BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Review CS#1

Introduction

- Software development lifecycle
- The Waterfall approach : Advantages & Disadvantages
- Agile Methodology
- Operational Methodologies: ITIL
- Problems of Delivering Software
- Principles of Software Delivery
- About DevOps
- Need for DevOps
- DevOps Practices in Organization
- The Continuous DevOps Life Cycle Process
- Evolution of DevOps
- Case studies

Agenda

DevOps : Process Dimension

- DevOps Misconception
- DevOps Anti-Patterns
- Three Dimensions of DevOps
  - Process
  - People
  - Tools
- DevOps : Process Dimension
  - DevOps and Agile
  - Team Structure
  - Agile Methodology
    - TDD
    - BDD
    - FDD

DevOps Misconception

DevOps Myths & Misconception

- DevOps is a Team
- CI/CD is all about DevOps
- Non-Compliant to Industry Standards

An additional team is likely to cause more communication issues

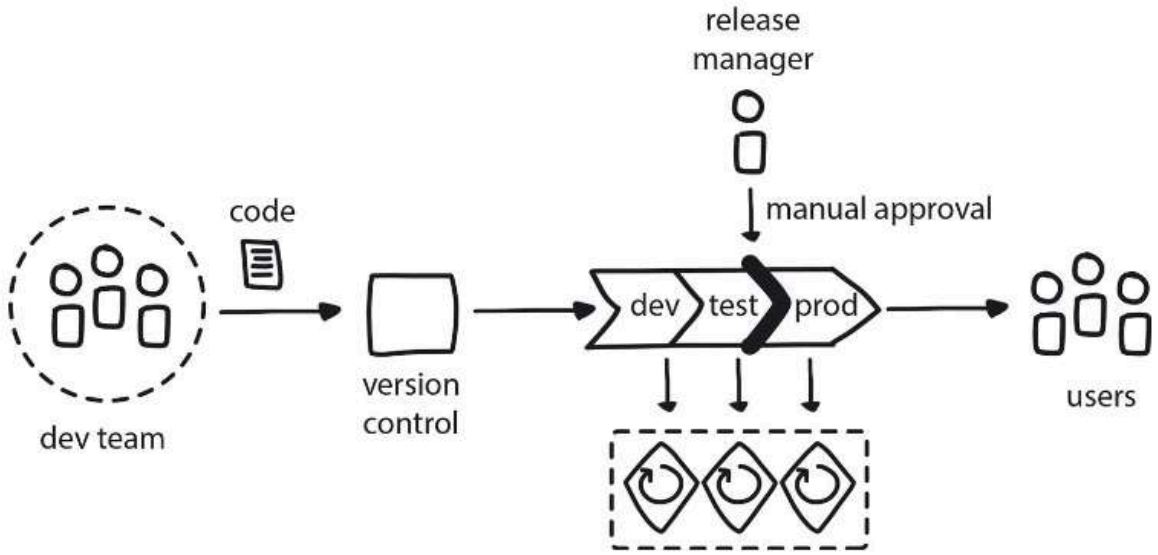
Scalability
Consistency
Reliability

Regulations implement control in order to be compliant
Controls reduce the risk that may affect the confidentiality, integrity, and availability of information

DevOps Misconception

DevOps Myths & Misconception

- Automation eliminates ALL bottlenecks
- “Universal" Continuous Delivery pipeline
- All about Tools
- Release as fast as Amazon / Facebook



Improve the partnership b/w dev test & Ops

It increases feedback loops
However Hand off processes can often add to process wait states

Empowered smaller self-sufficient teams and Continuous Delivery Engineer

Processes to fit organizational and busin needs — not vice-versa

Mandating tools developers can use.
Prioritizing tools over people

DevOps Anti-Patterns

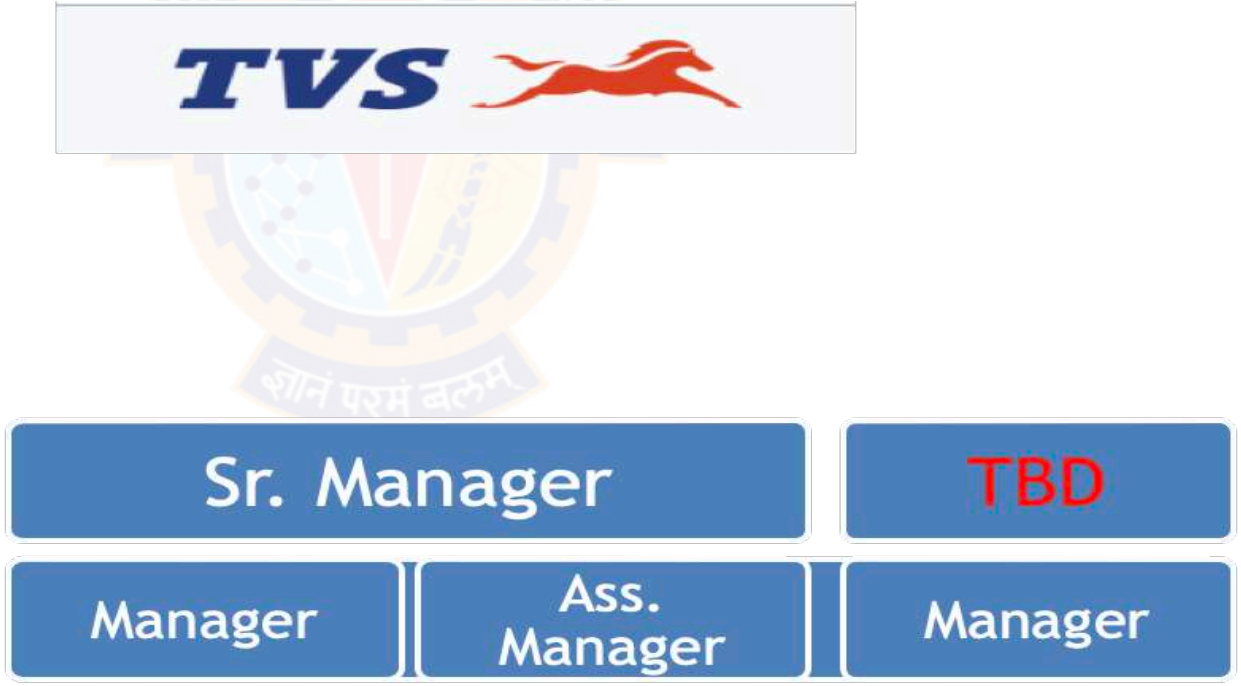
Anti-Patterns

- Blame Culture
  - A blame culture is one that tends toward blaming and punishing people when mistakes are made, either at an individual or an organizational level
- Silos
  - A departmental or organizational silo describes the mentality of teams that do not share their knowledge with other teams in the same company
- Root Cause Analysis
  - Root cause analysis (RCA) is a method to identify contributing and “root” causes of events or near-misses/close calls and the appropriate actions to prevent recurrence
- Human Errors
  - Human error, the idea that a human being made a mistake that directly caused a failure, is often cited as the root cause in a root cause analysis

Lessons of History

Example for Silos

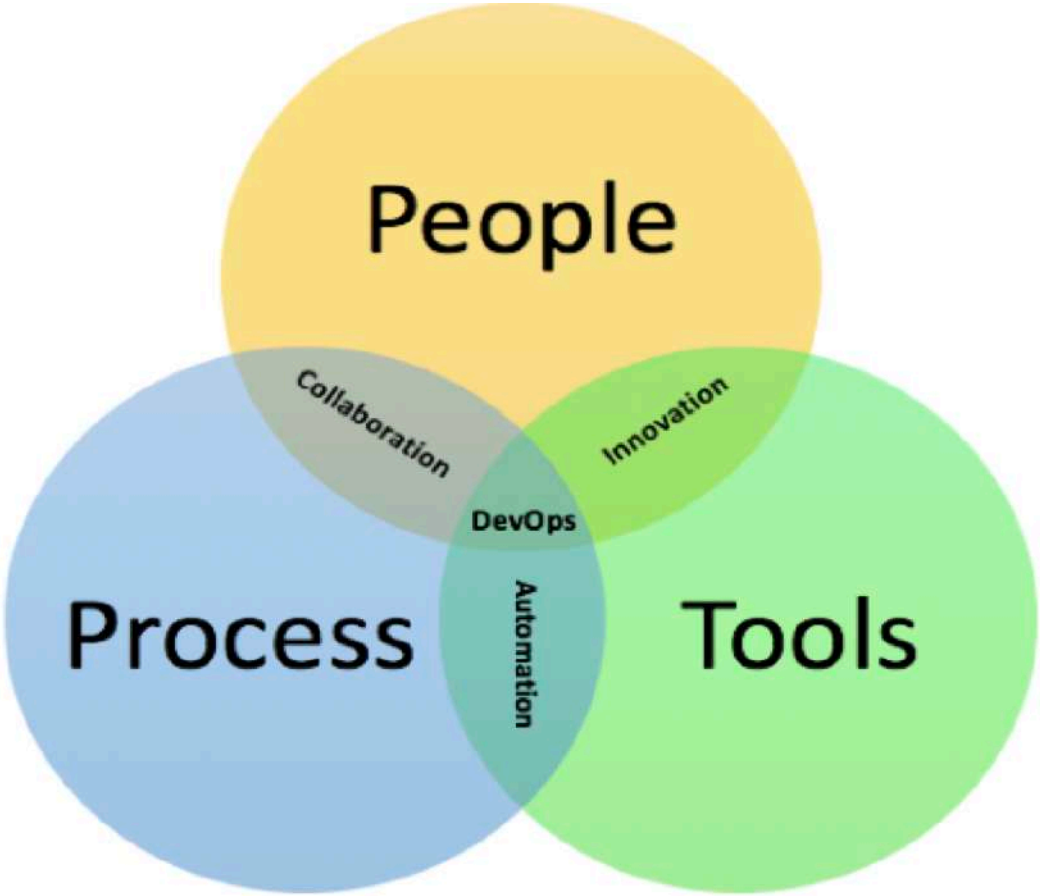
- TVS India
  - T. V. Sundaram Iyengar
  - Founded in 1978



DevOps Dimensions

Three dimensions of DevOps

- People
- Process
- Tools / Technology



DevOps - Process

DevOps and Agile

- We need processes and policies for doing things in a proper way and standardized across the projects so the sequence of operations, constraints, rules and so on are well defined to measure success
- One of the characterizations of DevOps emphasizes the relationship of DevOps practices to agile practices
- We will focus on what is added by DevOps
- We interpret transition as deployment

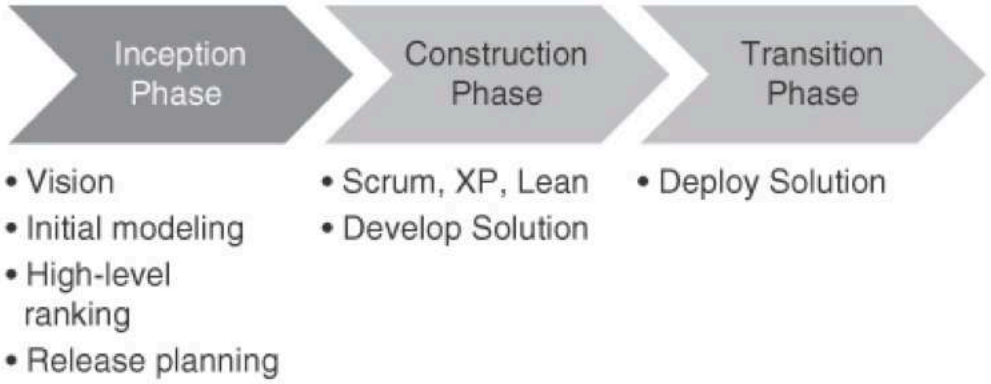


FIGURE 1.2 Disciplined Agile Delivery phases for each release. (Adapted from *Disciplined Agile Delivery: A Practitioner's Guide* by Ambler and Lines) [Notation: Porter's Value Chain]

DevOps and Agile

DevOps practices impact all three phases

- Inception phase : During the inception phase, release planning and initial requirements specification are done
  - Considerations of Ops will add some requirements for the developers
  - Release planning includes feature prioritization but it also includes coordination with operations personnel
- Construction phase: During the construction phase, key elements of the DevOps practices are the management of the code branches,
  - the use of continuous integration
  - continuous deployment
  - incorporation of test cases for automated testing
- Transition phase: In the transition phase, the solution is deployed and the development team is responsible for the deployment, monitoring the process of the deployment, deciding whether to roll back and when, and monitoring the execution after deployment

DevOps – Process

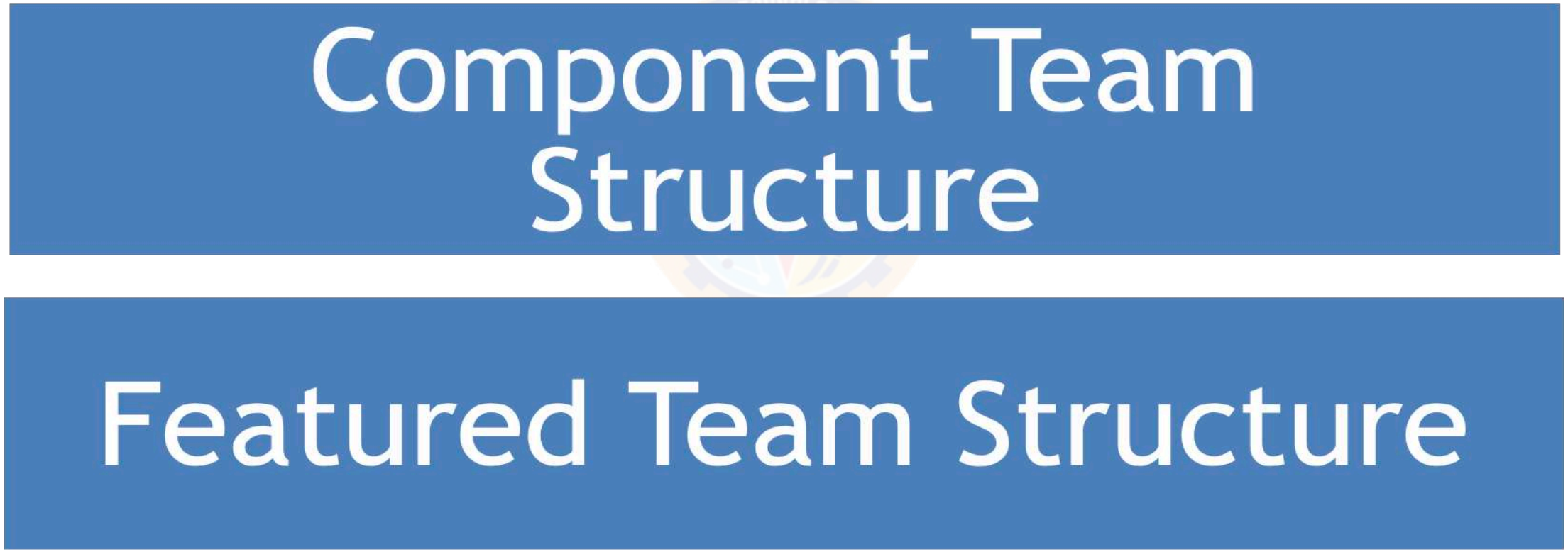
Process

Extreme Programming	SCRUM	Feature Driven Development
Kanban	Hybrid Agile	Distributed Agile

DevOps – Process

Team Structure

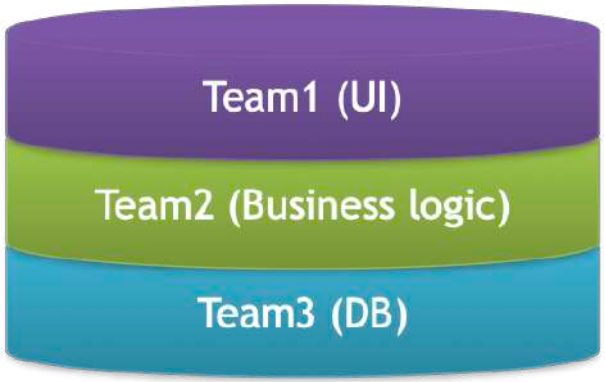
- Choosing appropriate team structure (Resource planning)



DevOps – Process

Component teams

- Made up of experts that specialize in a specific domain
- Architecture diagram can be represented as layer of the system
- Advantages:
  - work is always done by people specifically qualified to do the work
  - work is done efficiently with a low defect density
- Disadvantages
  - Working as Component Teams increases the amount of time it takes to deliver
  - working in Component Teams will have a negative effect on productivity that cannot be over emphasized

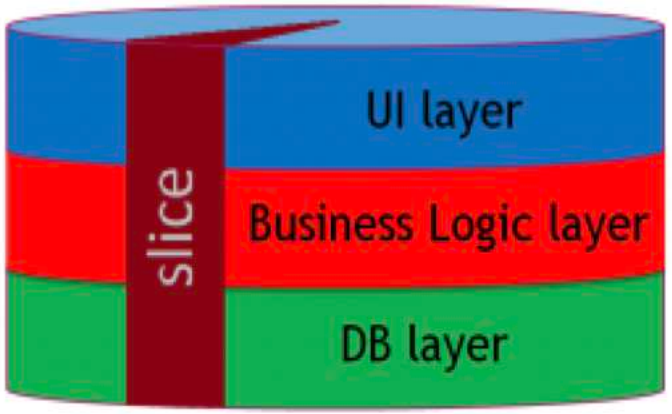


Architecture diagram

DevOps – Process

Feature teams

- Feature Teams contain multi-disciplined individuals that have the ability and freedom to work in any area of the system
- Feature Team usually has at least one member with specialist knowledge for each layer of the system
- This allows Feature Teams to work on what are known as ‘vertical slices’ of the architecture



DevOps – Process

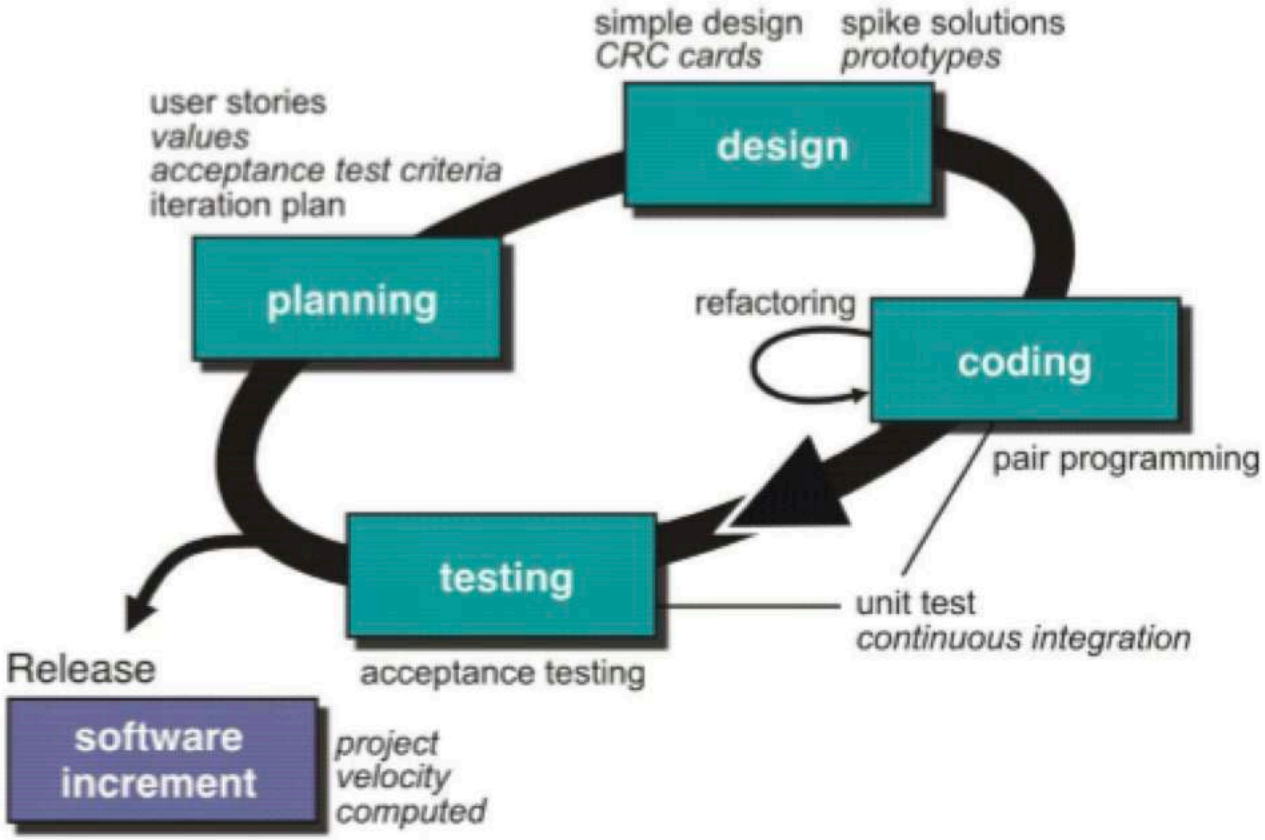
Feature teams

- The characteristics of a feature team are listed below:
  - long-lived—the team stays together so that they can ‘jell’ for higher performance; they take on new features over time
  - co-located
  - work on a complete customer-centric feature, across all components and disciplines (analysis, programming, testing, …)
  - composed of generalizing specialists
  - in Scrum, typically 7 ± 2 people
- Disadvantages
  - untrained or inexperienced employee to deliver a poorly designed piece of work into a live environment
  - Where do you get the personnel (‘generalizing specialists’)?

DevOps: Process

TDD

- Prerequisite :: XP



TDD

Understanding

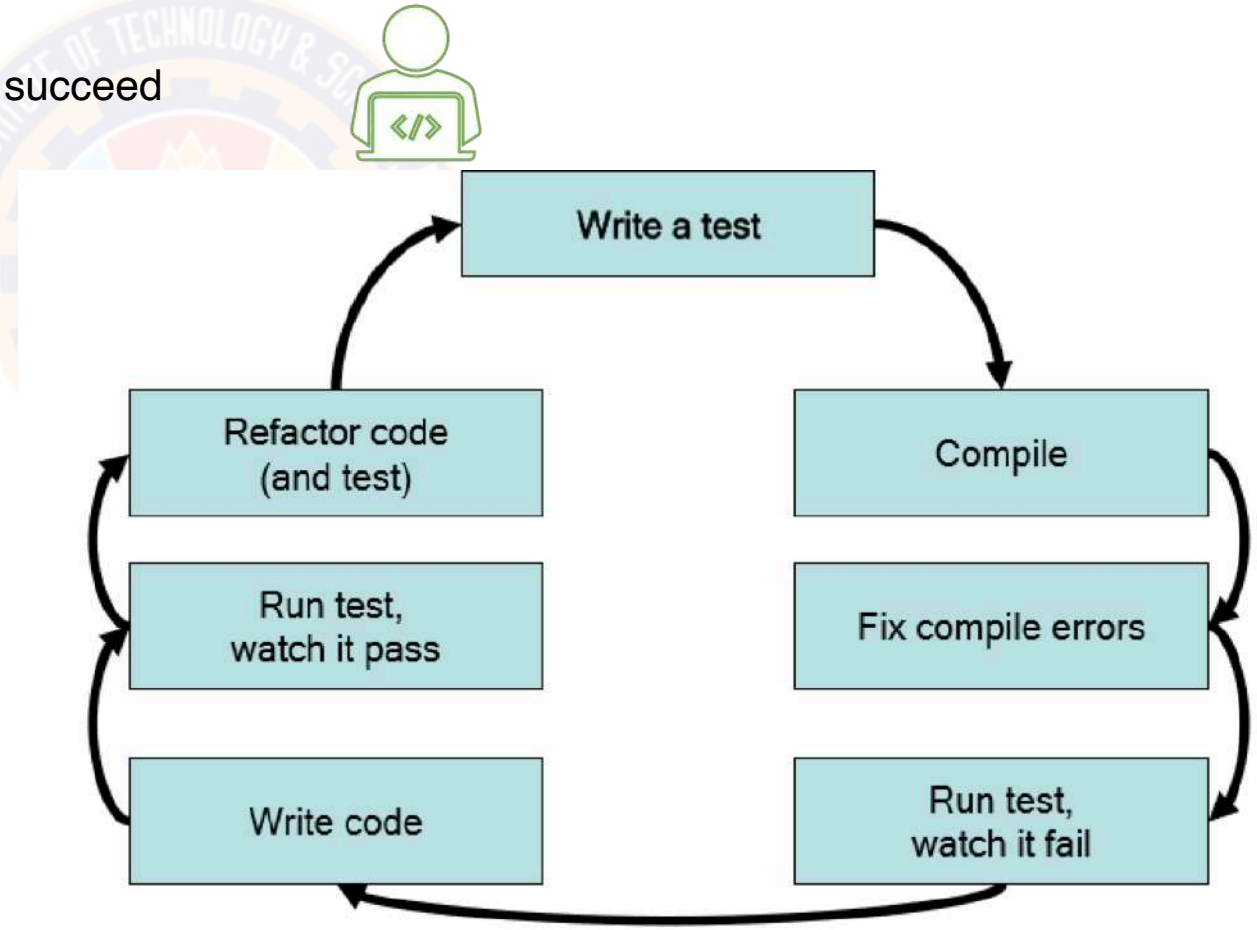
- Many Names
  - Test driven development
  - Test drive design
  - Emergent design
  - Test first development

- TDD Quotes
  - Kent Beck said “Test-first code tends to be more cohesive and less coupled than code in which testing isn’t a part of the intimate coding cycle”
  - “If you can’t write a test for what you are about to code, then you shouldn’t even be thinking about coding”

TDD

TDD Three steps

- RED
  - Write a new TEST which fails
- GREEN
  - Write simplest possible code to make it succeed
- REFACTOR
  - Refactor the code (including test code)



TDD

TDD Overview

- In Extreme Programming Explored (The Green Book), Bill Wake describes the test / code cycle:

1. Write a single test
2. Compile it. It shouldn't compile because you've not written the implementation code
3. Implement just enough code to get the test to compile
4. Run the test and see it fail
5. Implement just enough code to get the test to pass
6. Run the test and see it pass
7. Refactor for clarity
8. Repeat

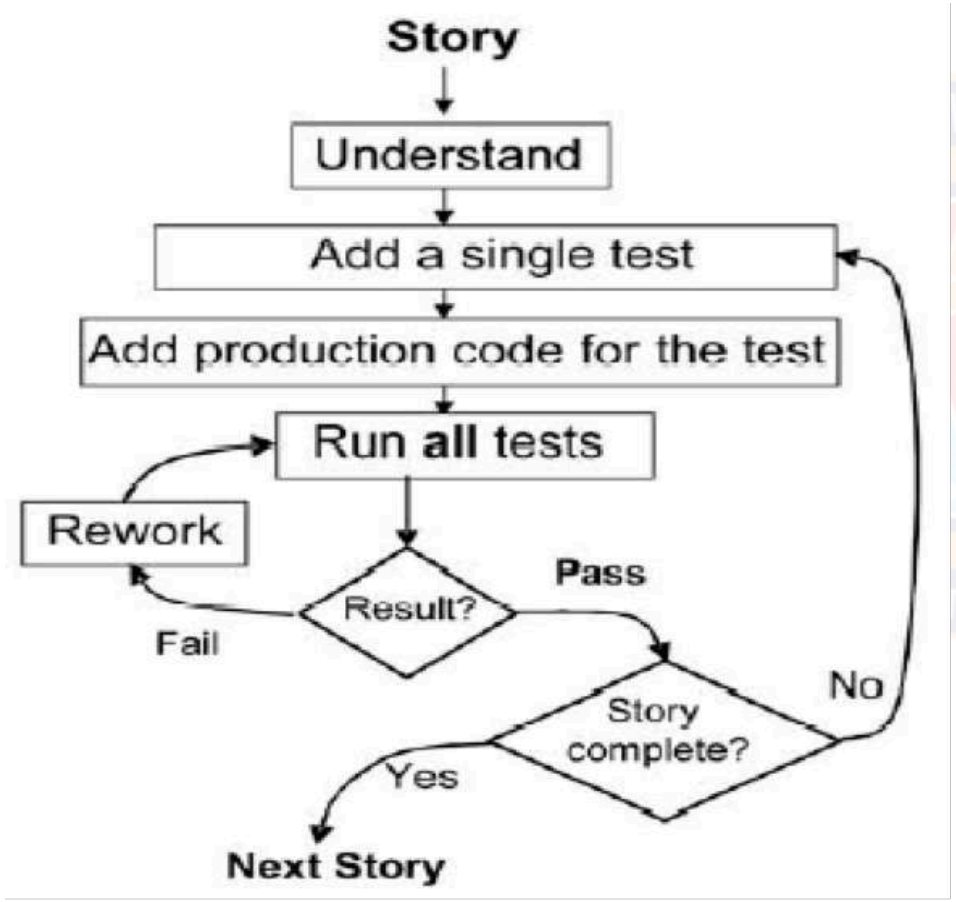
TDD

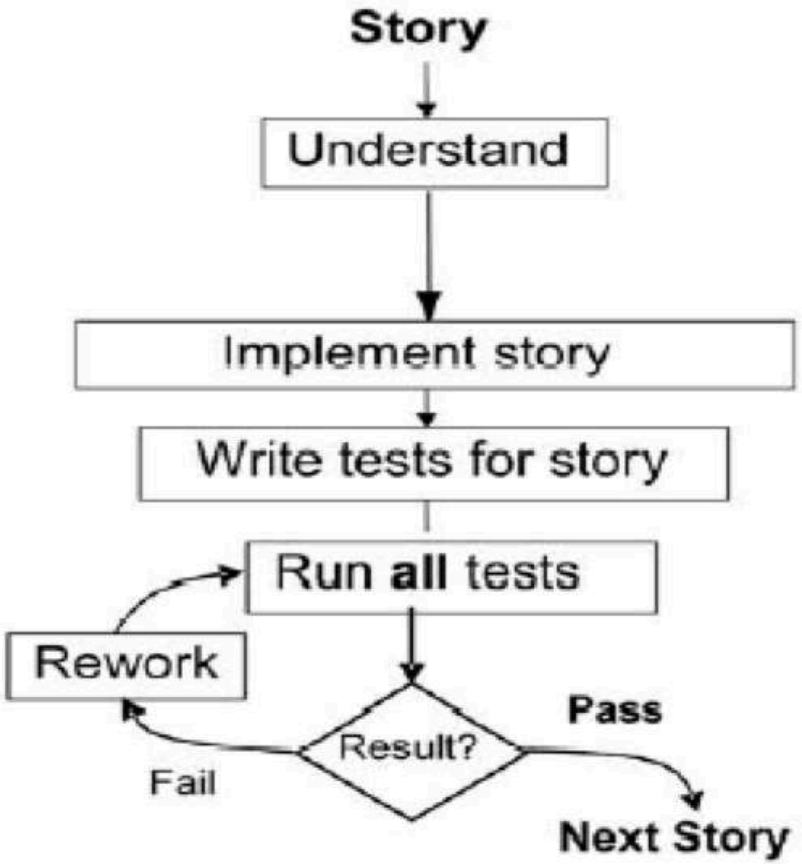
Why TDD

- TDD can lead to more modularized, flexible, and extensible code
- Clean code
- Better code documentation
- More productive
- Good design

TDD

Test First vs. Test Last





TDD

Lets Have TDD Overview

https://www.youtube.com/watch?v=uGaNkTahrIw&t=132s

DevOps: Process

FDD

- What is a Feature?
  - Definition: small function expressed in client-valued terms
  - FDD's form of a customer requirement

FDD

FDD Primary Roles

- Project Manager
- Chief Architect
- Development Manager
- Domain Experts
- Class Owners
  - This concept differs FDD over XP
  - Benefits
    - Someone responsible for integrity of each class
    - Each class will have an expert available
    - Class owners can make changes much quicker, if needed anytime
    - Easily lends to notion of code ownership
    - Assists in FDD scaling to larger teams, as we have one person available for complete ownership of feature.
- Chief Programmers

FDD

FDD Primary Roles

- Release Manager
- Language Guru
- Build Engineer
- Toolsmith
- System Administrator
- Tester
- Deployers
- Technical Writer

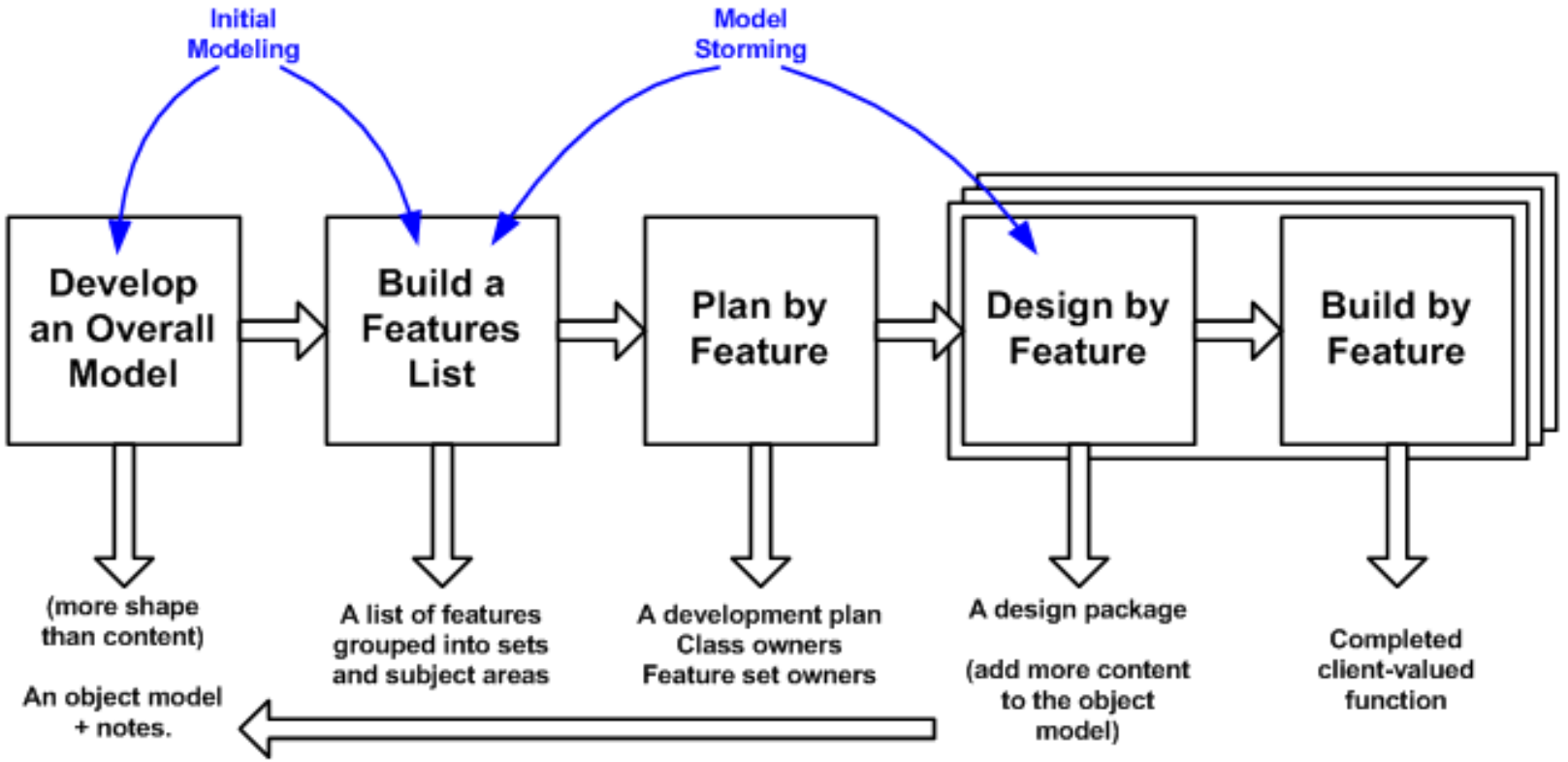
FDD

Feature Driven Development Process

- Process #1: Develop an Overall Model
- Process #2: Build a Features List
- Process #3: Plan By Feature
  - Constructing the initial schedule, Forming level of individual features, Prioritizing by business value , As we work on above factors we do consider dependencies, difficulty, and risks.
  - These factors will help us on Assigning responsibilities to team members, Determining Class Owners , Assigning feature sets to chief programmers
- Process #4: Design By Feature
  - Goal: not to design the system in its entirety but instead is to do just enough initial design that you are able to build on
  - This is more about Form Feature Teams: Where team members collaborate on the full low level analysis and design.
- Process #5: Build By Feature
  - Goal: Deliver real, completed, client-valued function as often as possible

FDD

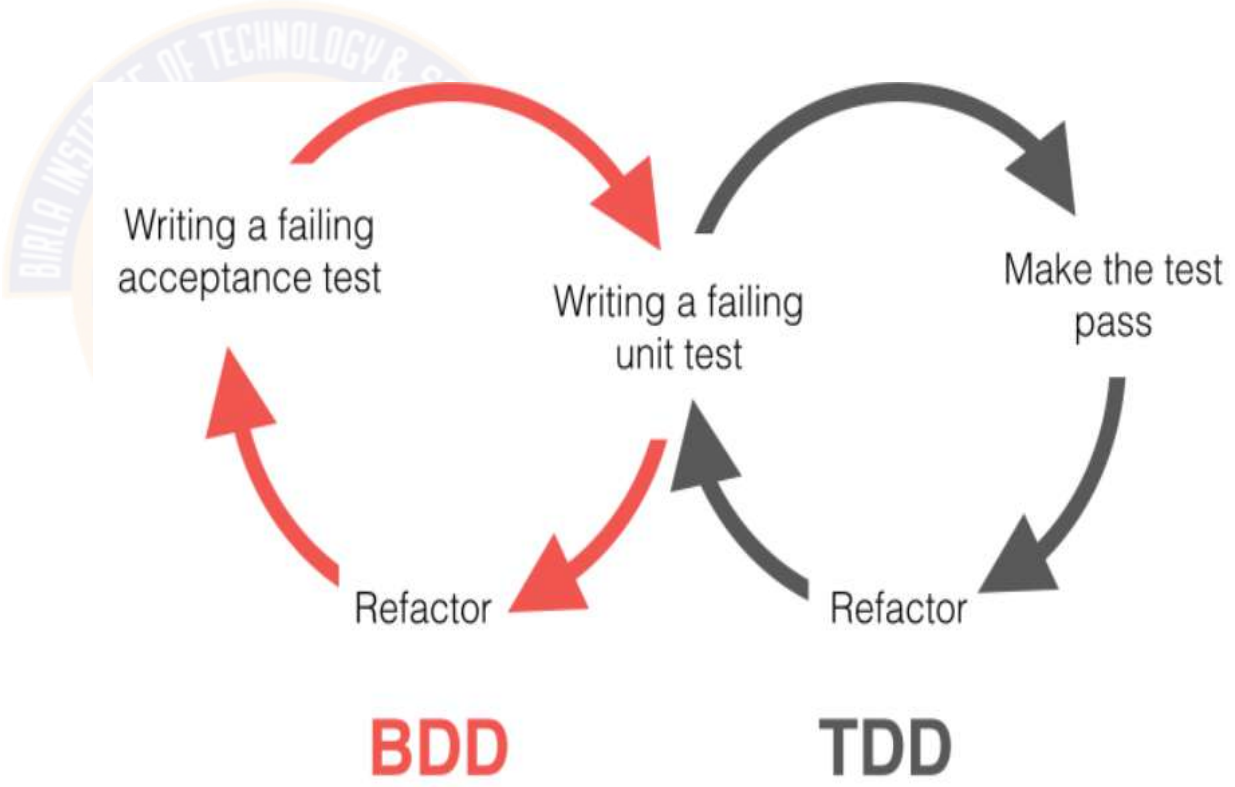
Feature Driven Development Process



DevOps: Process

BDD

- What is BDD?:
  - General Technique of TDD
  - Follows same principle as TDD
  - Shared tools
  - Shared Process



https://www.youtube.com/watch?v=4QFYTQy47yA

BDD

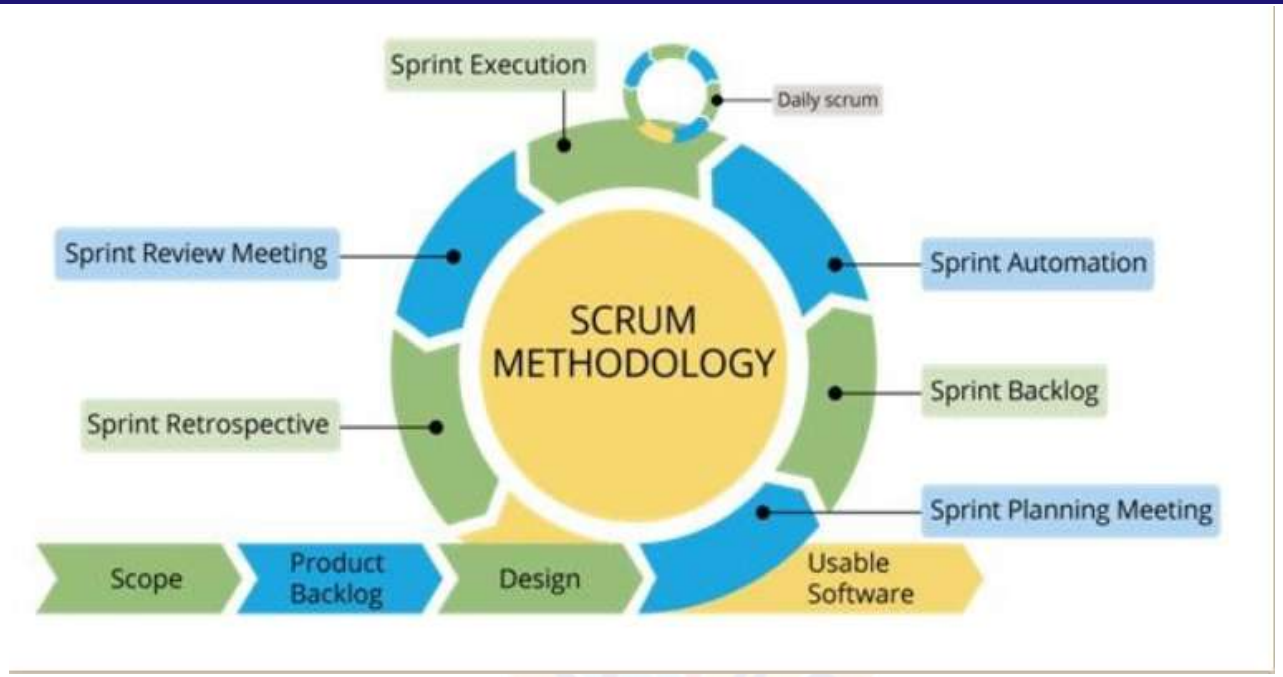
BDD Basic structure : Example

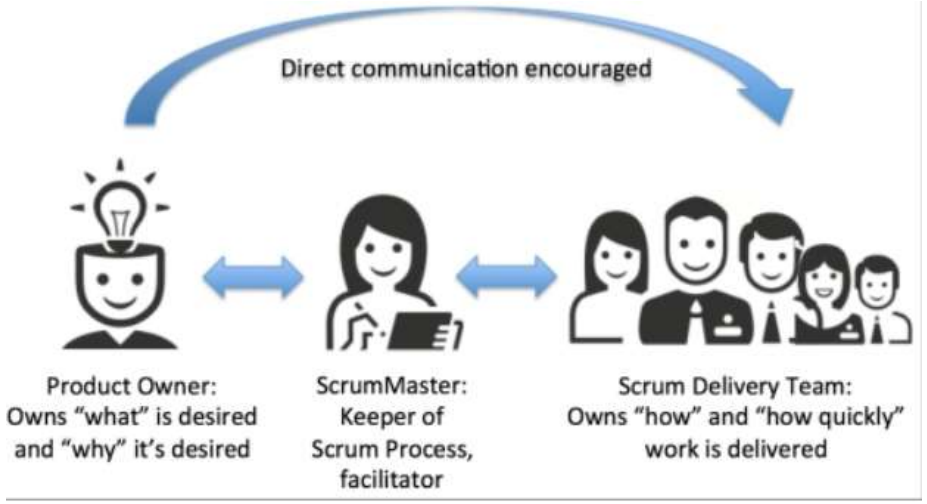
- User story:
  - As someone interested in using the Mobile app, I want to sign up in from the app so that I can enjoy my membership.

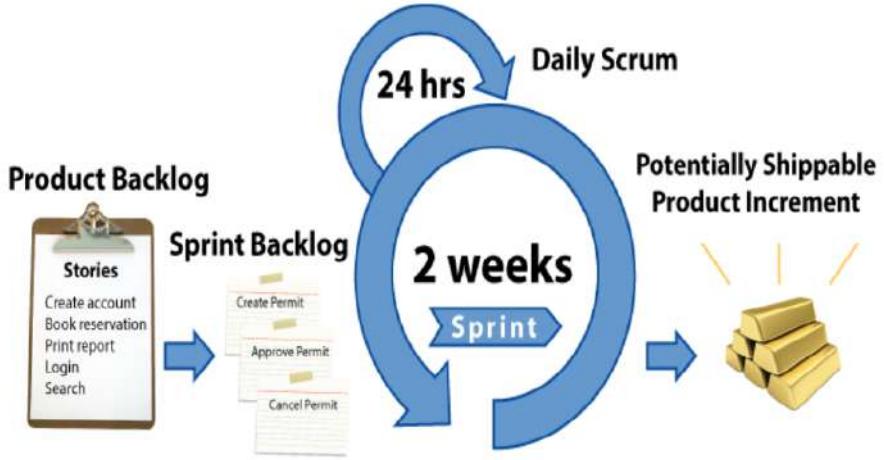
- Mobile App Signup:
- Scenario 1:
  - Given that I am on the app's "Create new account" screen
  - Then I should see a "Sign up using Facebook" button
  - And I should see a "Sign up using Twitter" button
  - And I should see a "Sign up with email" form field
  - Then I should see a new screen that asks permission to use my Facebook account data to create my new Mobile app account

DevOps: Process

SCRUM







SCRUM

SCRUM Overview

- You can refer below YouTube video for understanding the structure of organization to be agile.
- This Video is of First Bank in world to adopt Agile Scrum
- https://www.youtube.com/watch?v=uXg6hG6FrG0

Q&A

Thank You!

In our next session: