BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Text Books

T1 & T2

- DevOps: A Software Architect's Perspective (SEI Series in Software Engineering) by Len Bass, Ingo Weber, Liming Zhu , Publisher: Addison Wesley (18 May 2015)
- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation by Jez Humble, David Farley. Publisher: Addison Wesley, 2011

Reference Books

R1 & R2

- Effective DevOps: Building A Culture of Collaboration, Affinity, and Tooling at Scale by Jennifer Davis , Ryn Daniels. Publisher: O'Reilly Media, June 2016
- The DevOPS Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations by Gene Kim, Patrick Debois, John Willis, Jez Humble, John Allspaw. Publisher: IT Revolution Press (October 6, 2016)

Agenda

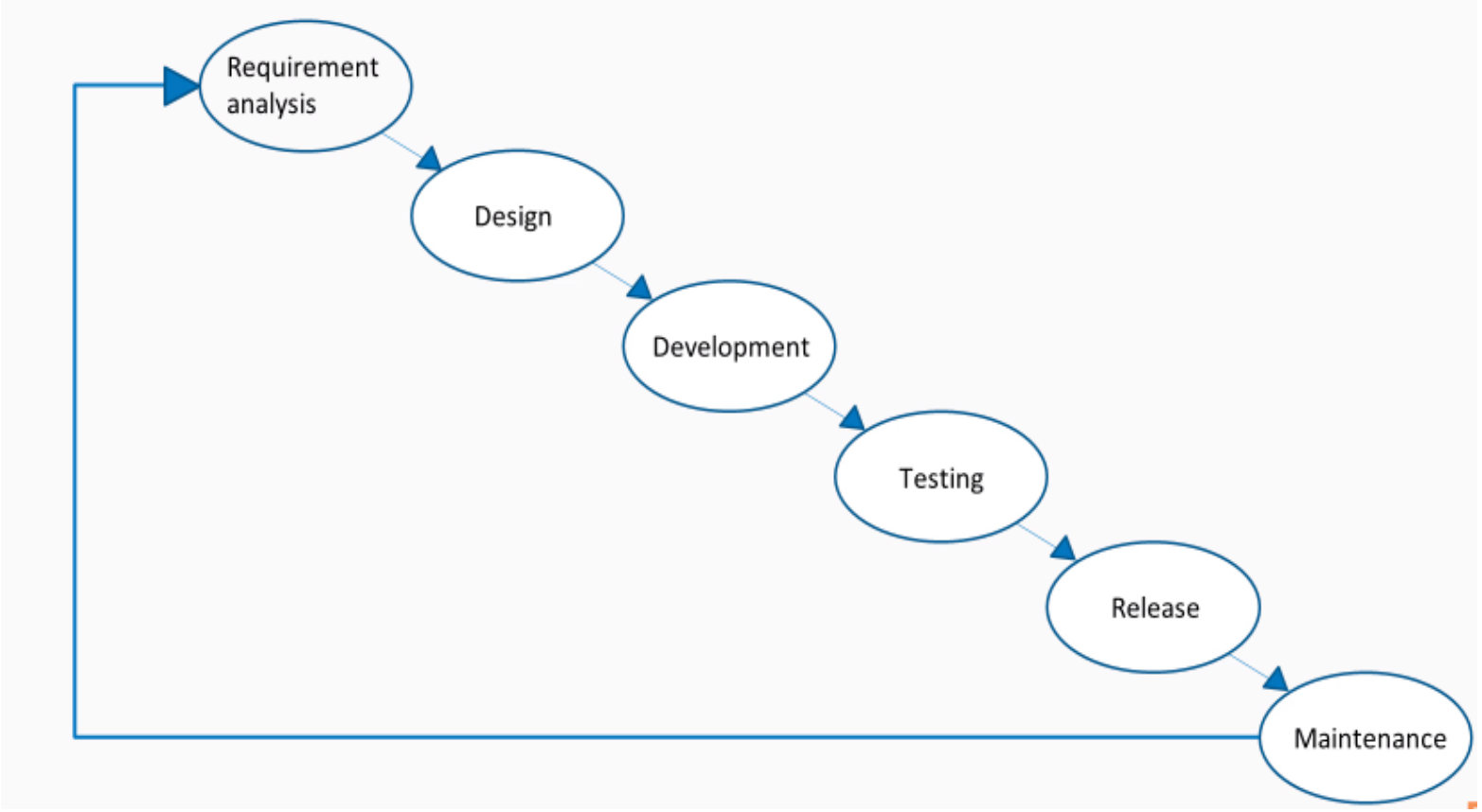
Introduction

- 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

Software Development Life Cycle

SDLC Phases

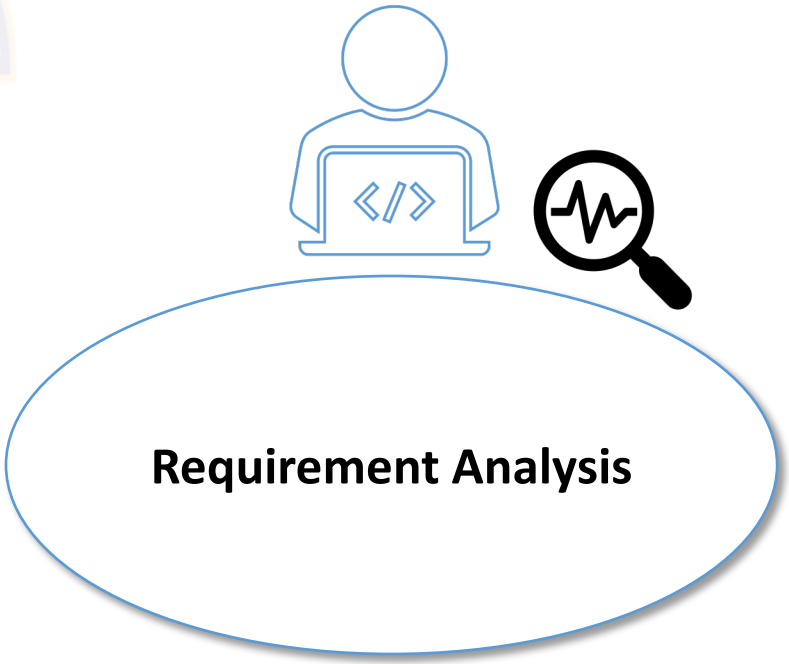
Diagram of our day-to-day activity as software engineers



Software Development Life Cycle

Requirement Analysis

- Encounter majority of problems
- Find common language between people outside of IT and people in IT
- Leads to different problems around terminology
- Business flow being capture incorrectly
- Iterative approach



Software Development Life Cycle

Requirement Analysis Contd..

Customer requirement

1. Have one trunk
2. Have four legs
3. Should carry load both passenger & cargo
4. Black in color
5. Should be herbivorous

Our Solution

1. Have one trunk ☑
2. Have four legs ☑
3. Should carry load both passenger & cargo ☑
4. Black in color ☑
5. Should be herbivorous ☑

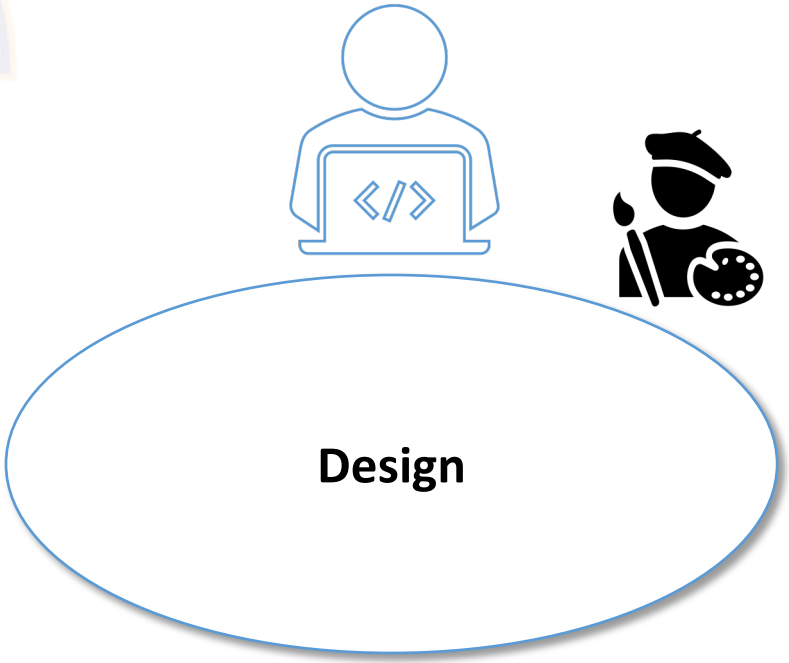
Our Value add:

Also gives milk ☺

Software Development Life Cycle

Design

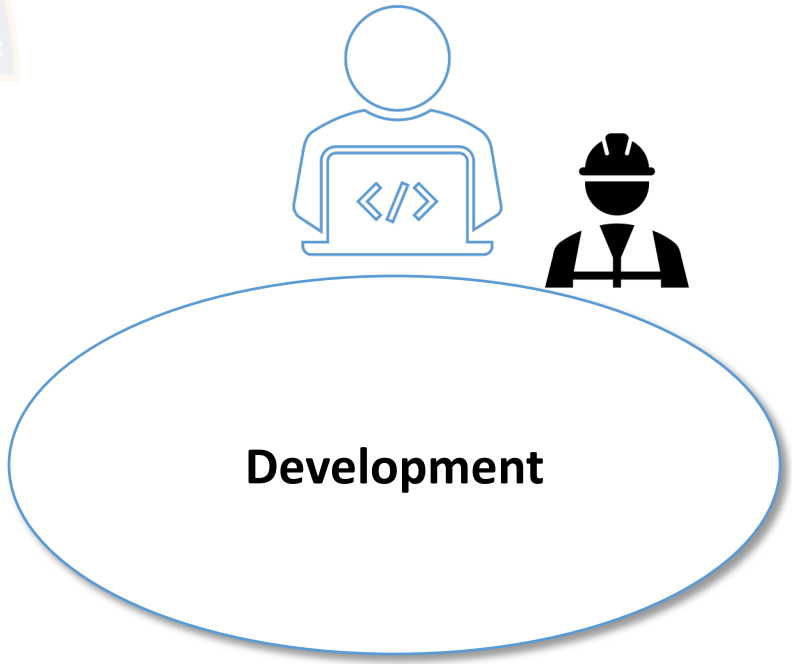
- Design our flows in language that IT crowd can understand straight away
- Overlaps with requirement analysis
- Desirable as diagrams are perfect middle language that we are looking for
- Minimal Value Product



Software Development Life Cycle

Development

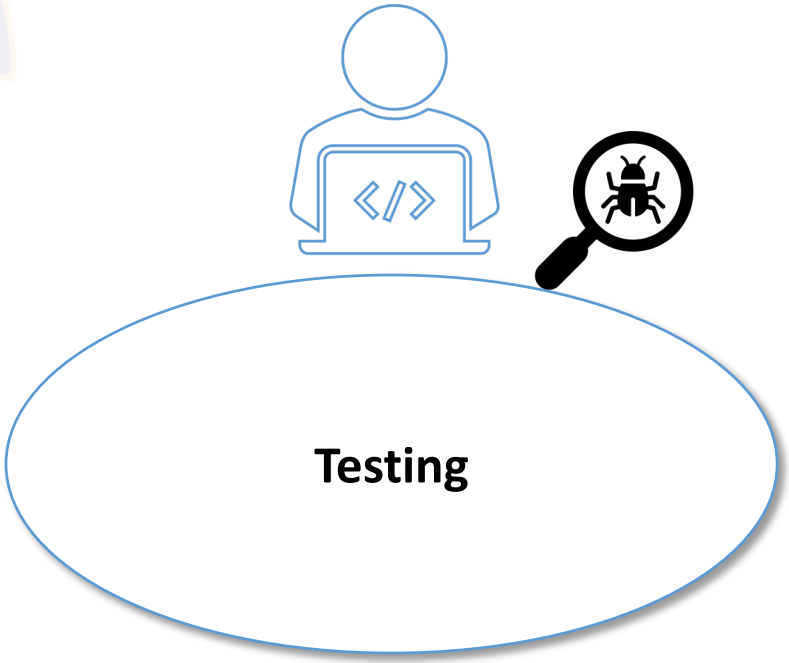
- Software is built
- Builds technical artifacts that work and according to potentially flawed specification
- Our software is going to be imperfect
- Deliver early and deliver often is mantra followed to minimize impact of wrong specification
- Stakeholders can test product before problem is too big to be tackled
- Involving stakeholders early enough is good strategy
- No matter what we do, our software has to be modular so we can plug and play modules in order to accommodate new requirements



Software Development Life Cycle

Testing

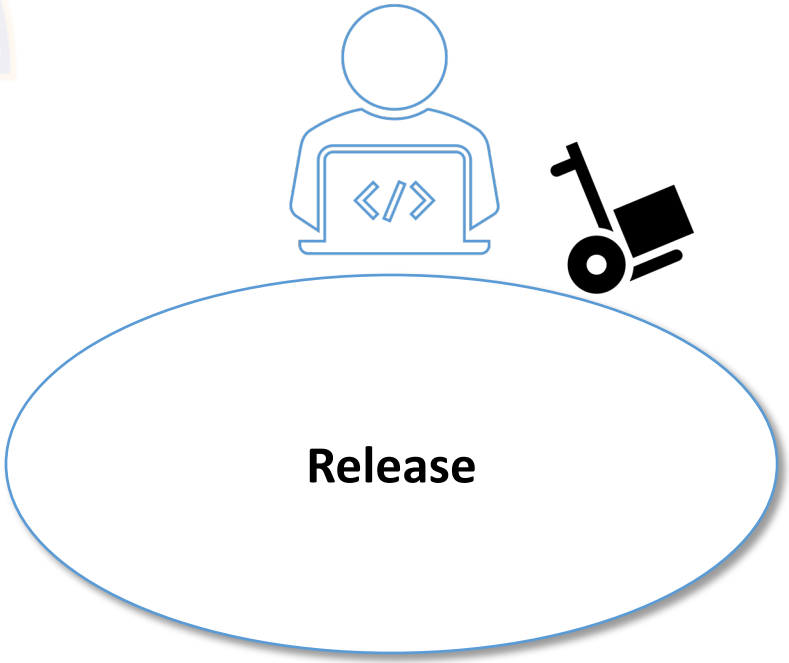
- Continuous Integration server will run testing and inform us about potential problems in application
- Depending on complexity of software, testing can be very extensive
- Continuous integration server focuses on running integration and acceptance



Software Development Life Cycle

Release

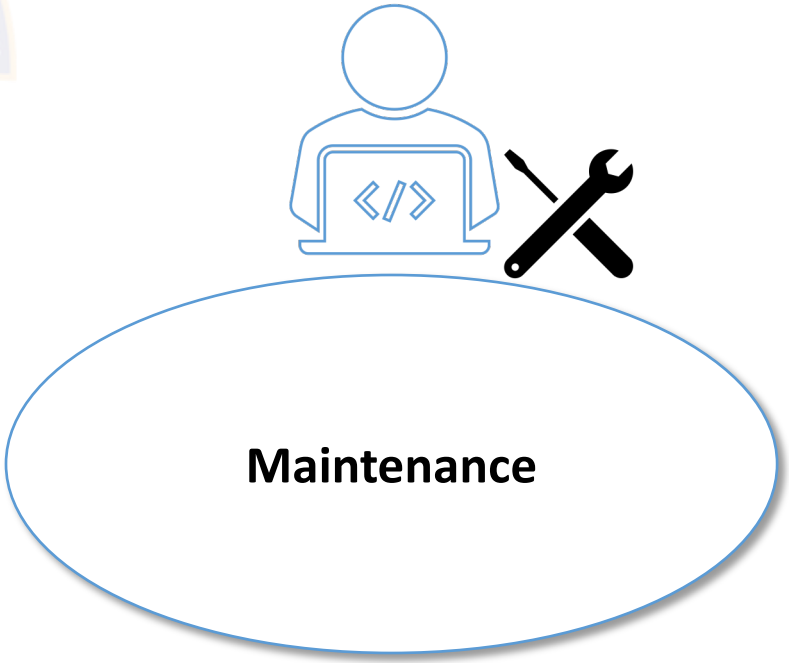
- Deliver software to what we call production
- There will be bugs and reason why we planned our software to be able to fix problems quickly
- Create something called as Continuous delivery pipelines
- Enables developers to execute build-test-deploy cycle very quickly
- Deploy = Release



Software Development Life Cycle

Maintenance

- There are two types of maintenance evolutive and corrective
- Evolutive maintenance – evolve software by adding new functionalities or improving business flows to suit business needs
- Corrective maintenance – One where we fix bugs and misconceptions
- Minimize latter but we can not totally avoid



Case Study

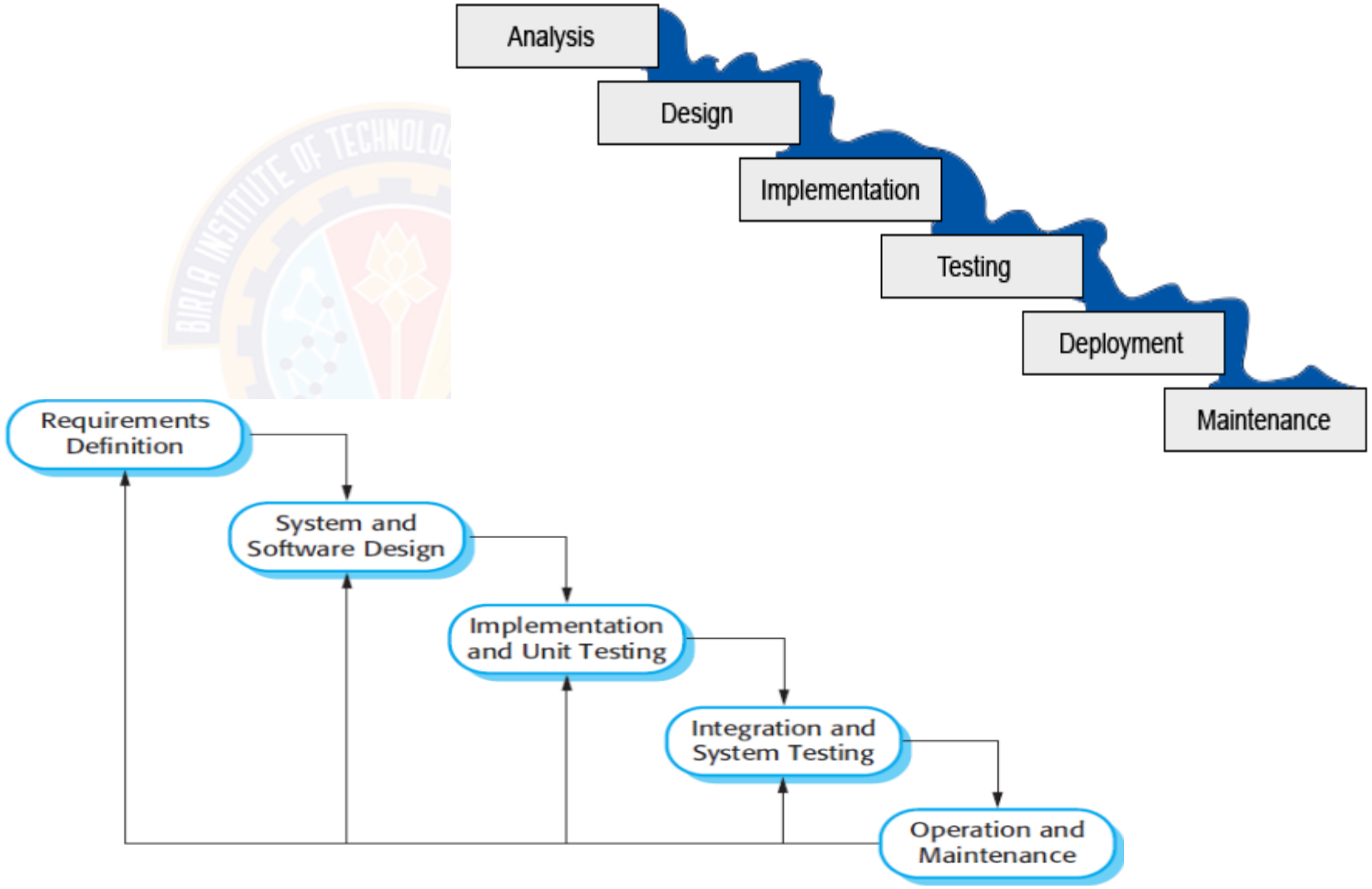
The Power of Automated Deployment form Continuous Delivery Book

- Large team dedicated to release
- High level of intervention
- Automated build, deploy, test and release system
- Only seven seconds to deploy the application into production
- Successful deployment of the system
- Roll back the change in the same amount of time

Waterfall Model

Waterfall Model and Feedback Amendment in Waterfall Model

- Classical Life cycle /Black Box Model
- Sequential in Nature
- Systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software



Waterfall Model Contd..

Advantages

- Easy to use and follow
- Cost effective
- Each phase completely developed
- Development processed in sequential manner, so very less chance of rework
- Easy to manage the project
- Easy documentation

Waterfall Model Contd..

Waterfall Model Problems

- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway
- In principle, a phase has to be complete before moving onto the next phase
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process
  - Few business systems have stable requirements

- Does Waterfall ends????
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work

Waterfall Model Contd..

Waterfall Model Problems

- Does Waterfall ends????
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work

Need of Agile

Why Agile?

- The project will produce the wrong product
- The project will produce a product of inferior quality
- The project will be late
- We'll have to work 80 hour weeks
- We'll have to break commitments
- We won't be having fun

- Storm called Agile
- According to VersionOne's State of Agile Report in 2017 says 94% of organizations practice Agile, and in 2018 it reported 97% organizations practice agile development methods

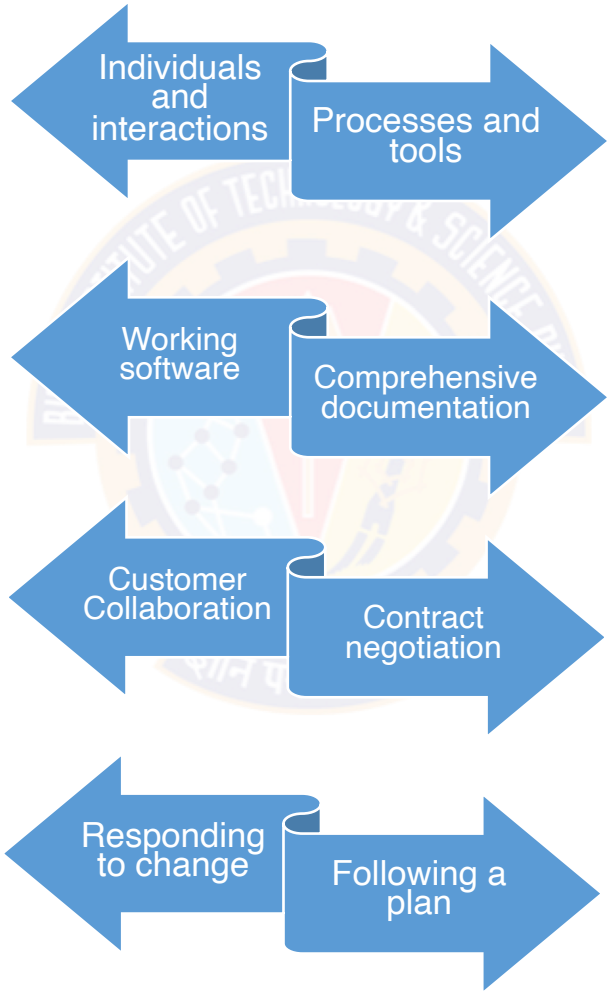
Principles of Agile Methodology

Twelve Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity--the art of maximizing the amount of work not done--is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Pillars of Agile Methodology

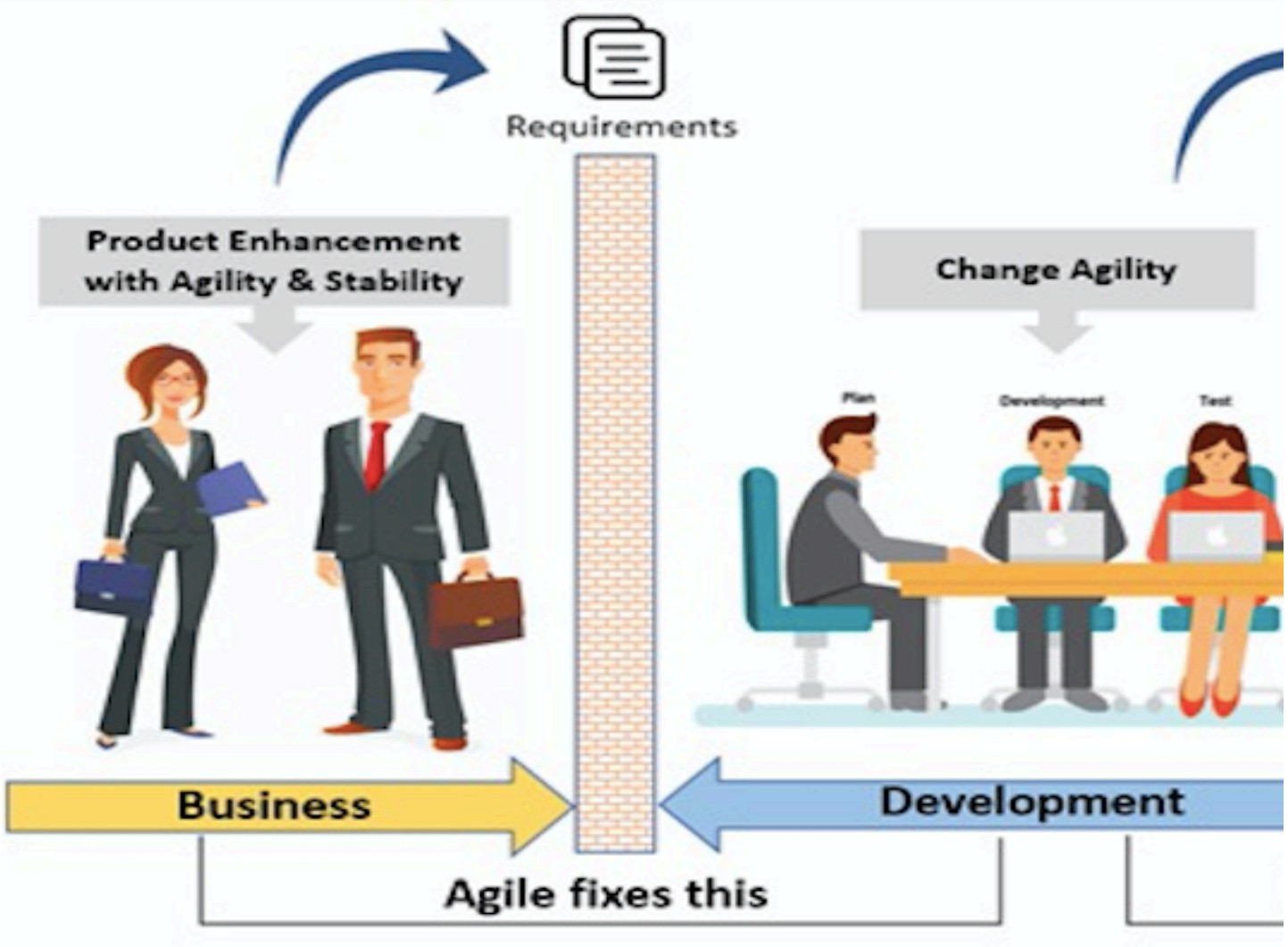
Agile focuses on



Agile Brings What??

Being Agile

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed



Agile Methodologies

Few of Agile Methodologies

- Scrum
- Extreme Programming [XP]
- Test driven Development [TDD]
- Feature Driven Development [FDD]
- Behavior-driven development [BDD]

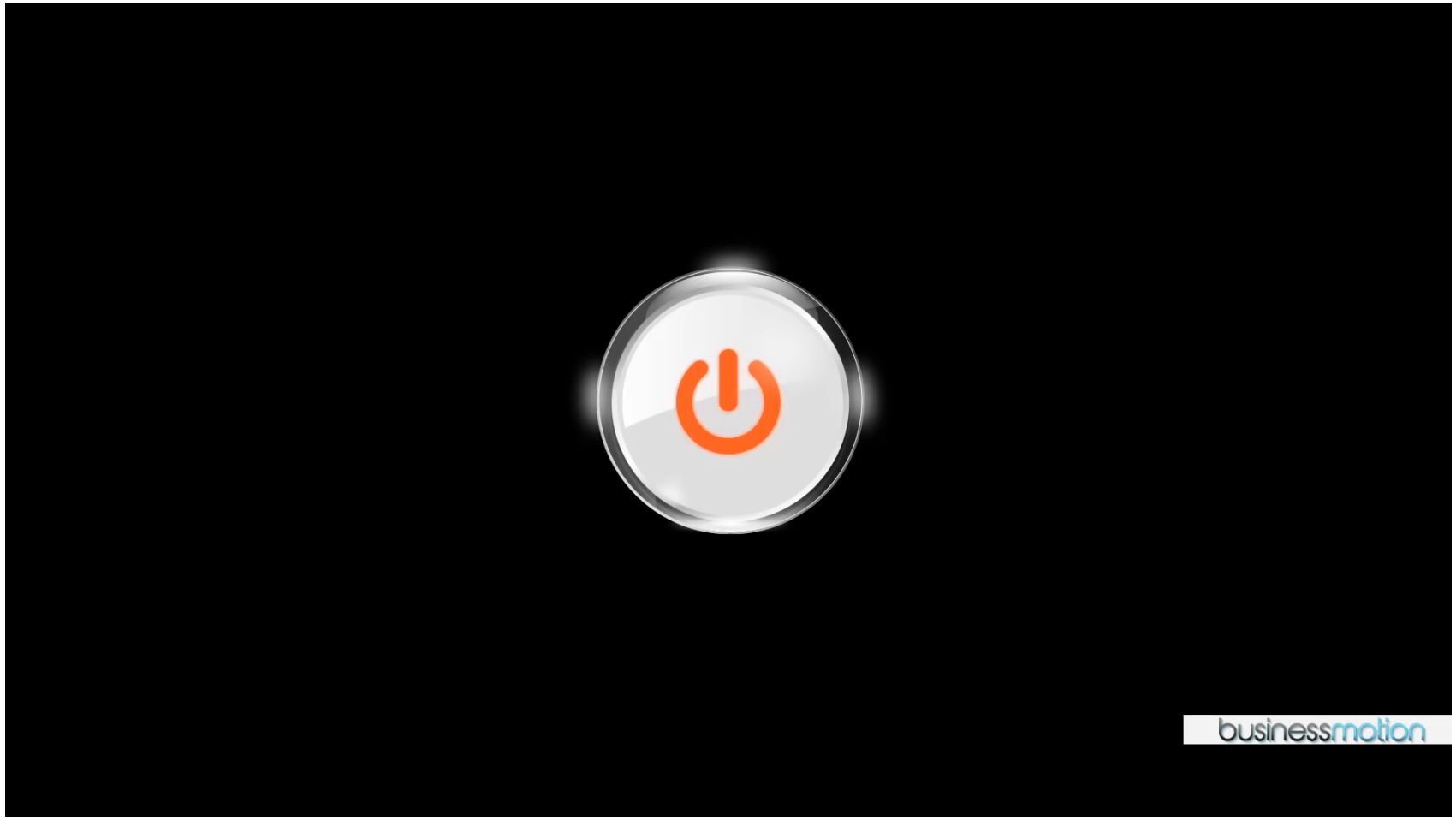
Roles in Agile

Basic roles involved

- User
- Product Owner
- Software Development Team

Agile Methodology

Summary



https://www.youtube.com/watch?v=1iccpf2eN1Q

Operational Methodology

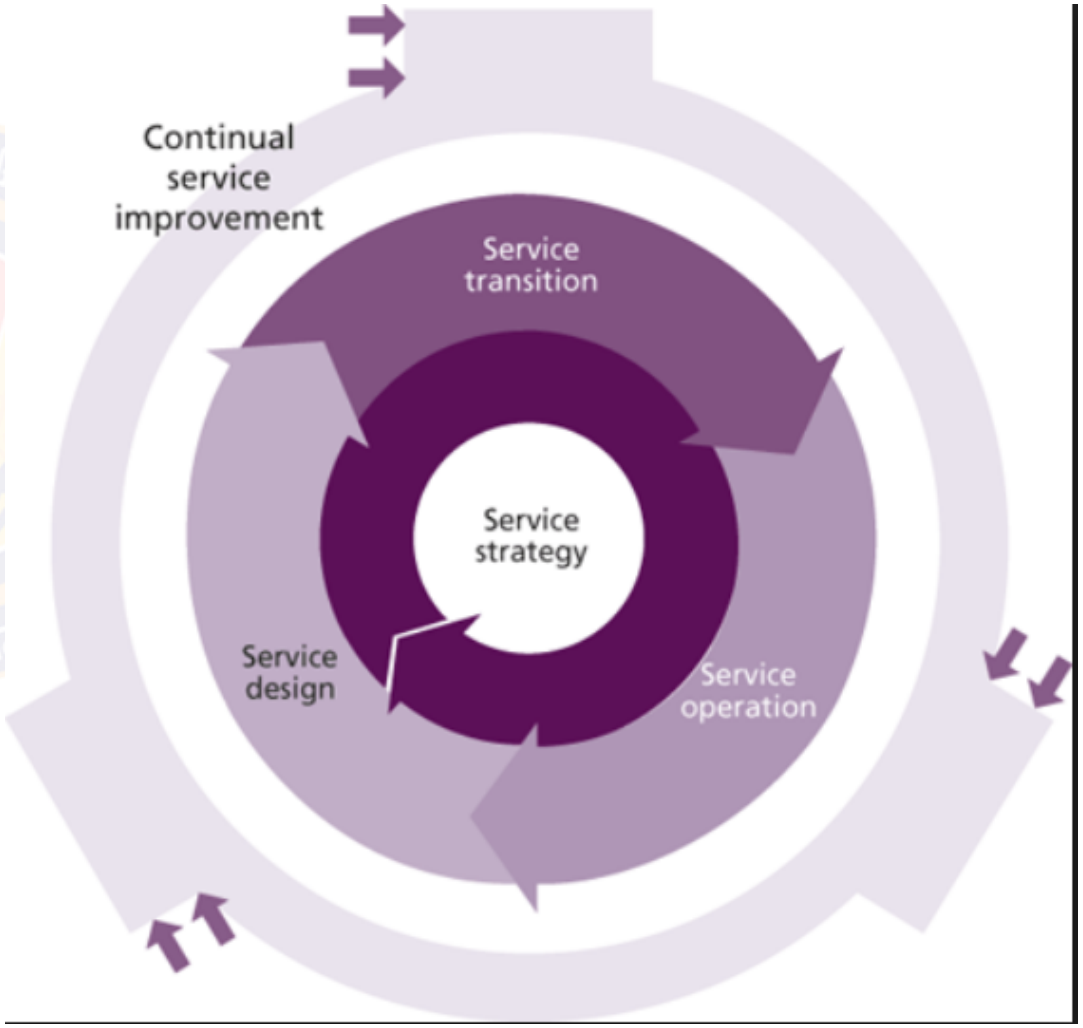
ITIL

- ITIL is a framework of best practices for delivering IT services
- The ITIL processes within IT Service Management (ITSM) ensure that IT Services are provided in a focused, client-friendly and cost-optimized manner
- ITIL's systematic approach to IT service management can help businesses
  - Manage risk
  - Strengthen customer relations
  - Establish cost-effective practices
  - And build a stable IT environment
- that allows for
  - Growth
  - Scale and
  - Change

IT Service Management (ITSM)

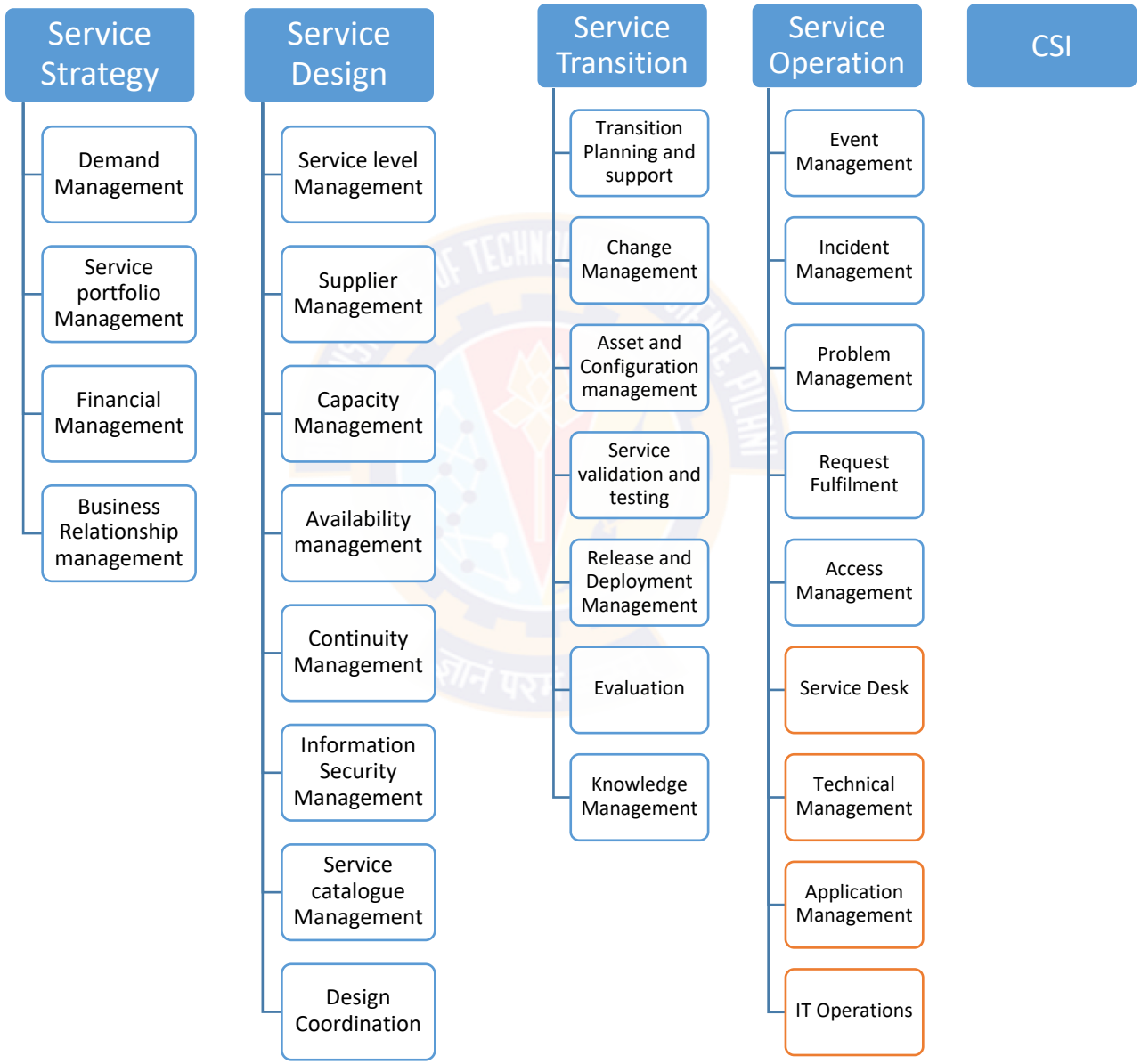
Lifecycle

- ITIL views ITSM as a lifecycle
- Five Phases:
  - Service Strategy
  - Service Design
  - Service Transition
  - Service Operation
  - Service Improvement [create and maintain value]



ITIL

Framework



ITIL

Is a Project???

- ITIL is not a “Project”
- ITIL is ongoing journey
- Benefits of ITIL:
  - Pink Elephant reports that Procter and Gamble saved about $500 million over four years by reducing help desk calls and improving operating procedures
  - Nationwide Insurance achieved a 40 percent reduction in system outages and estimates a $4.3 million ROI over three years
  - Capital One reduced its "business critical" incidents by 92 percent over two years

ITIL

Lets Review

https://www.youtube.com/watch?v=FSfgovmPHO8

DevOps

Definition

- “DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality”

Implications of this definition

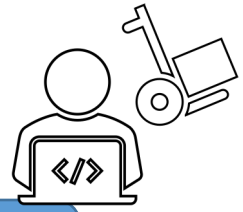
- Practices and tools
- Do not restricted scope of DevOps to testing and development

DevOps

Perspective involved

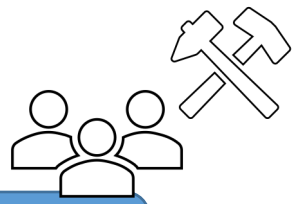
Business

Ideas
High Service Level
Expectation



Dev

Speed



Ops

Stabilize

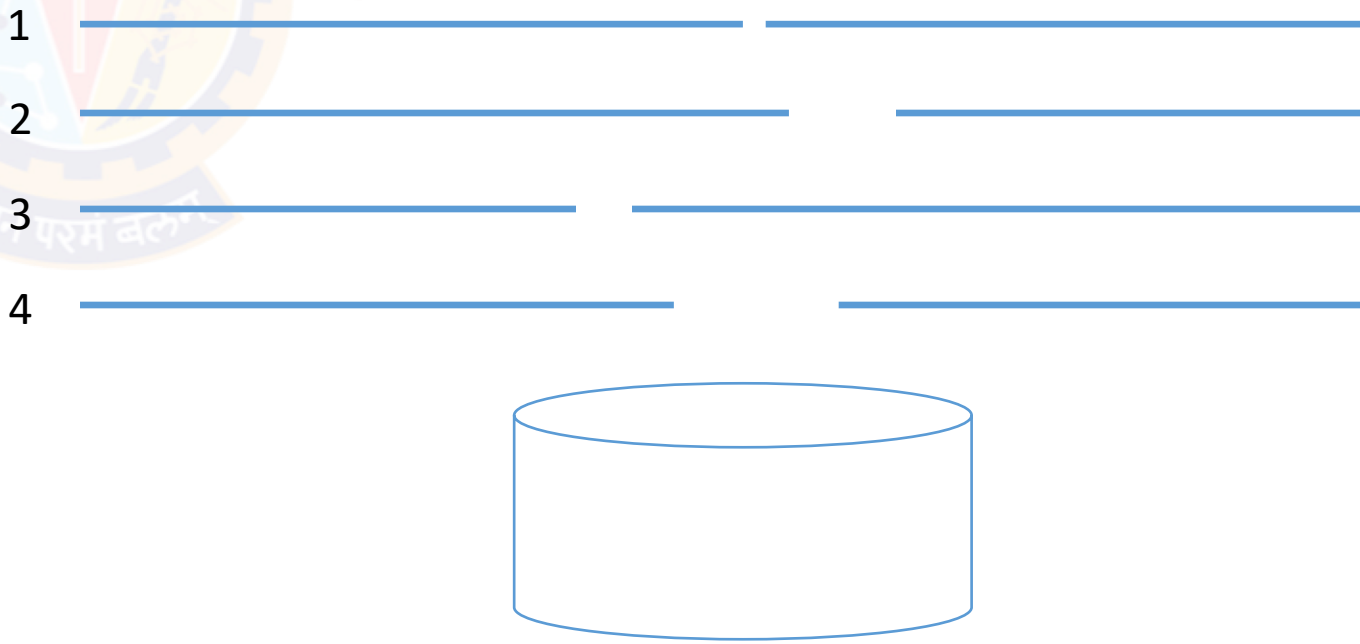
Trust

Quality:

- Checklist
- Approval
- Release Note
- Audit
- Compliance

Problems of Delivering Software

- Converting Idea to Product / Service?
- Reliable, rapid, low-risk software releases
- Ideal Environment
- Generic Methodologies for Software Methodologies
- More focus on Requirement Gathering
- Understanding the Value Stream Map



Common Release Antipatterns

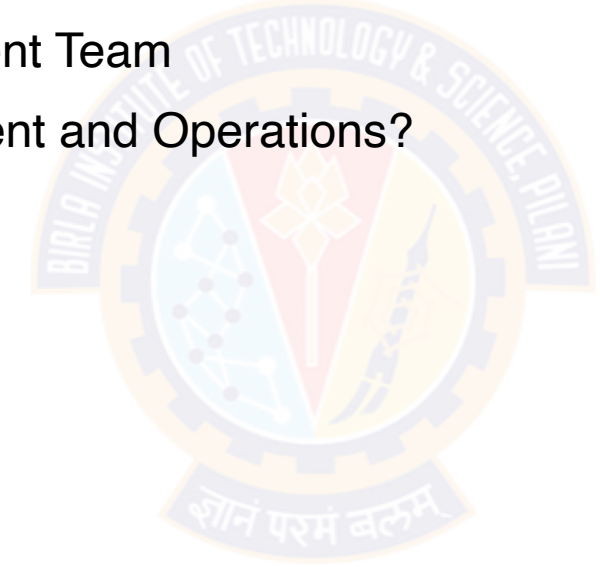
Deploying Software Manually

- Extensive and detailed documentation
- Reliance on manual testing
- Frequent calls to the development team to explain
- Frequent corrections to the release process
- Sitting bleary-eyed in front of a monitor at 2 A.M

Common Release Antipatterns

Deploying to a Production-like Environment Only after Development Is Complete

- Tester tested the system on development machines
- Releasing into staging is the first time that operations people interact with the new release
- Who Assembles? The Development Team
- Collaboration between development and Operations?



Common Release Antipatterns

Manual Configuration Management of Production Environments

- Difference in Deployment to Stage and Production
- Different host behave differently
- Long time to prepare an environment
- Cannot step back to an earlier configuration of your system
- Modification to Configuration Directly

Principles of Software Delivery

- Create a Repeatable, Reliable Process for Releasing Software
- Automate Almost Everything
- Keep Everything in Version Control
- If It Hurts, Do It More Frequently, and Bring the Pain Forward
- Build Quality In
  - “The Earlier you catch the defects, the cheaper they are to fix”
- Done, Means Released
- Everybody Is Responsible for the Delivery Process
- Continuous Improvement

DevOps Practices

Five different categories of DevOps practices

- Treat Ops as first-class citizens from the point of view of requirements
- Make Dev more responsible for relevant incident handling
- Enforce the deployment process used by all, including Dev and Ops personnel
- Use continuous deployment
- Develop infrastructure code, such as deployment scripts

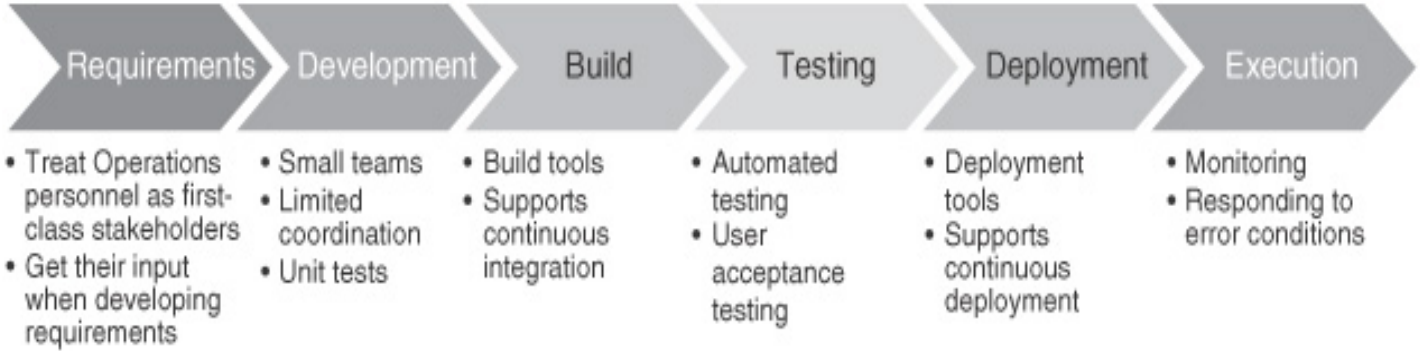
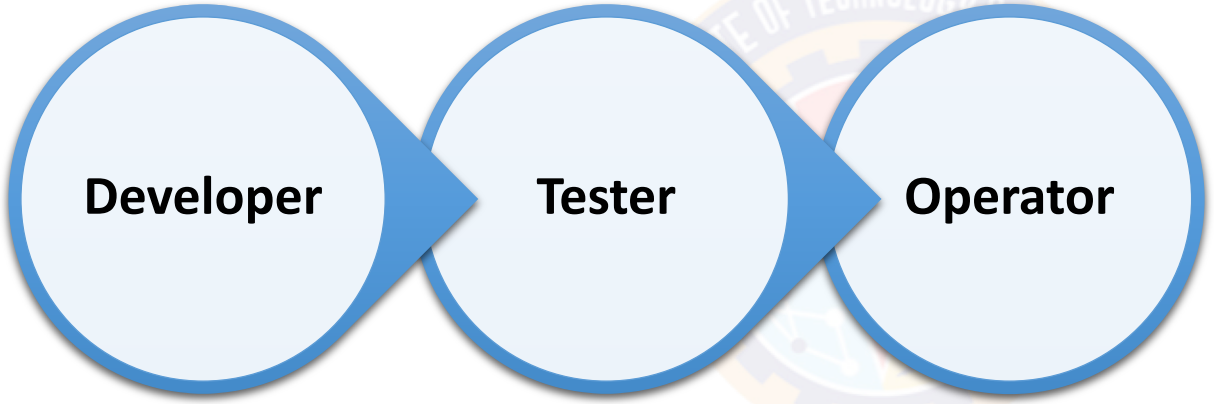


FIGURE 1.1 DevOps life cycle processes [Notation: Porter's Value Chain]

Need for DevOps

Timelines



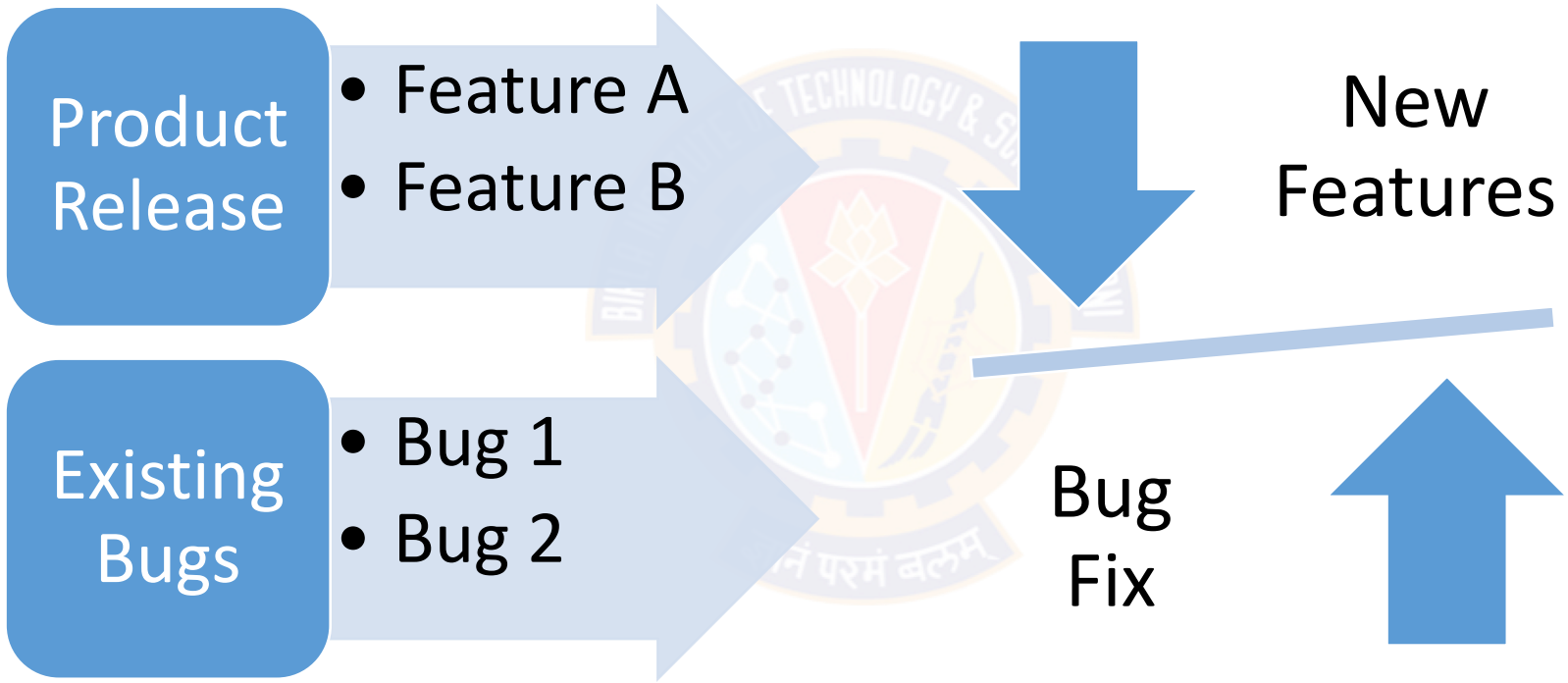
BUG

FIX

When

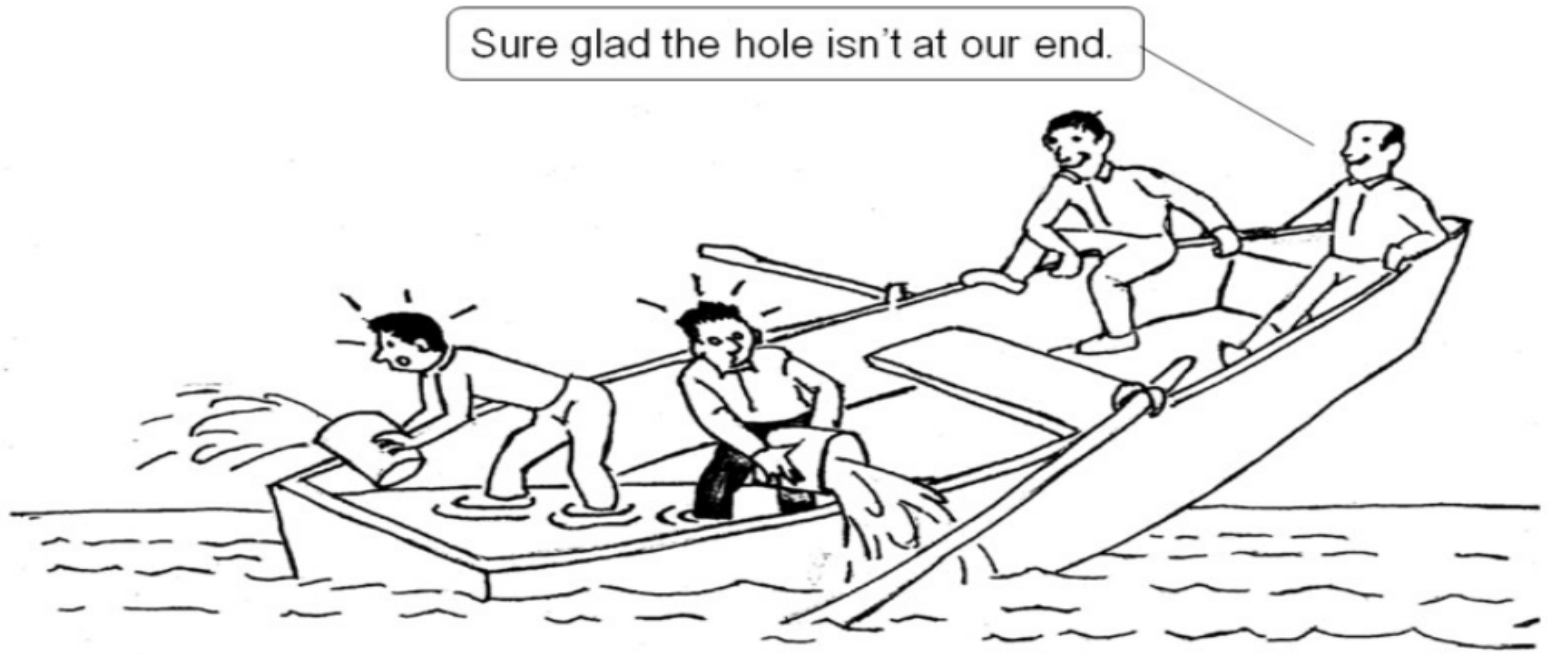
Need for DevOps

Imbalance



Need for DevOps

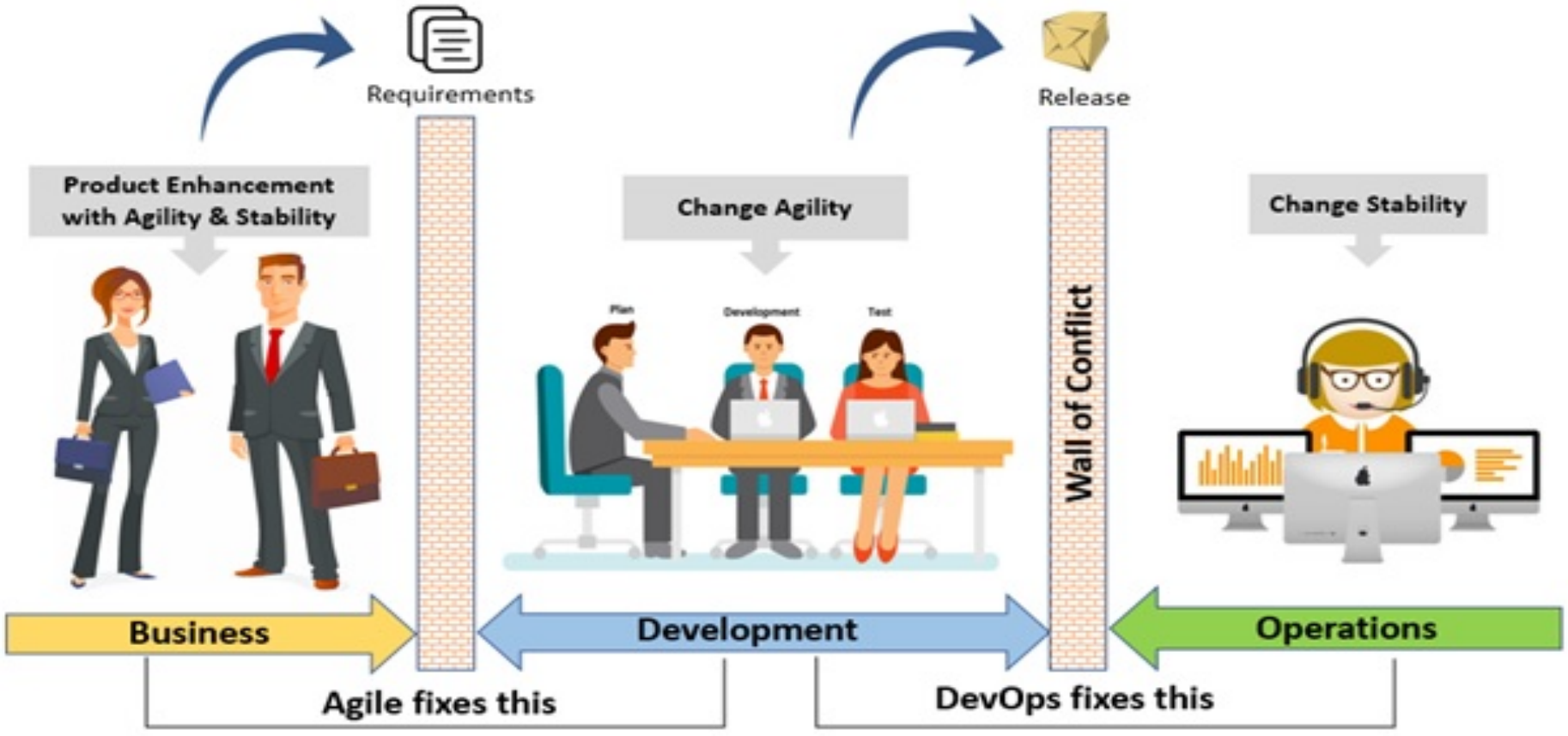
Blame Game



Need for DevOps

Where is Operations?

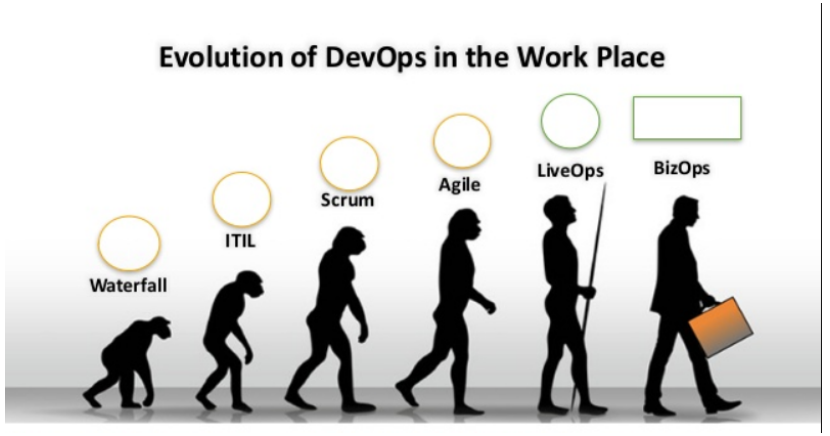
- Development is All Well (Waterfall, Agile)
- Where is Operations?



The evolution of DevOps

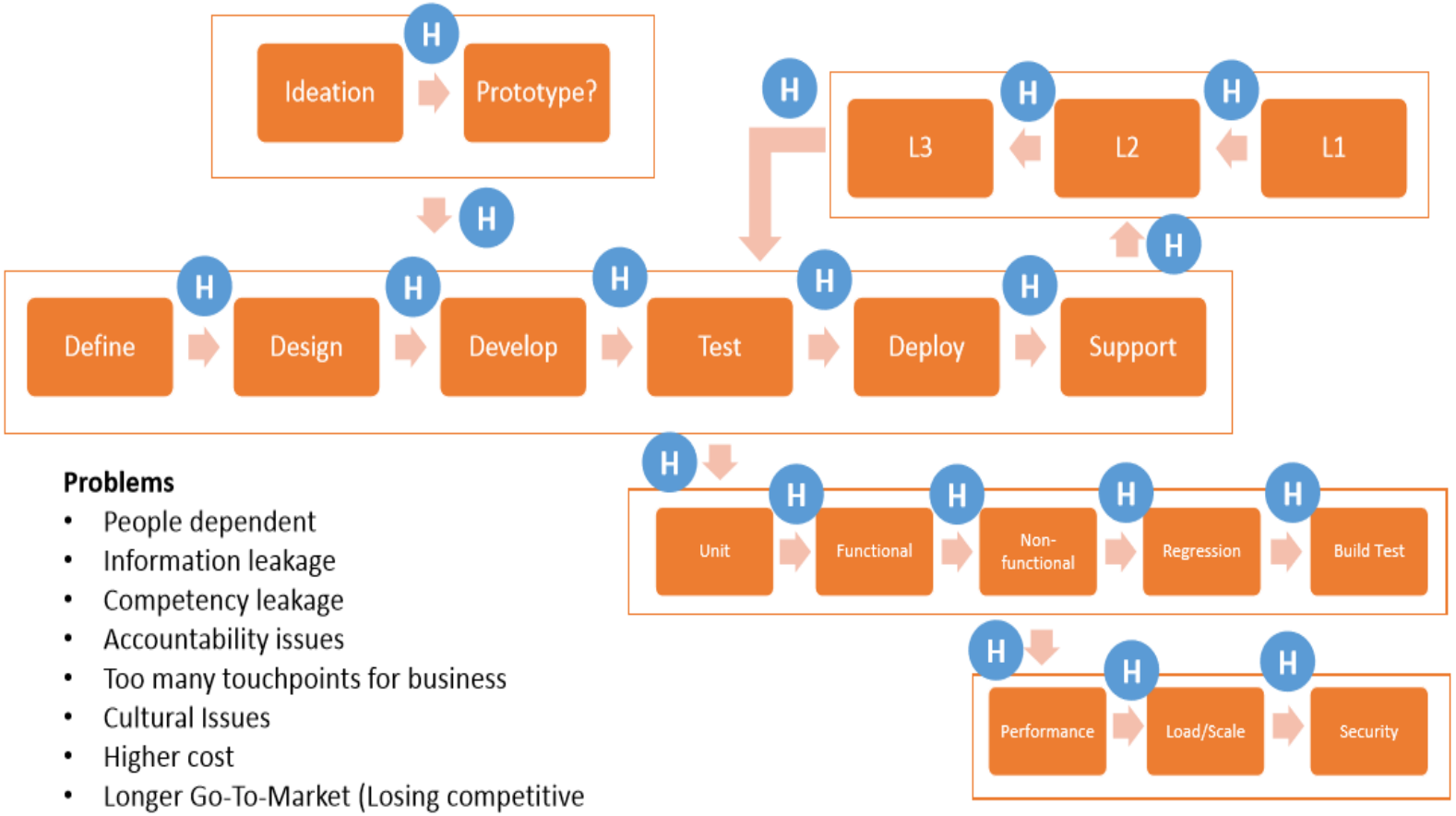
History

- Back in 2007
- Patrick Debois [Belgian Engineer]
- Initially it was Agile Infrastructure but later coined the phrase DevOps
- Velocity conference in 2008
- And if you see you may come across more tangential DevOps Initiative
  - WinOps
  - DevSecOps
  - BizDevOps



The old world before DevOps

More Handshakes

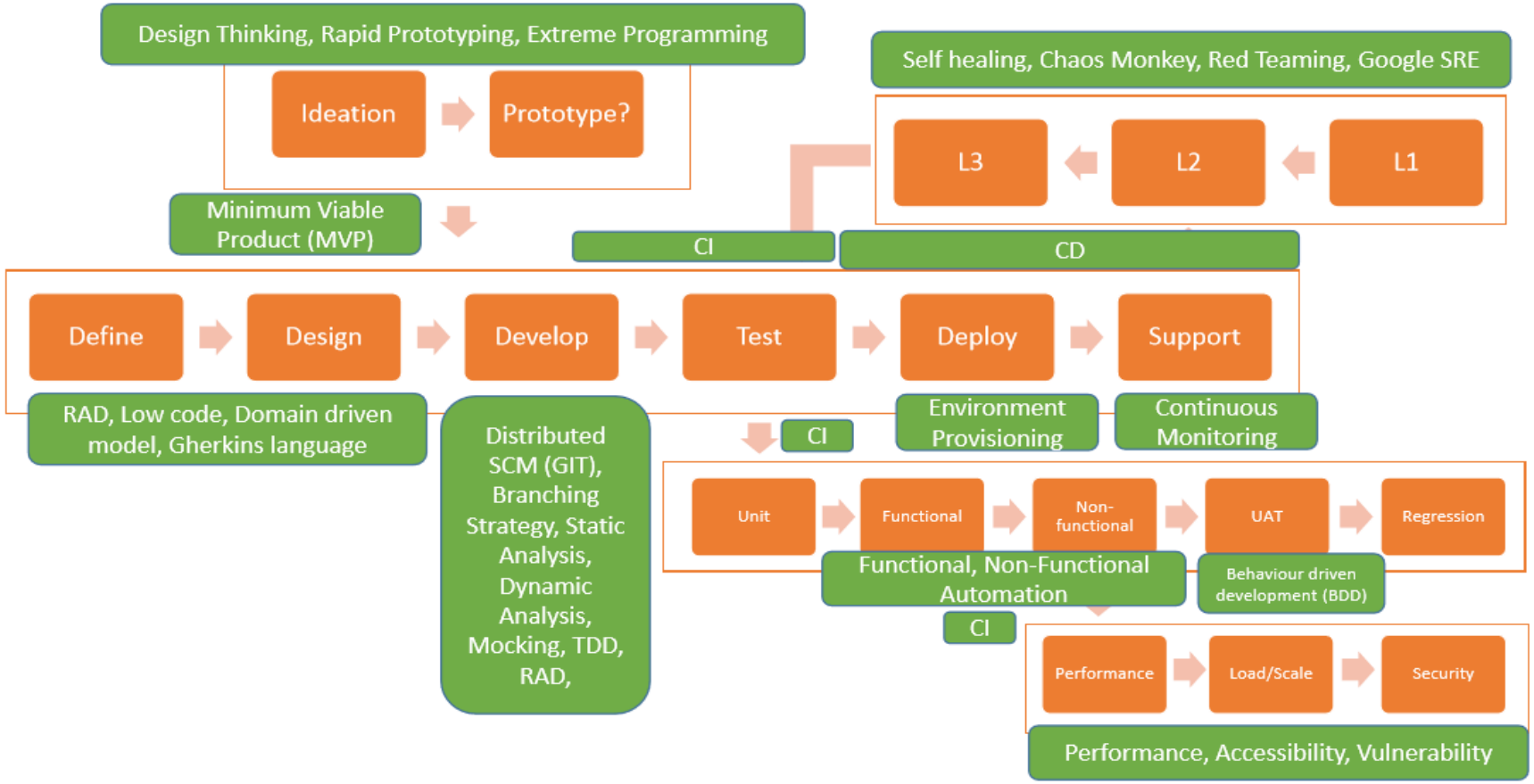


Problems

- People dependent
- Information leakage
- Competency leakage
- Accountability issues
- Too many touchpoints for business
- Cultural Issues
- Higher cost
- Longer Go-To-Market (Losing competitive advantage, Opportunity loss)

Evolution of DevOps :: New world

No More Handshakes



Case Study

Flickr / Yahoo

- Flickr was able to reach
  - 10+ Deploy per day after adopting DevOps
- In 2009
- You May Also Refer:
- YouTube Link: https://www.youtube.com/watch?v=LdOe18KhtT4

flickr

Case Study

Netflix

- Netflix's streaming service is a large distributed system hosted on Amazon Web Services (AWS)
- So many components that have to work together to provide reliable video streams to customers across a wide range of devices
- Netflix engineers needed to focus heavily on the quality attributes of reliability and robustness for both server- and client-side components
- Achieved this with DevOps by introducing a tool called Chaos Monkey
- Chaos Monkey is basically a script that runs continually in all Netflix environments, causing chaos by randomly shutting down server instances
- Thus, while writing code, Netflix developers are constantly operating in an environment of unreliable services and unexpected outages
- Unique opportunity to test their software in unexpected failure conditions

NETFLIX

Thank You!

In our next session: