BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Agenda

Introduction

- About DevOps
- Problems of Delivering Software
- Principles of Software Delivery
- Need for DevOps
- DevOps Practices in Organization
- The Continuous DevOps Life Cycle Process
- Evolution of DevOps
- Case studies

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

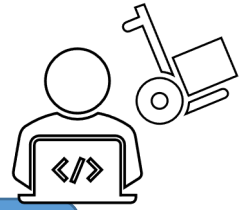
- Practices and tools
- Do not restricted scope of DevOps to testing and development

DevOps

Perspective involved

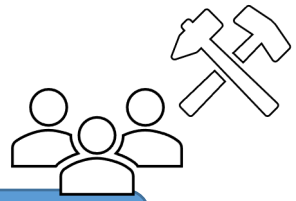
Business

Ideas
High Service Level
Expectation



Dev

Speed



Ops

Stabilize

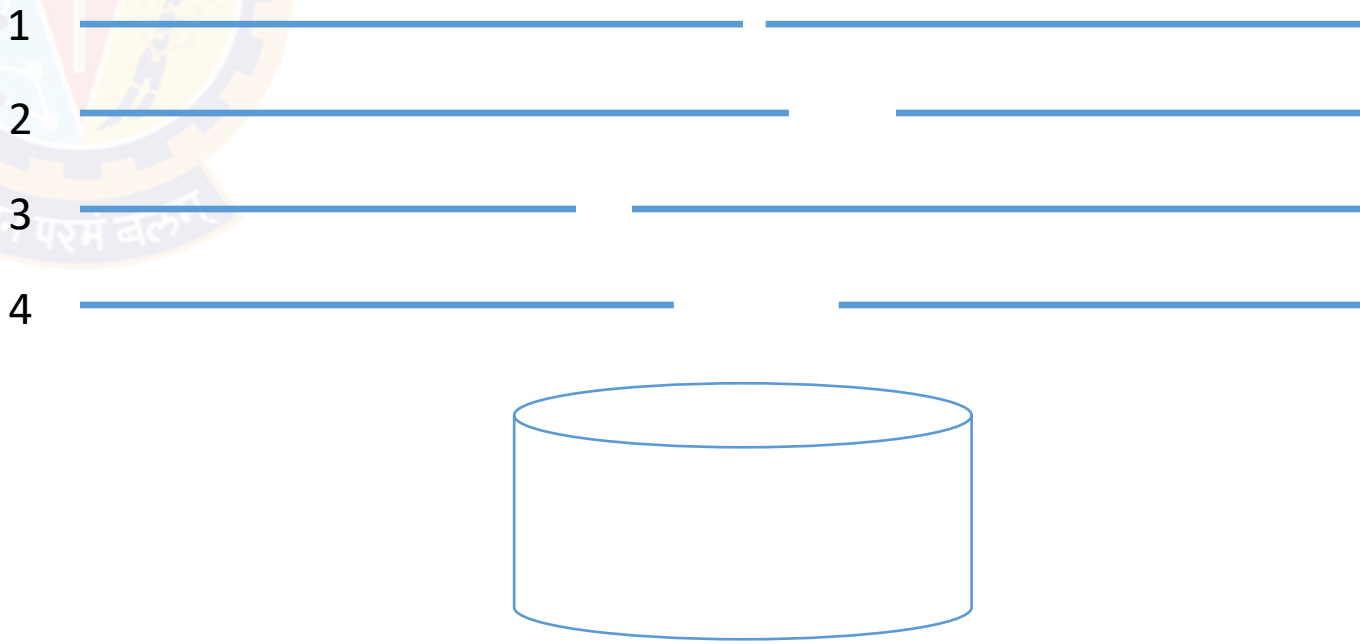
Trust

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

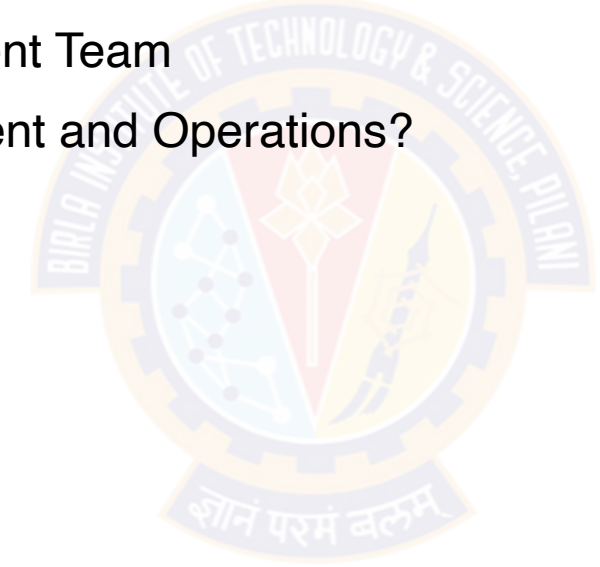
Deploying Software Manually

- 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

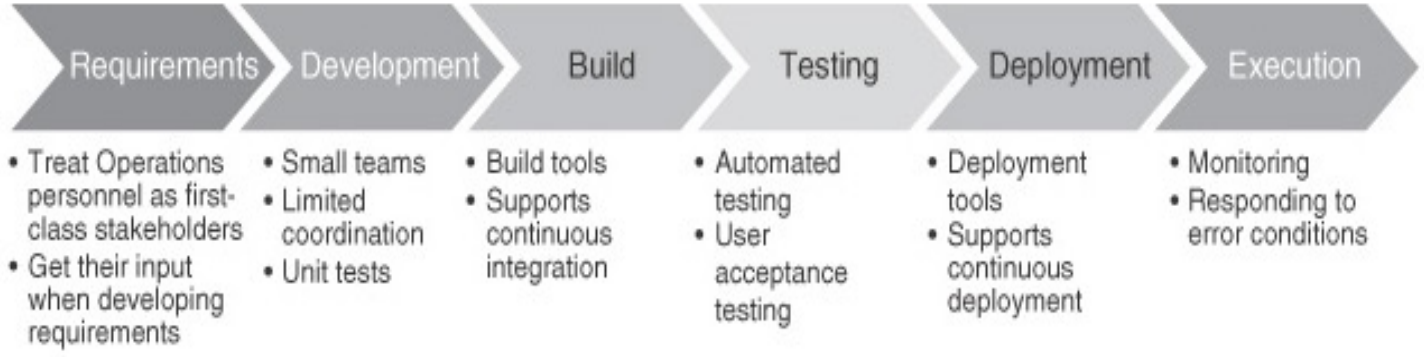
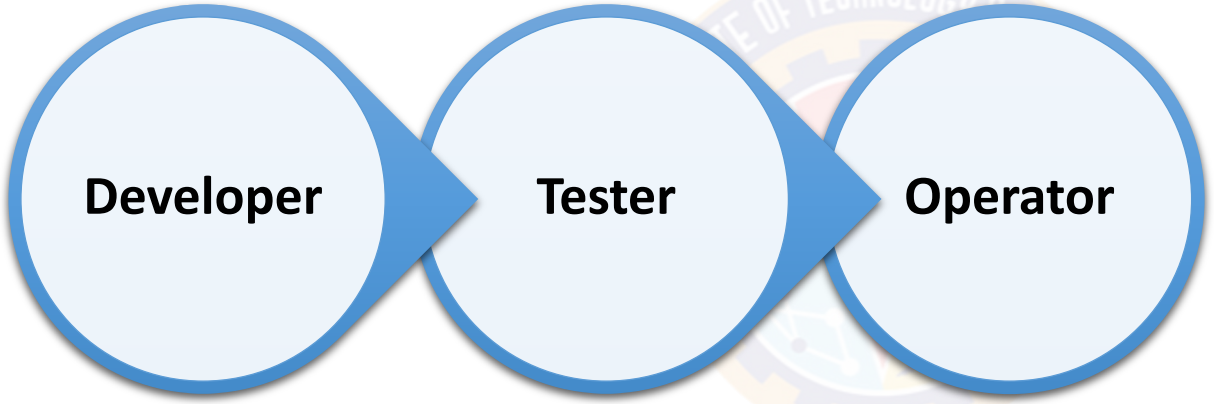


FIGURE 1.1 DevOps life cycle processes [Notation: Porter's Value Chain]

Need for DevOps

Timelines



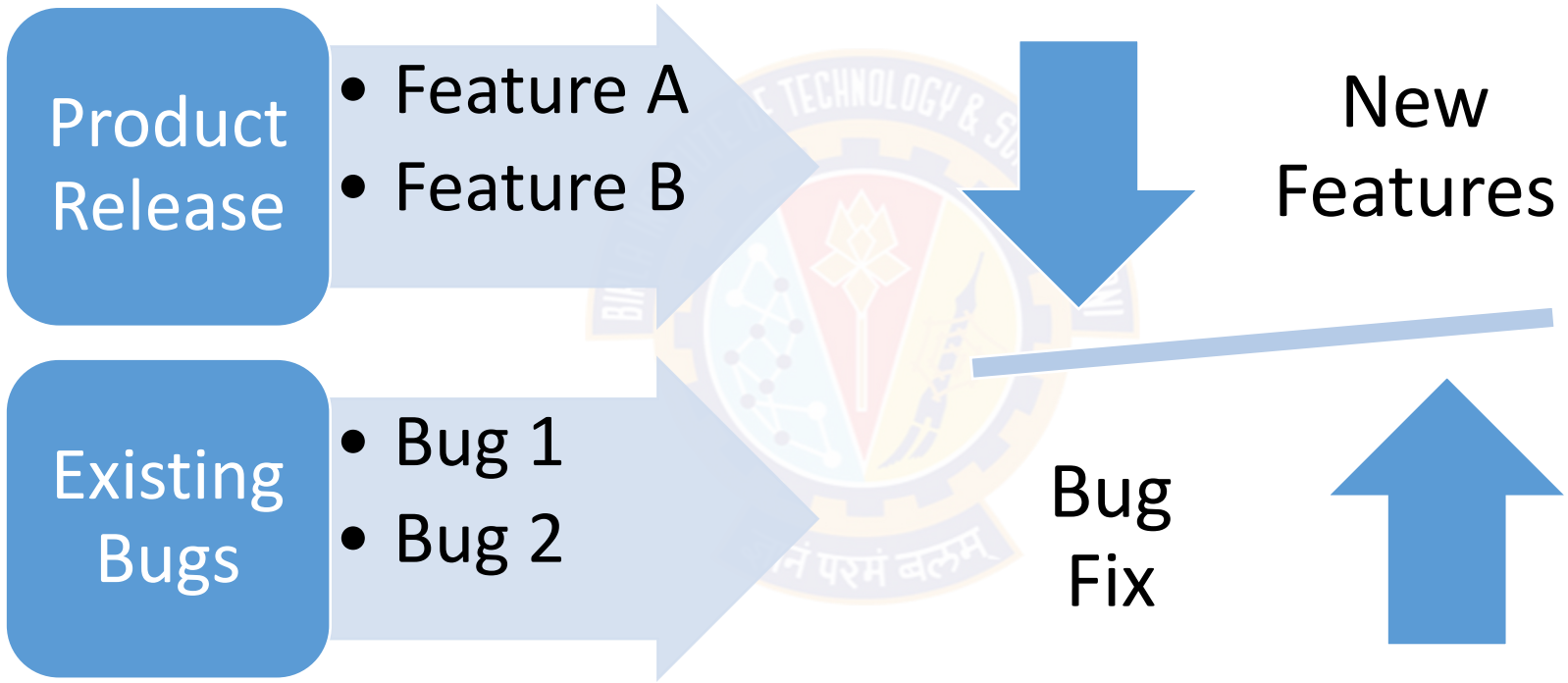
BUG

FIX

When

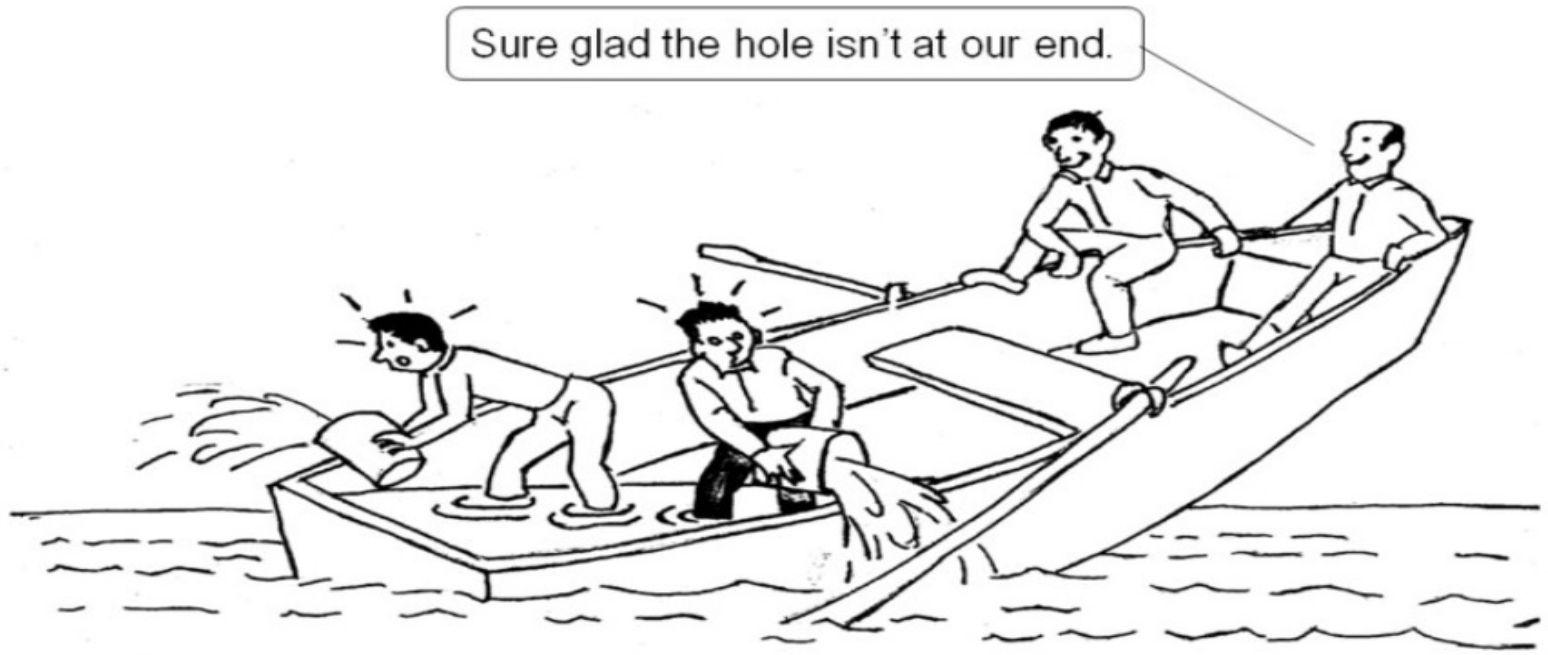
Need for DevOps

Imbalance



Need for DevOps

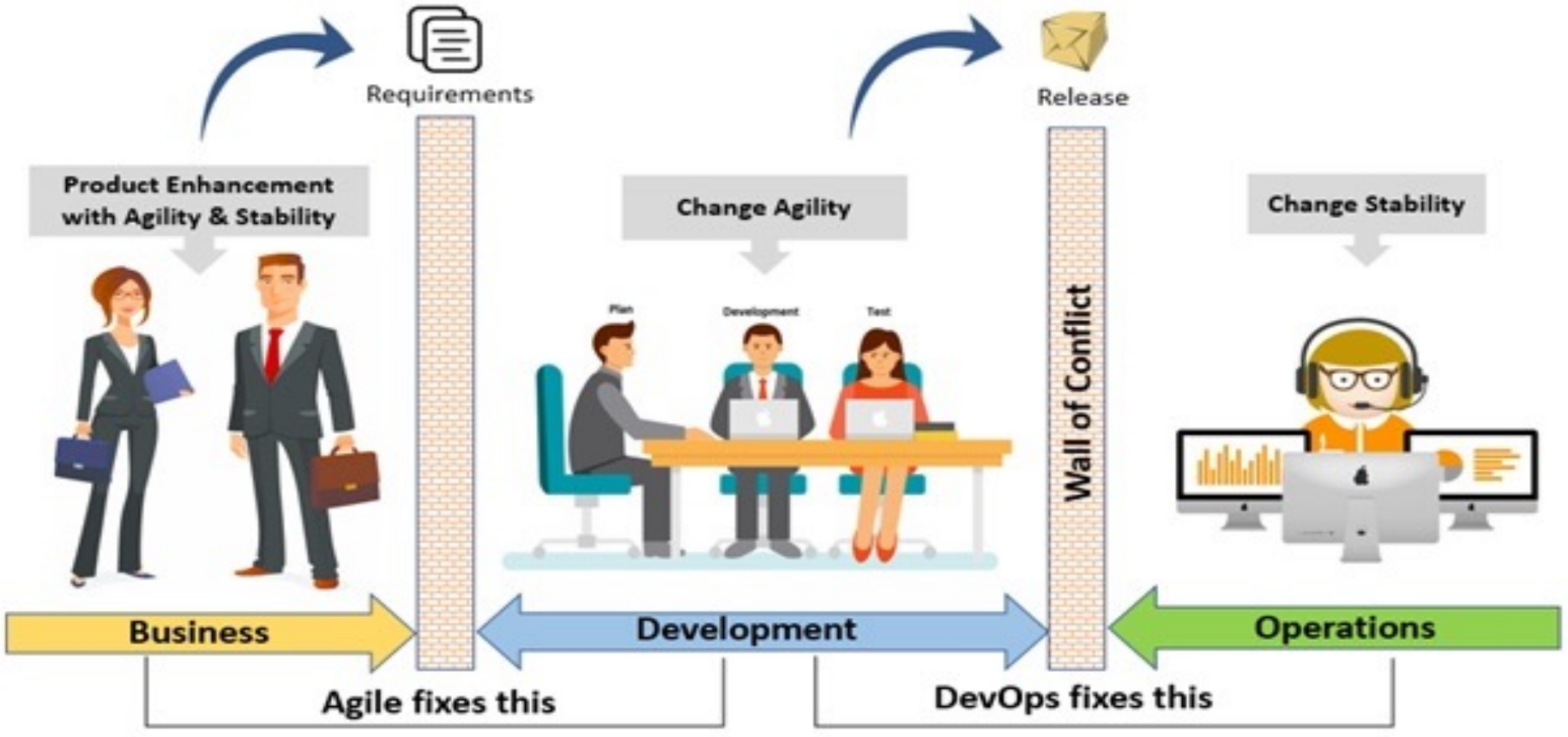
Blame Game



Need for DevOps

Where is Operations?

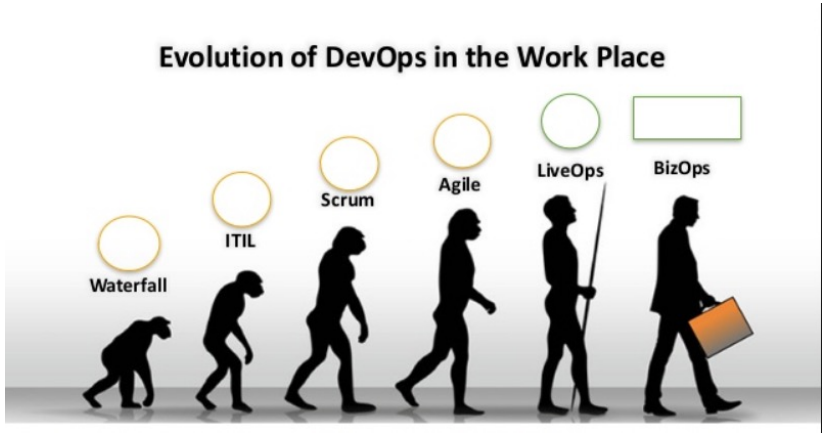
- Development is All Well (Waterfall, Agile)
- Where is Operations?



The evolution of DevOps

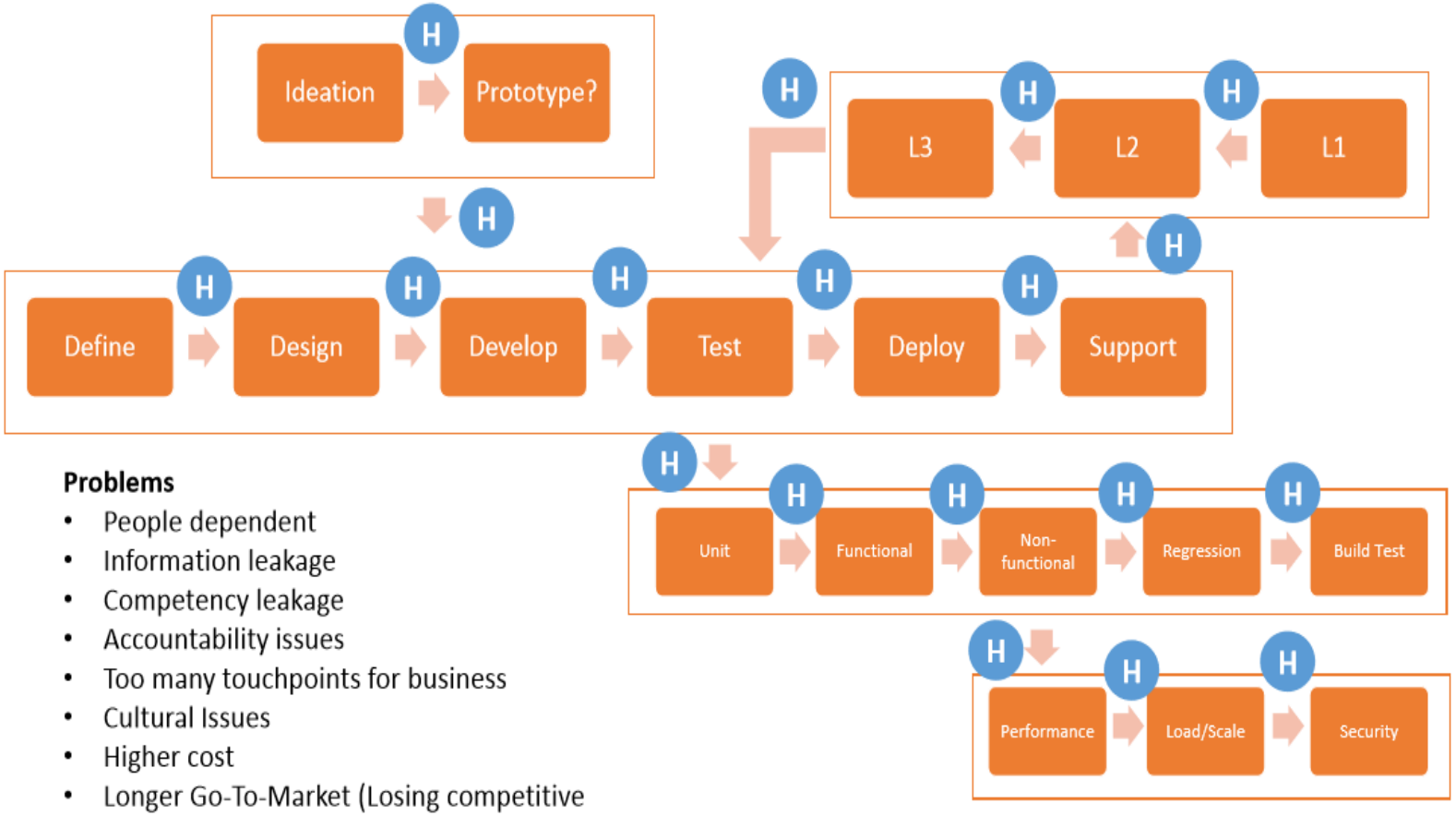
History

- 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

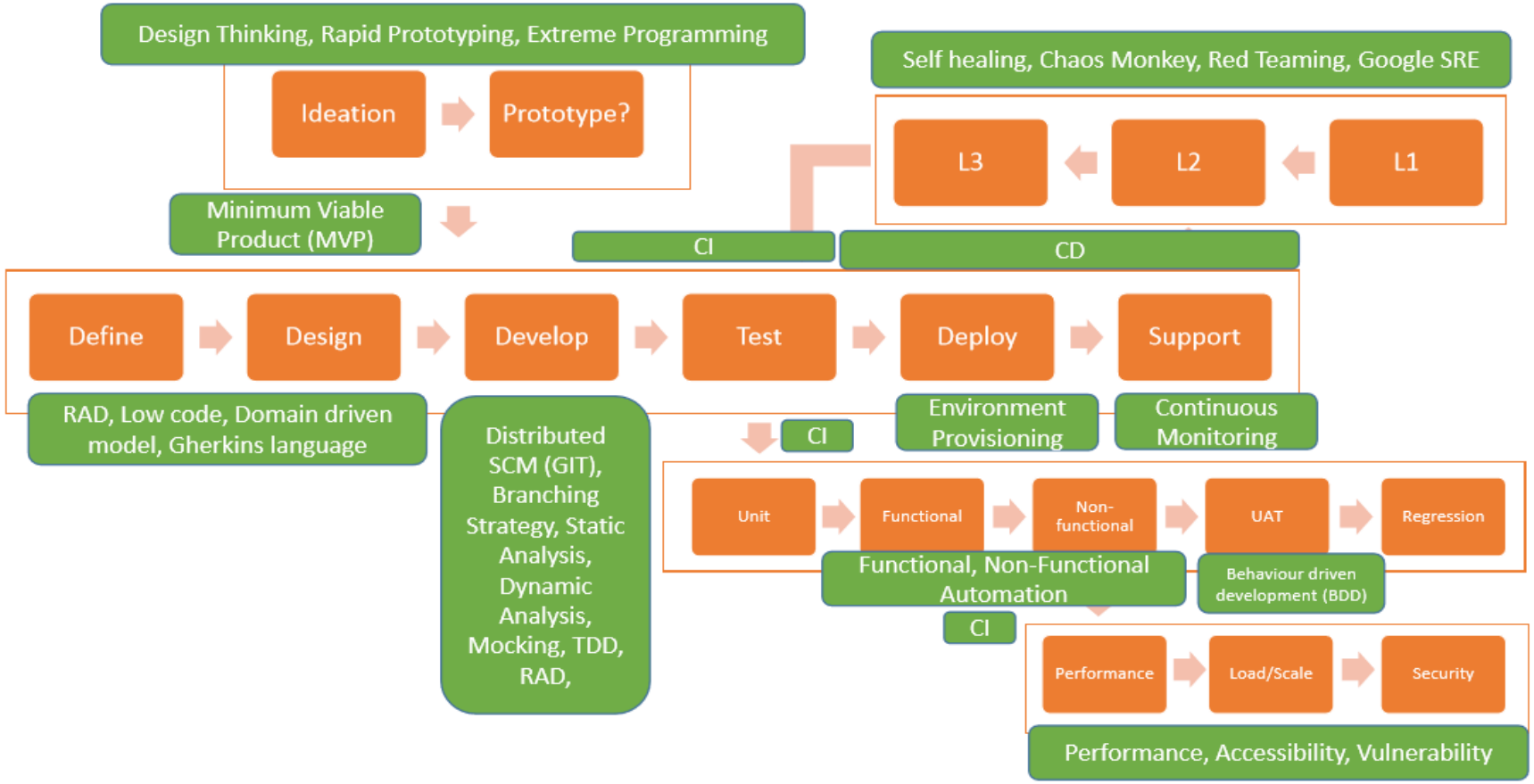


Problems

- 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

References

CS 1 & 2

- 4th chapter from Effective DevOps Building a Culture of Collaboration, Affinity, and Tooling at Scale by Jennifer Davis and Katherine Daniels
- 1st chapter Continuous Delivery by Jez Humble and David Farley and 5th Chapter from Effective DevOps Building a Culture of Collaboration, Affinity, and Tooling at Scale by Jennifer Davis and Katherine Daniels
- For ITIL: https://www.simplilearn.com/itil-key-concepts-and-summary-article,

Thank You!

In our next session: