Continuous Delivery

Ariel Alonso, IPC

About Me

- Ariel Alonso
- Systems Architect
- Independent Purchasing Cooperative, Inc.
- Software Engineer for 15 years
- Interests
  - Agile & XP
  - Test Driven Development
  - Automation
  - Testing
  - Distributed Systems
  - Big Data

Agenda

- What is Continuous Delivery?
- Implementing Continuous Delivery
- Continuous Delivery at IPC
- Q&A

What is Continuous Delivery?

How long would it take your organization to deploy a change [to production] that involves just one single line of code?

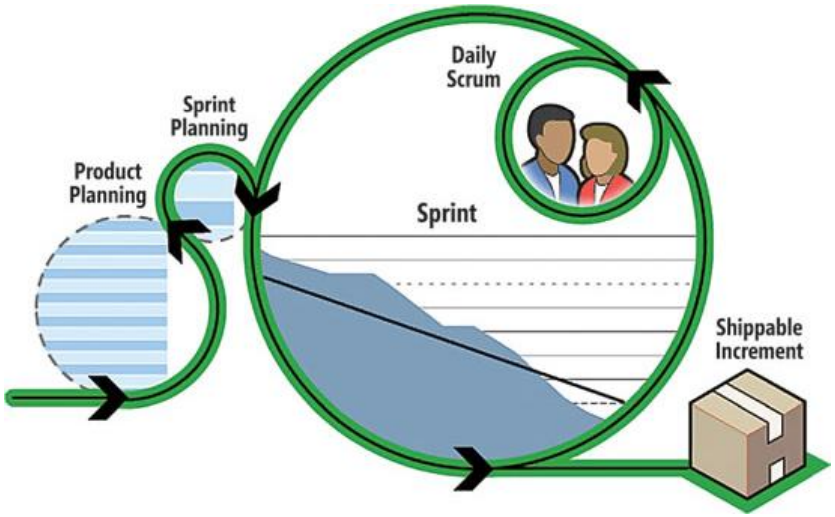
Do you do this on a repeatable, reliable basis?

- Mary and Tom Poppendieck, Implementing Lean Software Development


internal
architecture
management
refactoring
code
staging
formal
engineering
demo
specifications
edit
firewall
features
deployment
software
configuration
qa
uat
production
integration
documentation
requirements
implementation
release
bug
ccb
quality
cycle
external
analysis
development
maintenance
process
server
design
waterfall

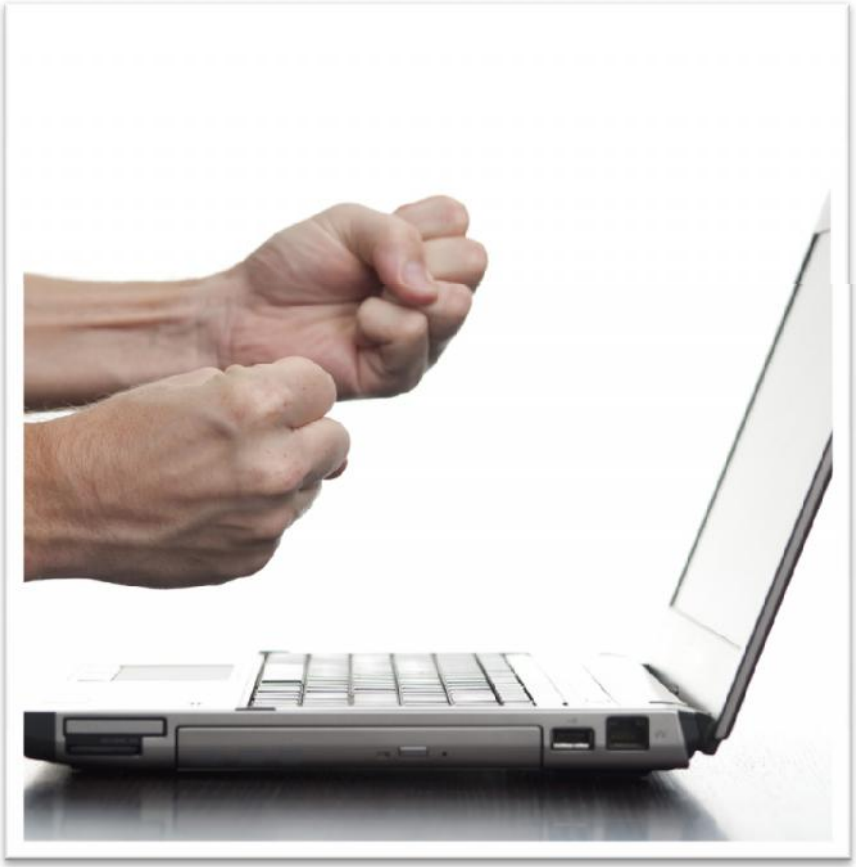
What Scrum Does For Us

- Smaller pieces of functionality
- More frequent feedback
- Shippable increments quicker



What About Deployments?

- Value is only realized in production
- Manual deployments are painful and error prone
- Poor configuration management leads to unpredictable results
- Infrequent deployments increase risk



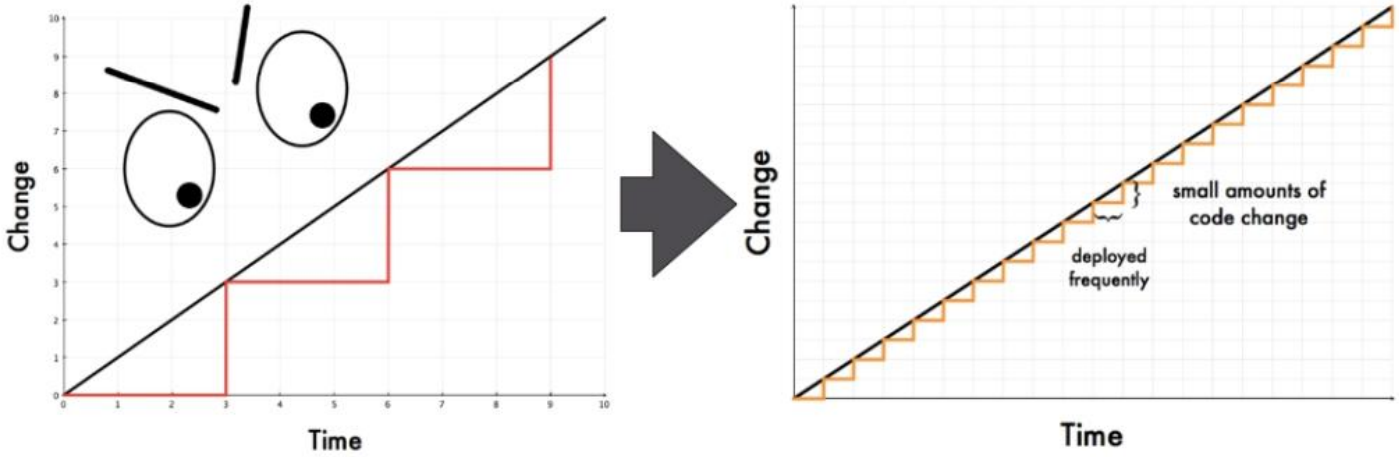
“If it hurts, do it more often.”

- Internet Wisdom

Frequent Deployments

- Deploying Frequently
  - Feedback from users
  - Reduce risk of deployment
  - Real project progress
  - Business value quicker
- More frequent changes with fewer lines of code reduce risk and help identify problems easier and sooner

Reduce Risk of Change



John Allspaw, Ops Meta-Metrics

What is Continuous Delivery?

- Set of software development practices and principles focused on building, testing, and deploying software faster and more frequently
- Goals
  - Improve Quality
  - Reduce Cycle Time
- Business value realized in production

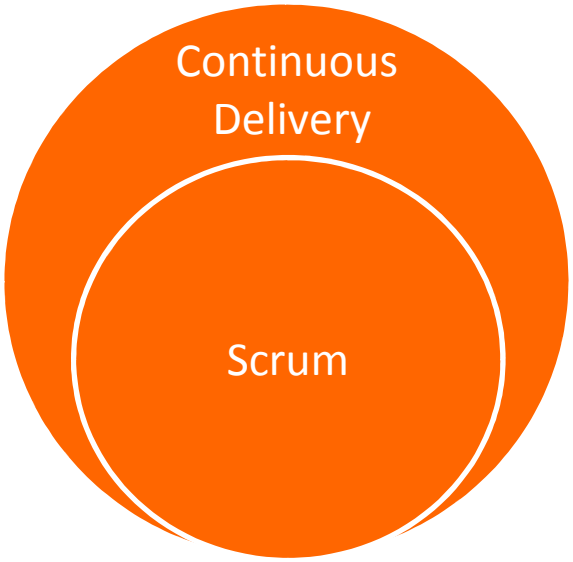
The Last Mile

Agile/Scrum

Gives us feedback on the quality of our requirements and the quality of our code

Continuous Delivery

Gives us feedback on the quality of our process to deliver software



“Just ship, baby.”

- Kent Beck

Implementing Continuous Delivery

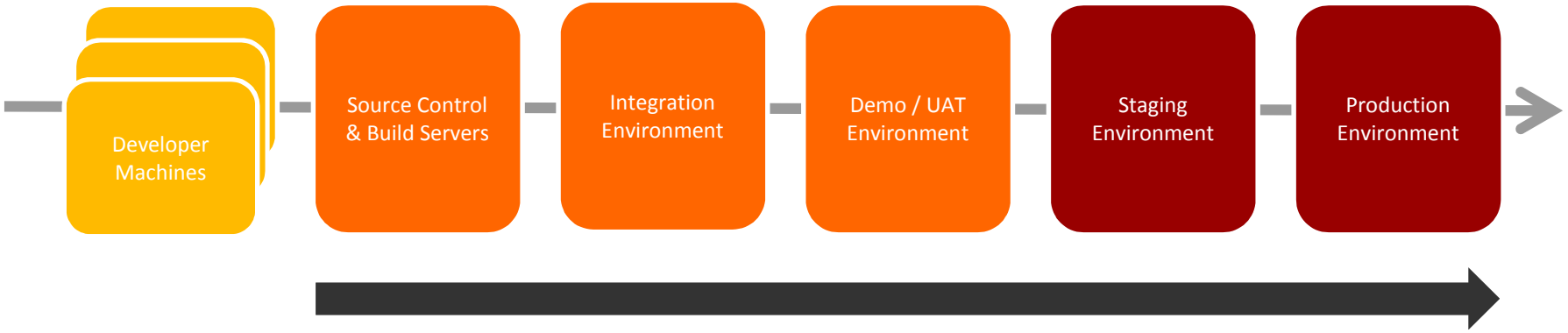
Principles of Continuous Delivery

- Process MUST be repeatable and reliable
- Automate everything
- If something is difficult or painful, do it more often
- Keep everything in source control
- Done means “in production”
- Build quality in to the product
- Everyone has responsibility for the release process
- Improve continuously

“Real artists ship.”

- Steve Jobs

Continuous Delivery Pattern



Single artifact is produced and advanced through the environments

Exactly the same mechanism is used to deploy to each environment

Continuous Delivery Practices

- Build binaries only once
- Use precisely the same mechanism to deploy to every environment
- Smoke test your deployment
- If anything fails, stop the line!

Key Components

- Source Control
- Test Driven Development
- Behavior Driven Development
- Automated Testing
- Continuous Integration
- Infrastructure as Code

Source Control

- Essential for development teams to work effectively on the same code base
- Ability to attach historical data to code, such as explanatory comments about the intent behind each change
- Control the “production line” with commit hooks

Continuous Integration

- Automatically builds entire application
- Automated unit and functional tests
- Exposes integration issues and conflicts early
- Everyone is responsible for the CI environment

Test Driven Development

- Red, Green, Refactor
- Permanent Regression
- Self Documentation
- More Maintainable Code

Behavior Driven Development

- Starts with a conversation
- Shared understanding of the features to be implemented
- Features are identified as user stories

```
Feature: Refund
  In order to make the customer whole
  As the product owner
  I want to make sure we know how to do math

  Scenario: Customer refund
    Given Fred has bought a microwave
      And the microwave cost 100 USD
    When we refund the microwave
    Then Fred should be refunded 100 USD
```

Automated Testing

- Unit Testing
- Functional Testing
- Performance Testing
- Stress Testing
- Security Testing
- Disaster Recovery Testing

Infrastructure as Code

- Automate the process of bringing the system to a working state
- Production like environments can be built from scratch in minutes
- Versioning environments along with applications
- Allows for continuous testing of our infrastructure

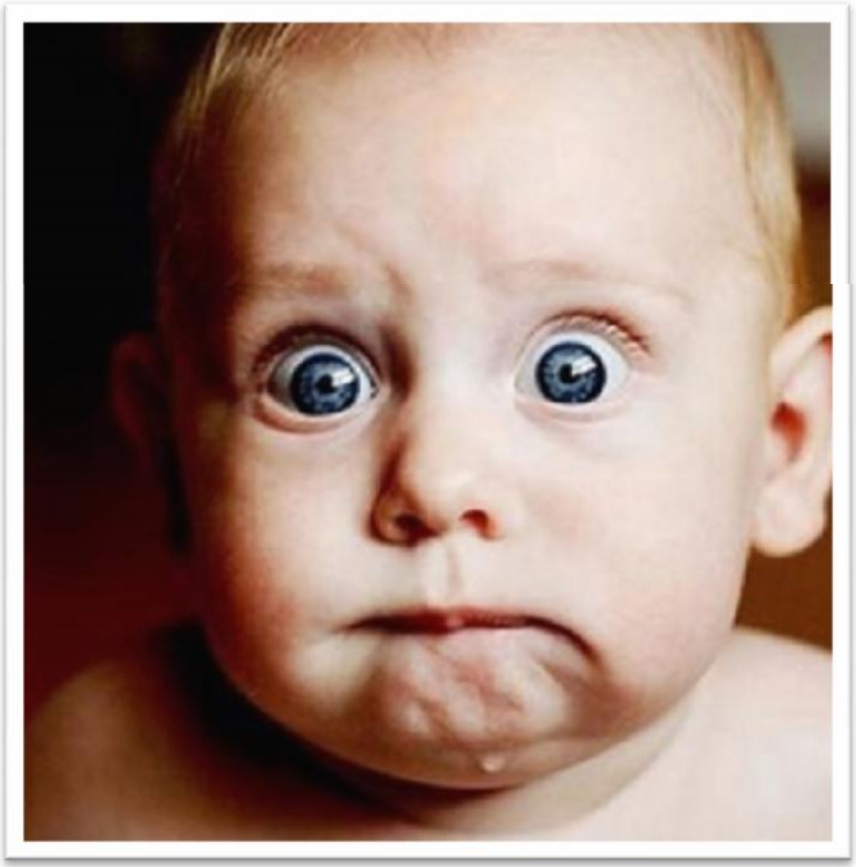
Tools

- Issue Tracking
  - JIRA, PivotalTracker, Bugzilla
- Source Control
  - Mercurial, Git, Subversion, CVS, TFS
- Virtualization
  - VMWare, OpenStack, Amazon EC2
- CI/Build Server
  - Bamboo, Hudson, Jenkins, TeamCity
- Automated Testing
  - jUnit, Nunit, RSpec, Cucumber, Selenium
- Infrastructure as Code
  - Puppet, Chef

Continuous Delivery at IPC

The Challenge

- Payment Processing Platform
- 28,000+ restaurants in US and Canada
- 2M transactions/day
- $5.8B/year in credit card sales
- 200M SUBWAY® Card accounts
- 99.999% Service Level Agreement
- **Release Weekly**

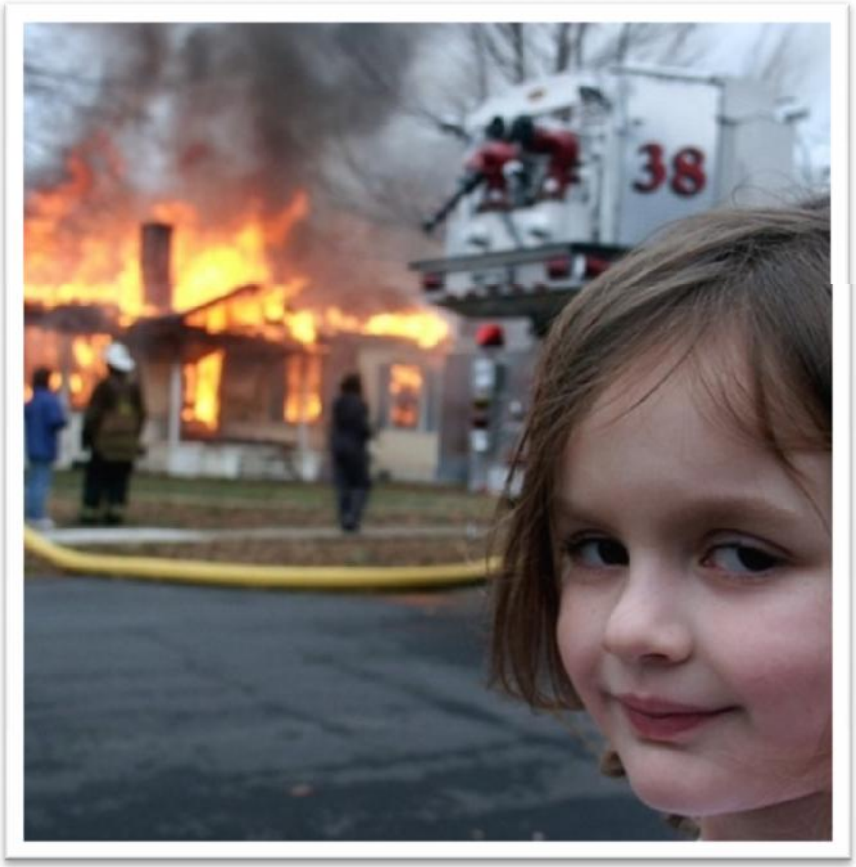


The Solution

- Transition from waterfall to Scrum
- Weekly sprints resulting in weekly production deployments
- Embrace DevOps and Continuous Delivery
- Invest in comprehensive testing approach

How Much Do We Test?

- 3,600+ Unit Tests
- 4,500+ Functional Tests
- 7,000+ Regression Tests
- Disaster Recovery Tests



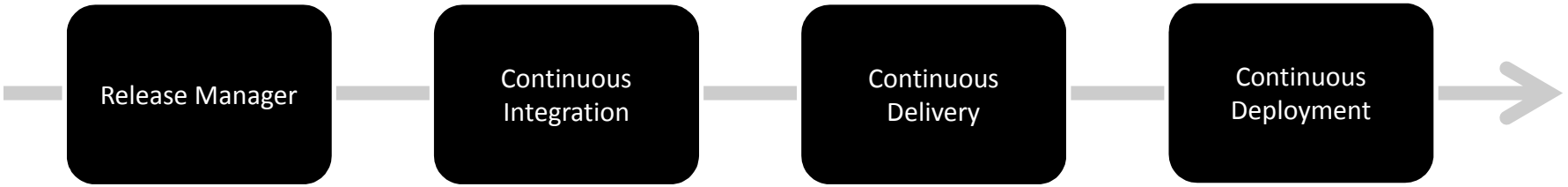
“Why do we never have time to do it right, but always have time to do it over? ”

- Anonymous

Keys to Success

- Organizational commitment to Agile
- Focused delivery team
- Embrace inevitability of change
- Treat agility as a corporate asset
- Favor cultural fit over technical skills when growing the team
- Continually improve

The Final Frontier



Manual builds produced by release managers

Automated builds combined with automated unit and functional testing

Repeatable and reliable deployments allowing for small, frequent releases

Near real-time deployment of production ready code

“The future is here. It’s just not evenly distributed yet.”

- William Gibson

Q&A