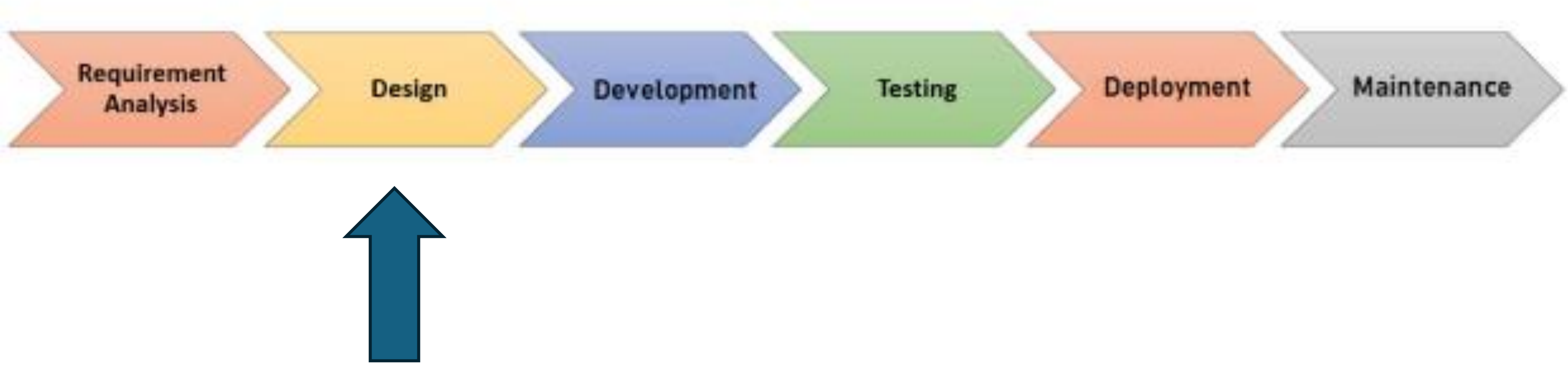
Introduction to Software Engineering

Day 4 - Session 1

Anjali Kulkarni

May 2024

Design Phase



What is a software design?

<u>Intermediate stage between conception of a software product and its development</u>

- Roadmap for developers
- Outline of software functions
- Elaboration of building blocks
- Interaction between components

Goal of Design Phase

Requirements →Actionable Steps

- To produce a model of the Software product
- To derive an architectural rendering of a system
- Detailed design
- Promotes Good Quality

Indications of a good design

- Accommodates all customer requirements (Use cases)
- Describes software from implementation perspective
    - Inputs and outputs
    - Functional aspects
    - Behavioral aspects
- Modular
- Scalable
- Well documented

Characteristics of a Good Design

Modularity

- Components with defined functions
- Loose coupling: Modules interact minimally
- High cohesion: Modules perform a single task well
- High Abstraction
- Single source of truth

Characteristics of a Good Design

Scalable

- Accommodates growth – users / data
- Flexible architecture - future expansion
- Consistent Performance
- Futuristic

Characteristics of a Good Design

Testable

- Designed to facilitate unit and integration testing
- Modules should be easily testable in isolation
- Clear separation of concerns aids testing
- Promotes early bug detection and higher quality software

Characteristics of a Good Design

Maintainable

- Clean and readable design instructions
- Easy to understand and modify
- Enforce meaningful notations
- Proper documentation

Actions ensuring good design

- Define clear goals and objectives
- Involve all stakeholders early
- Establish communication channels
- Brainstorming sessions
- User-centric design
- Wireframing and prototyping
- Modular design principles
- Coding standards and conventions
- Version control system

Requirements Phase

Design Approaches

- Function-oriented programming
  - Top-down approach
  - Breaking down a program into functions or procedures

- Object-oriented programming
  - Organizes code around objects that represent real-world entities.

Function-Oriented Programming

- Focuses on breaking down a program into functions or procedures
- Logic flow is typically sequential
- Each function performs a specific, well-defined task
- Functions can take inputs (parameters) and produce outputs

Strengths of FOP

- Simpler to understand for beginners.
- Easier to test and debug individual functions.
- Promotes code modularity by breaking down complex problems.
- Can be efficient for computationally intensive tasks.

Weaknesses of FOP

- Can lead to code duplication if similar functionality is needed in multiple places.
- Data can become difficult to manage and secure in a global context.
- Maintaining large codebases with many functions can become challenging.
- Lacks real-world modeling capabilities for complex systems.

Object-Oriented Programming

- Organizes code around objects that represent real-world entities.
- Objects encapsulate data (attributes) and behavior (methods).
- Objects interact with each other through messages.
- Inheritance allows for code reusability and specialization.
- Polymorphism enables flexible behavior based on object type.

OOP Design Principles

- SOLID
  - Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion
- DRY
  - Don't Repeat Yourself
- KISS
  - Keep It Simple, Stupid

Strengths of OOP

- Promotes code reusability through inheritance.
- Improves code maintainability with modular objects.
- Enhances data security through encapsulation.
- Models real-world entities effectively for complex systems.

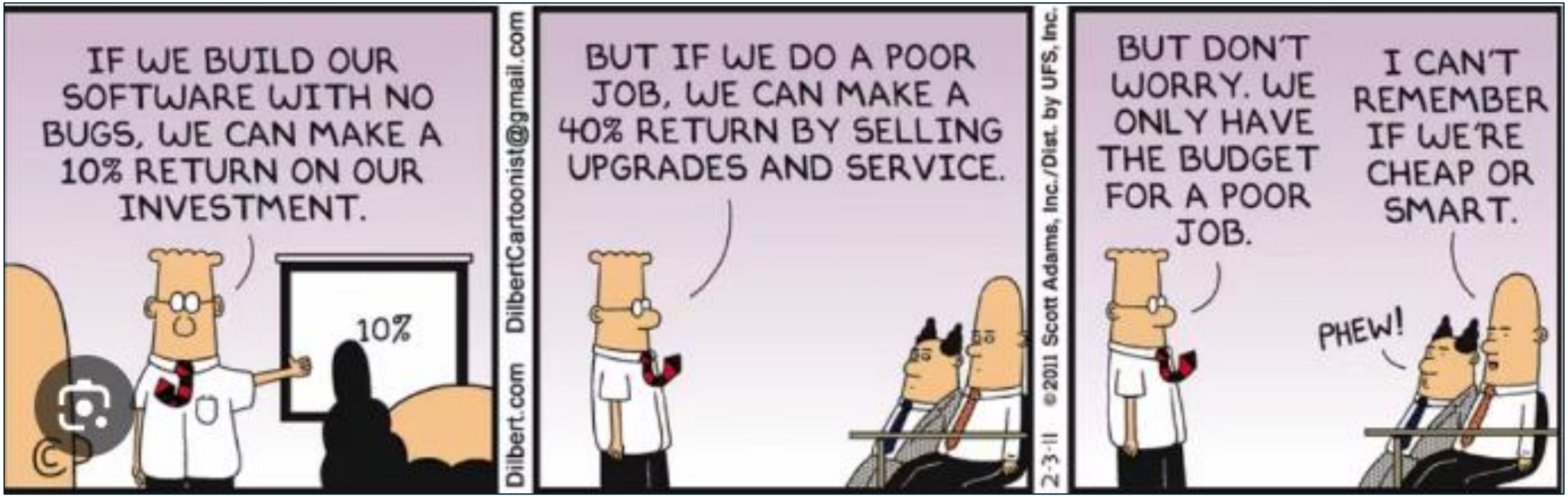
Weaknesses of OOP

- Increased Complexity
- Overengineering
- Performance Overhead
- Limited Suitability for All Problems

Not all problems are well-suited for OOP.

For tasks that are primarily data-driven or focus on algorithms, FOP might be a more natural fit.

Work Smart NOT …


IF WE BUILD OUR SOFTWARE WITH NO BUGS, WE CAN MAKE A 10% RETURN ON OUR INVESTMENT.
10%
Dilbert.com DilbertCartoonist@gmail.com
BUT IF WE DO A POOR JOB, WE CAN MAKE A 40% RETURN BY SELLING UPGRADES AND SERVICE.
2-3-11 © 2011 Scott Adams, Inc./Dist. by UFS, Inc.
BUT DON'T WORRY. WE ONLY HAVE THE BUDGET FOR A POOR JOB.
PHEW!
I CAN'T REMEMBER IF WE'RE CHEAP OR SMART.

Recap

- What are characteristics of a good design?
- Goal of a design phase?
- What are the 2 design Approaches?
- What are examples of OOP concepts?
- Where is FOP used?
- Where is OOP used?

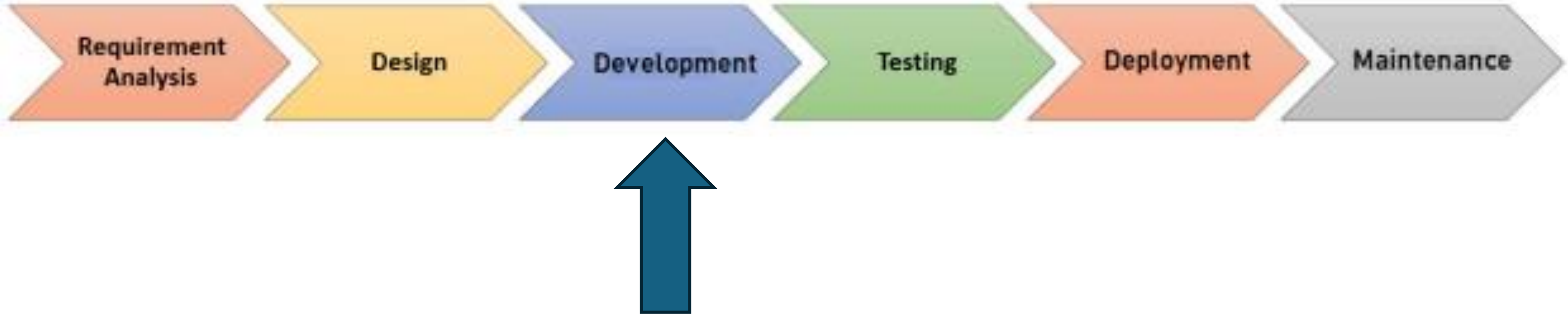
Introduction to Software Engineering

Day 4 - Session 2

Anjali Kulkarni

May 2024

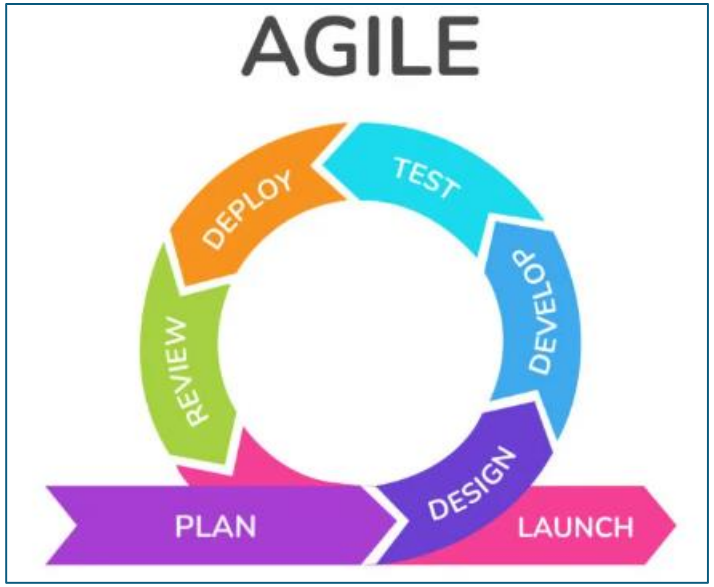
Design Phase



Agile Development Model

<u>Iterative, incremental approach to software development</u>

- Flexible
- Collaborative
- Continuous improvement



Agile Model – Core values

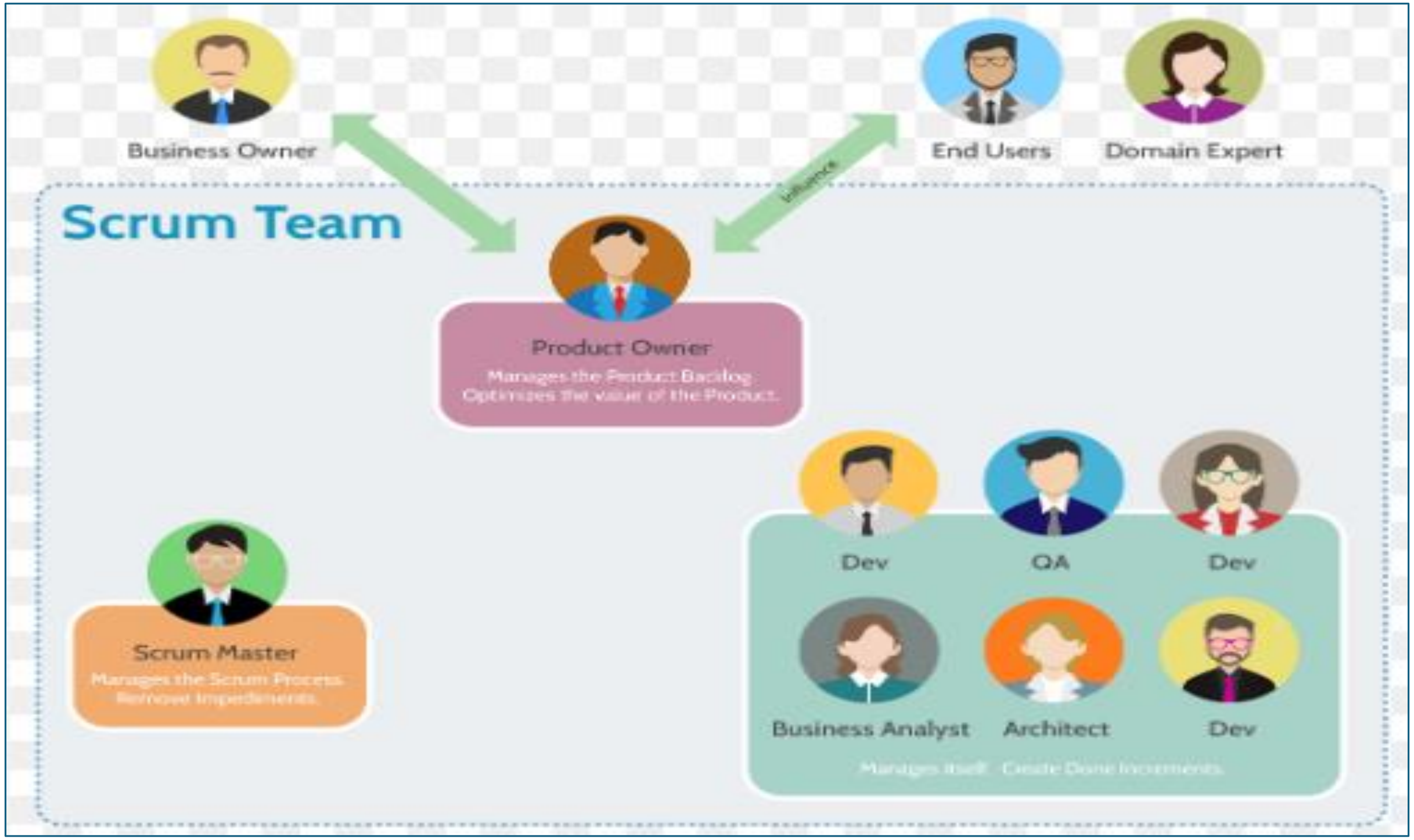
Core values form the foundation of Agile development

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 terminologies

- Projects vs Sprints
- User Stories
- Tasks and Subtasks
- Defects and Test Cases
- Scrum teams

Scrum Teams


Business Owner
End Users
Domain Expert
Influence
Scrum Team
Product Owner
Manages the Product Backlog
Optimizes the value of the Product.
Scrum Master
Manages the Scrum Process
Remove Impediments.
Dev
QA
Dev
Business Analyst
Architect
Dev
Manages itself . Create Done Increments.

Agile practices

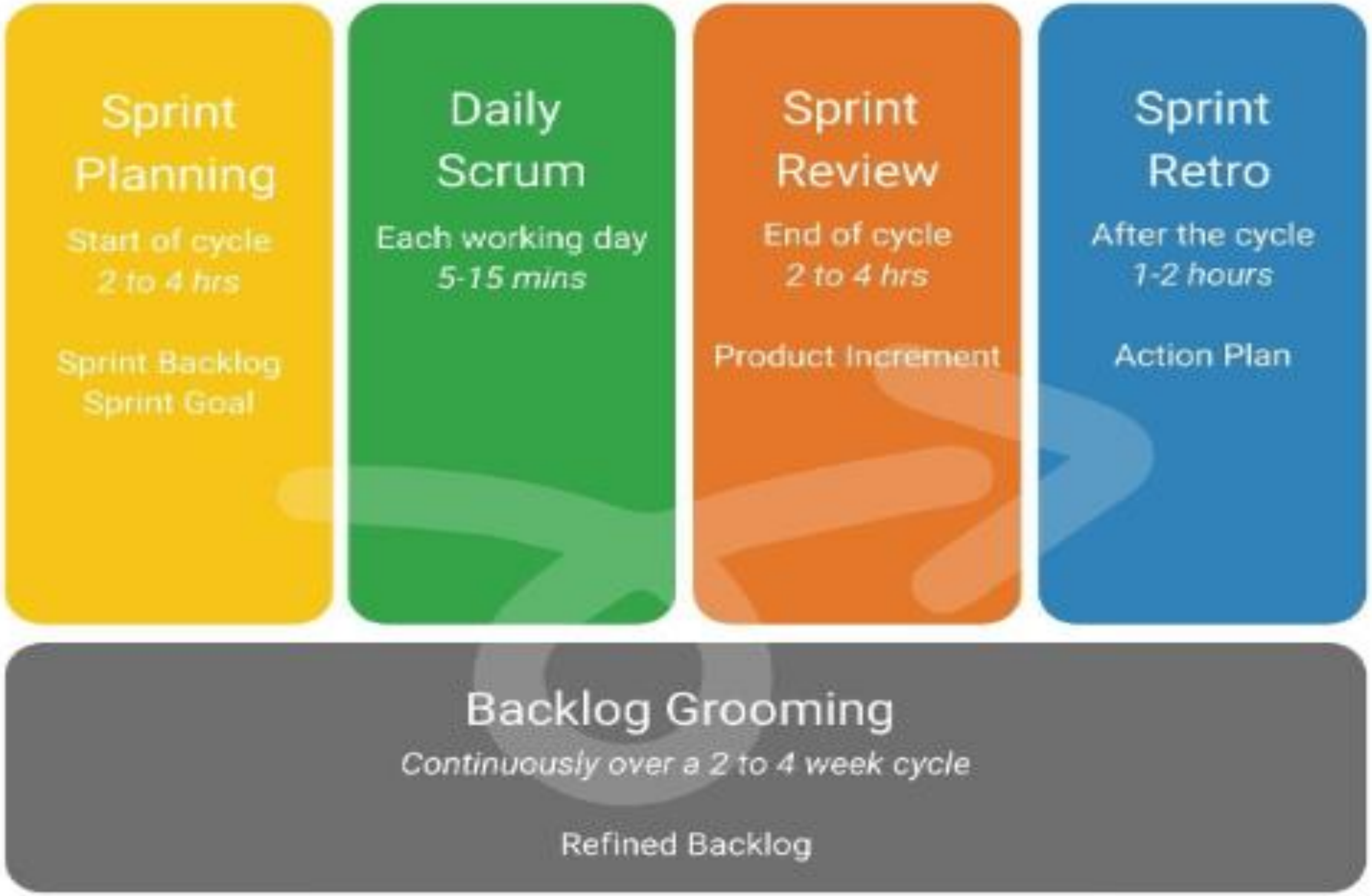
- Iterative Development
- Scrum Ceremonies
- Test-Driven Development (TDD)
- Continuous Integration (CI) and Continuous Delivery (CD)

Iterative development

- Adapt to changing requirements
- Identify and fix bugs early
- Get continuous feedback

DID I REMEMBER TO TELL YOU BEFORE YOU FINISHED THE CODING THAT THE USER'S SPECIFICATIONS CHANGED?
Dilbert.com DilbertCartoonist@gmail.com
AAAIII—YIIII—YIIII—YIIII—YIIIIII!!!!!!
5-16-11 ©2011 Scott Adams, Inc. Dist. by Universal Uclick.
SO, NO—ISH?
BAM! BAM! BAM!

Scrum ceremonies



Sprint Planning
Start of cycle
2 to 4 hrs

Sprint Backlog
Sprint Goal

Daily Scrum
Each working day
5-15 mins

Sprint Review
End of cycle
2 to 4 hrs

Product Increment

Sprint Retro
After the cycle
1-2 hours

Action Plan

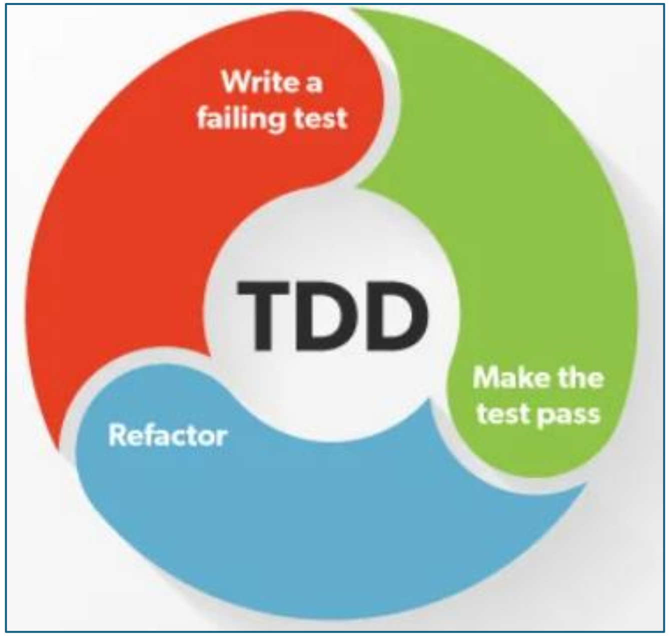
Backlog Grooming
Continuously over a 2 to 4 week cycle

Refined Backlog

Test Driven Development - TDD

Tests are written before the code that needs to be tested

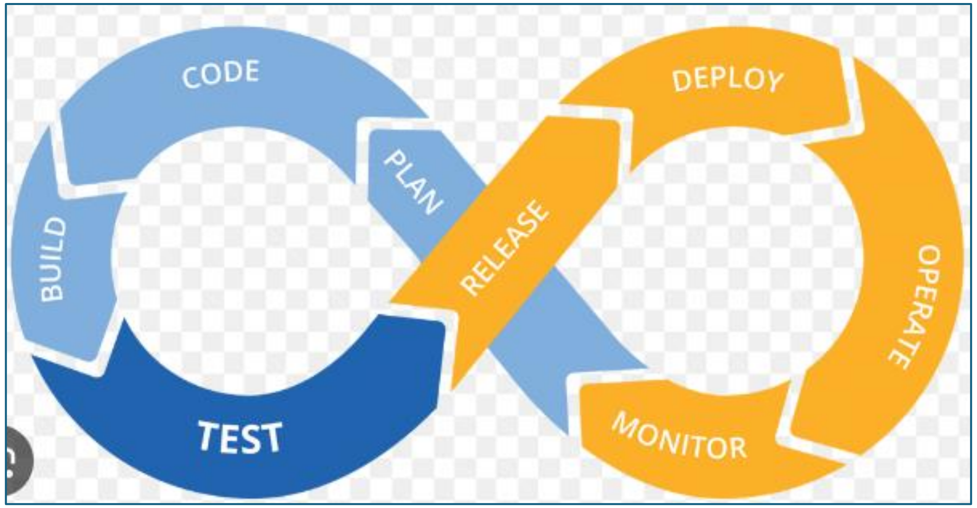
- Write a Test
- Run the Test
- Write Code
- Run the Test Again
- Refactor Code
- Repeat



CI / CD

- CI : Frequently integrate code into a shared repository

- CD : Code changes are automatically deployed to production after passing the testing phase



Benefits of Agile Development

- **Faster TTM**
  - Rapid and reliable deployment of features and bug fixes.
- **Improved Code Quality**
  - Automated testing and integration ensure high-quality code.
- **Reduced Risk**
  - Frequent deployments help identify and fix issues early.
- **Better Collaboration**
  - Encourages collaboration and integration across teams.

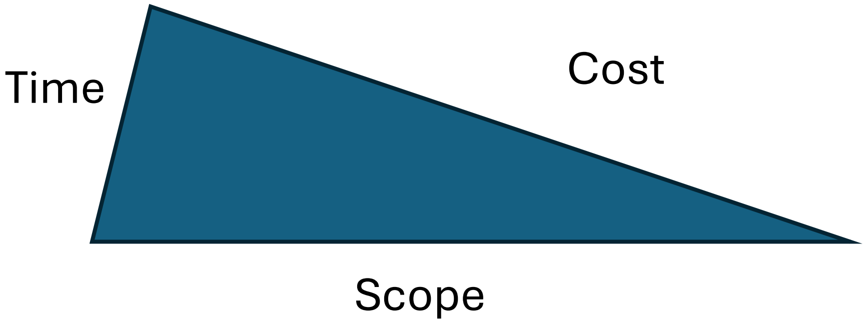
Agile Tools

Category	Tool Options	Activities	Features
Project Management	• Jira • Trello • Asana	• Manage tasks • Track progress • Ensure team collaboration	• Kanban boards • Stories and backlog mgmt • Sprint planning, bug tracking • Reporting and analytics
Version Control Systems	• Git • GitHub • GitLab	• Track code changes • Collaborate effectively • Revert to previous versions.	• Version history tracking • Branching and merging • Code collaboration
CI/CD Tools	• Jenkins • CircleCI • Travis CI • GitLab	• Automate software building, testing, and deployment process • Ensure frequent delivery	• Automated build and test execution • Building deployment pipelines • Integration with VCS and project management tools
Testing Tools	• Selenium • Cypress • Playwright	• Automated and manual testing • Cross browser, platform testing	• Automated testcase creation • UI and API testing • Error reporting

Quality Triangle

Balancing Scope, Time, and Cost

- Relationship between three key constraints:
  - **Scope:** The features and functionalities of the software being developed.
  - **Time:** The amount of time allocated to complete the project.
  - **Cost:** The financial resources available for the project.



Fun facts!!

Myth	Reality
While running behind schedule, add more people.	Adding more people at a later stage in adhoc manner reduces productivity
Outsourcing will allow me to relax while the org builds the product	Organisations who cannot control and manage internal projects, will struggle managing outsourced products
Once we write the program and get it to work, our jobs are done	Sooner you begin writing code, the longer it‘ll take you to get done
Process will slow down if voluminous documentation is created	Documentation helps in creating quality software and faster delivery
QA can be done if we have time	Without QA product run into unforeseen issues at runtime, resulting into sky rocketing costs

Recap

- What are benefits of agile development?
- What is TDD?
- List Sprint Ceremonies
- What is ideal size of a scrum team?
- Who all are a part of Scrum team?
- Agile tool for Project Mgmt, CI/CD, Version control, CI/CD?
- What are the factors affecting / controlling quality?

Thank You!!