Effective Lean Systems Delivery

Harish Narayan

Presentation Flow

- About me
- What I hope to convey
- Some assumptions and goals
- Cornerstones of lean systems approach
- Discussion on each cornerstone
- Some key delivery principles
- Conclusion

Some Assumptions

- Business would typically leverage online channels to generate revenue
- Iterative / Agile development cycle would be the preferred methodology (as a start)
- Software could be released to production, as often as possible.

Goal of this approach

Maximize delivery of new business value

- Keep changes discrete
- Eliminate batch type processing of work, which tends to create lead times and WIP
- Deliver to production as soon as Quality Release Candidate (QRC) is available.

Leveraging lean systems development and testing to maximize business value…

What does it mean to you?

To me…

- It is to…

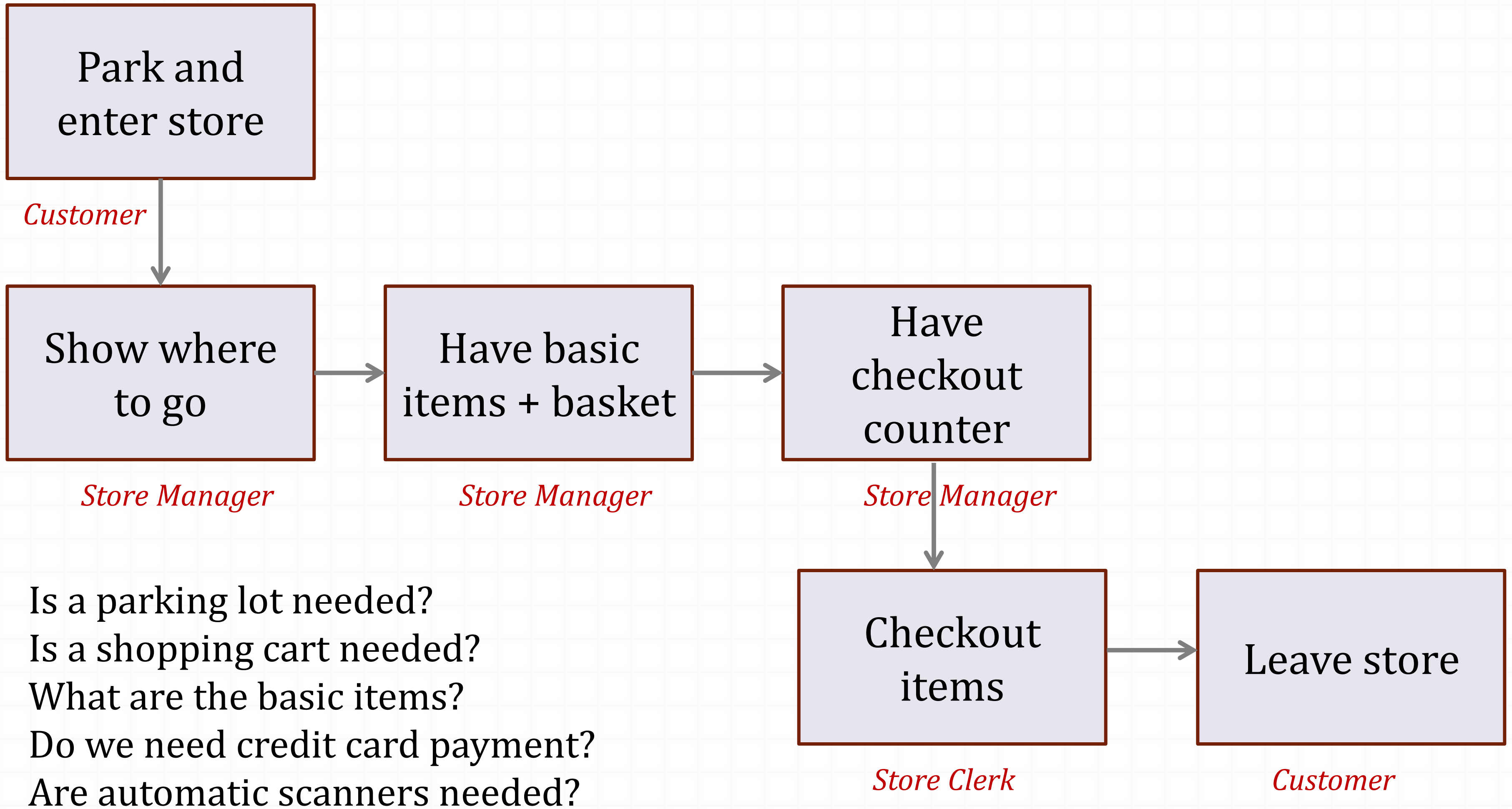
Leverage principles of lean, and strive for iterative but continuous delivery

- 4 cornerstones
  - Value stream orientation
  - Behavior Driven Development (BDD) / TDD
  - Continuous integration
  - High degree of test automation, and automated deployment.

What is a Value Stream?

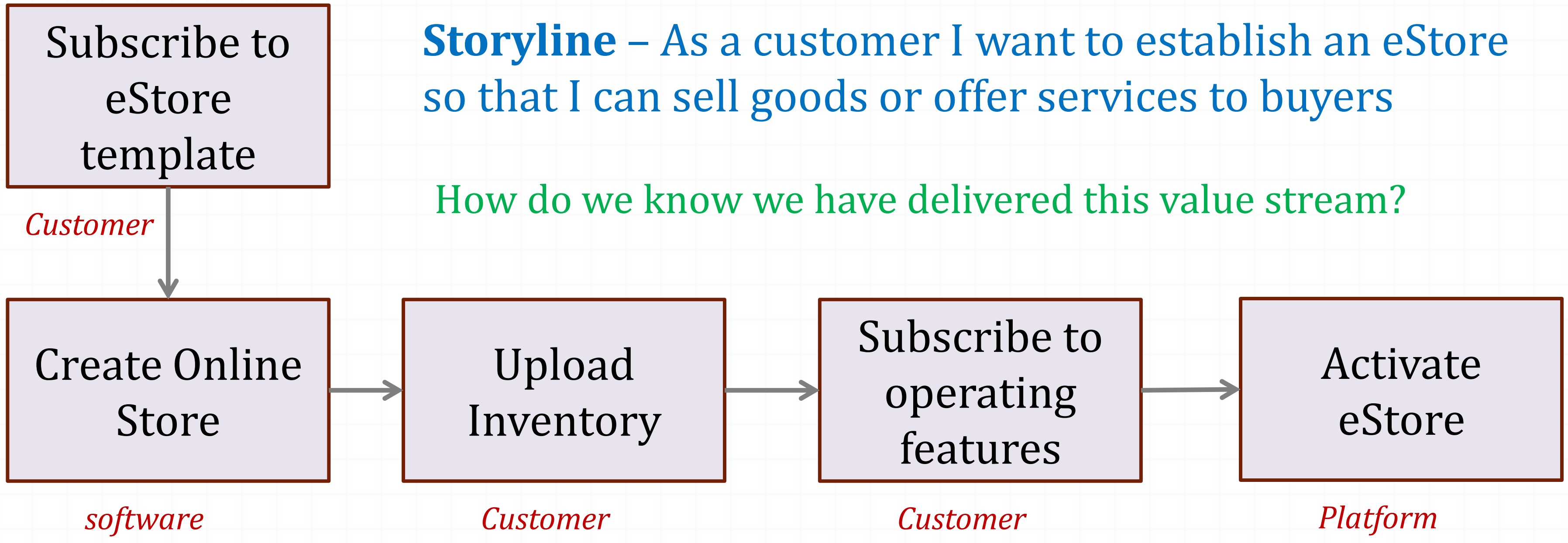
- It is a series of steps (devoid of any complexity), mostly value-added, required to take a product or service to any customer
- It can usually be identified where business repetition takes place. Ex:
  - PO through an acquisition process
  - ***Customer purchasing basic groceries in a grocery store***
- Useful for
  - Establishing a business ‘storyline’ with ‘boundaries’
  - Identifying features (or core business flows) for MVP & beyond
  - Estimating and budgeting development and testing
  - Prioritizing what to work on, and when to work on it
  - Validating features, estimates and priorities.

Ex: Grocery Store Value Stream



Is a parking lot needed?
Is a shopping cart needed?
What are the basic items?
Do we need credit card payment?
Are automatic scanners needed?
Is a store clerk needed?

Ex: eStore Value Stream



Storyline – As a customer I want to establish an eStore so that I can sell goods or offer services to buyers

How do we know we have delivered this value stream?

Can eStore be customized? Should it be customized?
What type of formats are supported for inventory upload?
What operating features are available? Needed in the first iteration?
What level of security is provided for the eStore?

What is the role of 'quality'? What is the role of a 'tester'?

BDD

Behavior Driven Development – A simple way to model a system based on business domain - help permeate information directly to technical development and the code base. Uses…

- Expand on the scenarios that make up the value stream
- Help determine where I can start development, and frame progress and completeness
- Better assess what should I test and how much should I test
- Manage tests better based on behavior

Specified using a language called 'Gherkin' – with a set of key words (Given, When, Then)

As a (role)
I want this (feature)
To deliver this (value)

Ex: eStore BDD

Feature 1: Customer can create an online Store
Given *Customer is a registered member of 'my company'*
And *Is a Seller*
When *Customer subscribes to the eStore software*
Then *Software platform can create an online Store*

Feature 2: Customer has an active eStore to conduct business
Given *Customer has an eStore*
When *Customer uploads inventory*
And *Customer subscribes to operating platform features*
Then *Software platform can activate an operational eStore.*

```
public class CustomerIsAMember implements Given {
      public void CheckSomething (SoftwarePlatform platform) {
      …
      }
}
public class CustomerSubscribesToEStore implements Event {
      public void DoSomething (SoftwarePlatform platform) {
      …
      }
}
```

How can we test these features?
- As they are being developed?
- What techniques can be used?

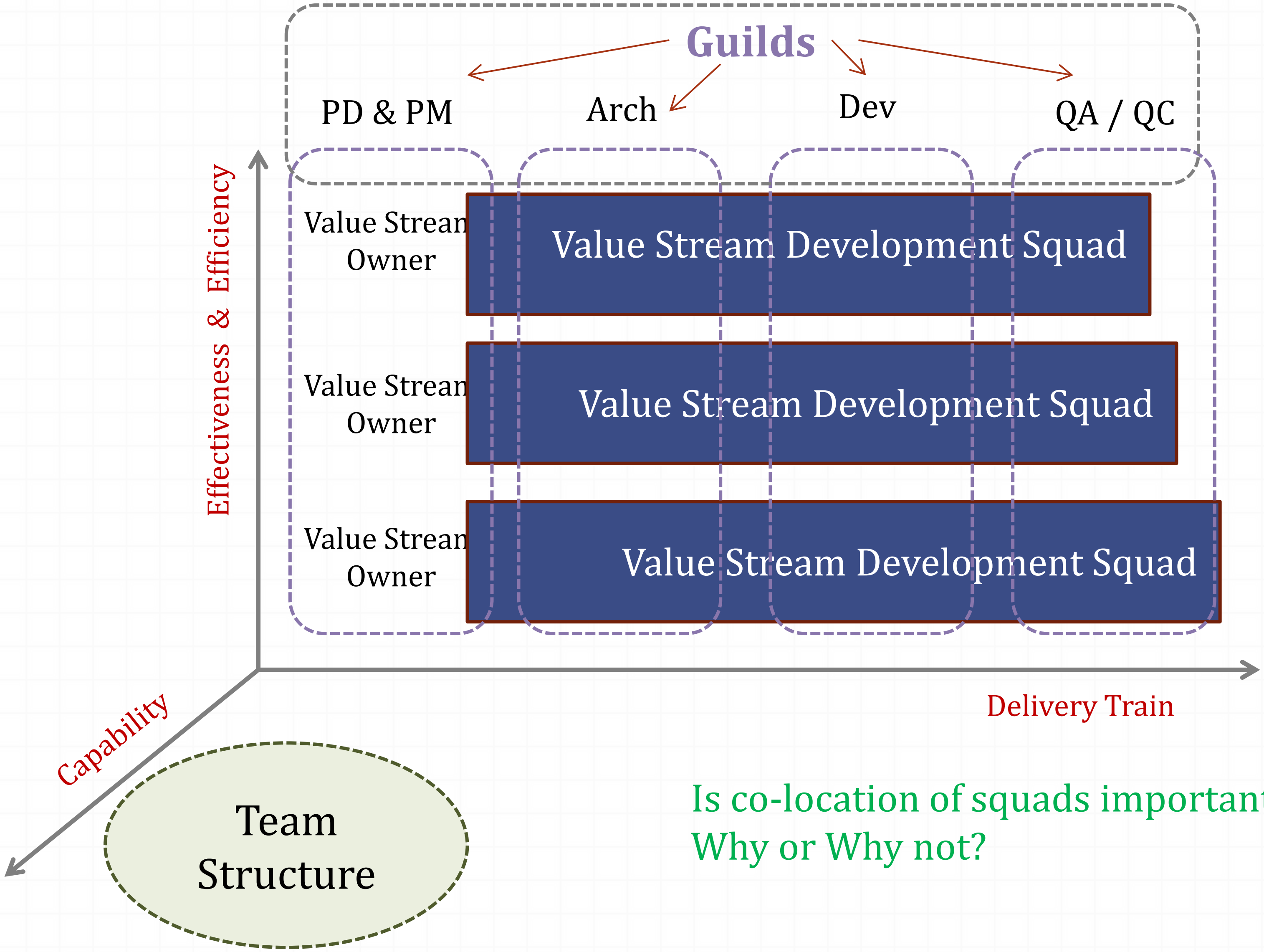
Ex: Elaborated Requirements

Requirements from the features elaborated…

1. *Customer (Seller) is provided an option to create an online store*
2. *The software platform shall provide*
   1. *templates to create an online estore*
   2. *a wizard to create an online estore*
3. *The software platform shall create an online estore based on information from template or wizard*
4. *The customer can upload inventory*
   1. *from a flat file*
   2. *from a comma separated (CSV) file*
5. *The customer can upload*
   1. *images to the online store*
   2. *videos to the online store*
6. *The customer can offer secure payments through the platform's payment gateway.*

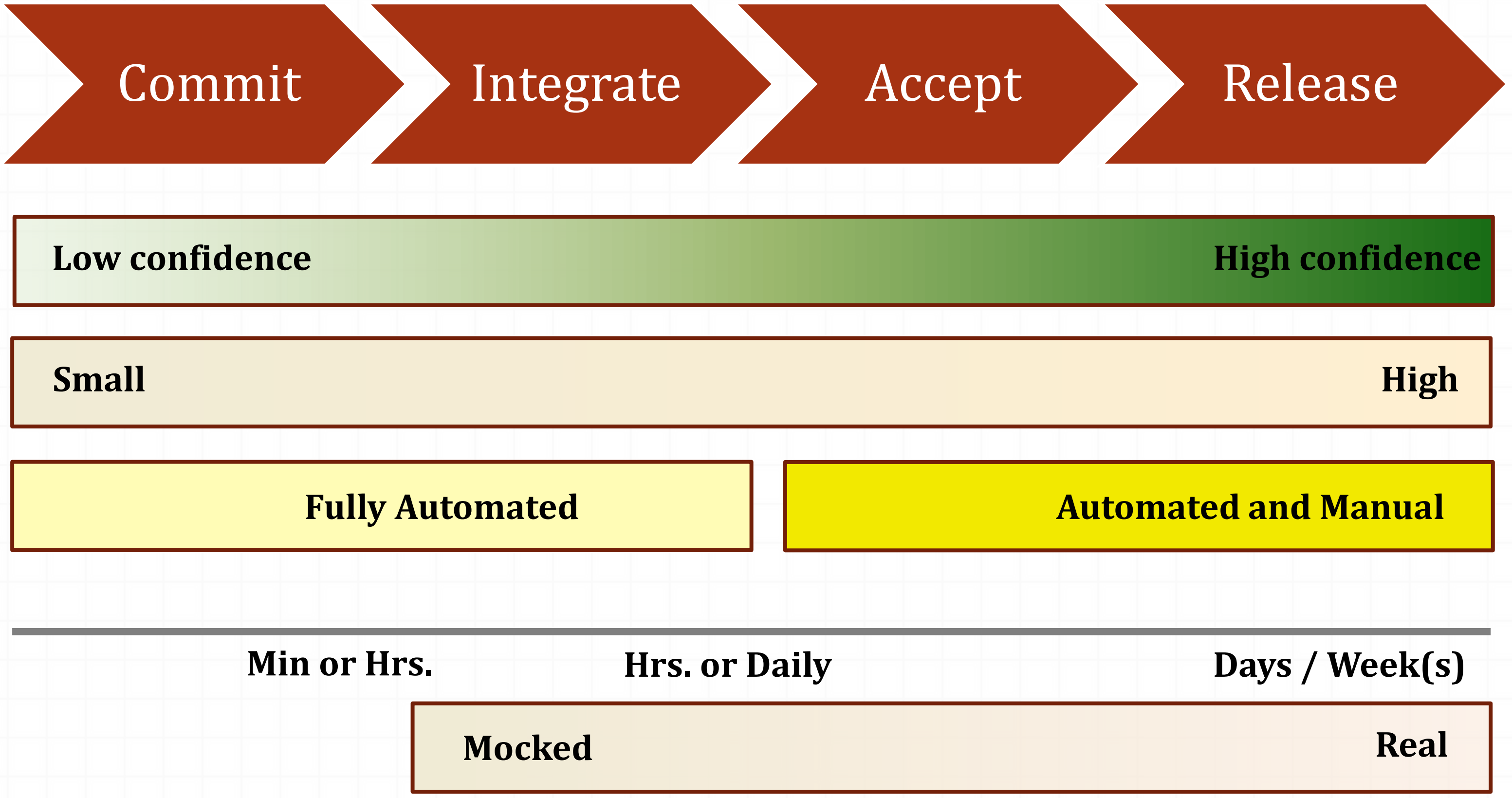
Which of these are needed for the first iteration?
How do we decide?

Teams – Squads - Guilds

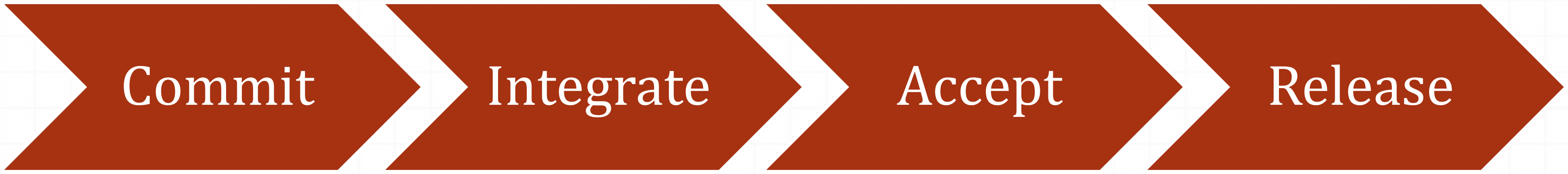


Is co-location of squads important?
Why or Why not?

Release Pipeline



Pipeline… Expanded



Commit	**Integrate**	**Accept**	**Release**
• Commit Code • Build and package binaries • Run unit tests • Store artifacts for later use	• Deploy and configure env. • Run smoke tests • Run automated integration tests	• Select candidates to be tested (functionality, usability, performance, etc.,) • Deploy and configure env. • Run smoke tests • Run manual and automated tests	• Select release candidate based on business need and confidence in quality • Deploy to production

Key Delivery Principles

- Build often (but only once)
  - Binaries are constructed at check-in time and the output (artifacts) stored
  - Later use of that version uses those artifacts and does not attempt to re-build
- Automate everything early – *else lead times increase*
  - All unit and integration testing
  - Deployment, configuration of environment, provisioning of infrastructure
- Maintain high fidelity – *else lead and debugging times increase*
  - Between environments – dev to integration to test to production

More Key Principles

- Enhance Visibility… *reduce lead times*
  - @Build stages - visible with error information
  - Deployment status providing view into different environments
  - Metrics around all levels of testing, including way to decide on confidence in quality
- Fix broken build above all… *reduce WIP*
- Fail fast
  - Minimize cost of defects and do not perform any value-added effort on defective version (stop testing for regression)
  - Perform environment checks and test validation before commit
- Version Version Version
  - Version everything that is not reproducible from something else.

Leverage tools to fit process

- Development
  - Project / Ticket Management
  - Defect Management
  - BDD
- Unit testing and test automation
- Source and version control
- Build
- Continuous integration and deployment
- Environment configuration and infrastructure provisioning
- Other Testing
  - Functional
  - Performance
  - Test Management

In Conclusion

Lean systems development and testing enables

- Organizations to make critical decisions about incrementally delivering of business value
- Reduction in process risk (waste and lead times), which tends to reduce cost of delivery
- Continuous delivery of features to production, significantly improving time-to-market.