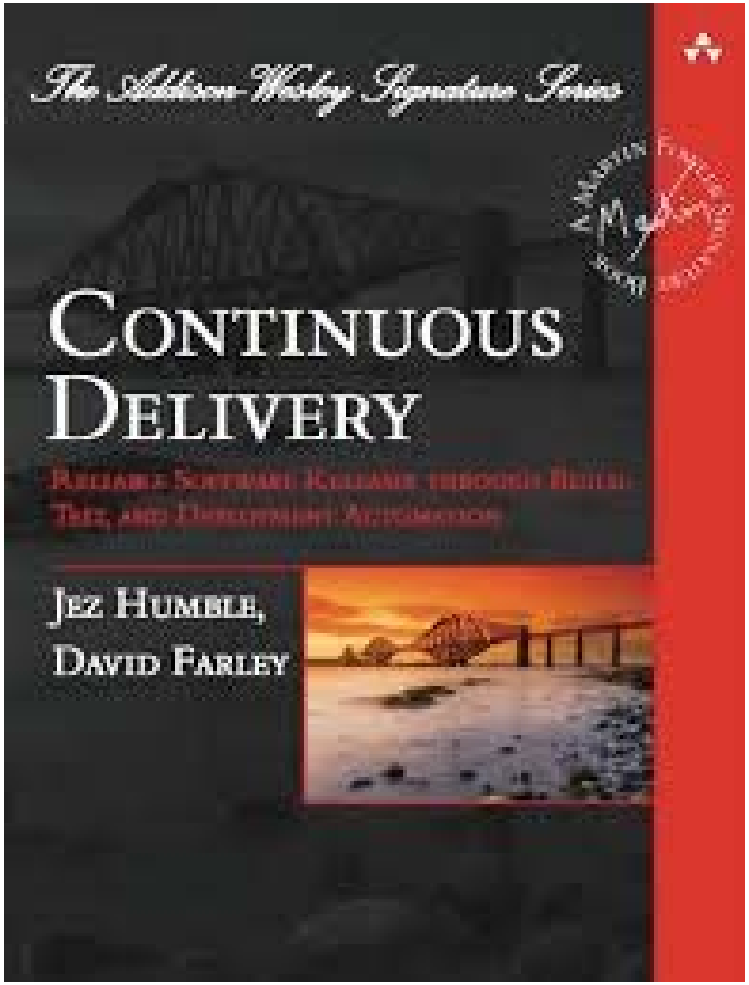
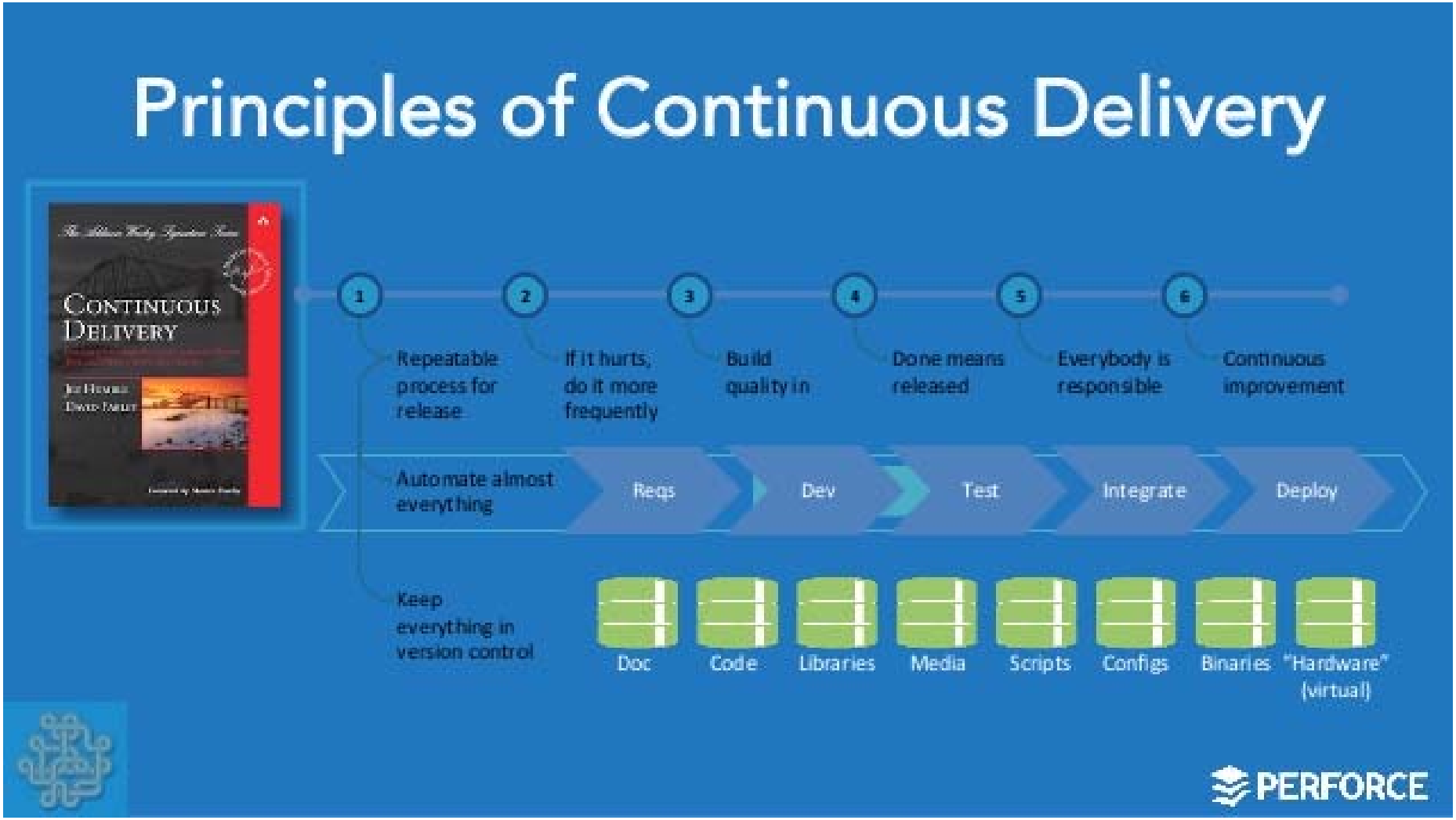
RLMCA202
Application Development and Maintenance

Module 1

Text Book –Mod 1



Principles of Software Delivery



Principles of Software delivery

- 1) **Create a Repeatable, Reliable Process for Releasing Software**
- Releasing software should be easy. It should be easy because you have tested every single part of the release process hundreds of times before. It should be as simple as pressing a button. The repeatability and reliability derive from two principles:
- **a)automate almost everything**
- **b)keep everything you need to build, deploy, test, and release your application in version control.**

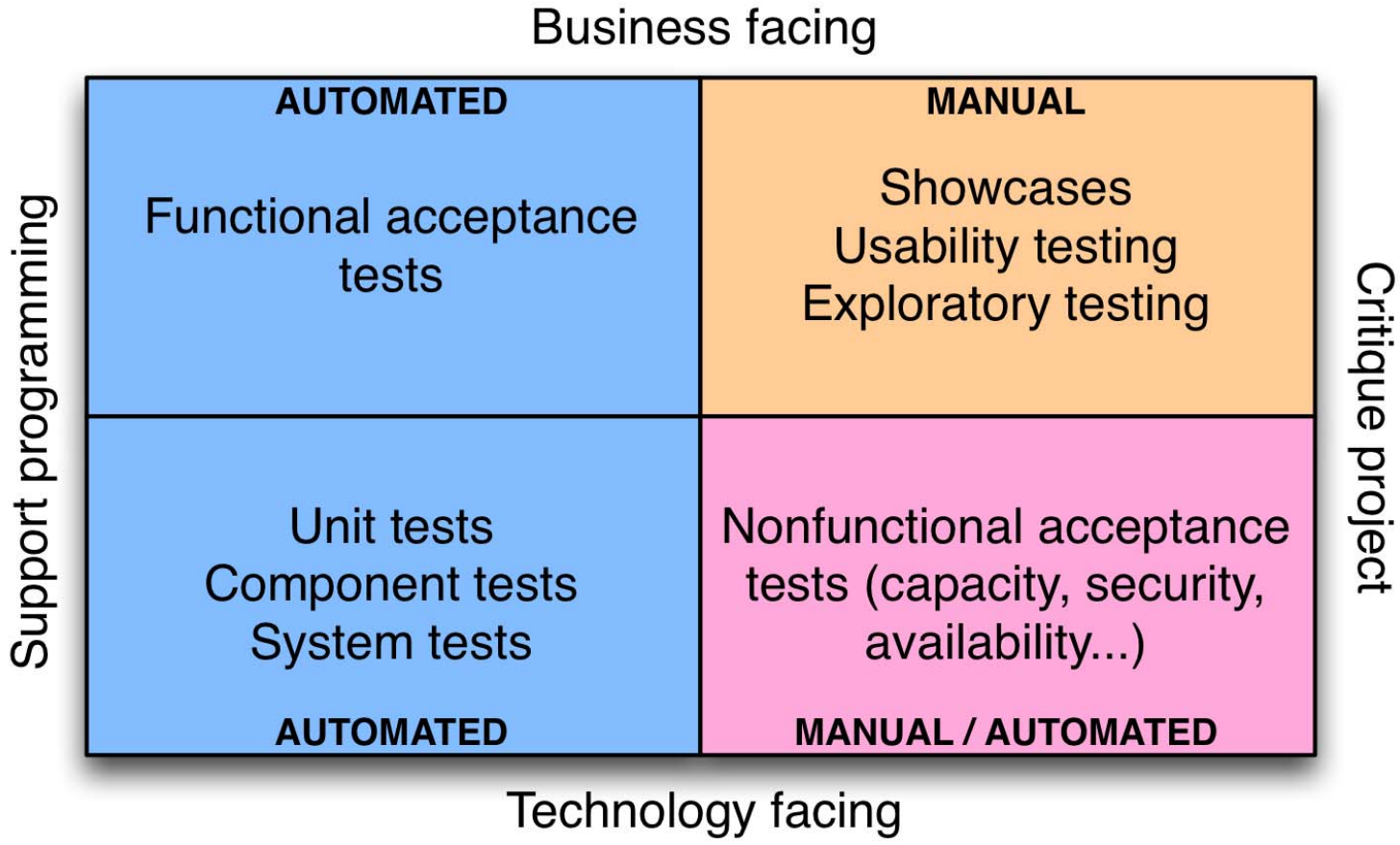
Automate almost Everything

- There are some things it is impossible to automate. Exploratory testing relies on
- experienced testers.
- Approvals for compliance purposes by definition require human intervention.
- your build process should be automated up to the point where it needs specific human direction or decision making

Automated v/s Manual Tasks

- Automated Tasks
- Functional Acceptance Testing
- Unit Test
- Computation Test
- System Test

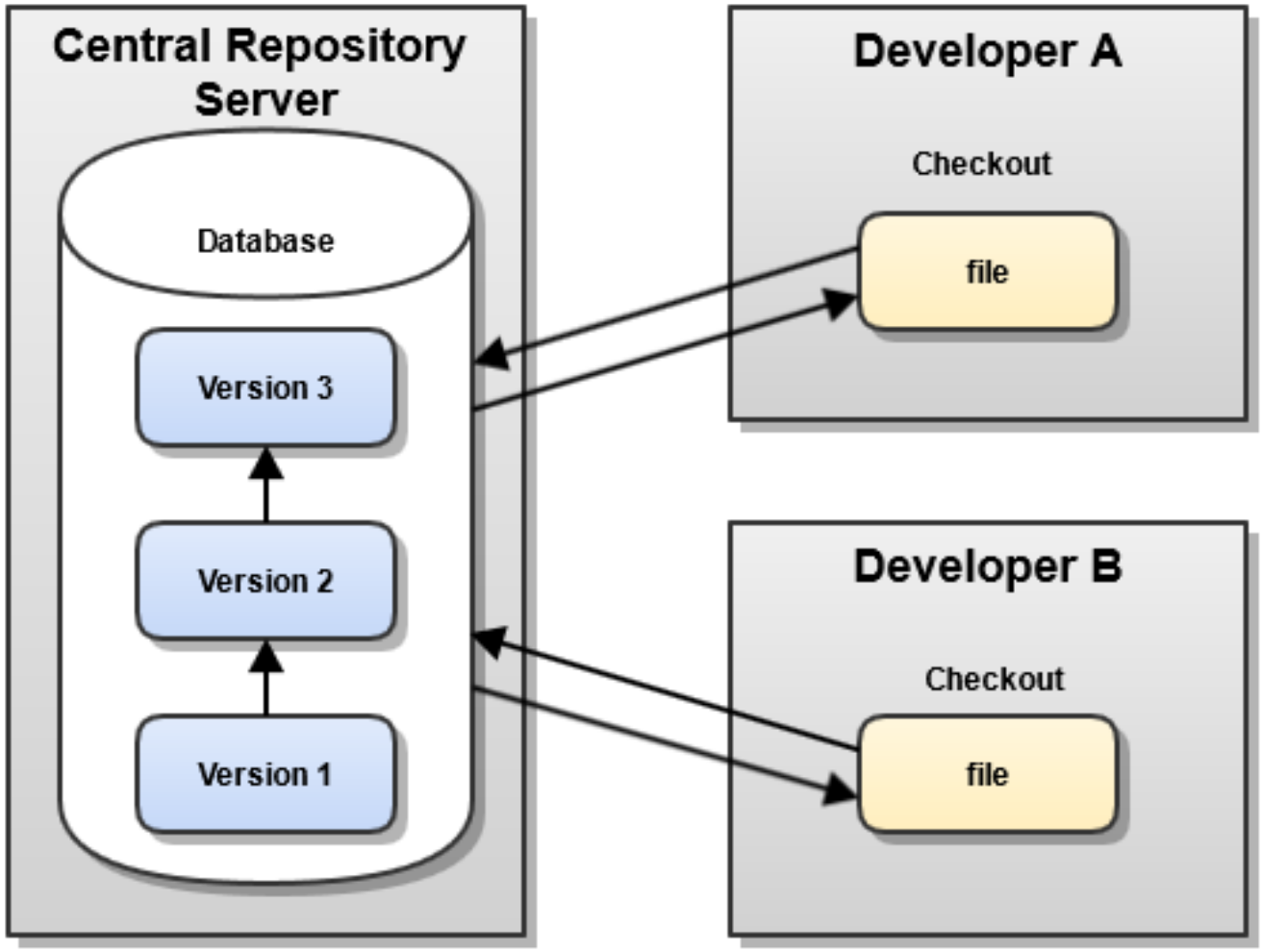
- Manual Tasks
- Usability Tests
- Exploratory testing
- Non Functional Acceptance Test(Availability, Security etc)


Business facing
AUTOMATED
Functional acceptance tests
MANUAL
Showcases
Usability testing
Exploratory testing
Support programming
Critique project
Unit tests
Component tests
System tests
AUTOMATED
Nonfunctional acceptance tests (capacity, security, availability...)
MANUAL / AUTOMATED
Technology facing

Keep Everything in Version Control

- Everything you need to build, deploy, test, and release your application should be kept in some form of **versioned storage**.
- This includes requirement documents,test scripts, automated test cases, network configuration scripts, deployment scripts, database creation, upgrade, downgrade, and initialization scripts, application stack configuration scripts, libraries, tool chains, technical documentation,and so on.
- All of this stuff should be version-controlled, and the relevant version should be identifiable for any given build.

Version Control



If It Hurts, Do It More Frequently, and Bring the Pain Forward

- Integration is often a very painful process. If this is true on your project, integrate every time somebody checks in, and do it from the start of the project.
- If testing is a painful process that occurs just before release, don’t do it at the end. Instead, do it continually from the beginning of the project.

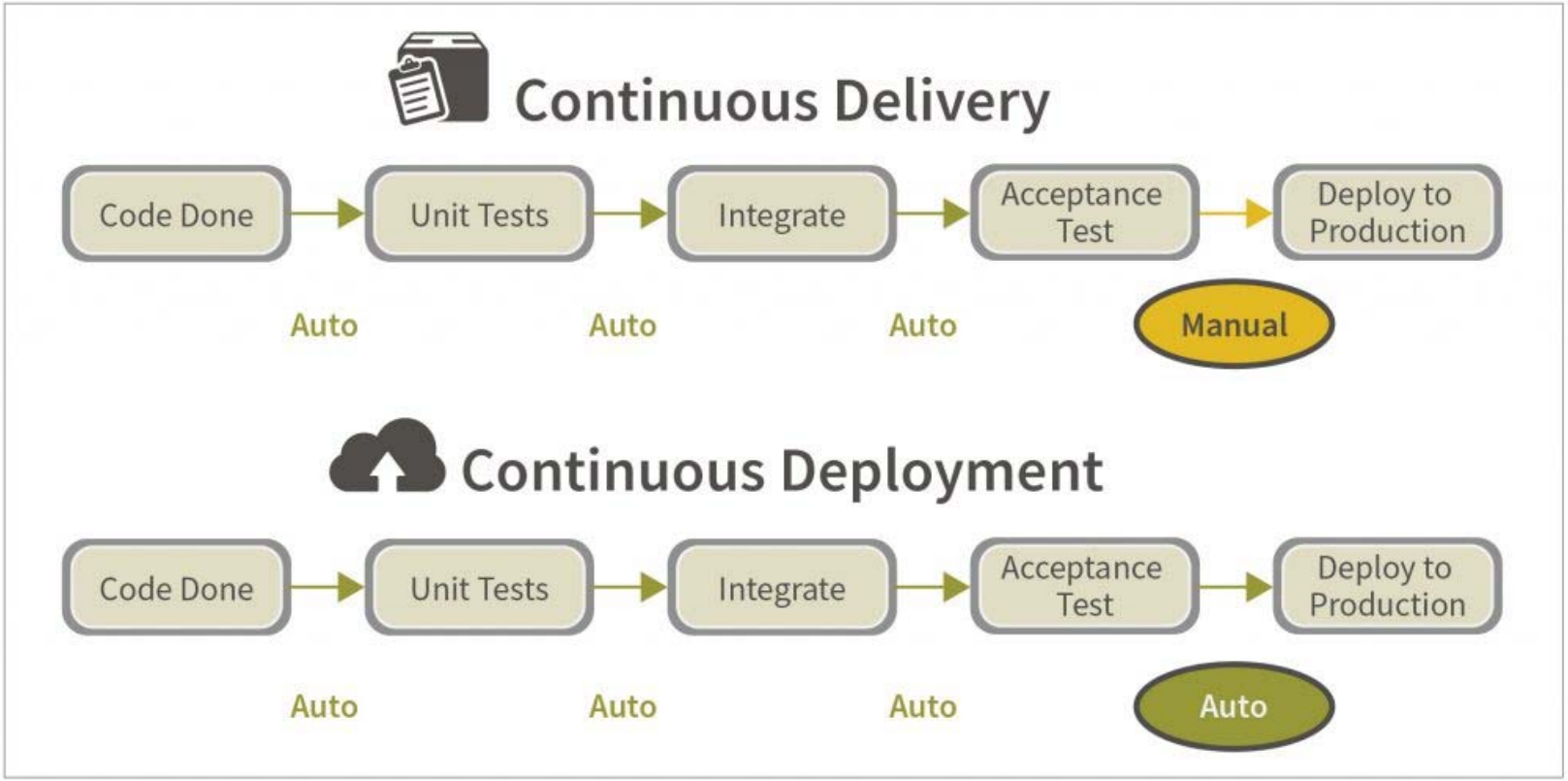
- If releasing software is painful, aim to release it every time somebody checks in a change that passes all the automated tests.
- If creating application documentation is painful, do it as you develop new features instead of leaving it to the end.
- Make documentation for a feature part of the definition of done, and automate the process as far as possible.

- In short,
- 1.Reduce lead time for changes
- 2.Increase Release Frequency
- 3.Reduce time to restore service

DEPLOYMENTS AT AMAZON.COM

11.6s	1,079	10,000	30,000
Mean time between deployments (weekday)	Max number of deployments in a single hour	Mean number of hosts simultaneously receiving a deployment	Max number of hosts simultaneously receiving a deployment

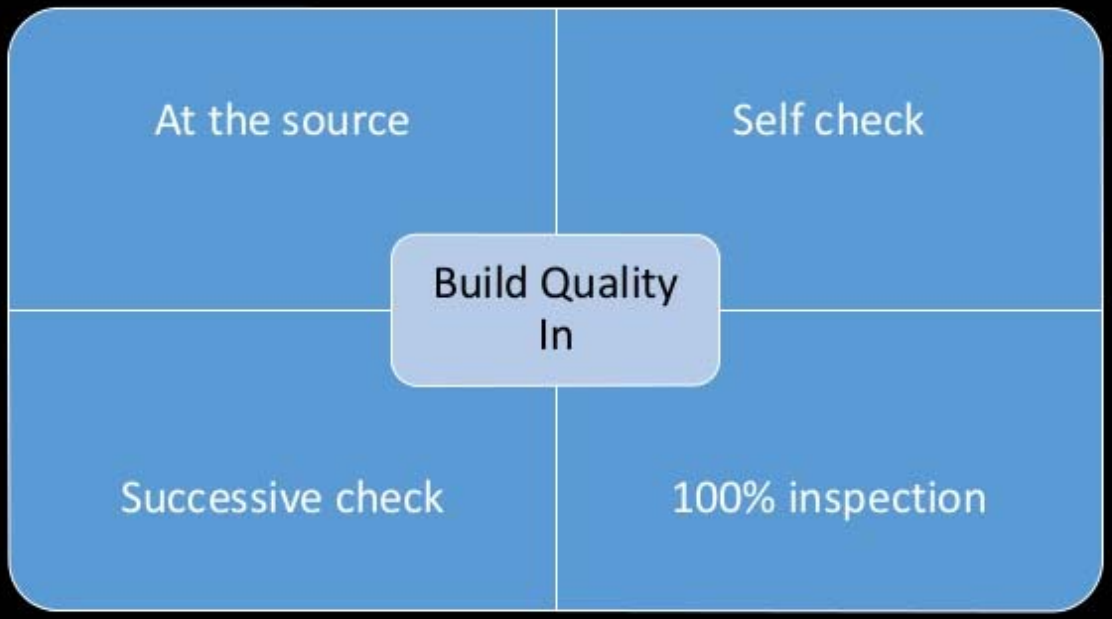
Continuous Delivery v/s Continuous Deployment

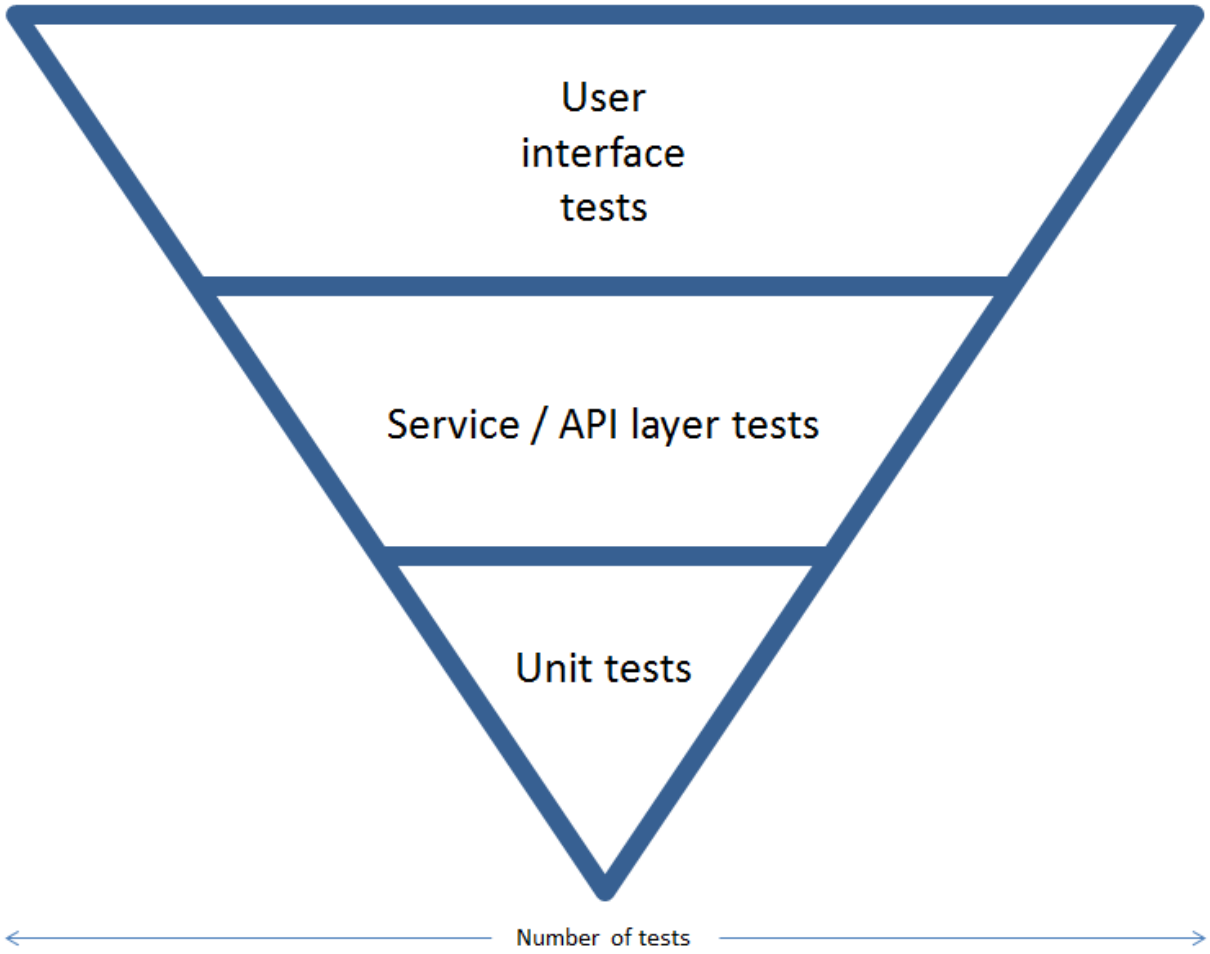


Build Quality In

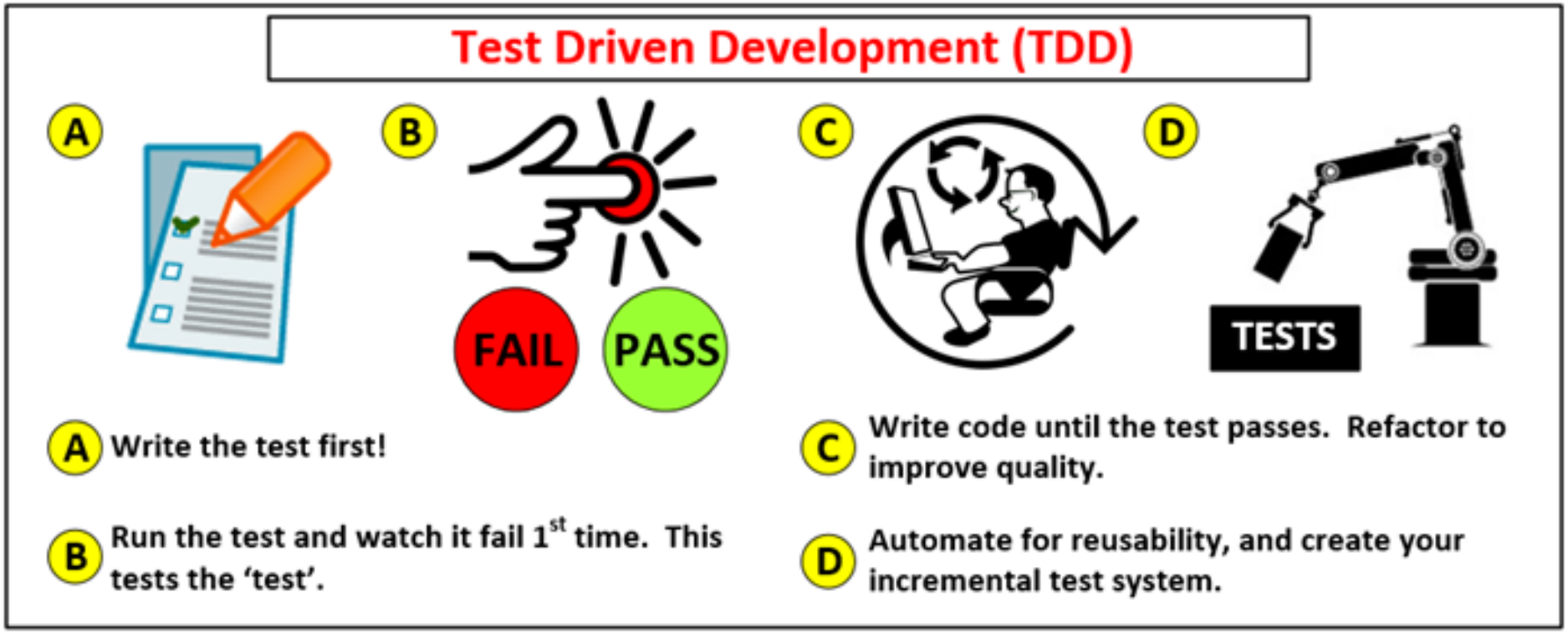
- The earlier you catch defects, the cheaper they are to fix.
- There are two other corollaries of “Build quality in.”
- Firstly, testing is not a phase, and certainly not one to begin after the development phase.
- If testing is left to the end, it will be too late. There will be no time to fix the defects.
- Secondly, testing is also not the domain, purely or even principally, of testers.
- Everybody on the delivery team is responsible for the quality of the application all the time

TDD fits "*Build Quality In*"




User
interface
tests
Service / API layer tests
Unit tests
Number of tests

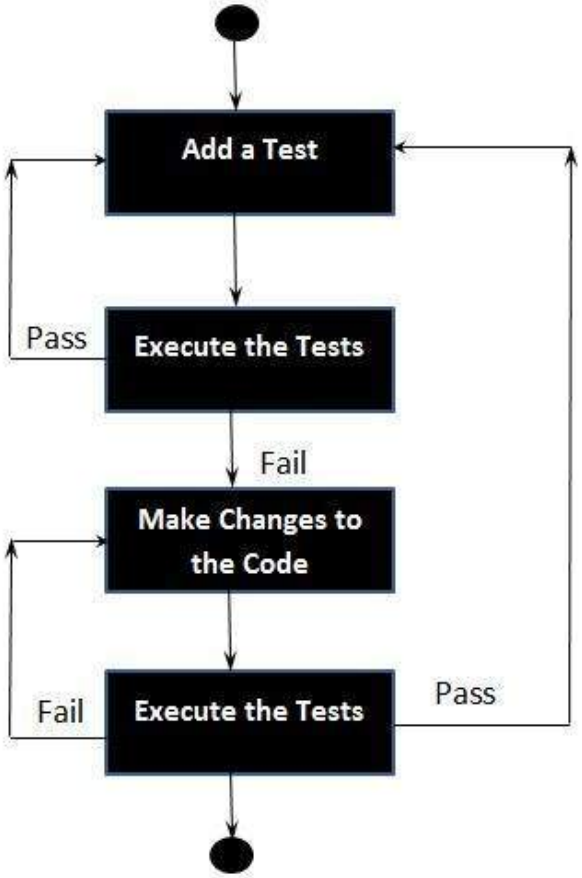
Test Driven Development



Test Driven Development

- Test-driven development starts with developing test for each one of the features. The test might fail as the tests are developed even before the development. Development team then develops and refactors the code to pass the test.

- **Test-Driven Development Process:**
- Add a Test
- Run all tests and see if the new one fails
- Write some code
- Run tests and Refactor code
- Repeat



Pass. Development Stops

Done Means Released

- A feature is only done when it is delivering
- value to users.
- There is no “80% done.” Things are either done, or they are not.
- It is possible to estimate the work remaining before something is done—but those will only ever be estimates

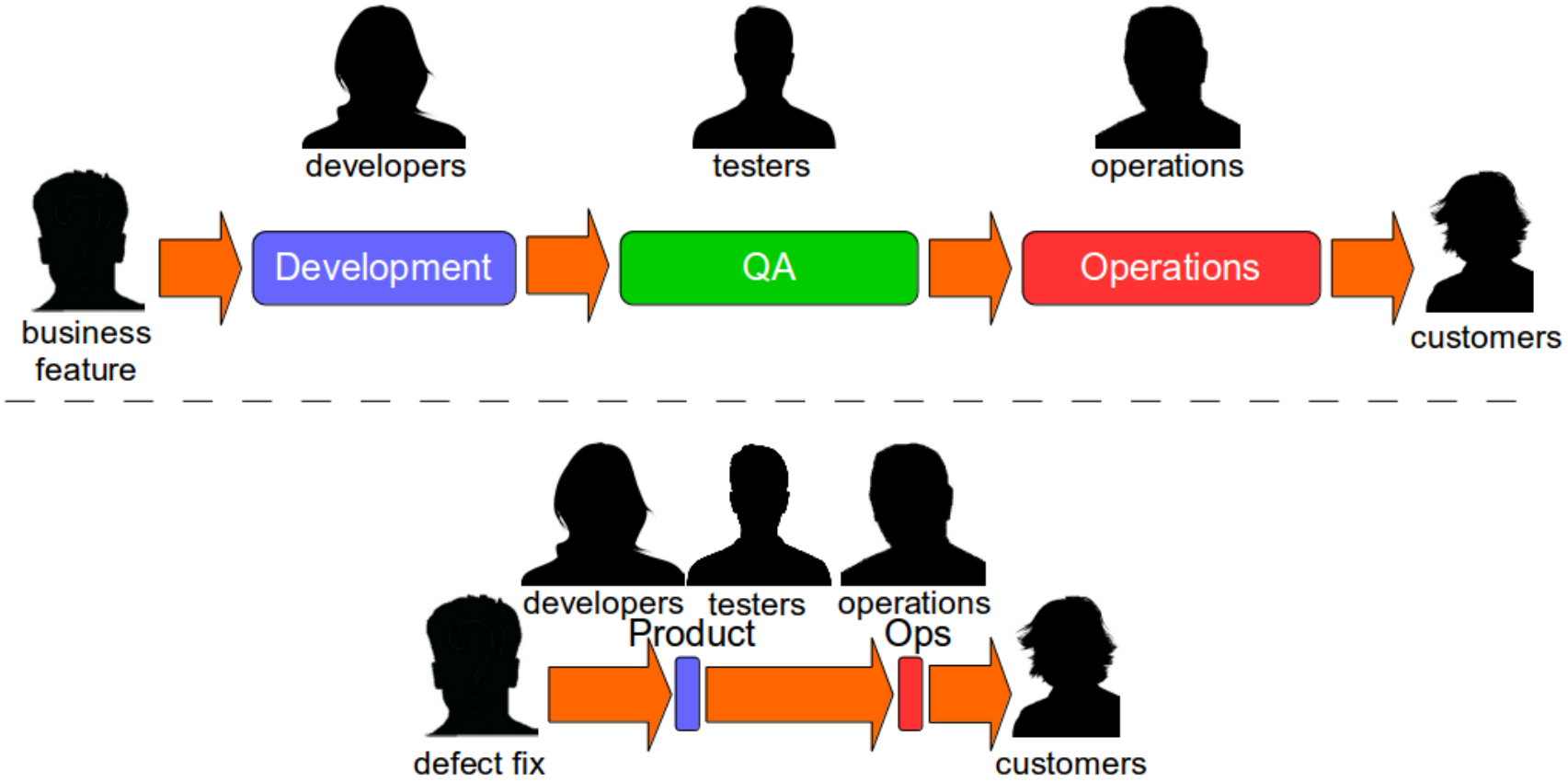
Done means released

Unreleased code is inventory and inventory is waste

Everybody Is Responsible for the Delivery Process

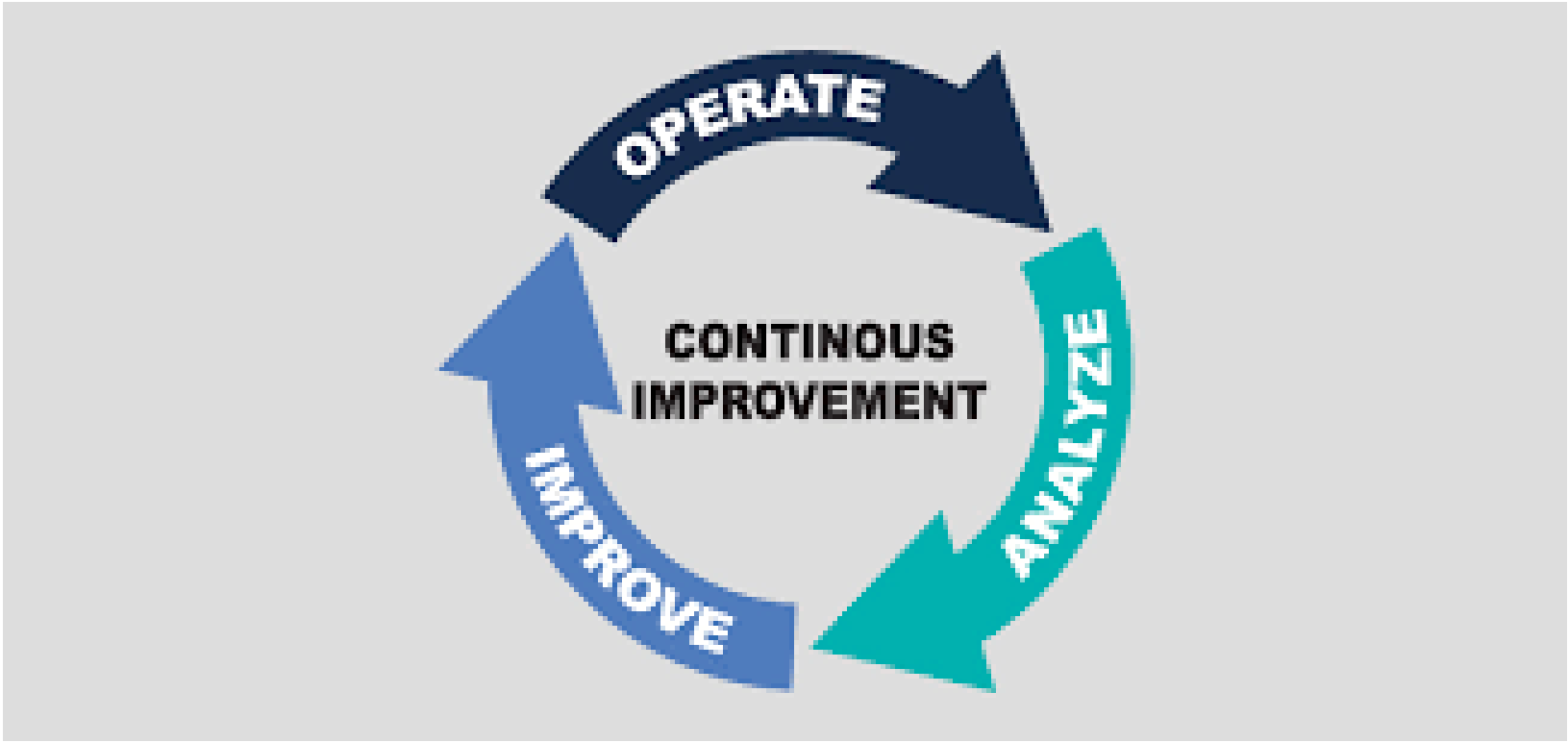
- Ideally, everybody within an organization is aligned with its goals, and people work together to help each to meet them. Ultimately the team succeeds or fails as a team, not as individuals.

Everyone is responsible !!!



Continuous Improvement

- It is worth emphasizing that the first release of an application is just the first stage in its life. All applications evolve, and more releases will follow.
- It is important that your delivery process also evolves with it.


OPERATE
ANALYZE
IMPROVE
CONTINOUS
IMPROVEMENT