



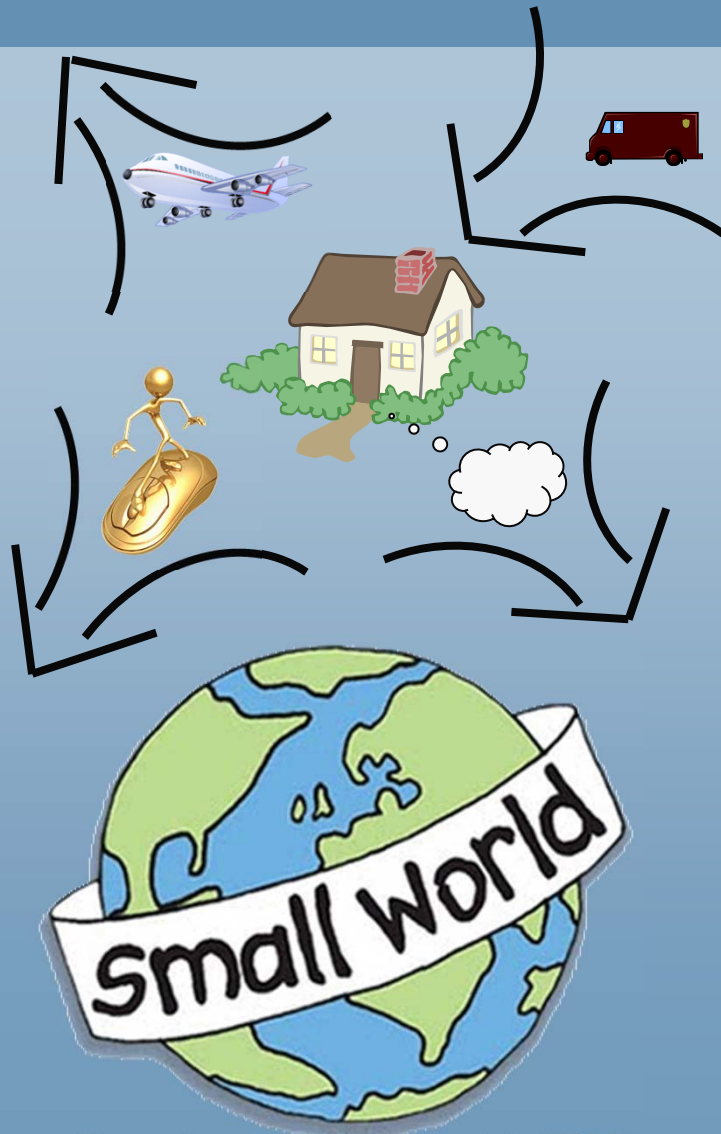
l e a n

software development

The Pace of Delivery

The Fastest Learner Wins

The Shrinking World



“The fact is that because of the cloud, today a young upstart can take market share without an incumbent having time to react.” – Werner Vogles, Amazon.com



The Learning Cycle:

Make a little.

Sell a little.

Learn a little.

How do we Learn?



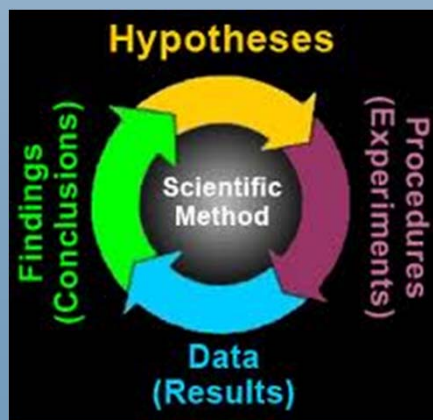
If you aren't making mistakes, you aren't learning.

Cooks' Illustrated

Old-Fashioned Chocolate Layer Cake



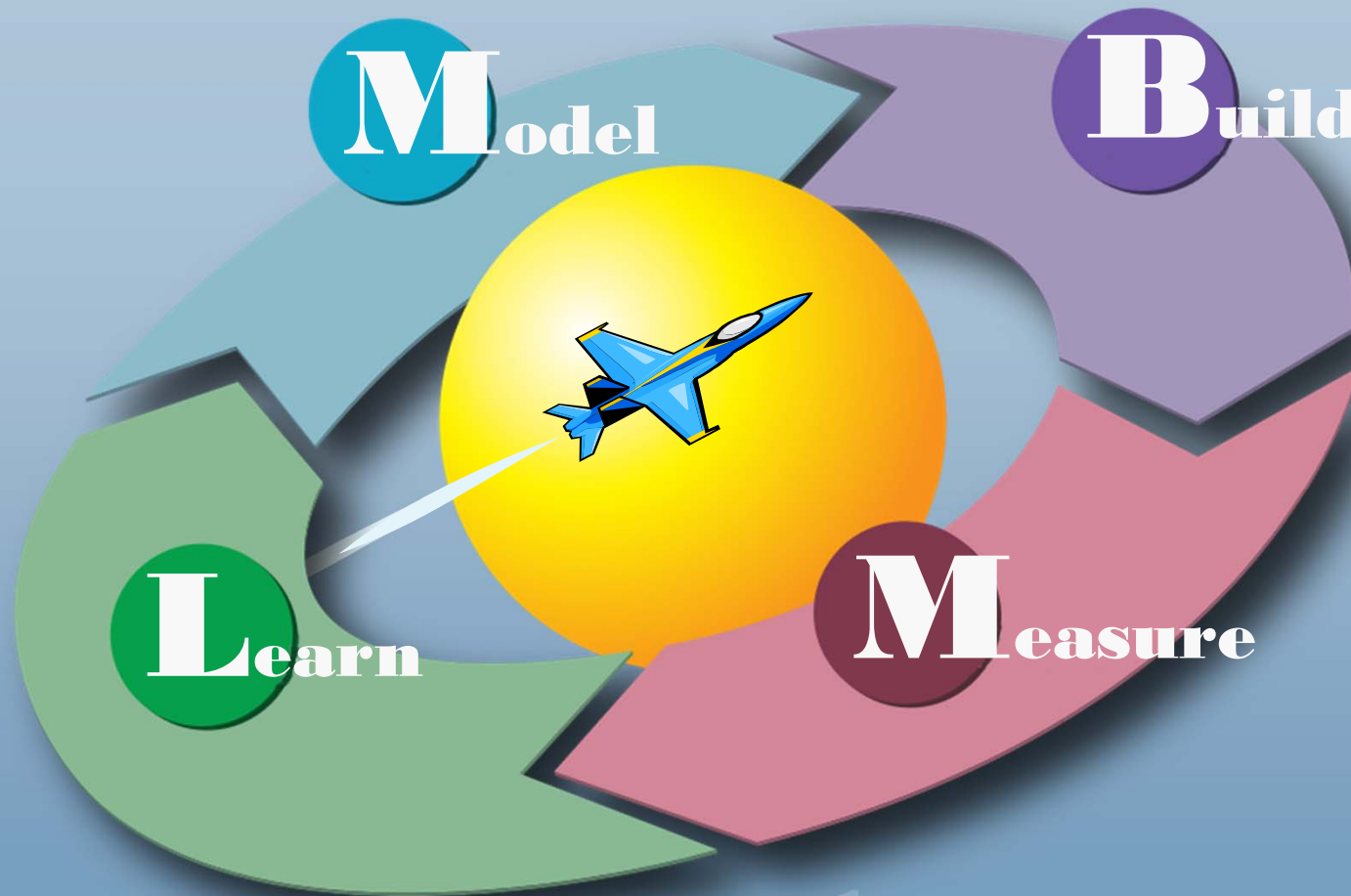
“We baked 130 cakes in search of the perfect wedge.”



Learning Requires:

- ✓ Hypothesis
- ✓ Experimentation
- ✓ Do it wrong lots of times
- ✓ Keep track of learning

The Fastest Learner Wins



Fast Companies



Deploys gmail 2 times/week.



Deploys major releases weekly,
minor releases daily.

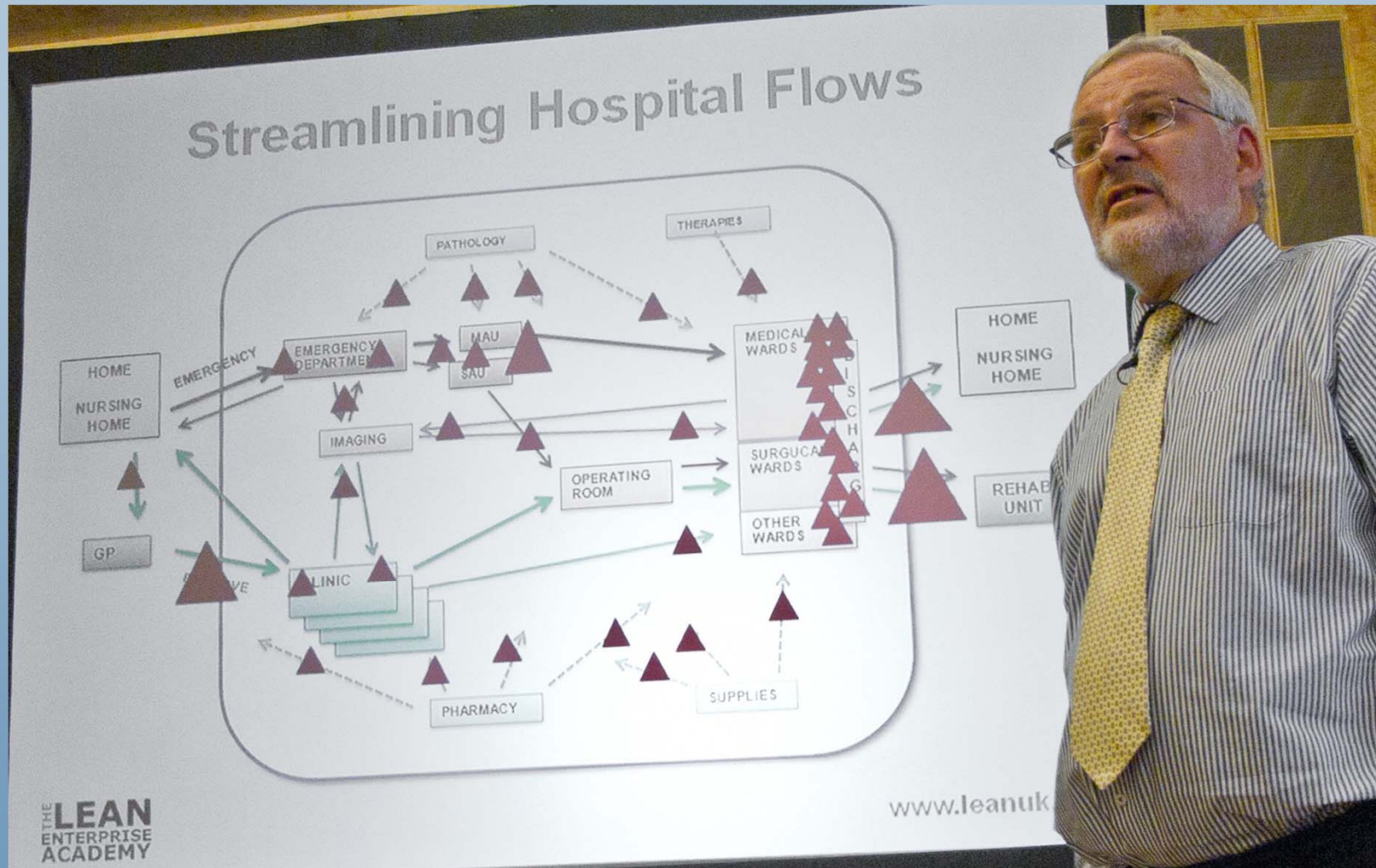


Two-Pizza teams deploy services
independent of other teams



App Store – A whole new world.

The Bottleneck: Hospitals



The Bottleneck: Software

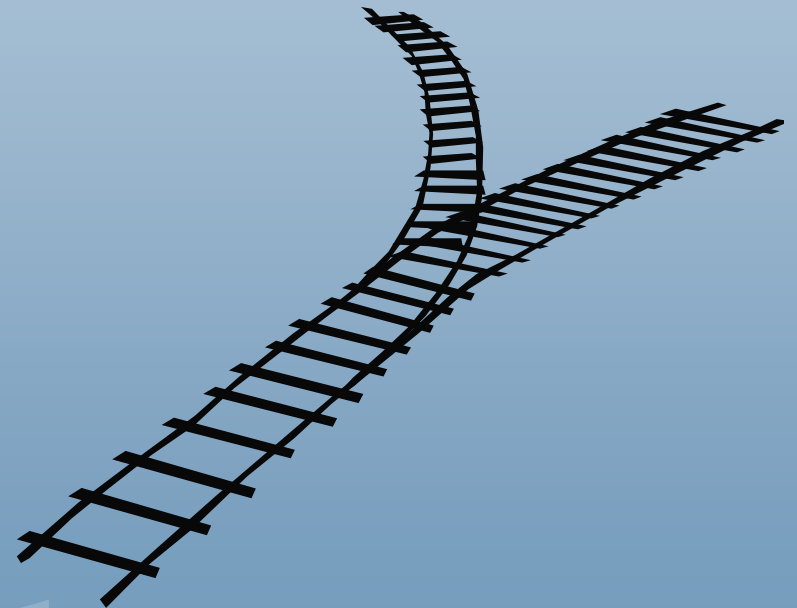


Big Bang System Integration



How can this be a good idea?

The longer two branches are apart, the harder they are to merge.



So why do we wait?

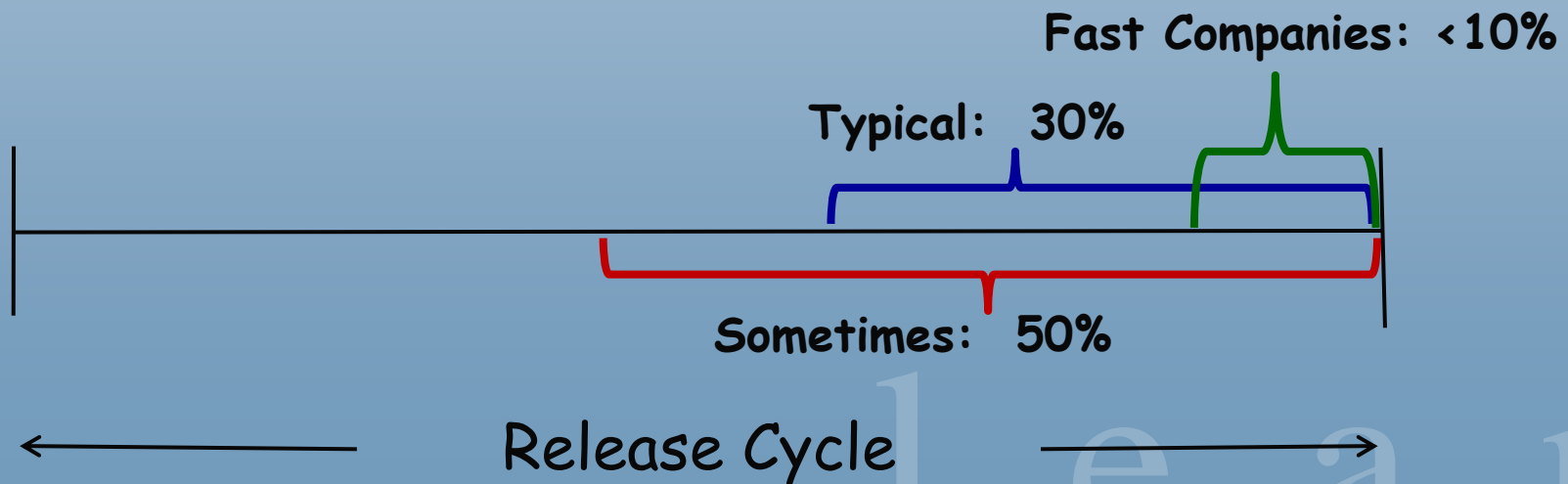
Build Quality In



Every software development process ever invented has had the same primary goal – find and fix defects as early in the development process as possible. If you are finding defects at the end of the development process – your process is not working for you.

How good are you?

When in your release cycle do you try to freeze code and test the system?
What percent of the release cycle remains for this “hardening”?



Build the Right Thing



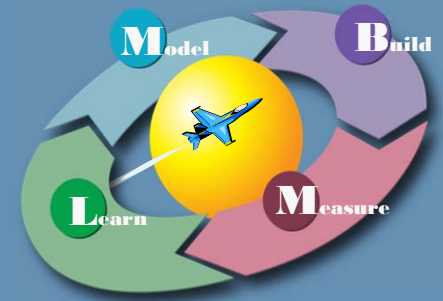
When in the release cycle do you choose the features for the release?



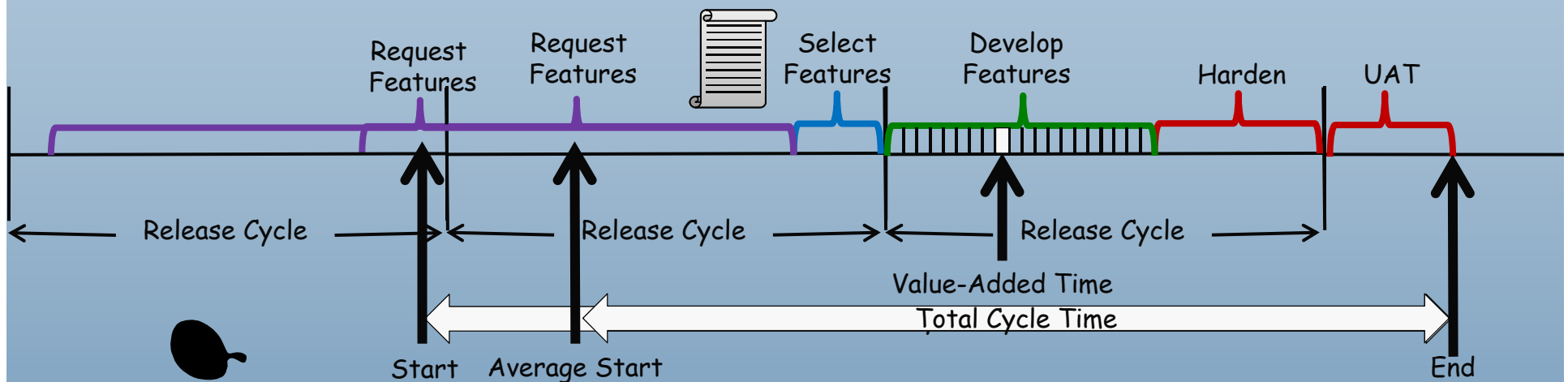
Governance: Project Metrics

The plan is always right,
even though it was made when you had the least information.

Release Cycle ≥ 6 Months



Quick & Dirty Value Stream Map:



Development Model:

- × Releases are very painful
- × Avoid releases!

It is NOT okay to find defects during Integration!



*If you find defects at the end of your process
– your process is defective!*

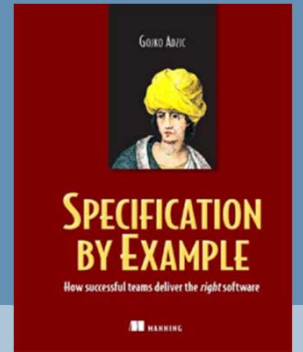
Getting Started: Contract Tests

Each team writes tests for any software that might use their service.

Other teams pull tests into their environment to find integration issues in advance.



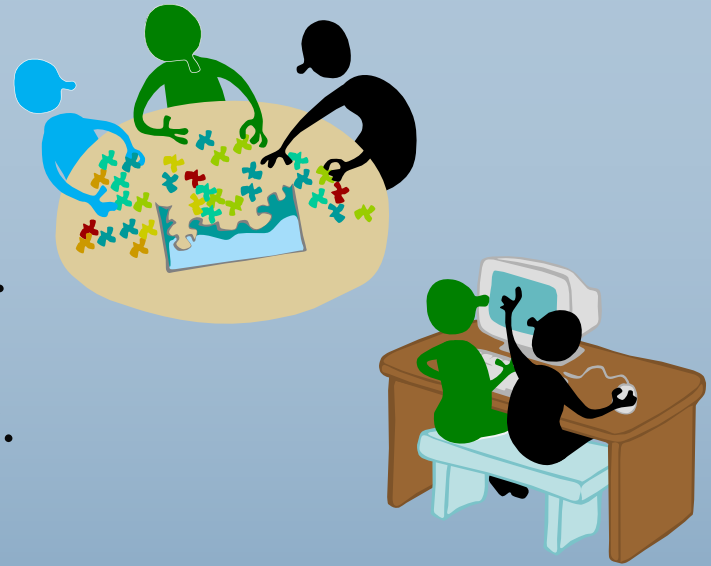
Specification by Example



For each feature:

1. Design

- a. Specify: Discuss and agree on examples of intended behavior.
- b. Automate: Put the examples in a framework such as cucumber.



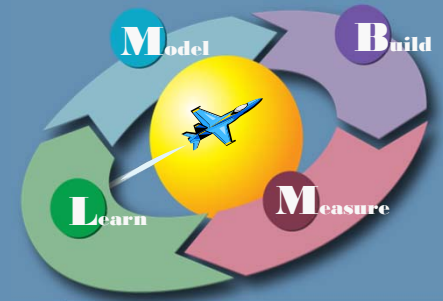
2. Implement

- a. Develop: Write feature code using TDD.
- b. Refactor: Clean up the code to keep it simple.
- c. Regression: Check that all completed examples work.

3. Deploy

- a. Feedback: Determine value for stakeholders ASAP.

Release Cycle Quarterly



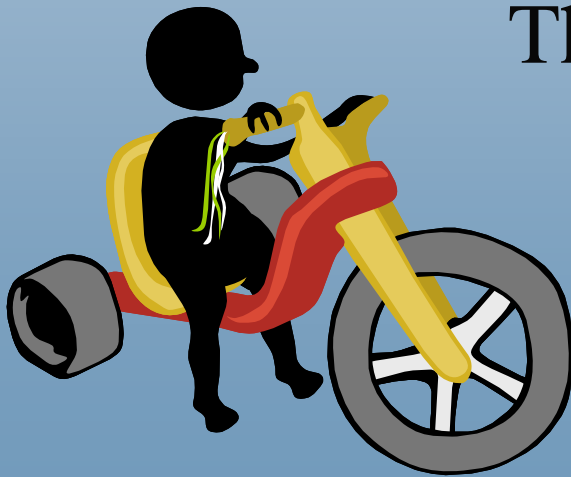
Hardening must be ≤ 2 weeks.

Typically: 2-4 week iterations

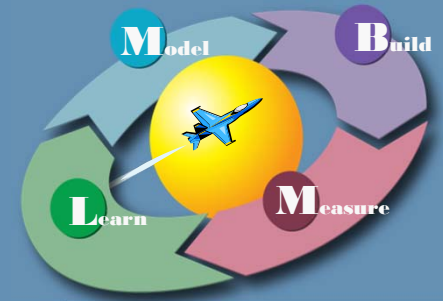
Code from each iteration goes to integration testing

Automated integration testing becomes necessary

The Big Bang becomes obsolete



Release Cycle Quarterly

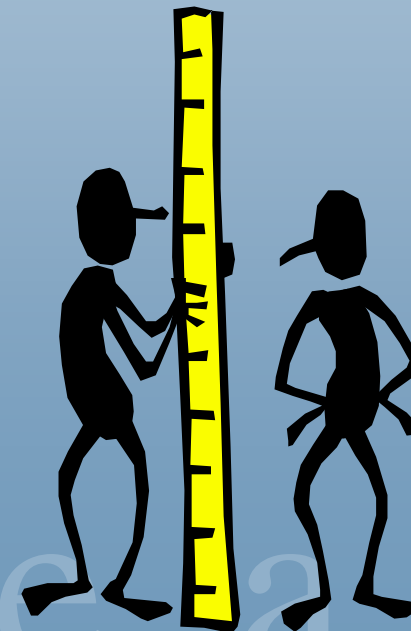


Change the Governance System:

- ✓ Teams stay with a system – there isn't time to change
- ✓ There isn't time for variance-based (project) metrics
- ✓ Success is measured based on customer satisfaction, revenue improvement, etc.

Business issues:

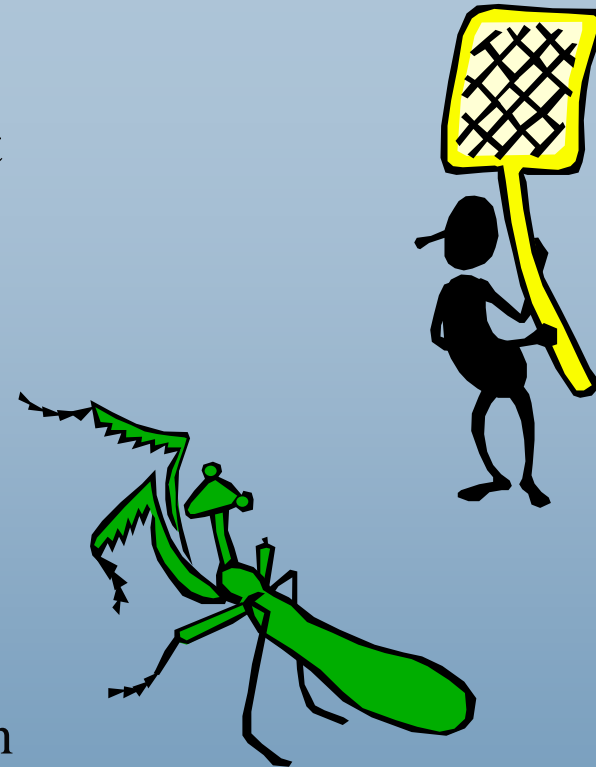
- ✓ How to price and sell releases?
- ✓ Which releases to support?
 - ✗ Supporting multiple branches can create a support nightmare



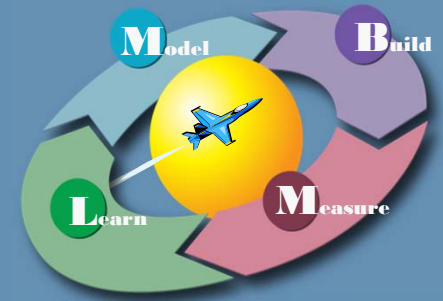
Basic Disciplines



1. Low Dependency Architecture
2. Coding Standards
3. Source Control / Configuration Mgmt
4. Automated Specification Examples
5. Automated Unit Tests
6. **STOP** if the tests don't pass
7. Design/Code Reviews
8. Refactoring is a Habit
9. Continuous Integration
10. System Testing / UAT – early & often
11. Escaped Defect Root Cause Analysis
12. Validation: Did we build the right thing?



Release Cycle Monthly



Now you need:

- ✓ Cross Functional Team
- ✓ Visualization
- ✓ Short Daily Meetings
- ✓ SBE/TDD working!
- ✓ Hardening ≤ 3 days

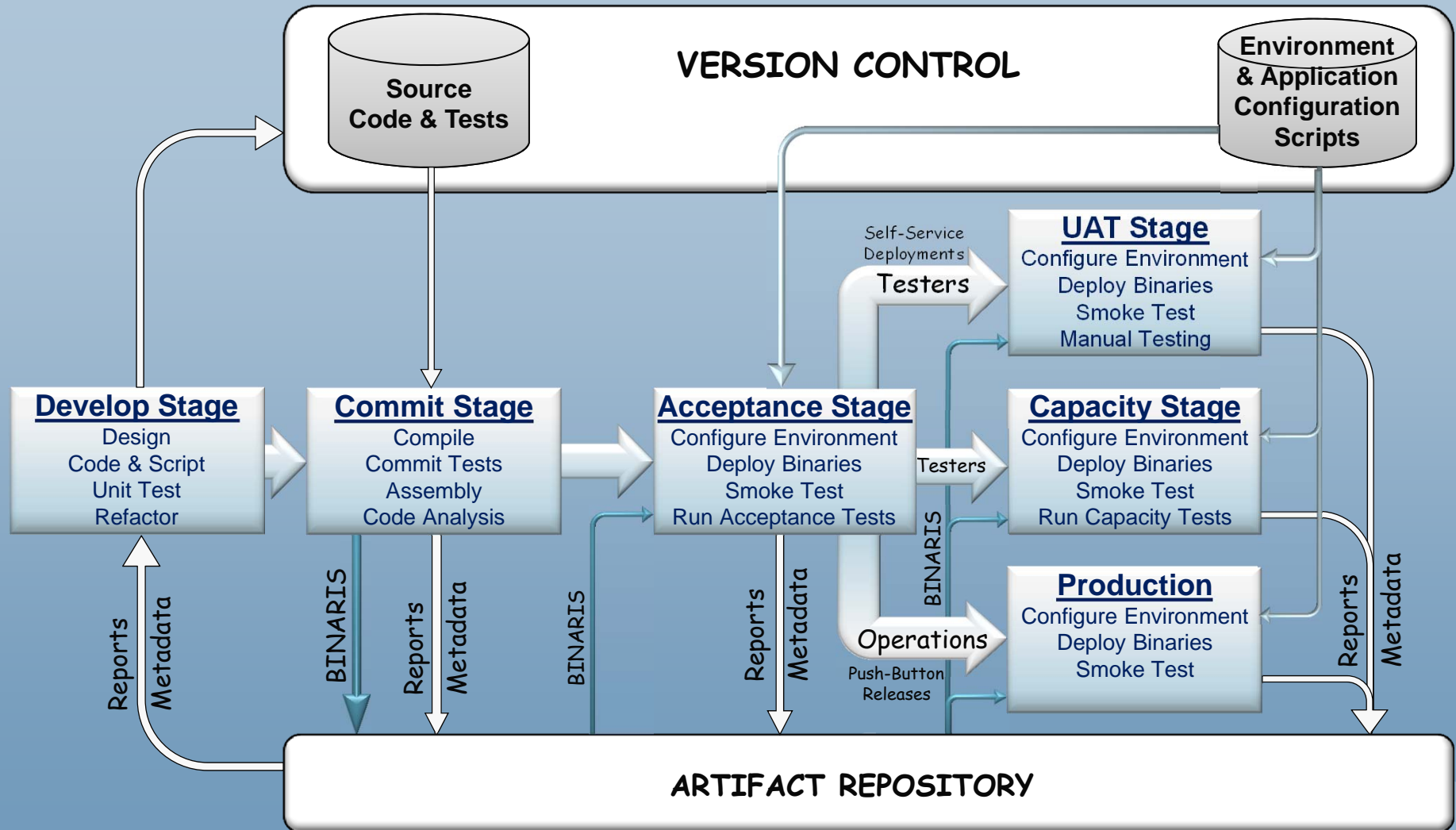


Business Environment
Works best for:

- ✓ Software as a Service (SaaS)
- ✓ Internal Software

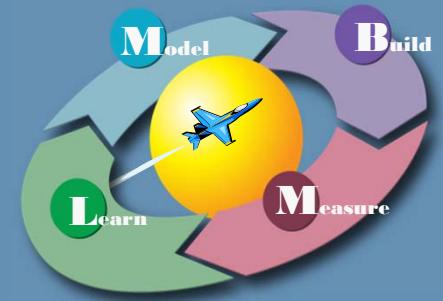


A Typical Deployment Pipeline



Release Cycle

Weekly/Daily/Continuous



Kanban works well

The team is everyone.

Iterations become irrelevant

High discipline is fundamental

Estimating is largely unnecessary

Rapid cycles of learning drive portfolio decisions



DevOps:

Test & deployment automation is essential

Business Issues:

Increasingly common in startups

The Lean Startup

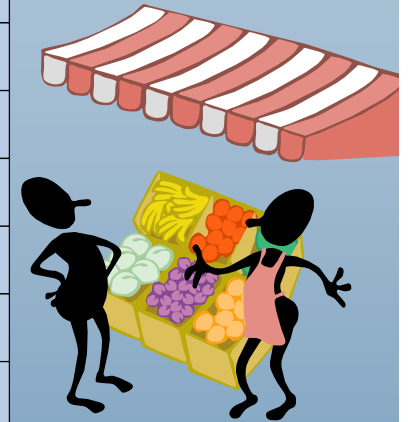


Agile Vs. Lean Startup

Adapted from similar chart posted by Joshua Kerievsky, Industrial Logic Blog† August, 2011



Agile	Lean Startup
Product Roadmap	Business Model Canvas
Product Vision	Product Market Fit
Release Plan	Minimal Viable Product
Iteration	Build-Measure-Learn Loop
Iteration Review	Persevere or Pivot
Backlog	“To Learn” List
User Story	Hypothesis
Continuous Integration	Continuous Deployment
Definition of Done	Validated Learning
Acceptance Test	Split Test
Customer Feedback	Cohort-based Metrics
On-Site Customer	“Get Out Of The Building”
Product Owner	Entrepreneur



†<https://elearning.industriallogic.com/gh/submit?Action=PageAction&album=blog2009&path=blog2009/2011/agileVsLeanStartup&devLanguage=Java>



l e a n

software development

Thank You!

More Information: www.poppendieck.com