Continuous Integration Best Practices in Agile Environments

Ben Reich @ AgileSparks

Who Am I?

years leadership experience in software development & startup operations with Gantt charts

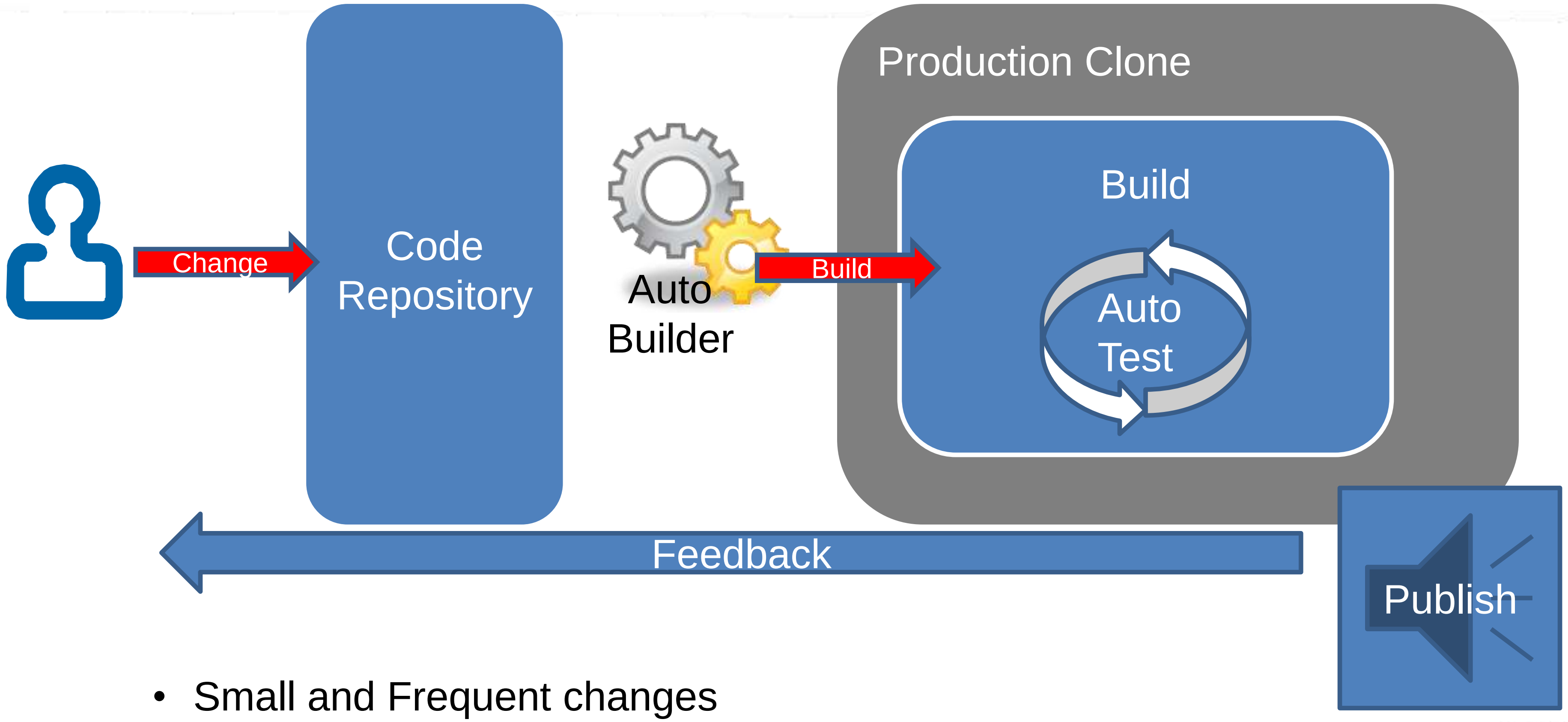
7 Years of searching and finding ways to improve life quality, predictability and efficiency

www.linkedin.com/in/benreich
E-mail: ben@agilesparks.com

What is this about?

Ideas and best practices on implementing continuous integration in an agile way

Continuous Integration 101



- Small and Frequent changes
- Fast and immediate build
- Immediate feedback

So, why is it Agile?

Small Iterations

Rapid Delivery

Continuous Integration

Encourage Cooperation

Working Software

AgileSparks

BUT: IS A BUNCH OF PROGRAMMERS CONSTANTLY MIXING SOME CODE TOGETHER REALLY GETTING US WHERE WE WANT TO BE?

The Challenges of CI

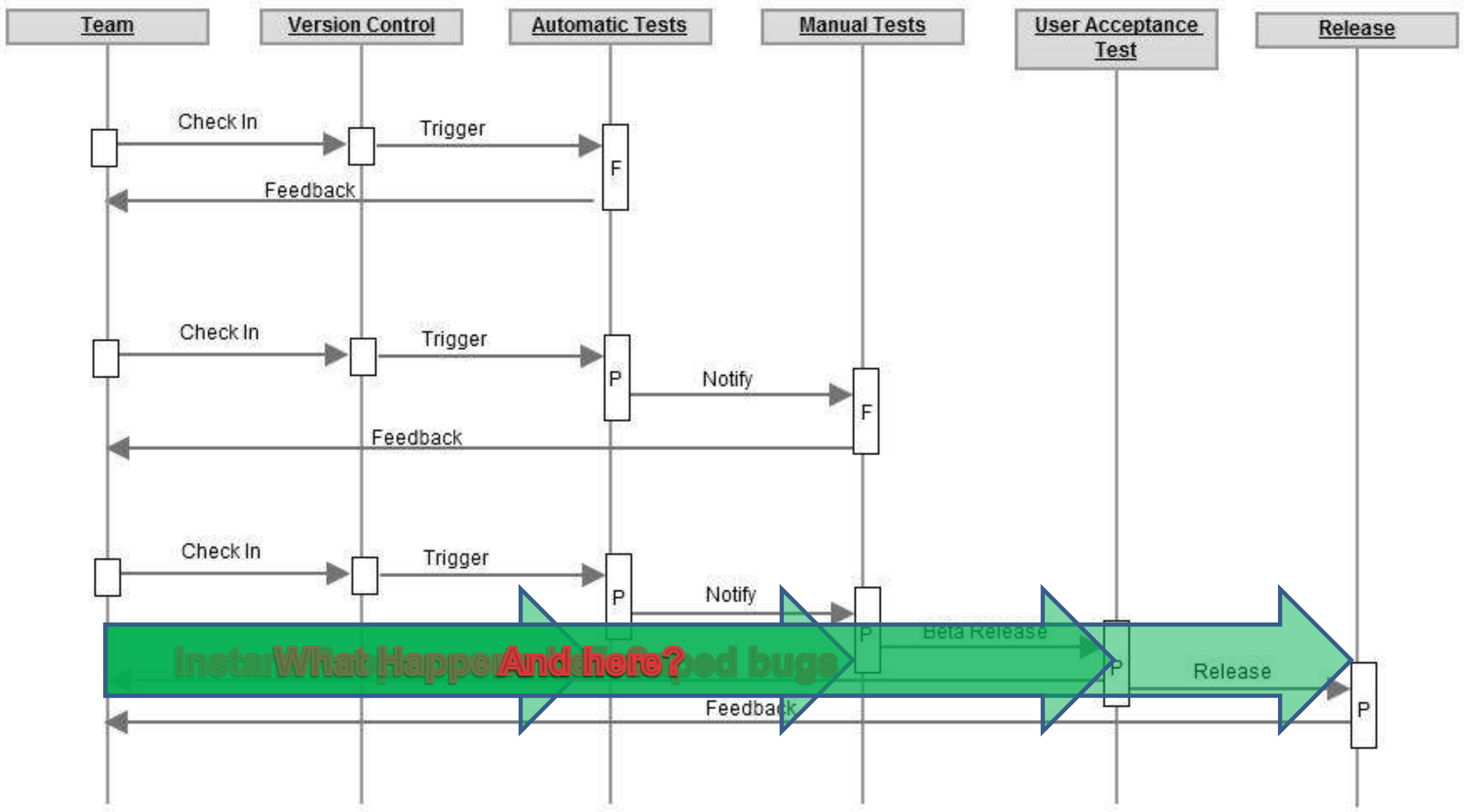
- Distance from the end user
- Automatic tests are difficult
- Large and complex products
- Diverse organization
- Insufficient review
- Inter-team synchronization issues

You know its becoming irrelevant if:

Nobody is consuming it and people are waiting for your “real release”

AgileSparks

Simple Value Stream

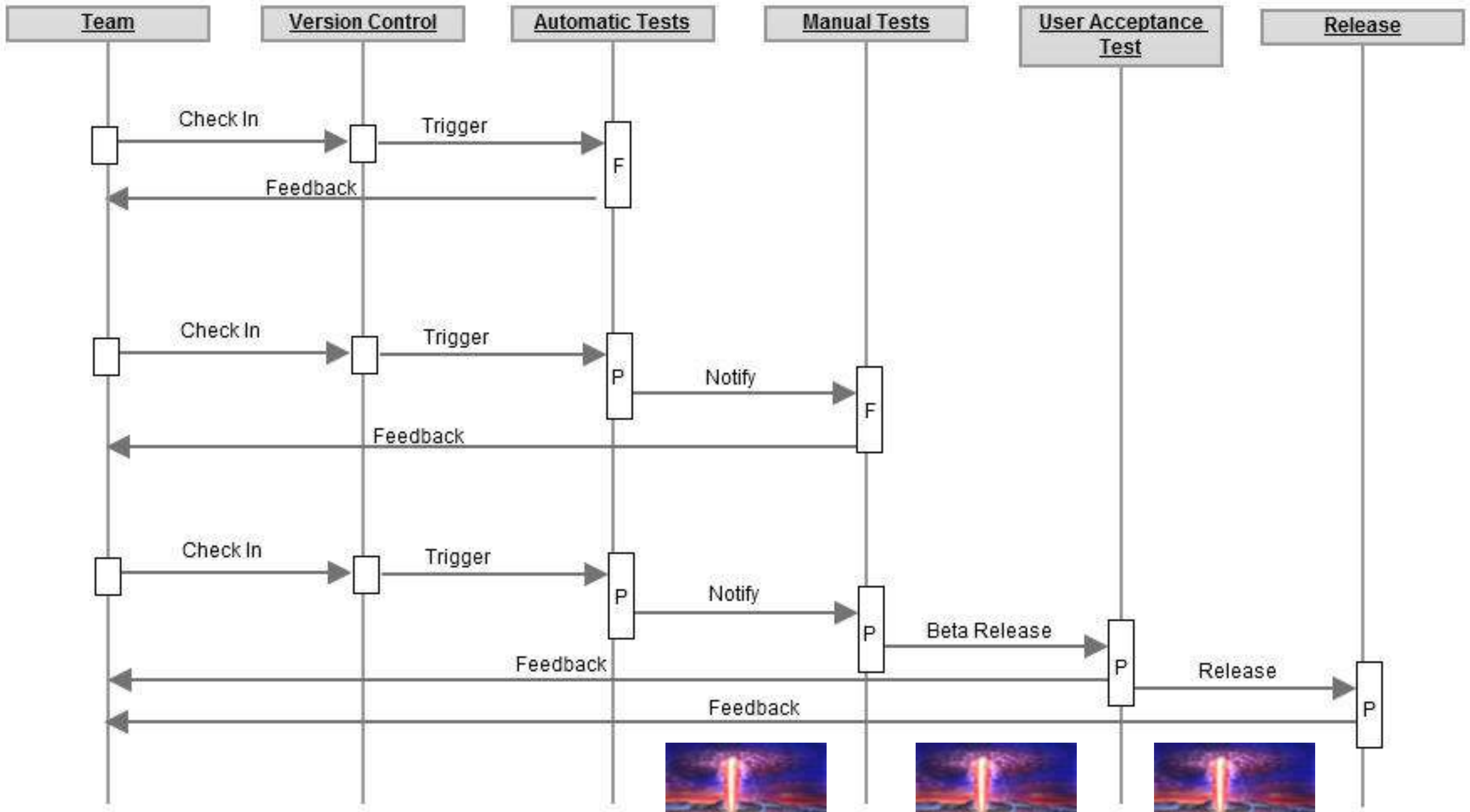


Best
Practice

Holy Place" Culture - Done means Released

- Identify the "holy place"
  - Designated delivery target
  - Always up and running
  - As Releasable as possible
  - Replica of Production
  - Demo-able and accessible
- Identify the Stakeholder
  - Represents the customer
  - Is motivated to succeed
  - Can deliver UAT
  - Has the bandwidth to review

Simple Value Stream

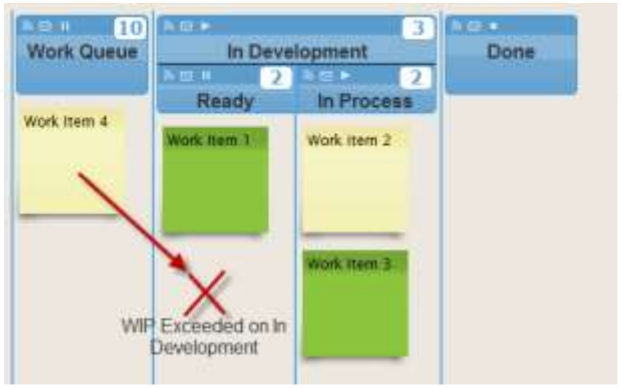


Put it where its accessible, demoable, won't break but will be updated regularly in days

The Delivery Gap

- Manual testing takes longer and there is no synchronization
- Developers need to program before testers need to test
- Large projects need constant integration testing
- Too many changes can disrupt the flow

Ingredients to Fill the Gap



WIP Limits

Time Box

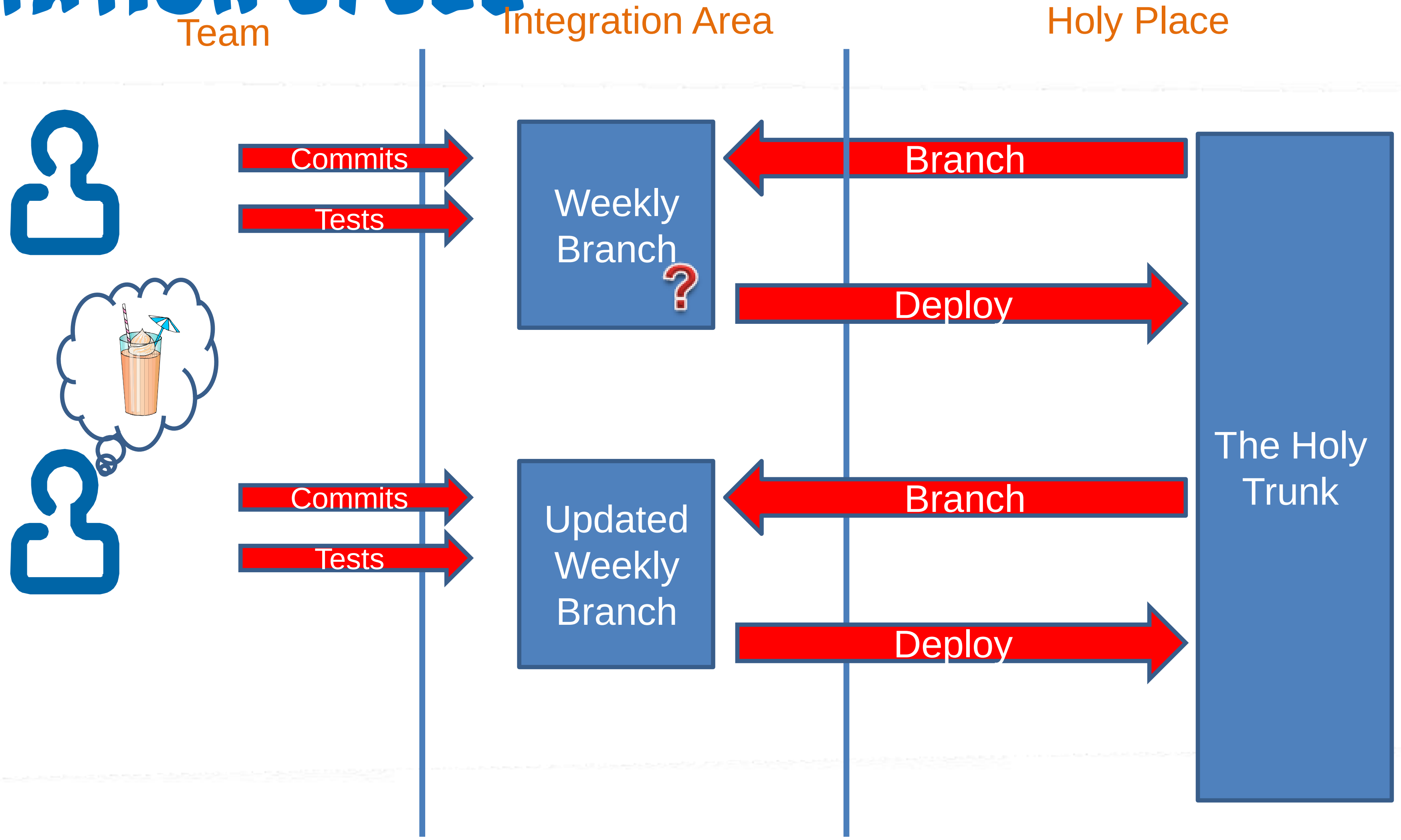
Agile Communication

Case Study – The Demo Cycle

- 8 man team
- 1 month sprint
- 1 week demo cycle
- 1 week deployment cycle
- Typically 2-4 user stories deployed per week

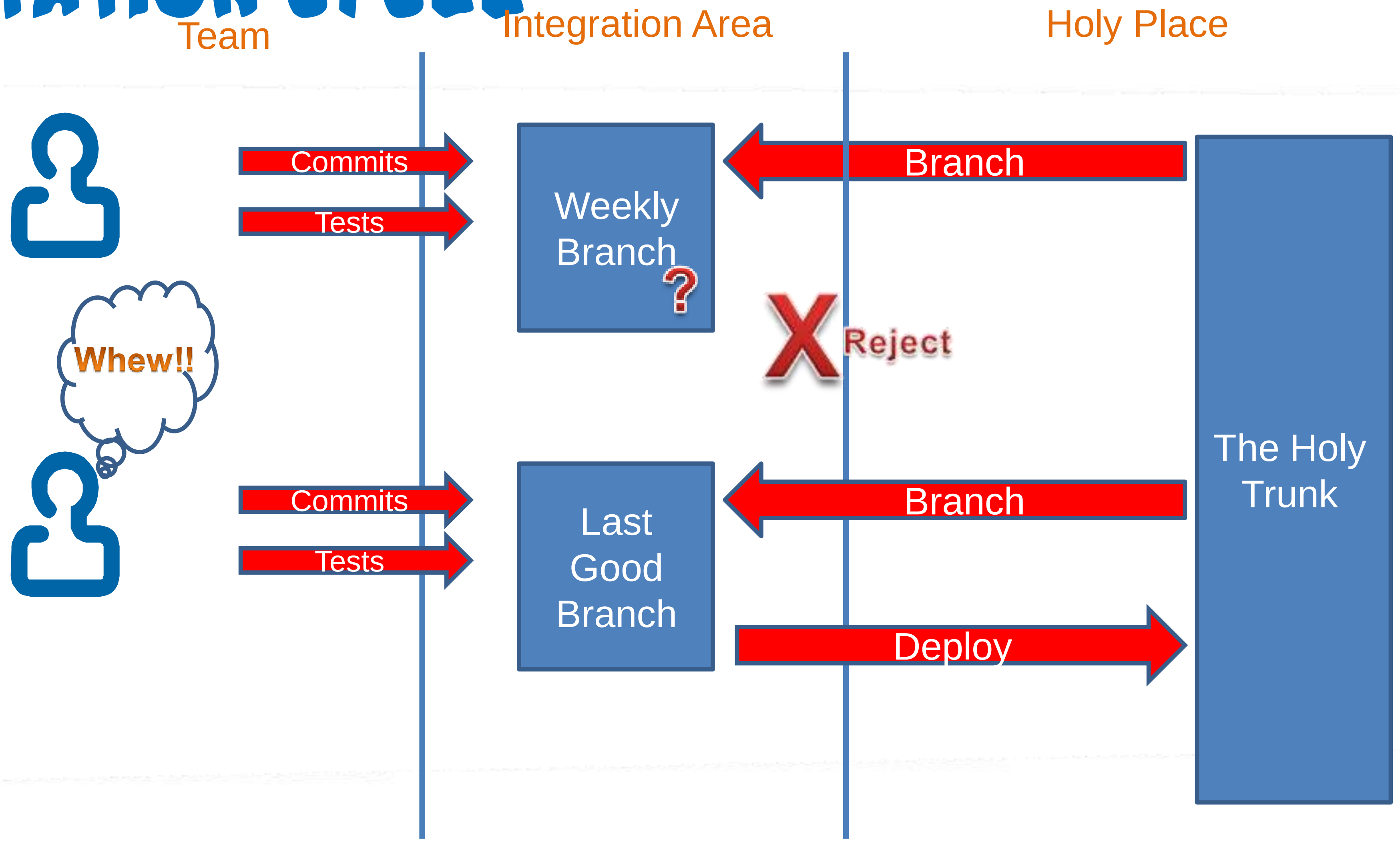
Day 1	Day 2	Day 3	Day 4	Day 5
Coordinate Commit Test	Commit Test	Commit Test	Commit Test Evaluate Freeze	Demo Decide Deploy

The Continuous Delivery Train Station Cycle



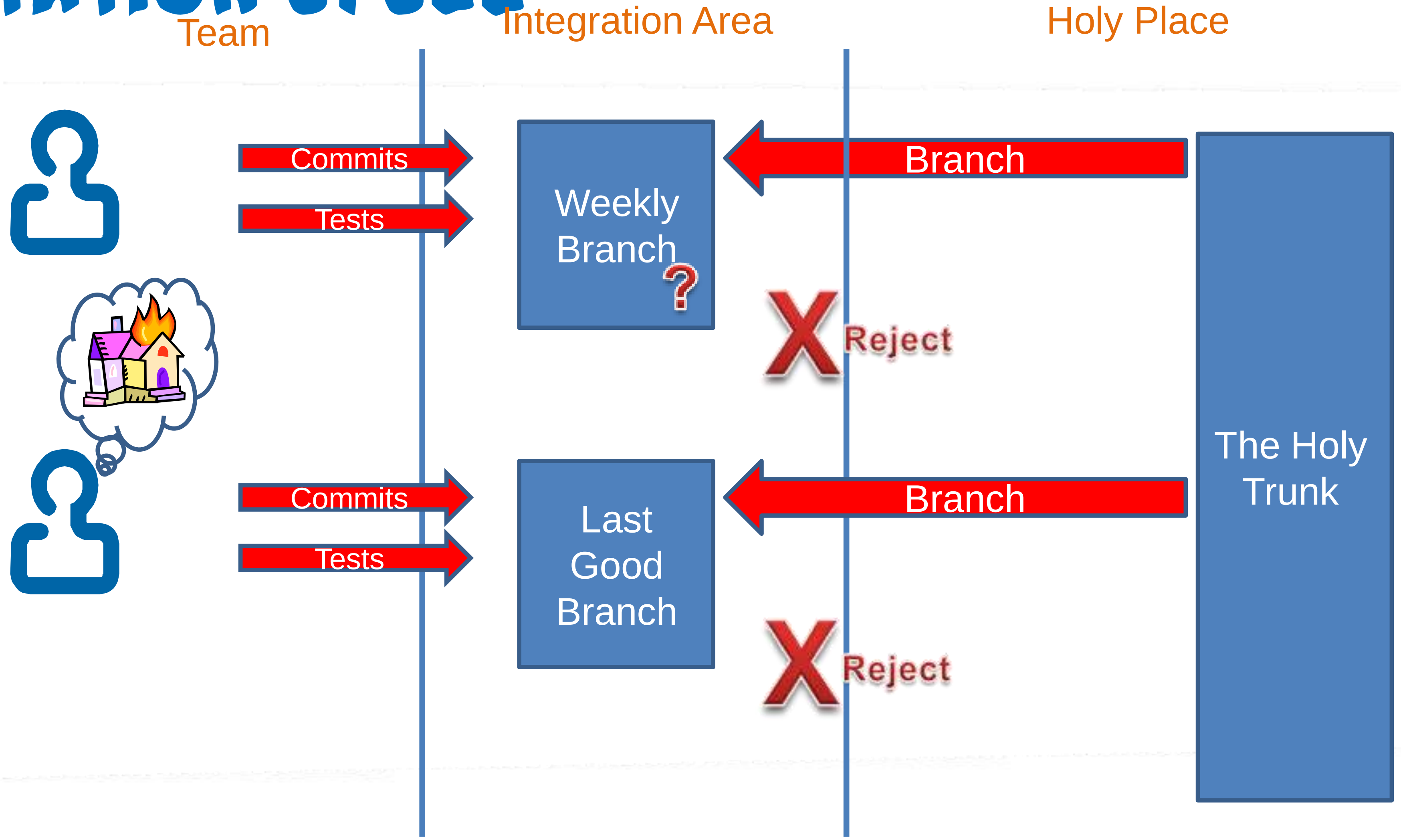
AgileSparks

The Continuous Delivery Train Station Cycle



AgileSparks

The Continuous Delivery Train Station Cycle



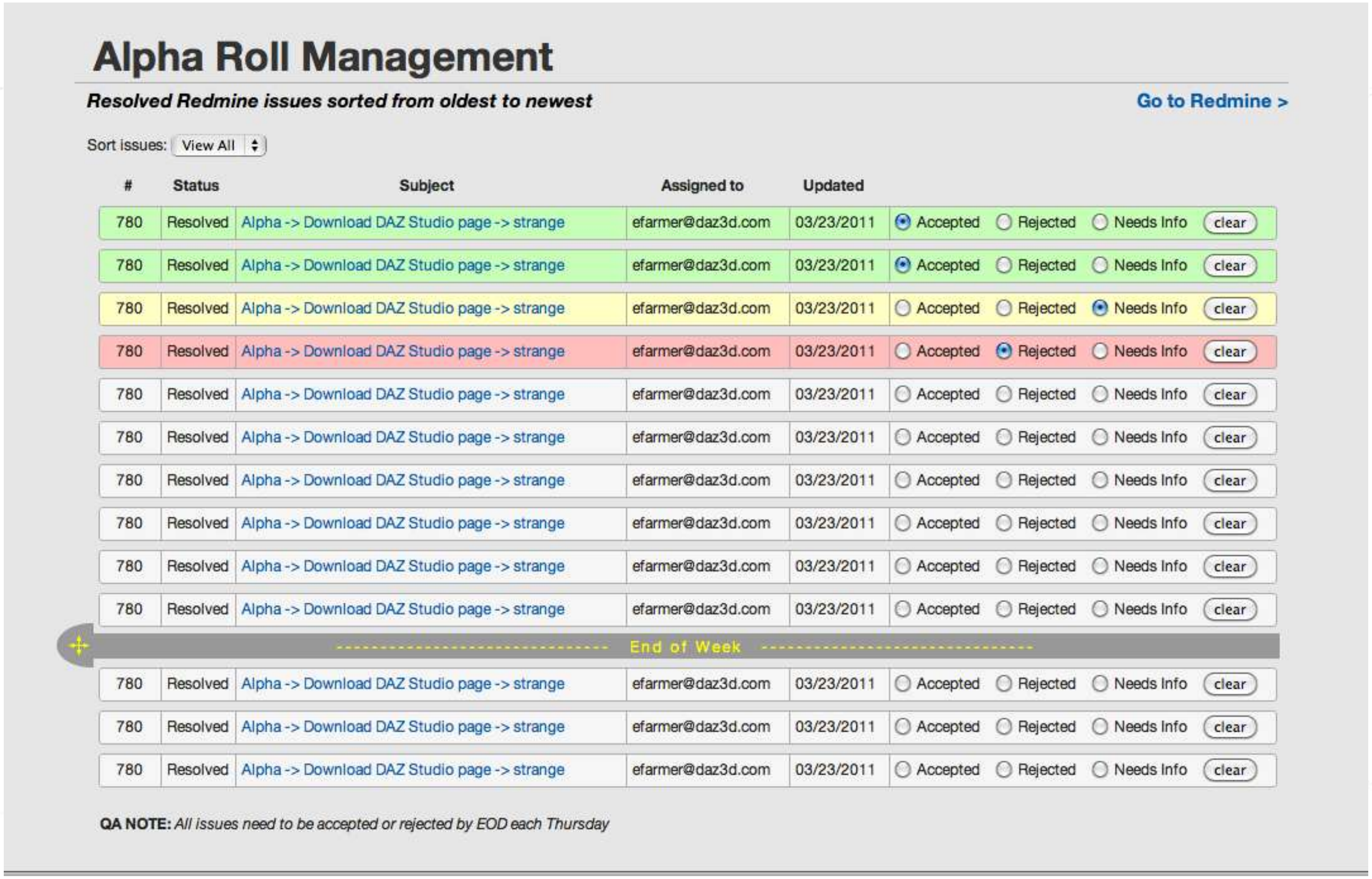
ALERT

CONDITION: RED

STOP

Swarm!!

Electronic WIP Dashboard



Alpha Roll Management

Resolved Redmine issues sorted from oldest to newest

Go to Redmine >

Sort issues: View All

#	Status	Subject	Assigned to	Updated				
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	[✓ Accepted]	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	[✓ Accepted]	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	[✓ Needs Info]	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	[✓ Rejected]	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
		---------------------------- End of Week ----------------------------						
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear
780	Resolved	Alpha -> Download DAZ Studio page -> strange	efarmer@daz3d.com	03/23/2011	Accepted	Rejected	Needs Info	clear

QA NOTE: *All issues need to be accepted or rejected by EOD each Thursday*

AgileSparks

Large Projects - Scrum of Scrum

- Frequency – 1 to 3 days
- What is discussed?
  - What has your team done since we last met?
  - What will your team do before we meet again?
  - Is anything slowing your team down or getting in their way?
  - Are you about to put something in another team’s way?
- The last two items will feature:
  - Broken APIs
  - Synchronization issues
  - Current build status
  - Story and Epic issues

Working with Large Remote Teams

Linux

jQuery

Is it Impossible?

Rails

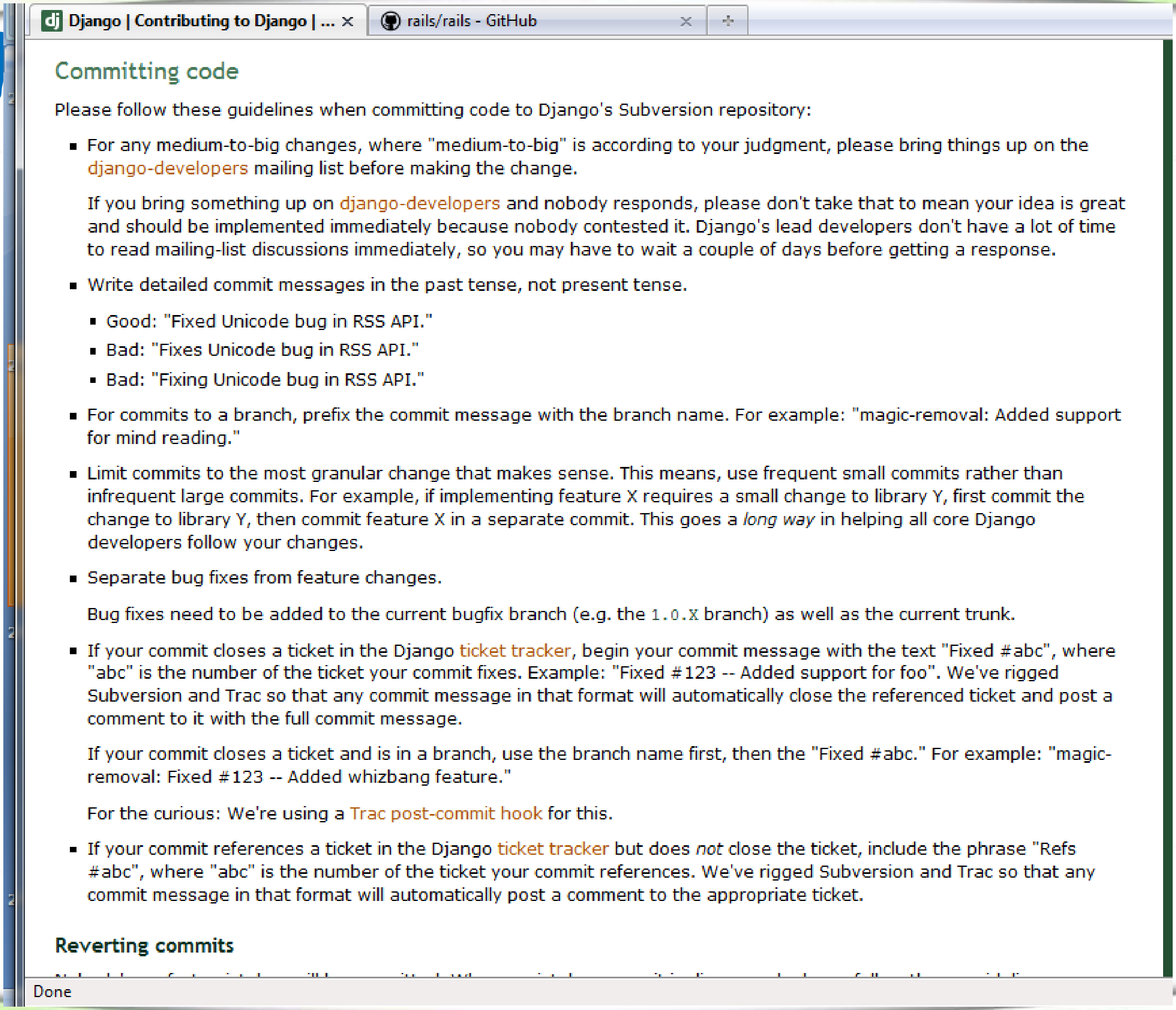
Python

Django

Learning from Open Source

- Manage by committee
- Dedicate resources to CI
- Set strict rules and enforce them
- Enforce and encourage communication
- Never go home with a broken build

Rules of Engagement - Exa



Committing code

Please follow these guidelines when committing code to Django's Subversion repository:

- For any medium-to-big changes, where "medium-to-big" is according to your judgment, please bring things up on the django-developers mailing list before making the change.

  If you bring something up on django-developers and nobody responds, please don't take that to mean your idea is great and should be implemented immediately because nobody contested it. Django's lead developers don't have a lot of time to read mailing-list discussions immediately, so you may have to wait a couple of days before getting a response.

- Write detailed commit messages in the past tense, not present tense.
  - Good: "Fixed Unicode bug in RSS API."
  - Bad: "Fixes Unicode bug in RSS API."
  - Bad: "Fixing Unicode bug in RSS API."
- For commits to a branch, prefix the commit message with the branch name. For example: "magic-removal: Added support for mind reading."
- Limit commits to the most granular change that makes sense. This means, use frequent small commits rather than infrequent large commits. For example, if implementing feature X requires a small change to library Y, first commit the change to library Y, then commit feature X in a separate commit. This goes a *long way* in helping all core Django developers follow your changes.
- Separate bug fixes from feature changes.

  Bug fixes need to be added to the current bugfix branch (e.g. the `1.0.X` branch) as well as the current trunk.

- If your commit closes a ticket in the Django ticket tracker, begin your commit message with the text "Fixed #abc", where "abc" is the number of the ticket your commit fixes. Example: "Fixed #123 -- Added support for foo". We've rigged Subversion and Trac so that any commit message in that format will automatically close the referenced ticket and post a comment to it with the full commit message.

  If your commit closes a ticket and is in a branch, use the branch name first, then the "Fixed #abc." For example: "magic-removal: Fixed #123 -- Added whizbang feature."

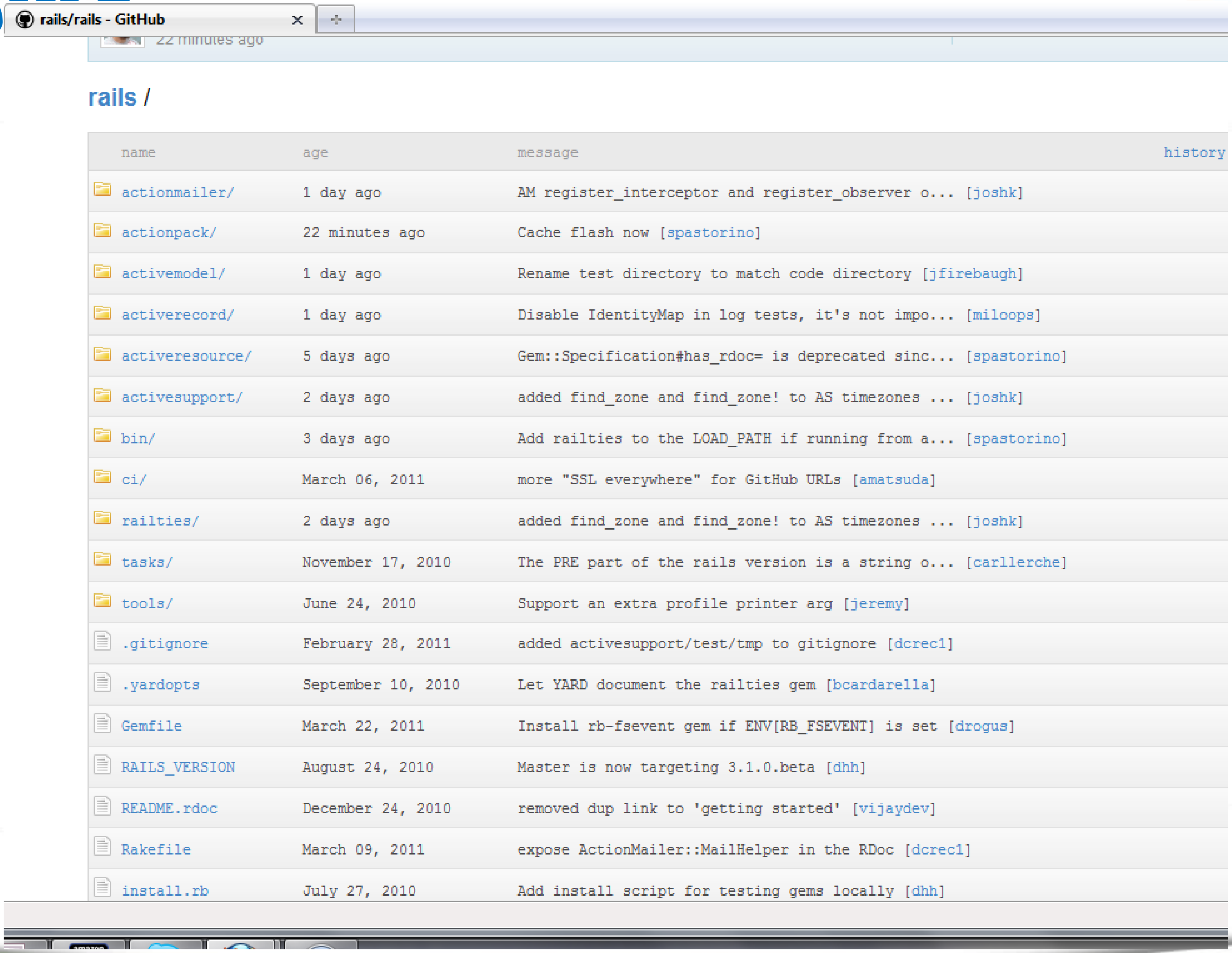
  For the curious: We're using a Trac post-commit hook for this.

- If your commit references a ticket in the Django ticket tracker but does *not* close the ticket, include the phrase "Refs #abc", where "abc" is the number of the ticket your commit references. We've rigged Subversion and Trac so that any commit message in that format will automatically post a comment to the appropriate ticket.

Reverting commits

AgileSparks

GitHub change log - Social coding



rails /

name	age	message	history
actionmailer/	1 day ago	AM register_interceptor and register_observer o... [joshk]	
actionpack/	22 minutes ago	Cache flash now [spastorino]	
activemodel/	1 day ago	Rename test directory to match code directory [jfirebaugh]	
activerecord/	1 day ago	Disable IdentityMap in log tests, it's not impo... [miloops]	
activeresource/	5 days ago	Gem::Specification#has_rdoc= is deprecated sinc... [spastorino]	
activesupport/	2 days ago	added find_zone and find_zone! to AS timezones ... [joshk]	
bin/	3 days ago	Add railties to the LOAD_PATH if running from a... [spastorino]	
ci/	March 06, 2011	more "SSL everywhere" for GitHub URLs [amatsuda]	
railties/	2 days ago	added find_zone and find_zone! to AS timezones ... [joshk]	
tasks/	November 17, 2010	The PRE part of the rails version is a string o... [carllerche]	
tools/	June 24, 2010	Support an extra profile printer arg [jeremy]	
.gitignore	February 28, 2011	added activesupport/test/tmp to gitignore [dcrec1]	
.yardopts	September 10, 2010	Let YARD document the railties gem [bcardarella]	
Gemfile	March 22, 2011	Install rb-fsevent gem if ENV[RB_FSEVENT] is set [drogus]	
RAILS_VERSION	August 24, 2010	Master is now targeting 3.1.0.beta [dhh]	
README.rdoc	December 24, 2010	removed dup link to 'getting started' [vijaydev]	
Rakefile	March 09, 2011	expose ActionMailer::MailHelper in the RDoc [dcrec1]	
install.rb	July 27, 2010	Add install script for testing gems locally [dhh]	

Source Control Management Recipes

Quiz - Which Tree Looks More Target Oriented?

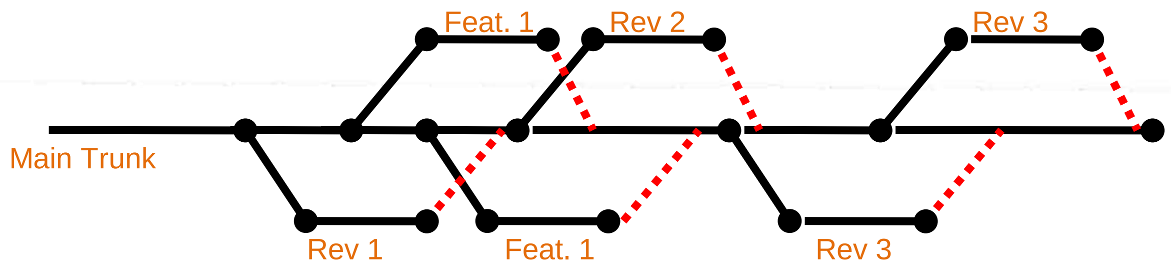
Eshel

Brosh

1 trunk – No Deviations

AgileSparks

Incremental Changes



- Rule no. 1 – Do not branch for features.
- Rule no. 2 – If you breached Rule no. 1:
  - Merge quickly – Put an expiration date on branches
  - Only one level
  - Police the branches
  - Pull Trunk before merging
  - Test branches before merging

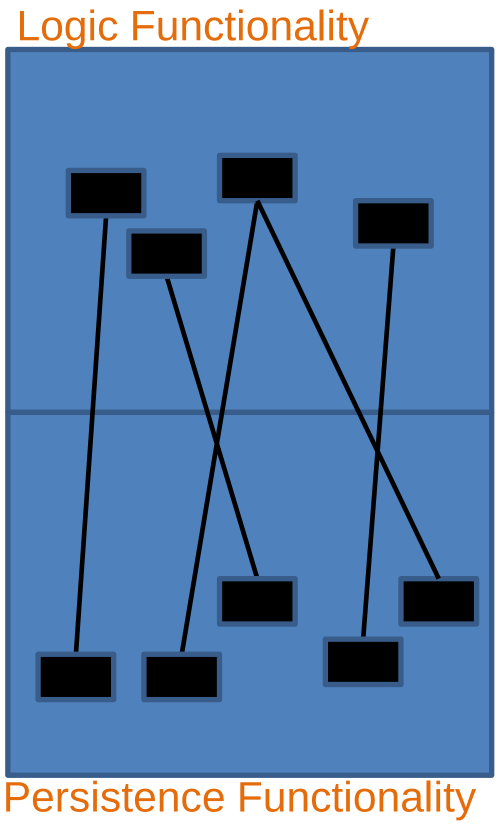
Hiding New Features

- Not usually recommended
- Better than Branching
  - Gradual integration
  - Passes some tests
  - Lower Stabilization Cost

AgileSparks

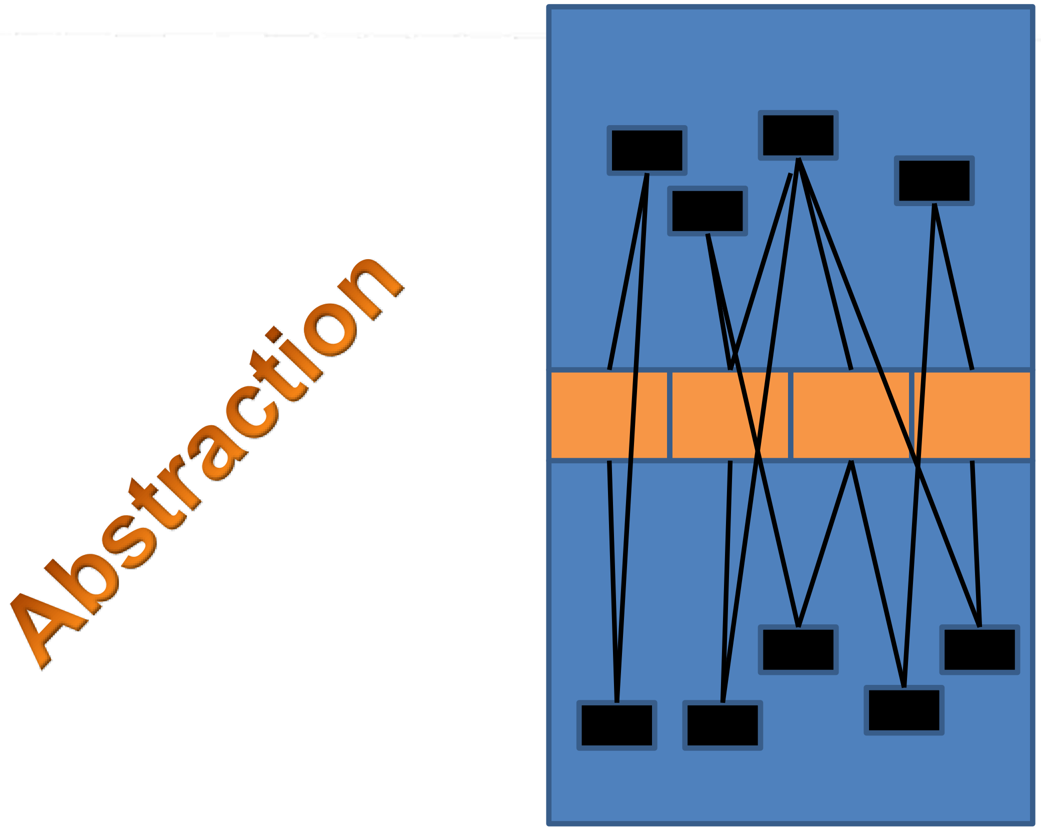
Branching by Abstraction

Spaghetti



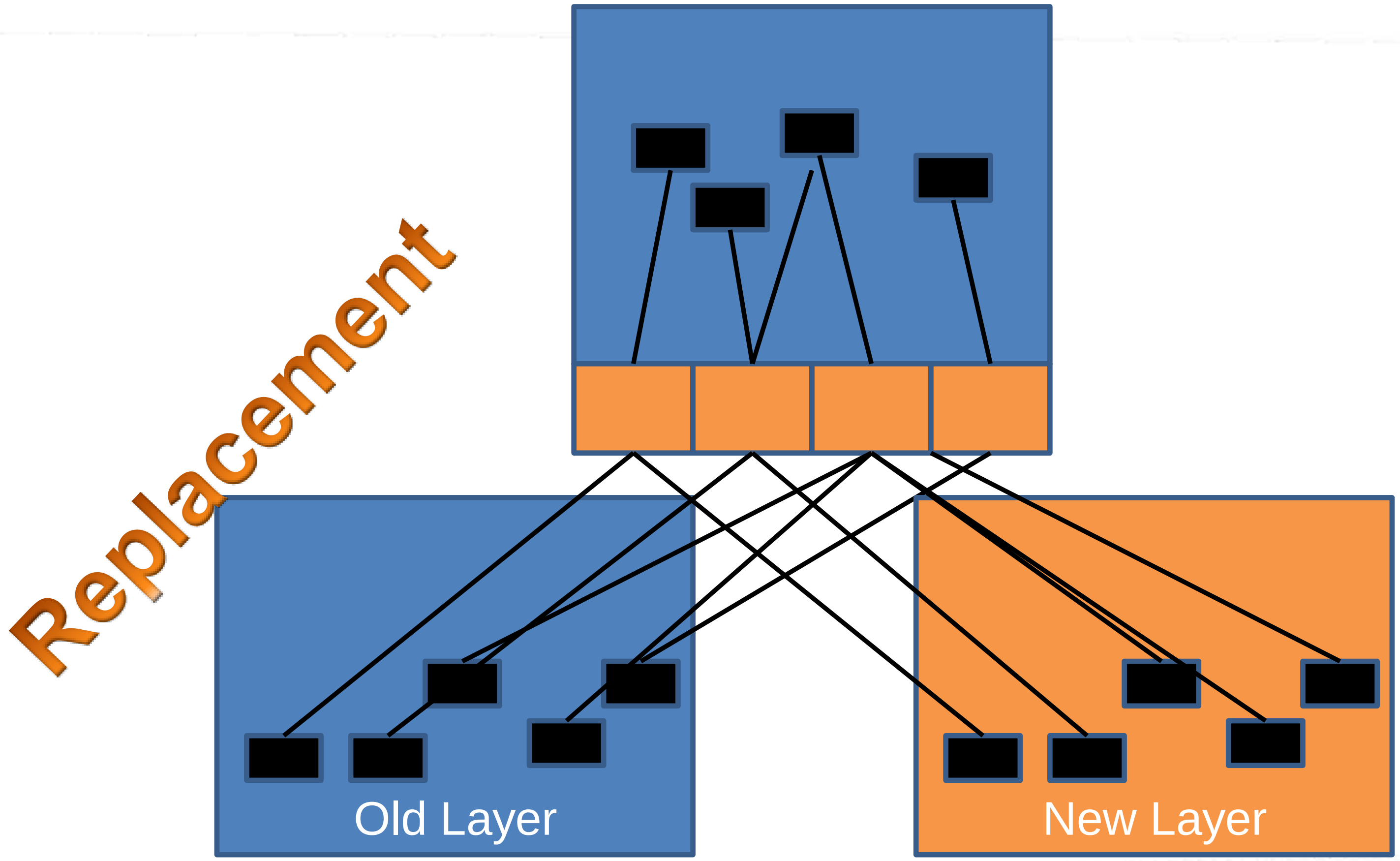
AgileSparks

Branching by Abstraction



AgileSparks

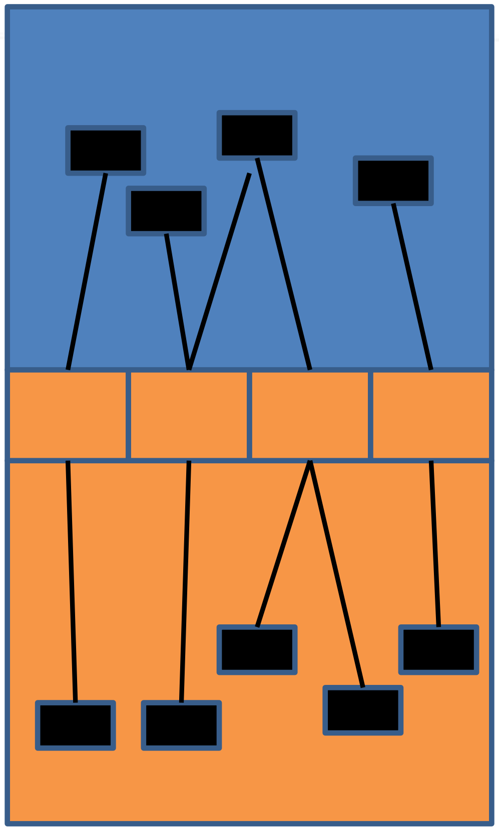
Branching by Abstraction



AgileSparks

BRANCHING BY ABSTRACTION

Replacement



AgileSparks

Auto testing – The Key to Success

Test	Auto-mated?
Build Tests	✓
Unit Tests	✓
Functional Test	✓
Non-Functional tests	✓
Acceptance tests	✓

- Automate in order
- Automate the happy path first
- Maximize coverage with data and not complex scripting
- Look for repeating manual tests
- Spec test together with PO
- Automate when functionality is frozen

What Next?

- Visualize your value stream
- Choose tools
  - Continuous Integration
  - Auto testing
  - Deployment
- Invest in automatic testing
- Tailor fit a process to your needs

Questions?

Thank You!

www.linkedin.com/in/benreich
E-mail: ben@agilesparks.com