over all architecture of the entire system from main module to all sub module. This is very useful for the developers to understand the flow of the system.
- For this the entry criteria are the requirement document that is SRS and the exit criteria will be the HLD projects standards the functional design and the database design documents

Low Level Design

- Specification for each program in the system is finalized
- In Low level design we create more detail and specific design of the system.
- The Low level document gives the design of the actual program code which is designed based on the HLD document
- Functionality and logic is designed in detail for sub-modules.

Coding Phase

- After design we do coding.
- The design must be translated into machine readable form.
- Different High level programming languages like c, c++, java are used for coding.
- With respect to the type of application, the right programming language is chosen.

Testing Phase

- Once code is generated , the software program testing begins.
- Testing is catching bugs on executing the program.
- Testing is the process of confirming whether developed software meets the customer expectations

Maintenance

- A good software product should effectively respond to the changing requirement of the customer or user.

- Any modification which needs to be taken care in the developed software will happen in maintenance phase

CONCEPT

SDLC Models

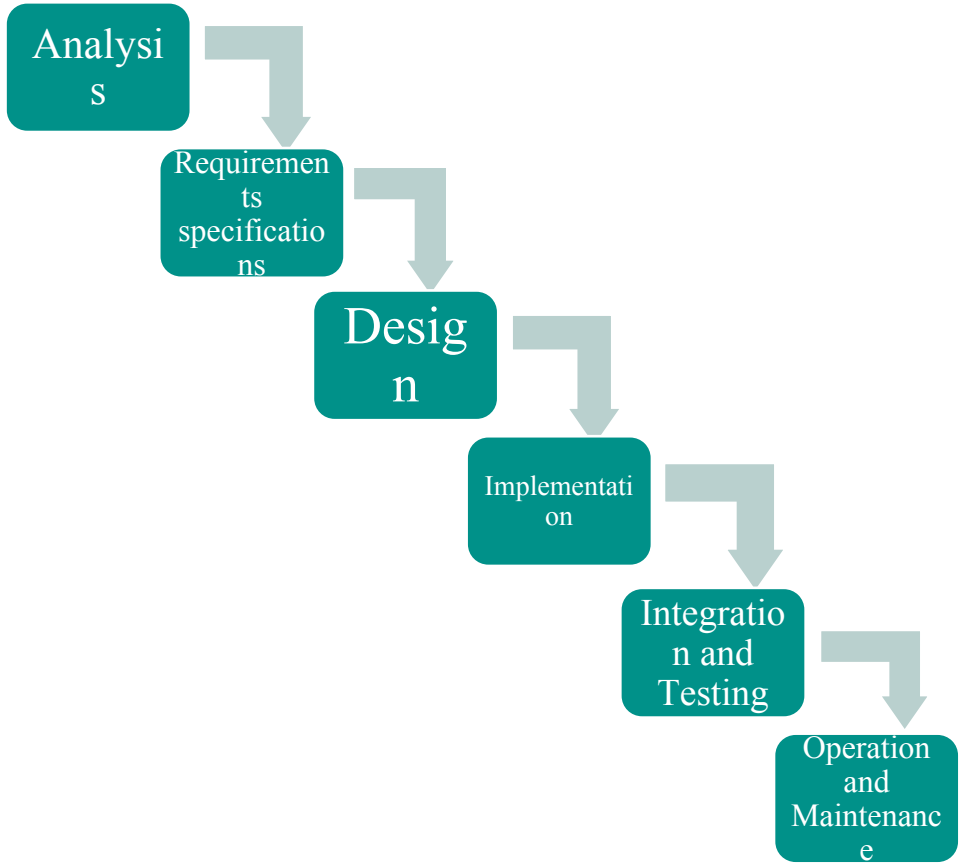
SDLC MODELS

Generic software process models

- Waterfall model
- Evolutionary models
  - Prototyping model
  - Spiral model
- Incremental models
  - Iterative model
  - RAD model
- Agile models
  - Scrum

WATERFALL MODEL

- Classical life cycle and linear sequential model is widely used
- Suggests a systematic sequential approach:
  - Conception & Initiation
  - Analysis
  - Design
  - Construction
  - Testing
  - Deployment
  - Maintenance / Support



WATERFALL MODEL

Phases of Waterfall Model

- Initial analysis
  - Specifying the problem background, business goals and success criteria
- Requirement analysis and definition
  - The goals and constraints of the system to be developed are established in consultation with system users
  - They are then defined in detail and serve as a system specification
- System and software design
  - Partitions the requirements to either hardware or software
  - Software design involves identification of necessary fundamentals of the software requirement and their relationship with those systems
- Implementation
  - Whole of software components are integrated as a set of programming units

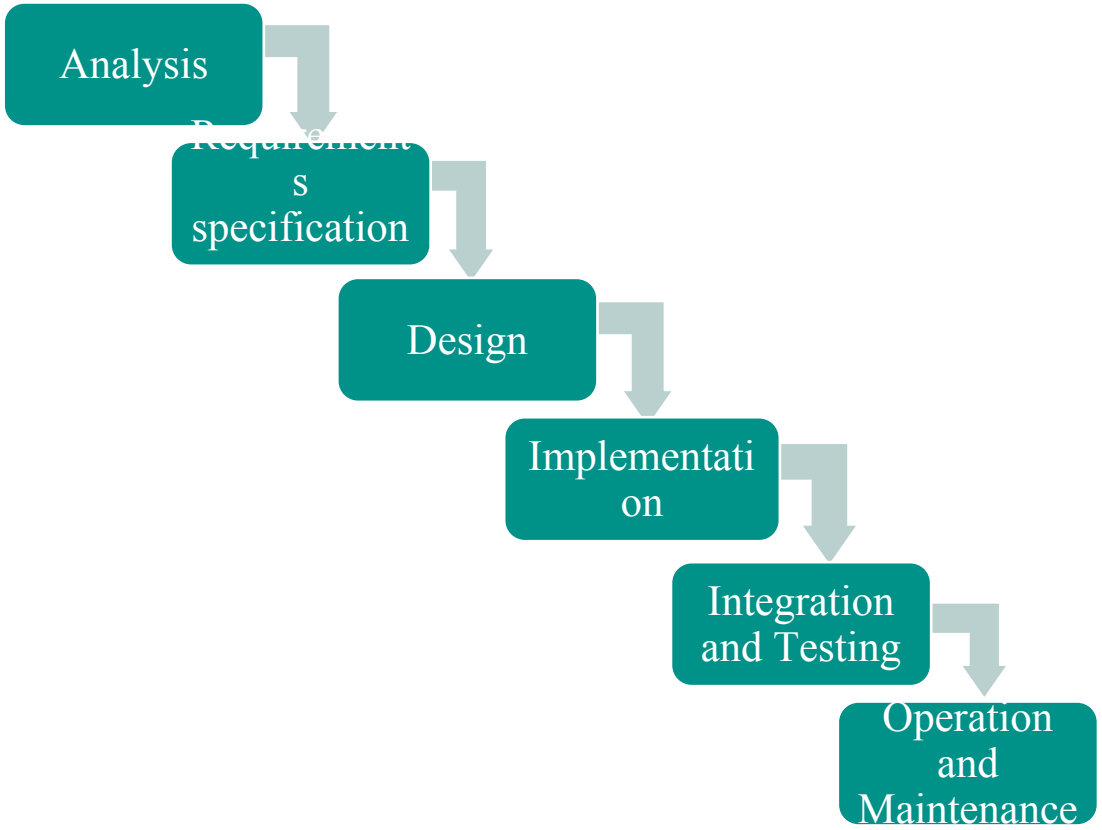
WATERFALL MODEL

Phases of Waterfall Model (contd …)

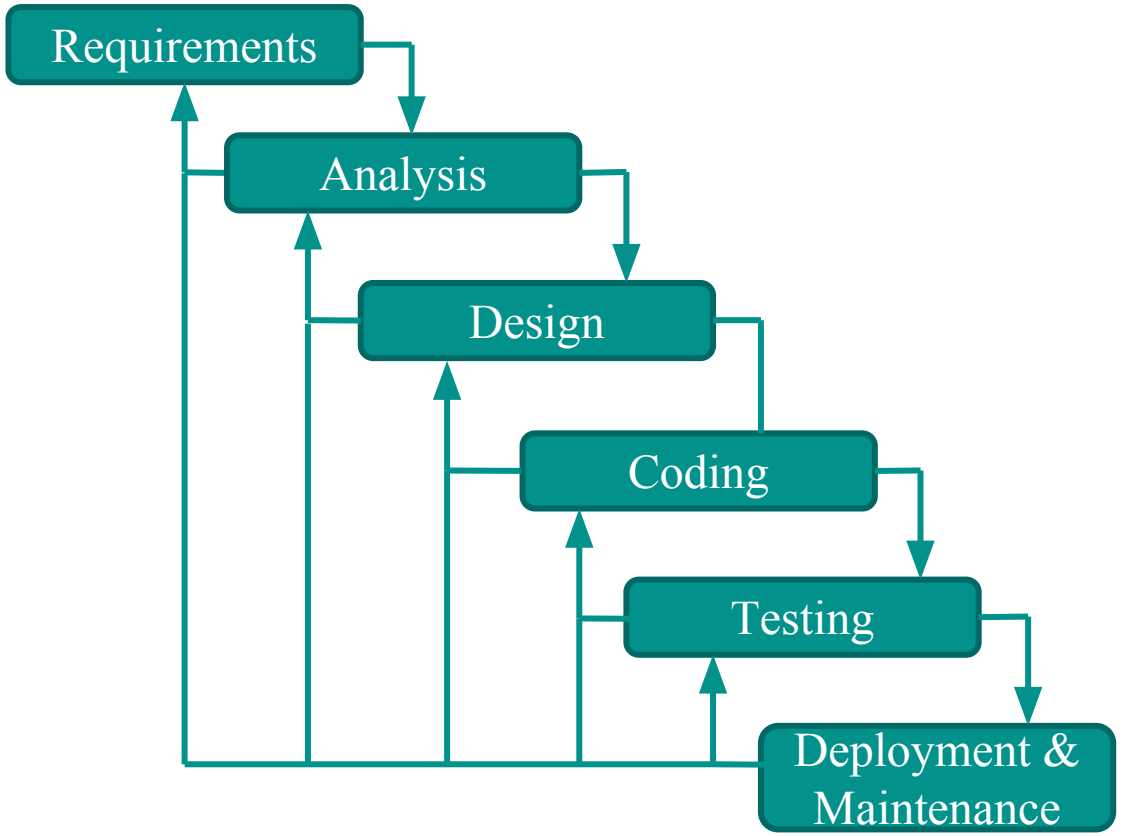
- Unit testing
  - Involves verifying that each unit meets its necessary requirement
- Integration and system testing
  - The individual program units are integrated and tested as a one complete system to ensure that all the necessary requirements have been met.
- Deployment/installation
  - After testing the system is installed in the final environment
- Operation and maintenance
  - Longest life cycle phase
  - Involves correcting the encountered errors which were not discovered at earlier stages
  - Also by enhancing the systems services, new requirements are discovered

WATERFALL MODEL

Traditional Model



Practitioner's Model



WATERFALL MODEL

Advantages

- Easy to understand and Implement
- Widely used and known
- Reinforces good habits: define-before-design
- Identifies deliverables and milestones
- Document driven
- Works well on mature products and weak teams
- Fits other engineering process models

WATERFALL MODEL

Disadvantages

- Idealized, doesn't match reality well
- Does not reflect iterative nature of exploratory development
- Unrealistic to expect accurate requirements early in the project
- Delays discovery of errors
- Difficult to integrate risk management
  - Difficult to accommodate change after the process is underway
  - One phase has to be competed before moving to the next phase.
- Difficult and expensive to make changes to document

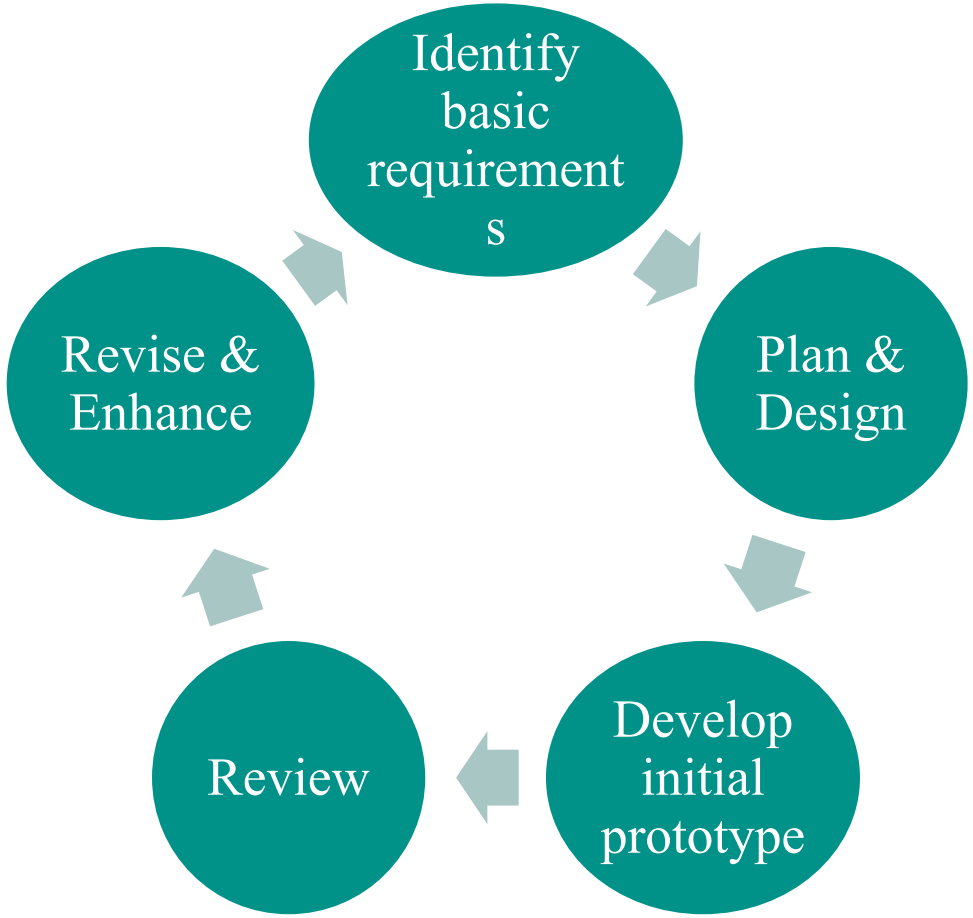
WATERFALL MODEL

When to use?

- Requirements are very well known
- Product definition is stable
- Technology is understood
- Creating a new version of an existing product
- Porting an existing product to a new platform

PROTOTYPING MODEL

- Allows users of the software to evaluate developers' proposals for the design of the final product by actually trying them out by means of a prototype of the screens
- Avoids the hardship of end users having to interpret and evaluate the design based on descriptions
- Users review and provide feedback on changes required for the prototype
- This is repeated until all requirements of system are identified



PROTOTYPING MODEL

Advantages

- Users are actively involved in the development
- Errors can be detected much earlier
- Quicker user feedback is available
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified

PROTOTYPING MODEL

Drawbacks

- Leads to implementation and then repairing way of building systems
- Practically this methodology may increase the complexity of the system, as scope of system may expand beyond original plans
- Incomplete or inadequate problem analysis

PROTOTYPING MODEL

When to use?

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development

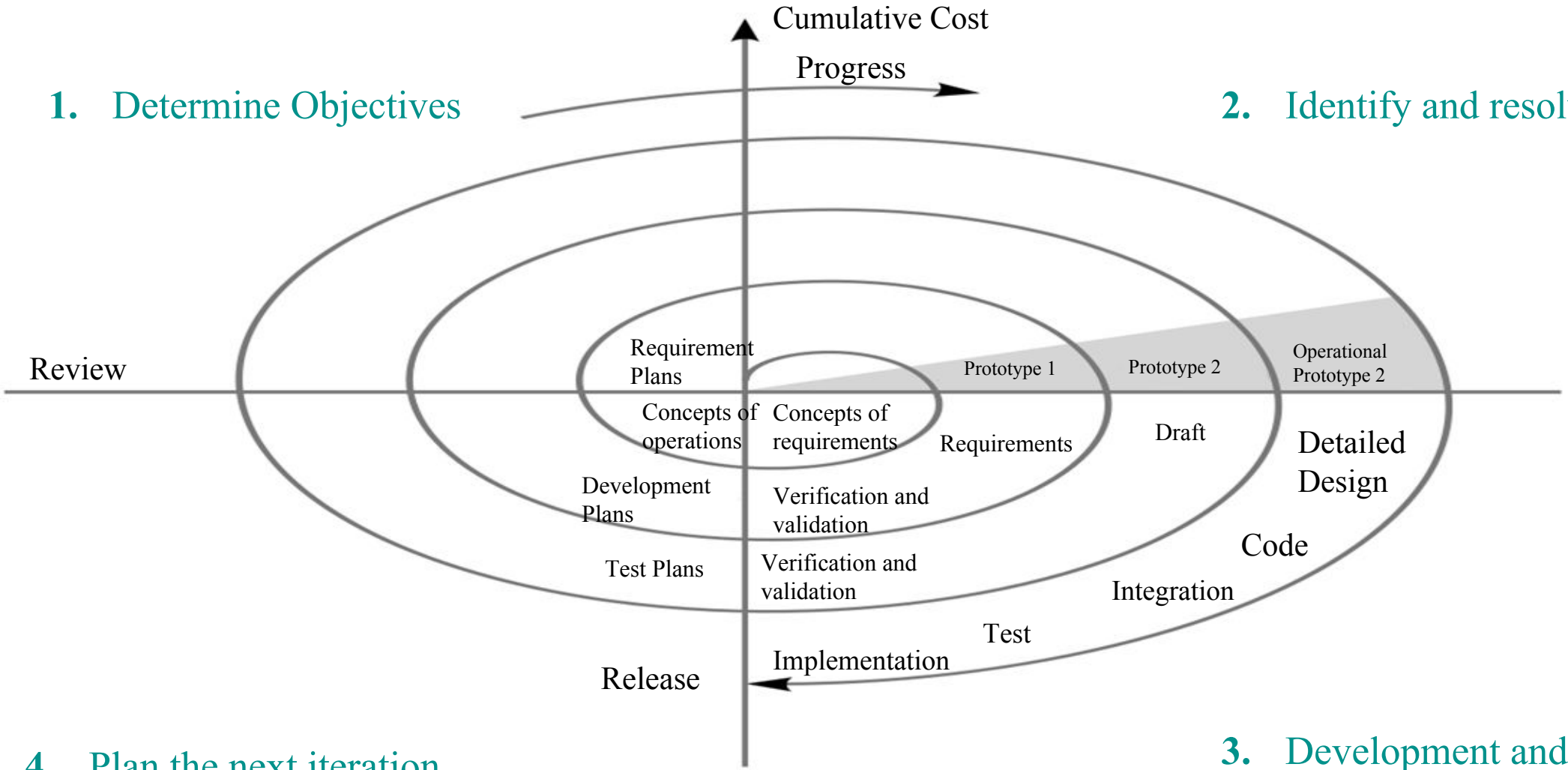
SPIRAL MODEL

- Combines the idea of iterative development with the systematic, controlled aspects of waterfall model.
- Has four phases:
  - Planning - requirements gathering phase (initial and also in subsequent spirals)
  - Risk analysis - identification of risk and alternative solutions. Creation of prototype at the end of this phase
  - Engineering - Development, testing & delivery of software
  - Evaluation - Evaluation of the software by the customer before the project continues to the next spiral

SPIRAL MODEL

1. Determine Objectives

2. Identify and resolve risk



4. Plan the next iteration

3. Development and Testing

SPIRAL MODEL

Advantages

- High amount of risk is resolved
- Software is produced at an early stage in the software life cycle
- Good for large and mission-critical projects
- Strong approval and documentation control

SPIRAL MODEL

Disadvantages

- It may be difficult to convince customers that the evolutionary approach is controllable
- It demands considerable risk assessment
- Can be a costly model to use
- Does not work well for small projects

SPIRAL MODEL

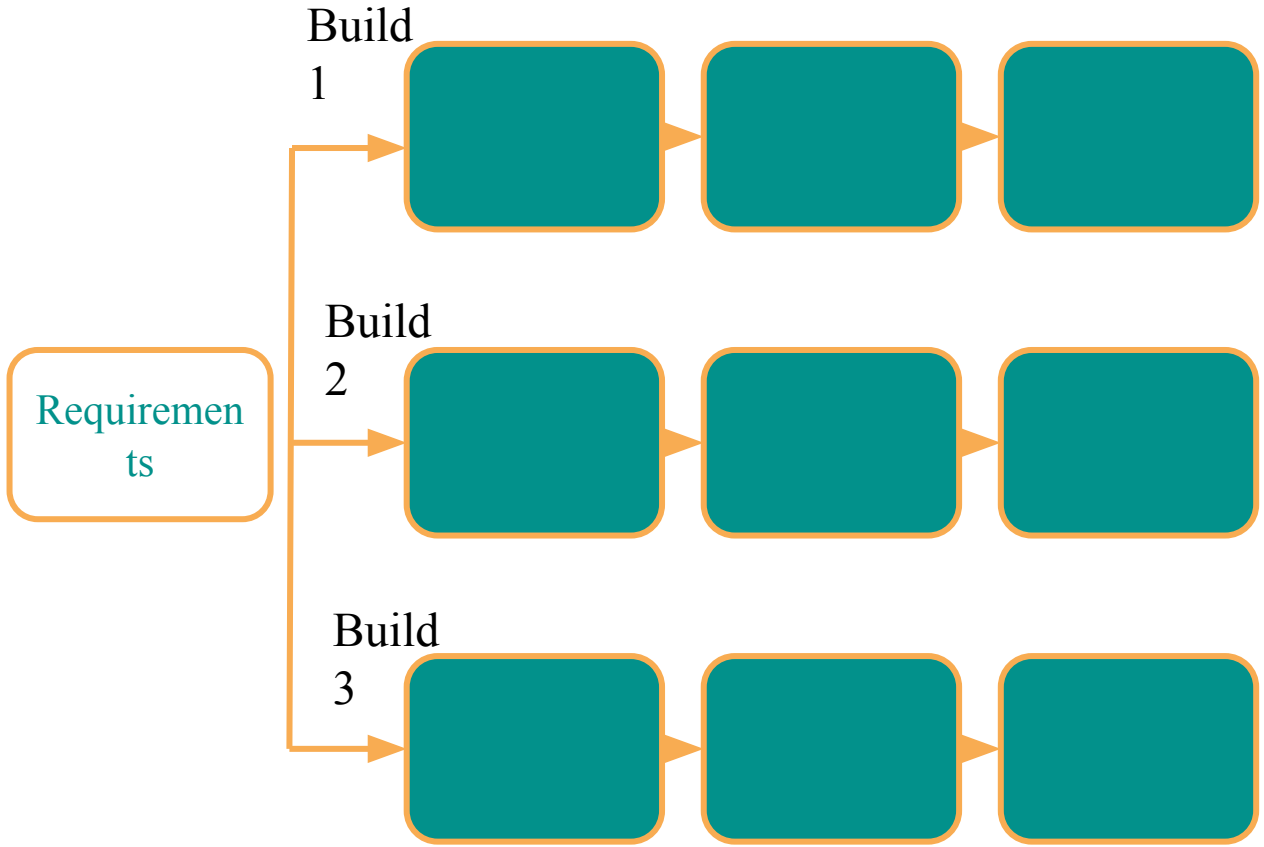
When to use?

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment is unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

INCREMENTAL MODELS

Iterative model

- Development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements
- The process is then repeated, producing a new version of the software for each cycle of the model
- The basic idea is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental)



- **Incremental/Overlapping Waterfall Model Overview:**
  - A type of evolutionary method.
  - The objective is to develop the system in incremental parts.
  - Requirements are prioritized and then implemented in groups; “divide and conquer”.
  - Each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.

- **Incremental/Overlapping Waterfall Model Assumptions:**
  - Requirements are first defined and then can be prioritized.
  - The product is capable of partial implementation delivery.

- **Incremental/Overlapping Waterfall Model Strengths:**
  - Provides the opportunity to incorporate user refinements, resulting from experience with earlier releases, into subsequent releases.
  - Provides returns for the investment in development as increments are delivered.
  - Satisfies changing user requirements with increments to the software released over time.
  - Delivers an operational product with each release.

- **Incremental/Overlapping Waterfall Model Weaknesses:**
  - Requires good planning and design and careful partitioning of the product.
  - Requires well-defined interfaces between the increments, especially if they will be developed in parallel.
  - Requires early definition of a fully functional product to define the increments.

Project Factors	Life Cycle			
	Waterfall	Incremental	Prototyping	Spiral
Technology / Architecture, Stable	✔	✔		
Technology / Architecture, New			✔	✔
Requirements, Stable	✔			
Requirements, Volatile		✔	✔	✔
Requirements, Defined	✔	✔		
Requirements, Vague			✔	✔
Risks, High		✔	✔	✔
Risks, Low	✔			
System, Small and Simple			✔	
System, Large and Complex				✔
User-Interface, High			✔	
Partial functionality needs to be delivered early		✔	✔	

1. WaterFall is _____ model

- ❑ Linear Sequential
- ❑ Iterative
- ❑ Incremental
- ❑ Both B & C

1. _____ model will be used when there is risk involved in the software needs to considered

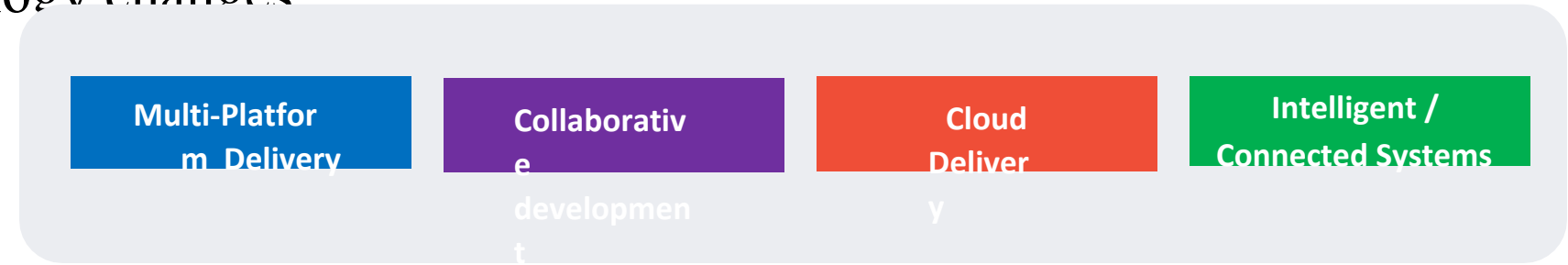
- ❑ WaterFall
- ❑ Spiral
- ❑ Prototype
- ❑ Agile

INTRODUCTION

Agile Methodology

Why Agile?

- Software development is to be adopted for new ways of working and technology changes



Impacts on Business and Software Development:

- Heavy demand for rapid product lifecycles
- Increasing interactions with customer
- Exploring adjacent or new markets
- New skills and competencies needed
- Increased security and privacy requirements
- Greater need to align with strategic priorities

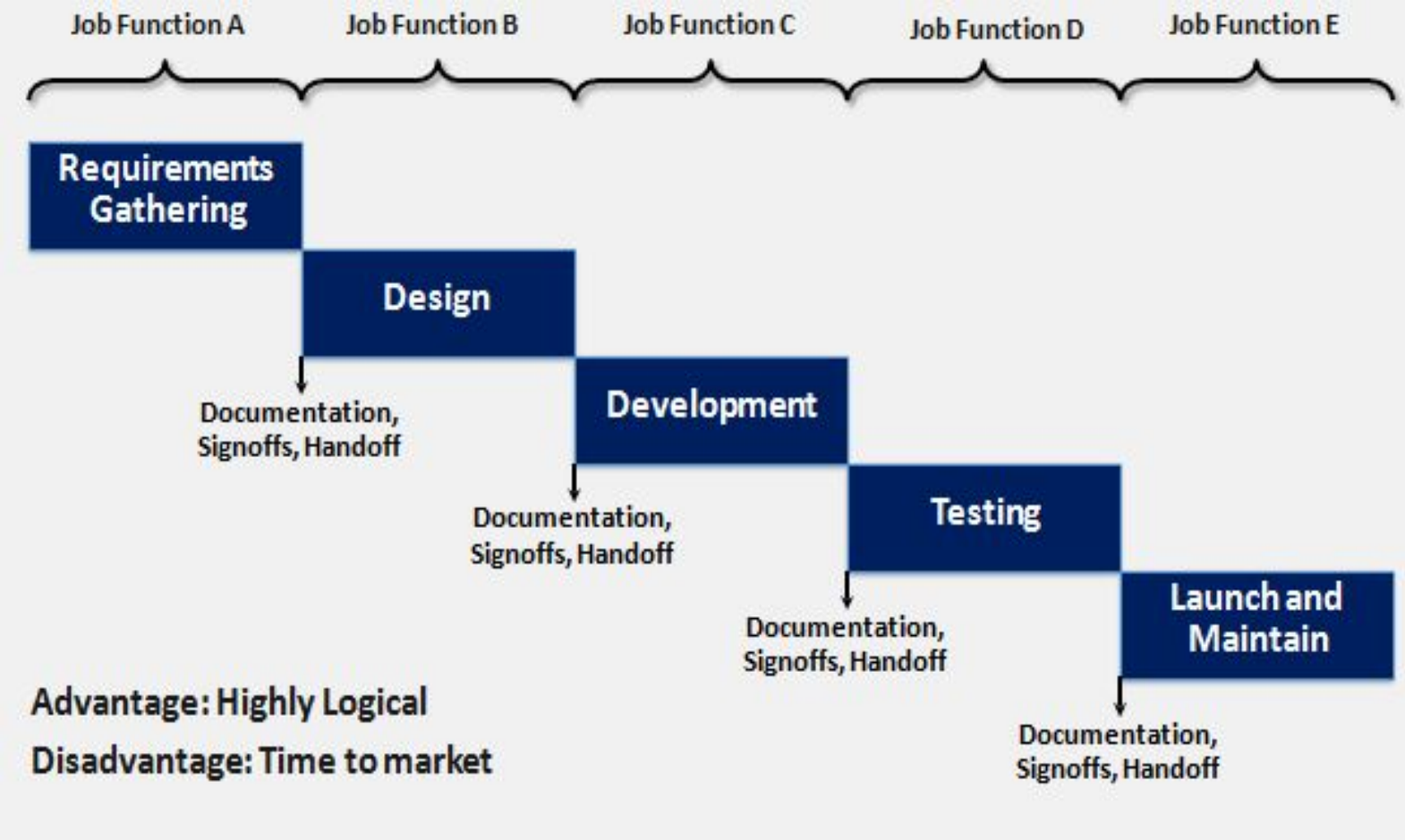
The Agile manifesto – 2001

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	>	processes and tools
Working software	>	comprehensive documentation
Customer collaboration	>	contract negotiation
Responding to change	>	following a plan

It means that while there is value in the items on the right, **we value the items on the left more**.

Waterfall model



CONCEPT

Scrum Basics

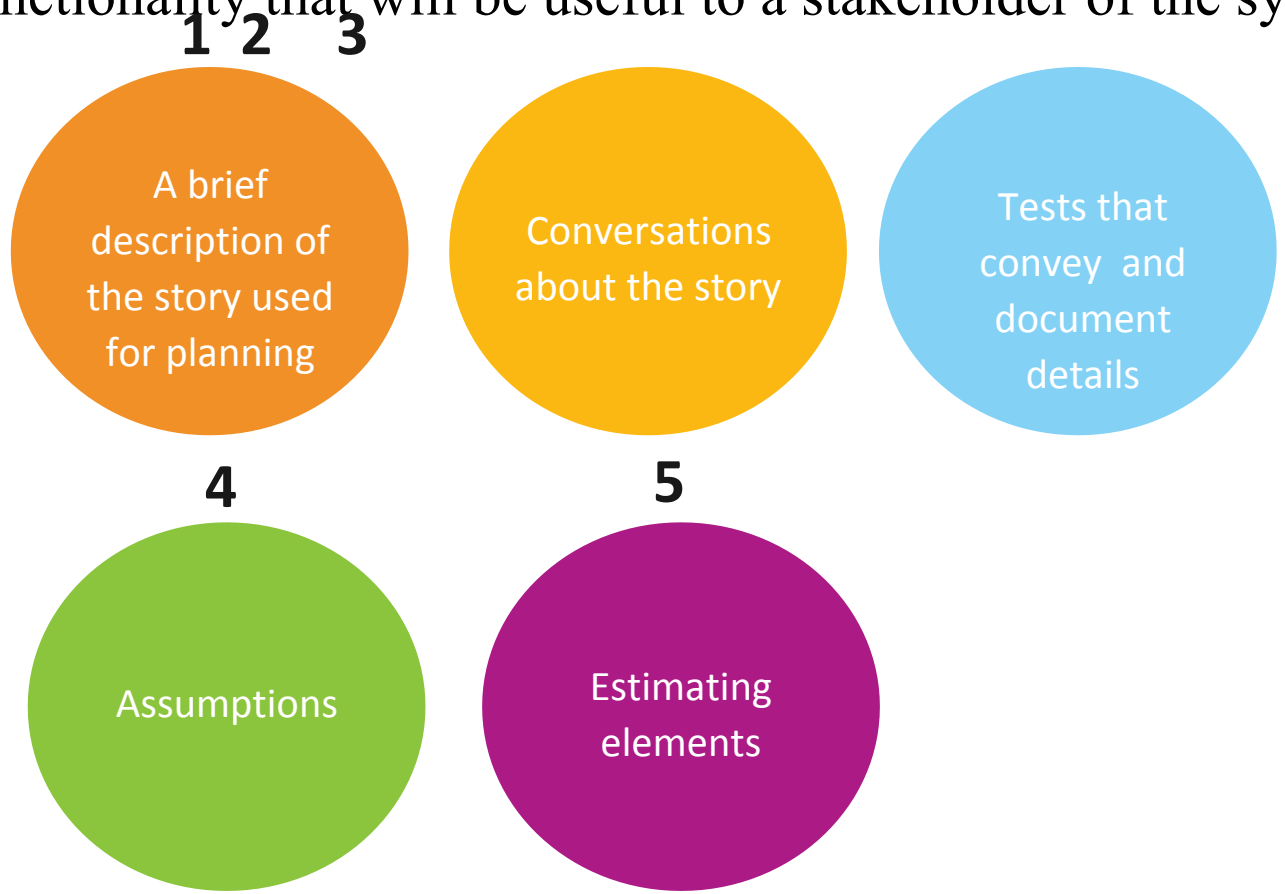
What is Scrum?

- Scrum refers to a holistic or “rugby” approach— where teams go the distance as a unit, passing the ball back and forth—as opposed to the traditional sequential or “relay race” approach for managing new product development.
- It is not individual effort but the best coordination of the team that will determine success.

User Stories: Components

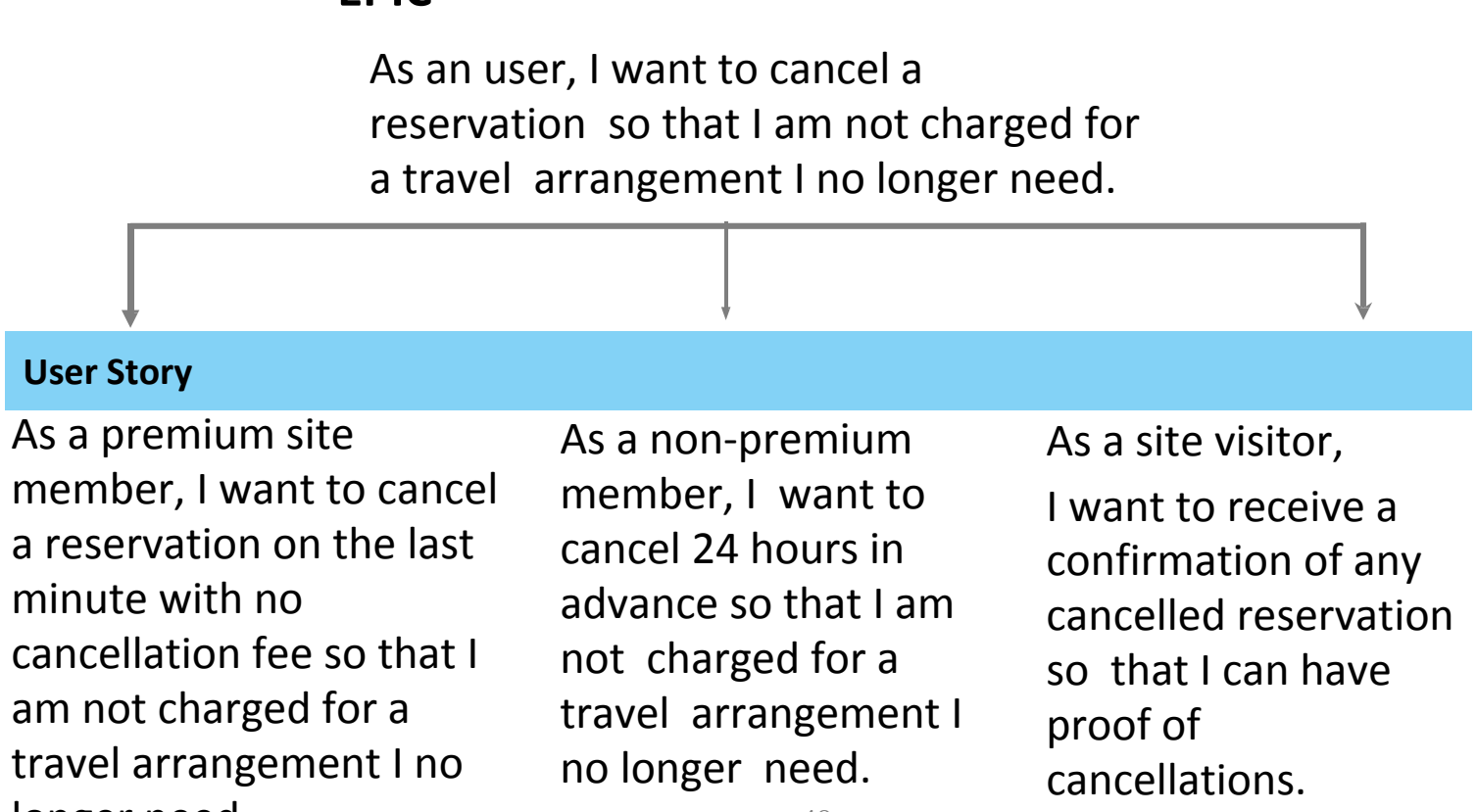
- A User Story describes functionality that will be useful to a stakeholder of the system.

User Stories consists of three things



What is an Epic?

- As a premium site member, I want to cancel a reservation on the last minute with no cancellation fee so that I am not charged for a travel arrangement I no longer need.



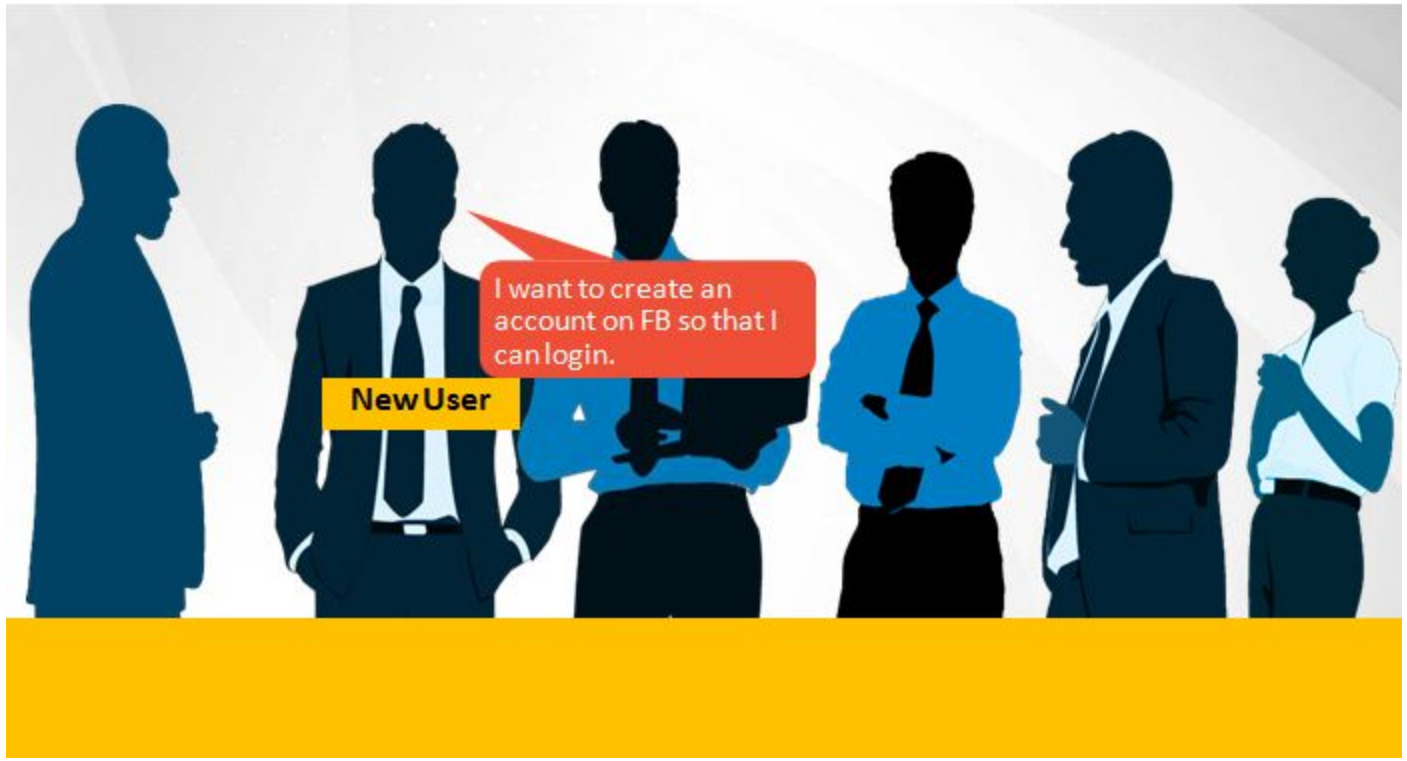
Product Backlog

To-Do List

ID	Story	Estimation	Priority
7	As an unauthorized user, I want to create a New account	3	1
1	As an unauthorized user, I want to login	1	2
10	As an authorized user, I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized user, I want to see the list of items so that I can select one	2	5
4	As an authorized user, I want to add a new item so that it appears in the list	5	6
3	As an authorized user, I want to delete the selected item	2	7
5	As an authorized user, I want to edit the selected item	5	8
6	As an authorized user, I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator, I want to see the list of accounts on login	2	10
Total		30	

Case study : Some possible user stories(1 of 6)

Case study : Some possible user stories(2 of 6)



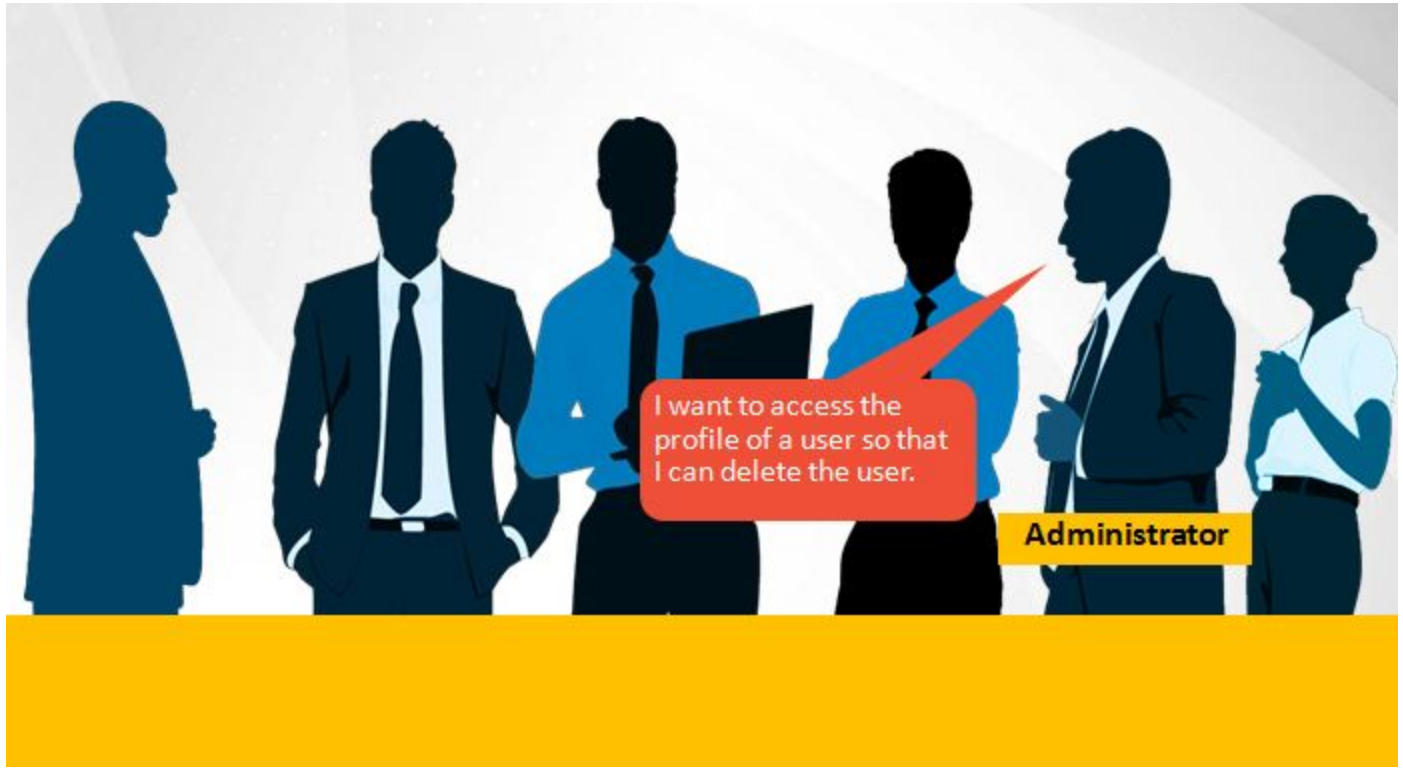
Case study : Some possible user stories(3 of 6)

52

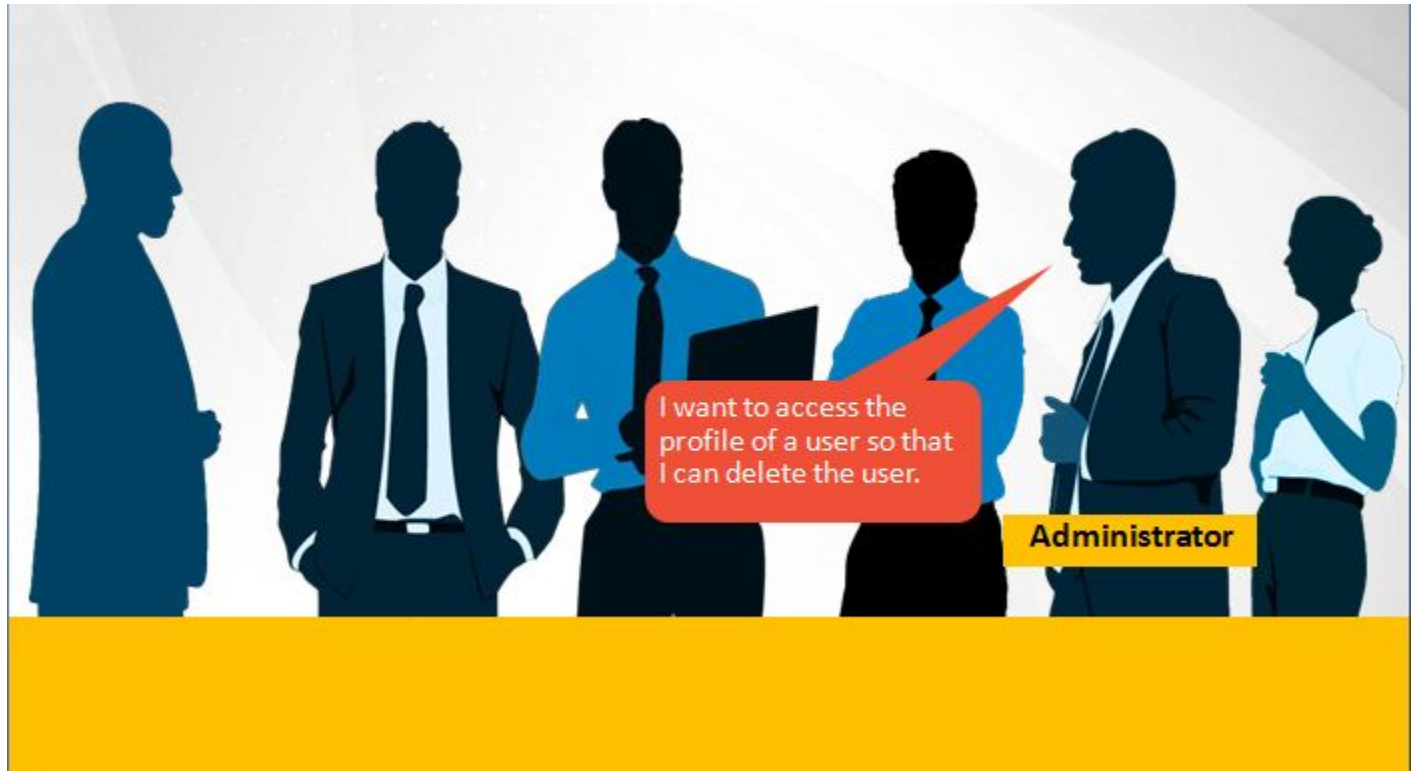
Case study : Some possible user stories(4 of 6)

Case study : Some possible user stories(4 of 6)

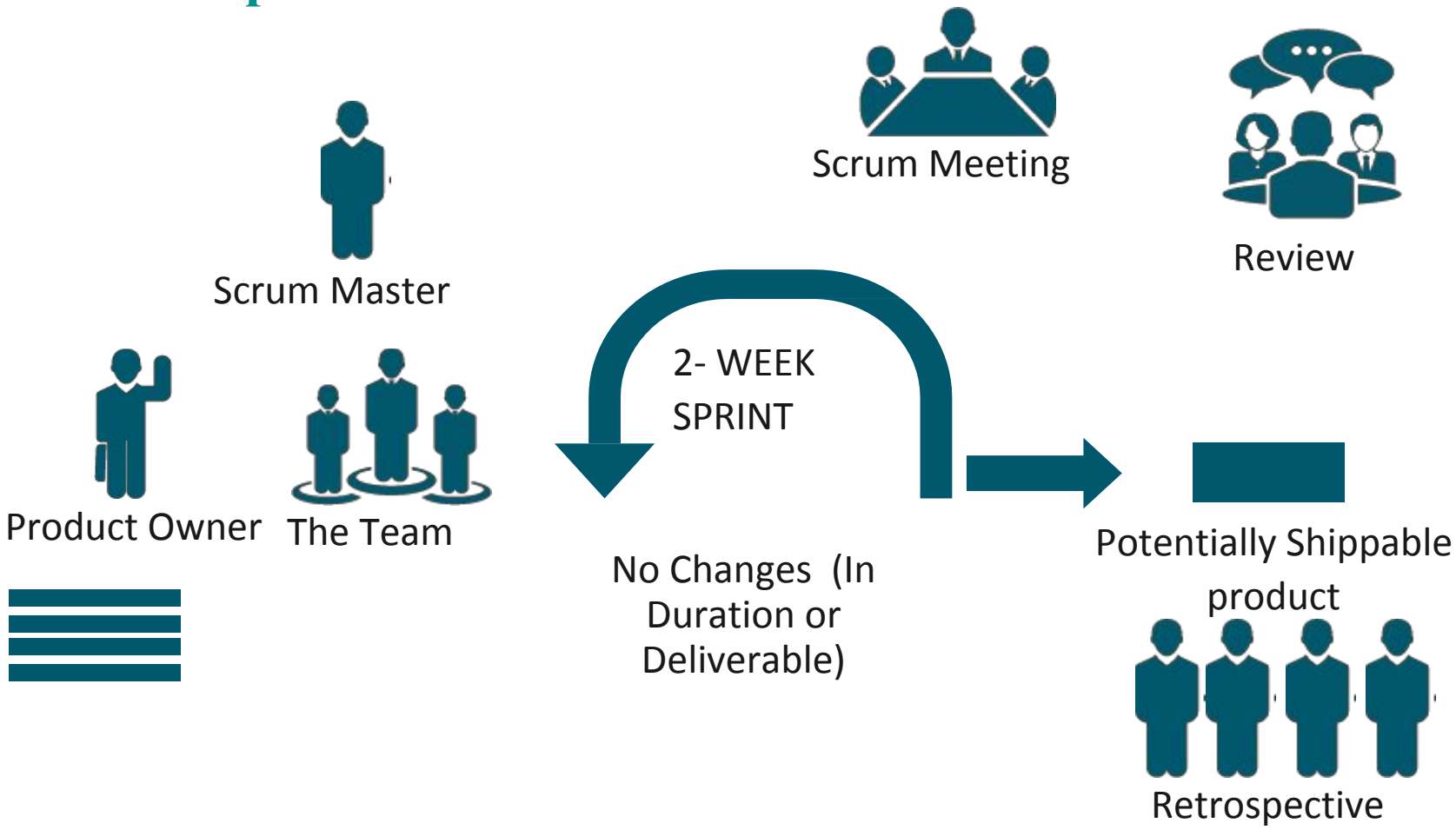
Case study : Some possible user stories(5 of 6)



Case study : Some possible user stories(6 of 6)



Scrum roles and responsibilities



Scrum roles and responsibilities—product owner

Scrum roles and responsibilities—the team

7 people + or – 2

- Can be shared with other teams (but it is better when not)
- Can change between Sprints (but better when they don't)
- Can be distributed (but better when collocated)

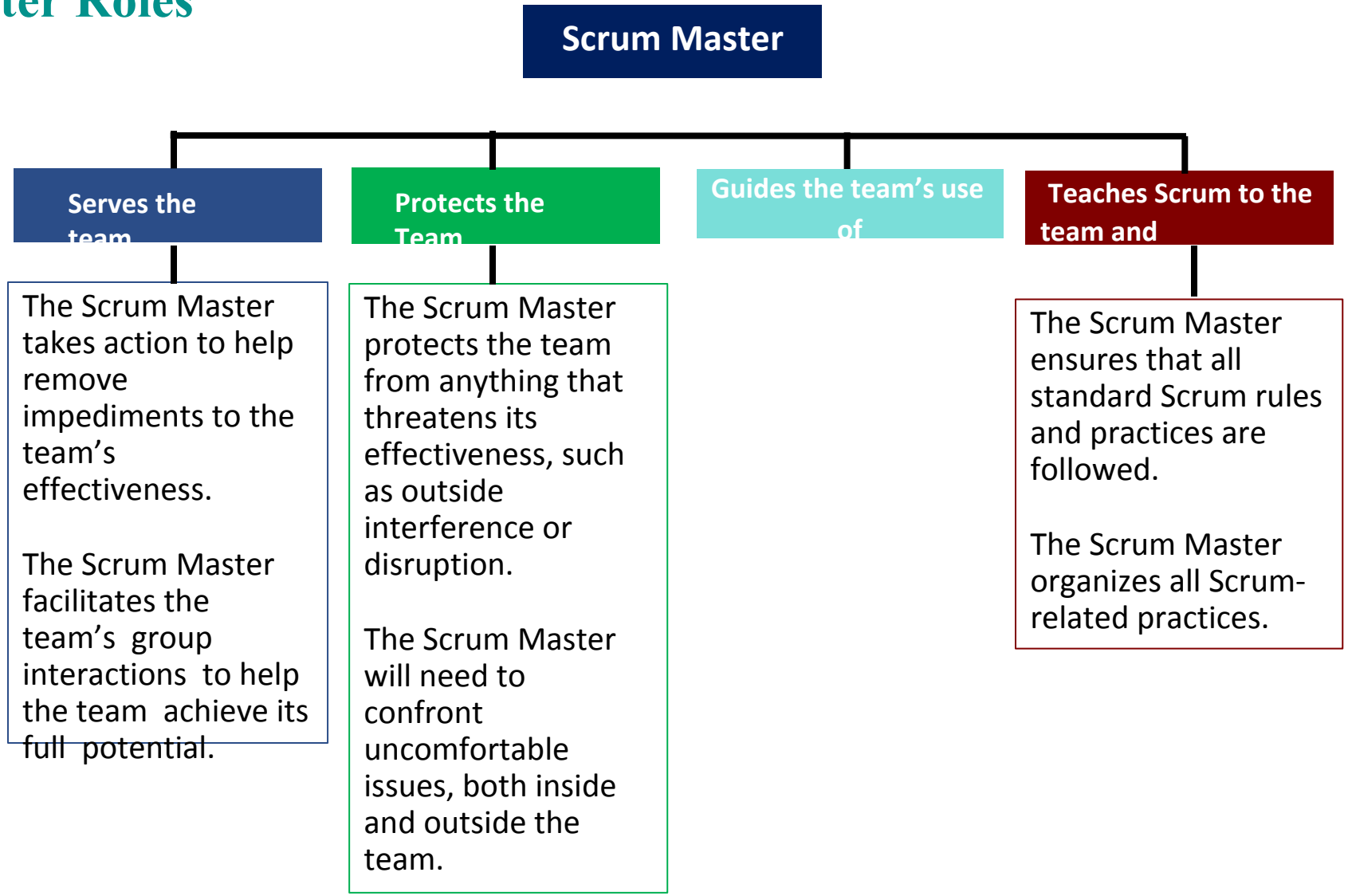
Cross-functional

- Possesses all the skills necessary to produce an increment of potentially shippable product
- Team takes on tasks based on skills, not just the official "role".

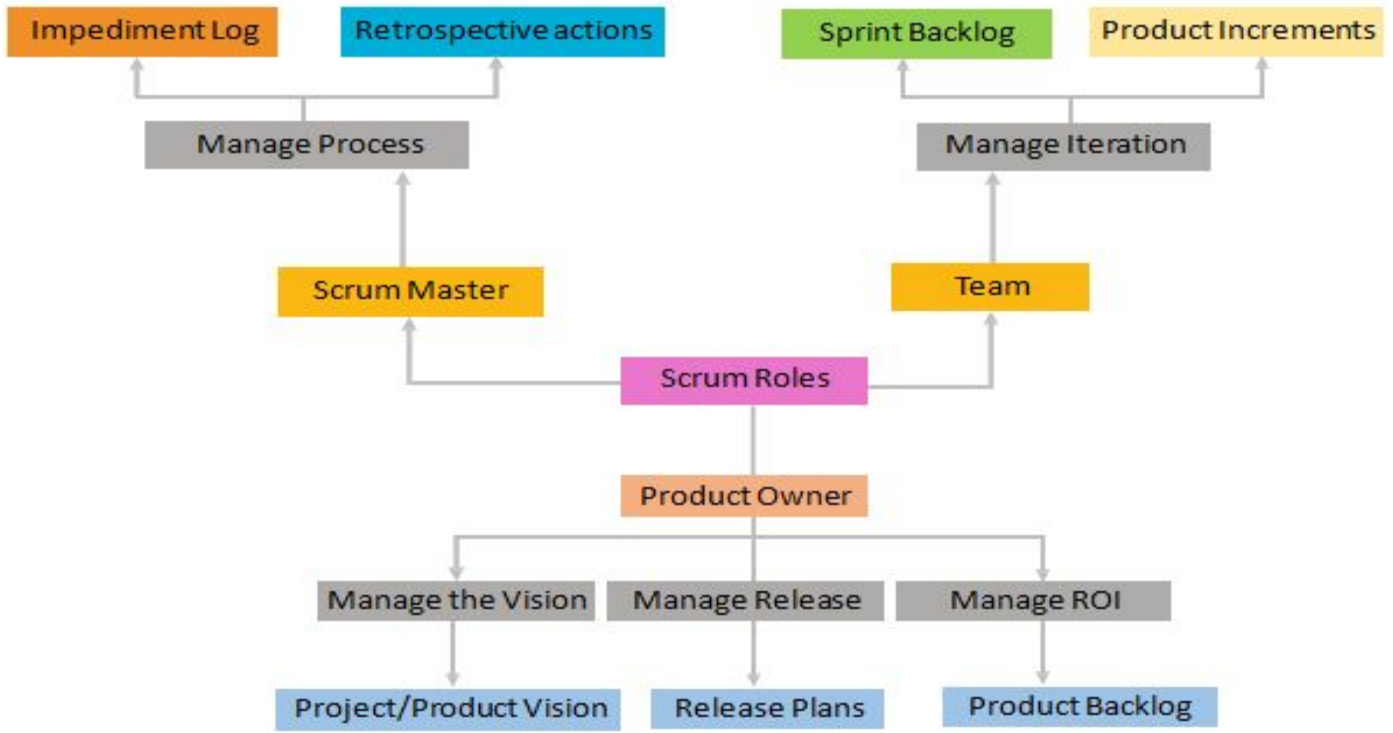
Self-managing

- Team manages itself to achieve the Sprint commitment.

Scrum Master Roles



Scrum project roles in a nutshell



Sprint planning

What is on the Product Owner's "shopping list"?

Team understands details of what Product Owner has prioritized on Product Backlog.

How much "money" do we have in our bank account?

Team decides how much productive time is available during the Sprint.

How many items on the shopping list can we afford to "buy" with that "money"?

Team decides how many Product Backlog items can be committed for completion during the Sprint.

Sprint Backlog

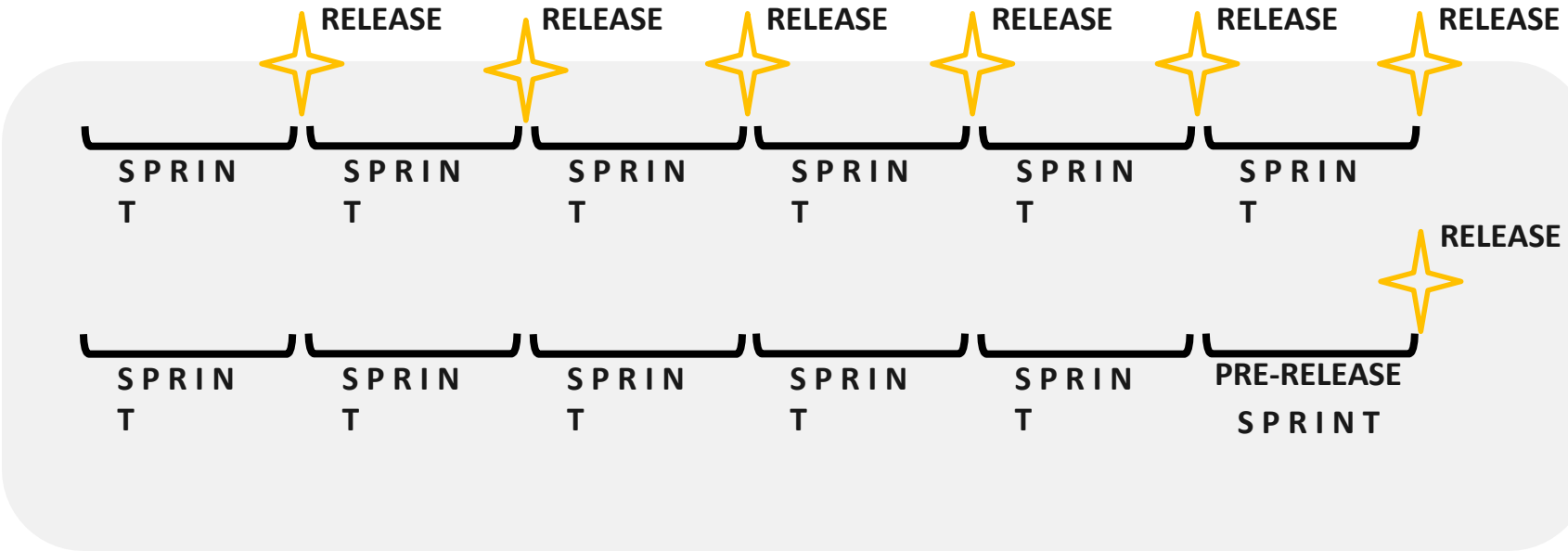
			Day of Sprint						
Backlog Item	Task	Owner	Initial Est.	1	2	3	4	5	6
Enable all users to place book in shopping cart	Design business logic	Sanjay	4						
	Design user interface	Jing	2						
	Implement back-end code	Tracy	2						
	Implement front-end code	Tracy	6						
	Complete documentation	Joe	8						
	Unit testing	Philip	4						
	Regression testing	Philip	2						
Upgrade transaction processing module	Implement back-end code	Tracy	5						
	Complete documentation	Joe	6						
	Unit testing	Philip	3						
	Regression testing	Philip	3						
Total			214						

Sprint Cycle: 2 Weeks

Mon	Tues	Weds	Thurs	Fri
5	6	1 1-Hr Pre-Meeting for next Sprint	2 8	3 Sprint Review & Retrospective 9
6 Sprint Planning Meeting	7 1	8 2	9 3	10 4
13 5	14 6	15 1-Hr Pre-Meeting for next Sprint 2	16 8	17 Sprint Review & Retrospective 4
20 Sprint Planning Meeting	21 1	22 2	23 3	24 4
27 5	28 6	29 1-Hr Pre-Meeting for next Sprint	30 8	31 Sprint Review & Retrospective 9

Definition of Done

In Scrum, “Done” is defined as “**Potentially Shippable**”.



CHECK YOUR UNDERSTANDING

1. Which one of these life cycle models does Agile follow?

- ❑ Waterfall Model
- ❑ Iterative and Incremental Model
- ❑ V Model

2. Which of the following prioritized items does the product backlog contain?

- ❑ Epics
- ❑ User Stories
- ❑ Both A & B

3. Which of the below roles is not defined by Scrum?

- ❑ Product Owner
- ❑ Project Manager
- ❑ Developer / Tester
- ❑ Scrum Master

CONCEPT

Scrum Execution

Daily Scrum Meeting

- Every weekday
- Whole team attends
- Everyone stands
- Lasts 15 minutes or less
- Product Owner can attend and report
- Update of artifacts after standup-Sprint backlog, Task board, and so on

71

Daily Scrum Meeting (continued)

Every Developer / Tester reports the following three things:

What task was I able to accomplish since yesterday? What task will I try to accomplish for the day?

Any issues blocking my work?

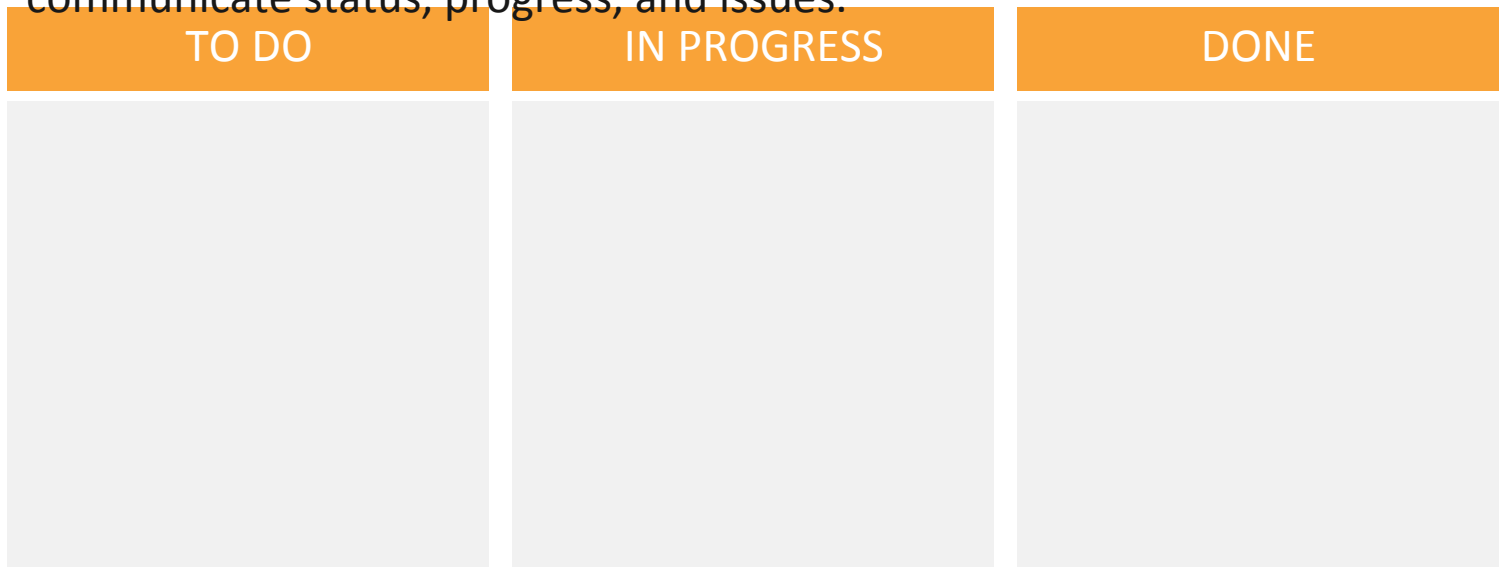
No discussions or conversations until meeting ends

Sprint Backlog Updation

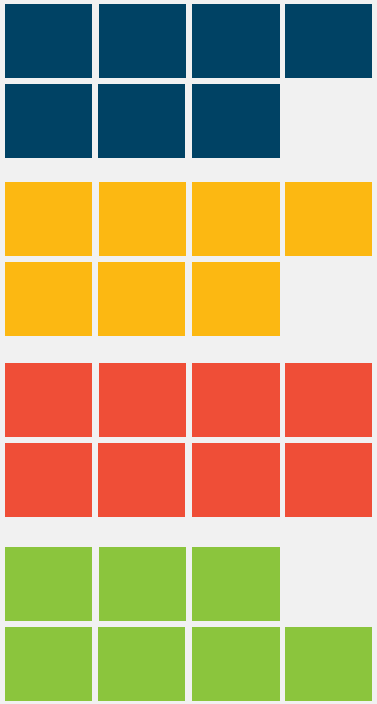
			Day of Sprint						
Backlog Item	Task	Owner	Initial Est.	1	2	3	4	5	6
Enable all users to place book in shopping cart	Design business logic	Sanjay	4	2	0				
	Design user interface	Jing	2	2	2				
	Implement back-end code	Tracy	2	4	2				
	Implement front-end code	Tracy	6	6	6				
	Complete documentation	Joe	8	6	6				
	Unit testing	Philip	4	3	3				
	Regression testing	Philip	2	2	2				
Upgrade transaction processing module	Implement back-end code	Tracy	5	10	8				
	Complete documentation	Joe	6	6	6				
	Unit testing	Philip	3	3	3				
	Regression testing	Philip	3	2	2				
Total			214	220	205				

Kanban Task Board (1 of 4)

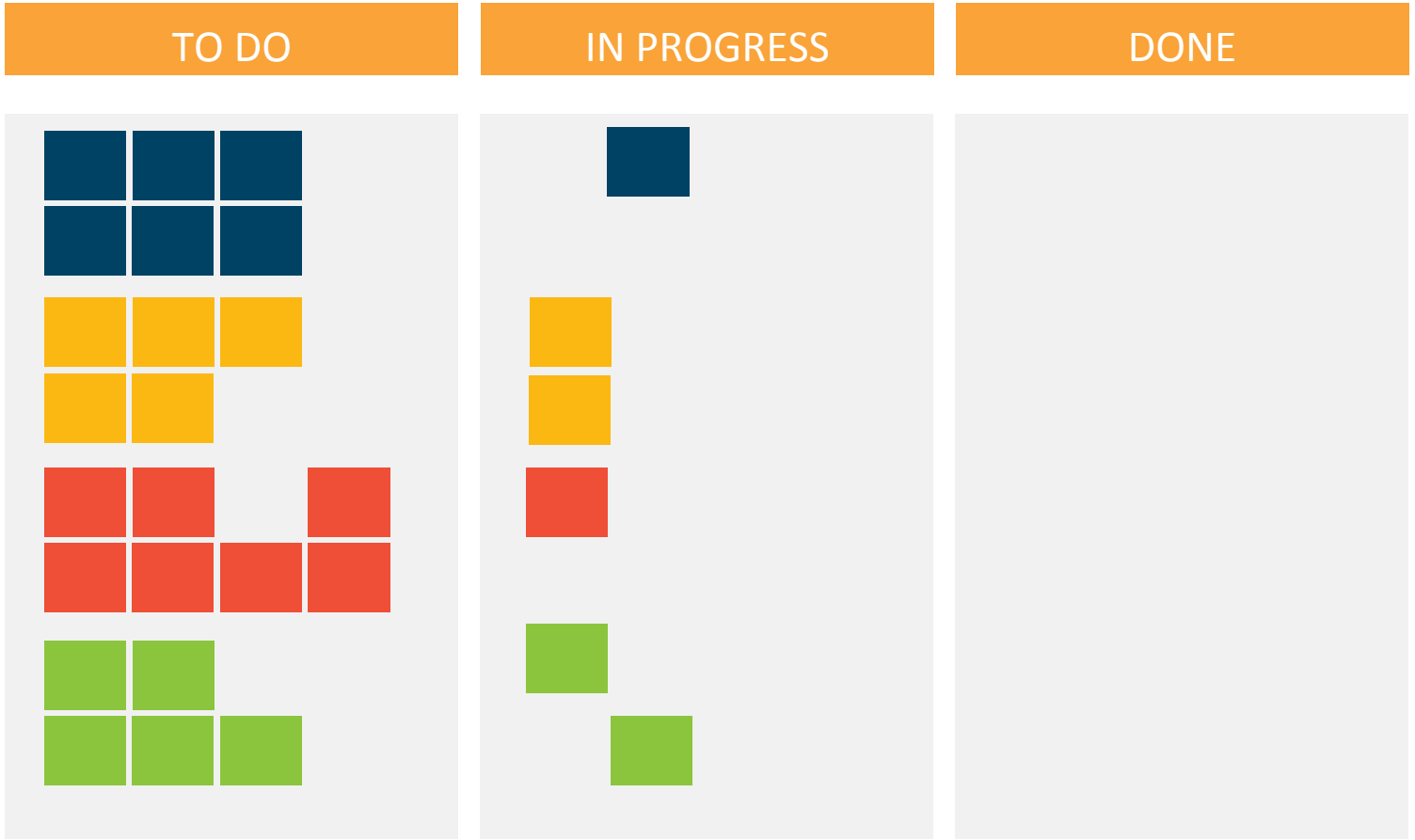
A Kanban board is a work and workflow visualization tool that enables you to optimize the flow of your work. Physical Kanban boards, like the one pictured below, typically use sticky notes on a whiteboard to communicate status, progress, and issues.



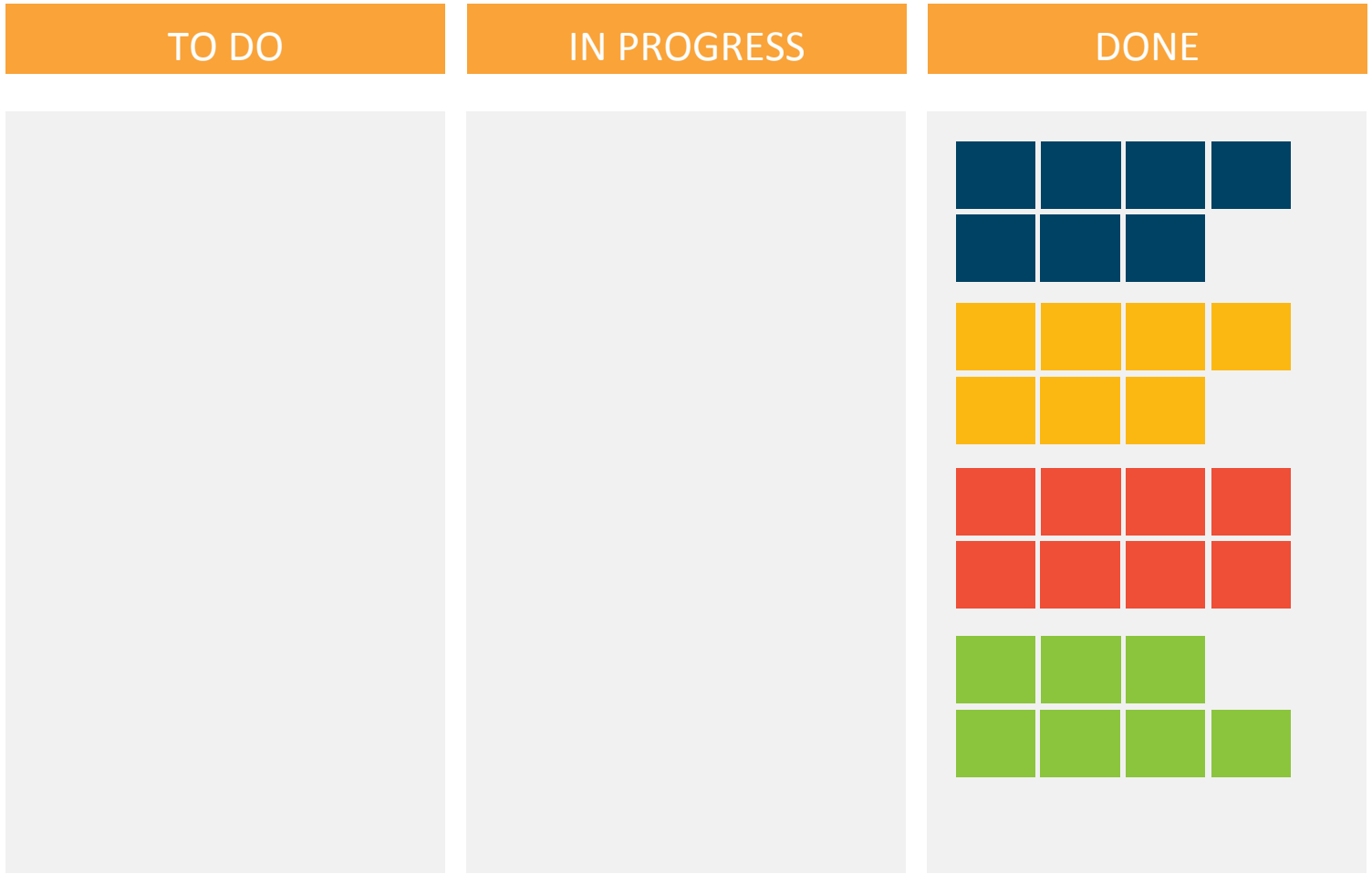
Kanban Task Board (2 of 4)

TO DO	IN PROGRESS	DONE
		

Kanban Task Board (3 of 4)



Kanban Task Board (4 of 4)



CHECK YOUR UNDERSTANDING

1. Product Owner has to mandatorily participate in Daily Scrum Meeting.

- ❑ Yes
- ❑ No
- ❑ Not required
- ❑ Need basis

2. Sprint backlog is updated by:

- ❑ Scrum Master
- ❑ Scrum Team
- ❑ Scrum Product Owner
- ❑ Any of the above

CONCEPT

Sprint Review and Retrospective

Sprint Review

- The purpose of the Sprint Review is:
  - Demo what the team has built
  - Generate feedback which the Product Owner can incorporate in the Product Backlog
- Attended by Team, Product Owner, ScrumMaster, functional managers, and any other stakeholders
- A demo of what's been built and not a presentation about what's been built:
  - No Power Points allowed!
- Usually lasts 1-2 hours
- Followed by Sprint Retrospective

Sprint Retrospective

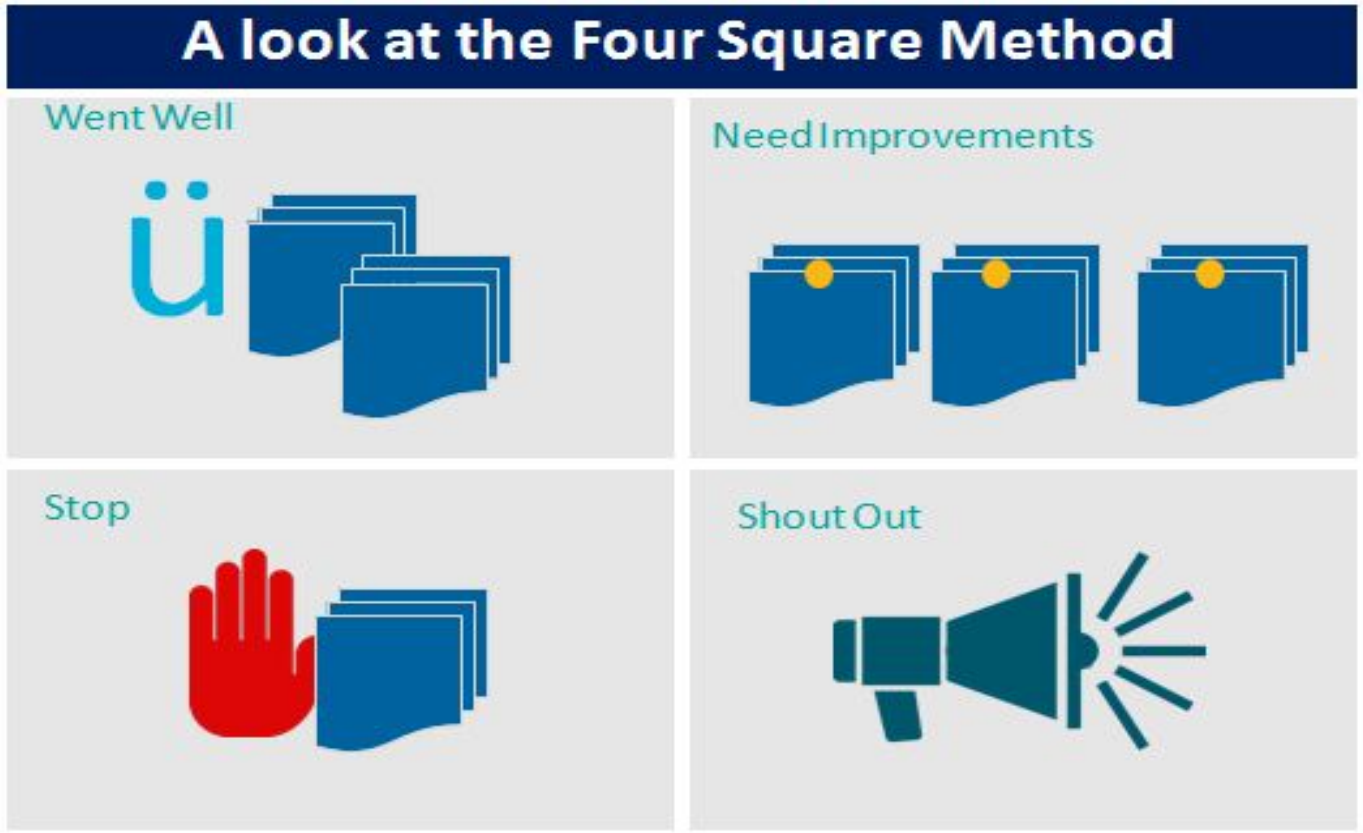
What is it?

- 1-2 hour meeting following each Sprint Demo
- Attended by Product Owner, Team, and ScrumMaster
- Usually a neutral person will be invited in to facilitate
- Presents what is working and what could work better

Why does the Retrospective matter?

- Accelerates visibility
- Accelerates action to improve

Sprint Retrospective: Four Square Method



CHECK YOUR UNDERSTANDING

1. Retrospective meetings help us to:

- ❑ Identify areas of improvement
- ❑ Recognize team members
- ❑ Inspect and Adapt
- ❑ A & B Only

QUIZ QUESTION

2. _____ is / are the one(s) who have legitimate interest in the project.

- ❑ Stakeholders
- ❑ Product owner
- ❑ Scrum master
- ❑ Scrum team

QUIZ QUESTION

3. Learnings of a sprint is incorporated into next sprint after a _____ meeting.

- ❑ Daily scrum
- ❑ Product backlog
- ❑ Sprint review
- ❑ Sprint retrospective

QUIZ QUESTION

4. A scrum team member should raise the issue about a software component not working, in a ____ meeting.

- ❑ Sprint planning
- ❑ Sprint review
- ❑ Daily scrum
- ❑ Customer review

SUMMARY

In this lesson, you've learned to:

- Software Development Life Cycle
- Different Phases of SDLC
- Different SDLC Models
- Agile