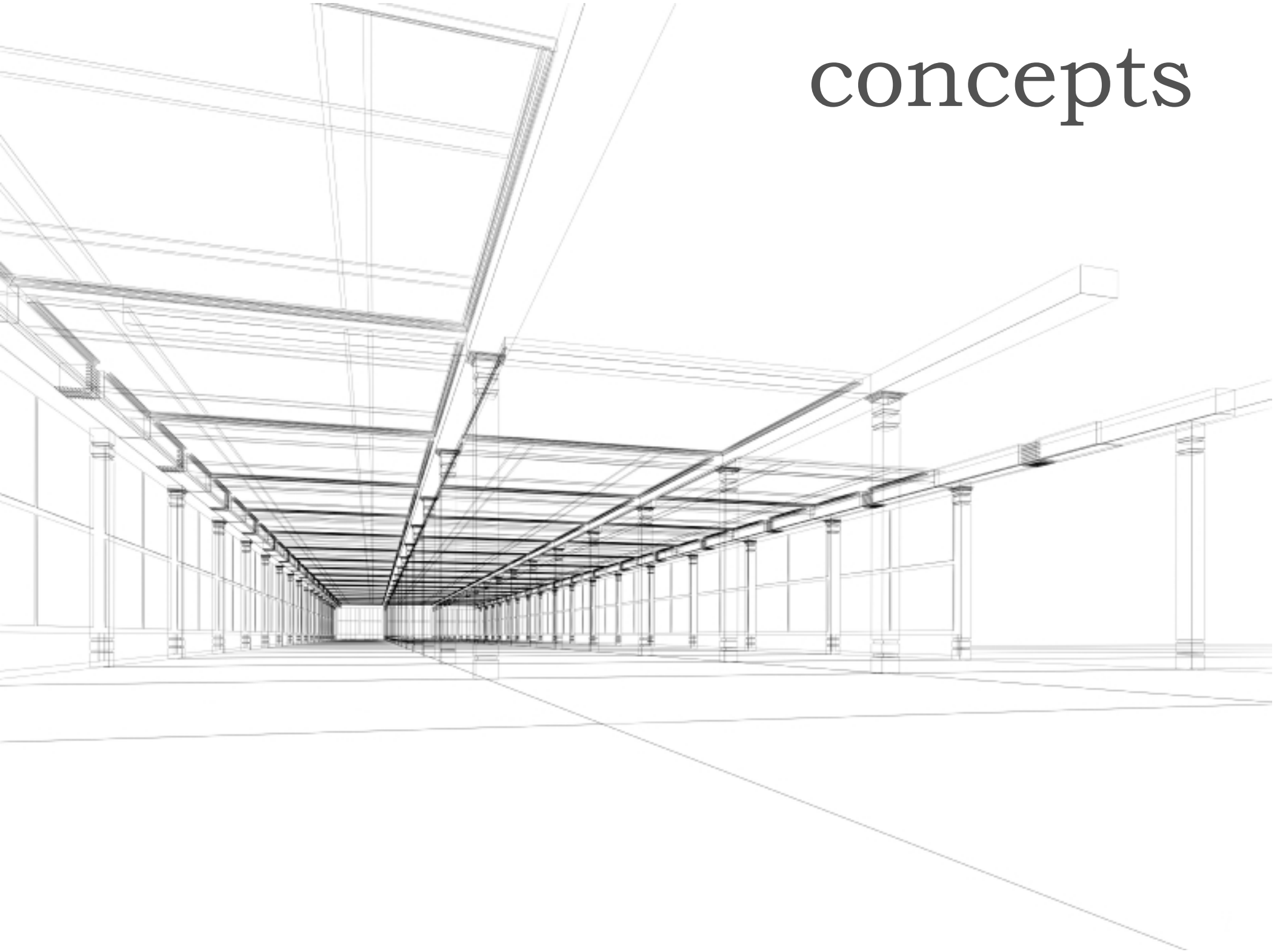
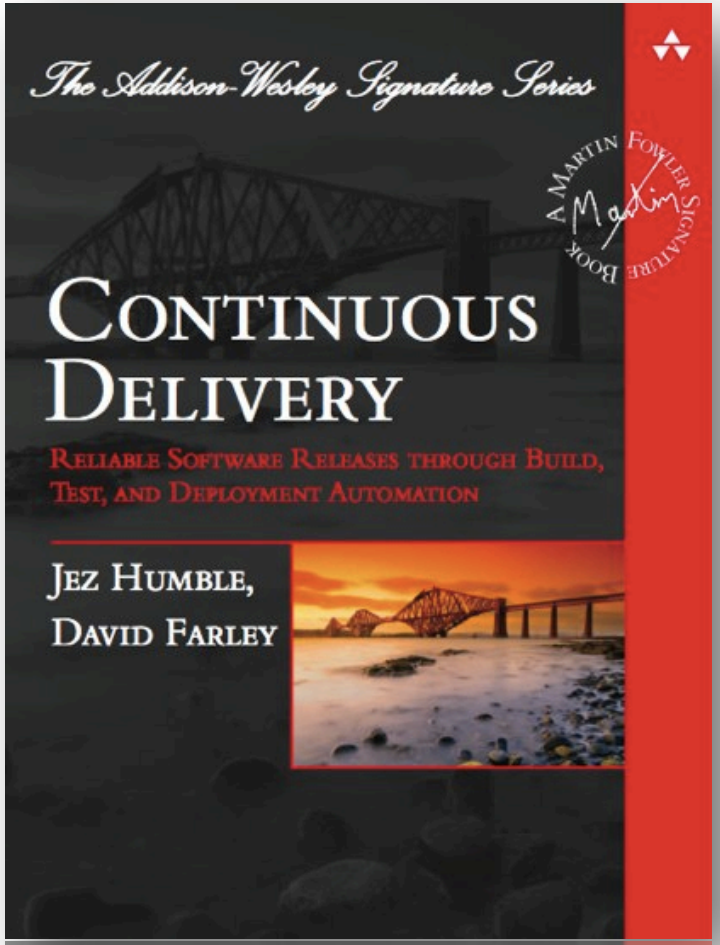
continuous delivery for architects part 1

concepts



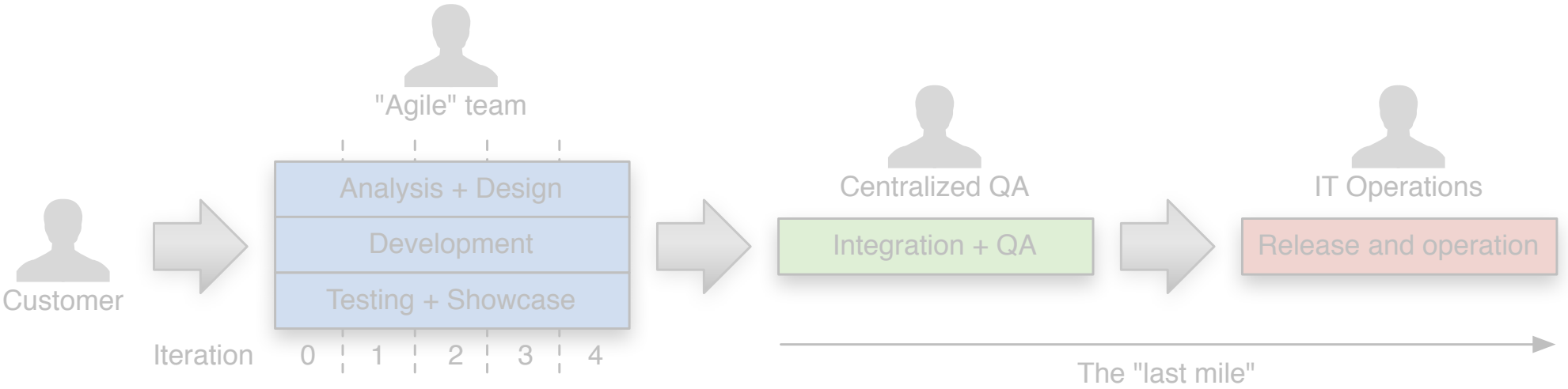


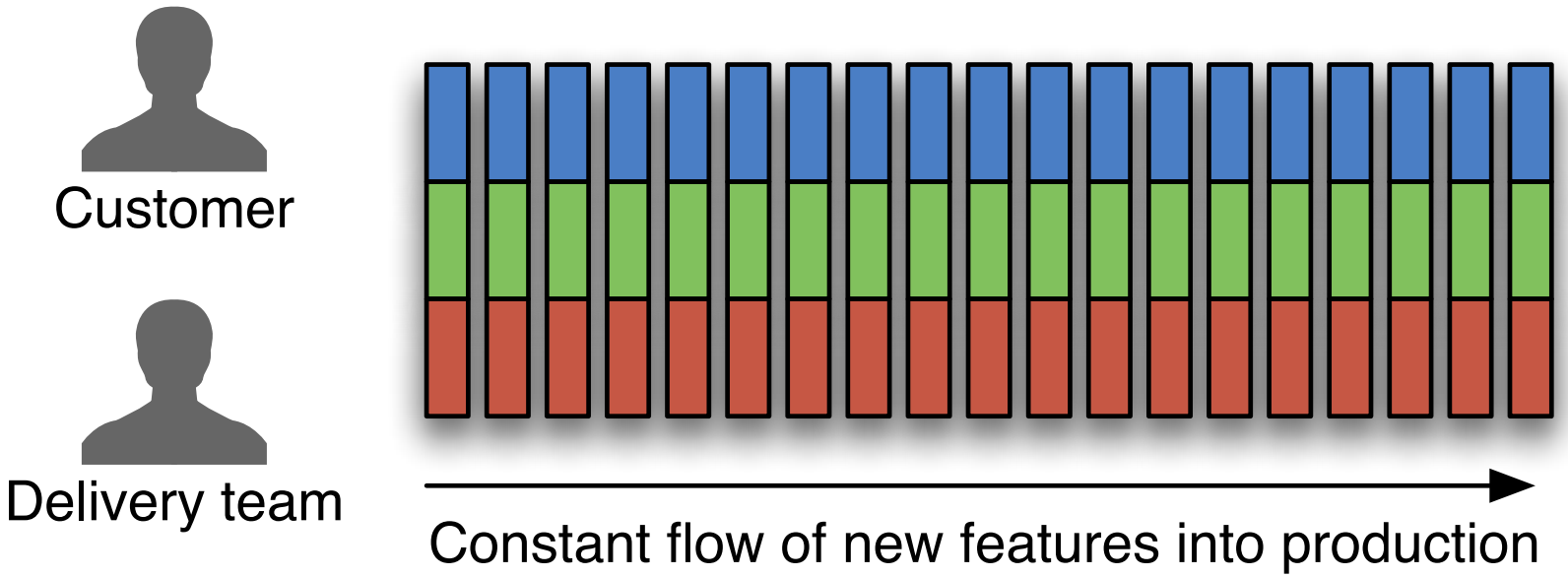
Continuous Delivery

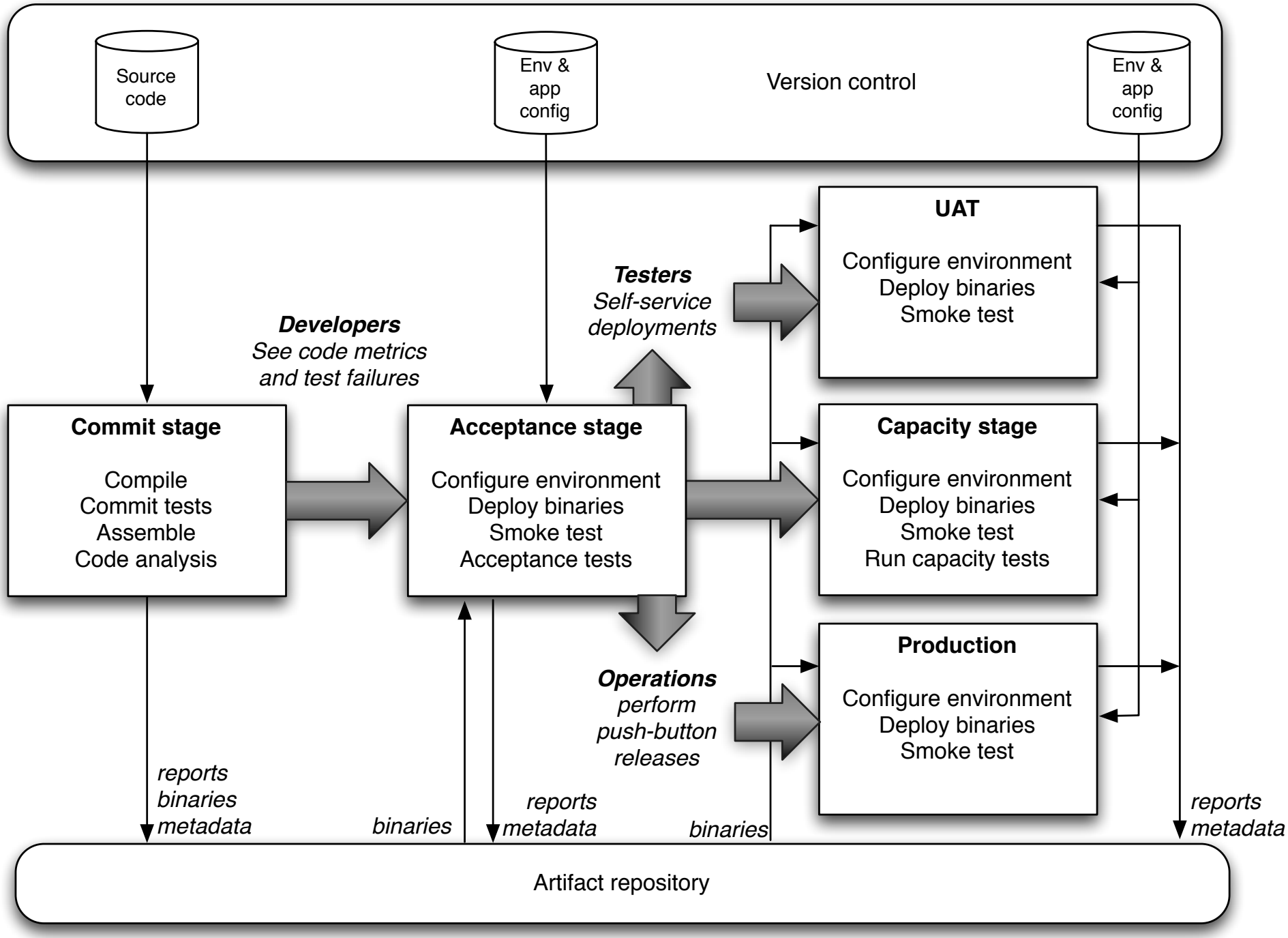
Jez Humble
David Farley

Fast, automated feedback on the production readiness of your application every time there is a change — to code, infrastructure, or configuration

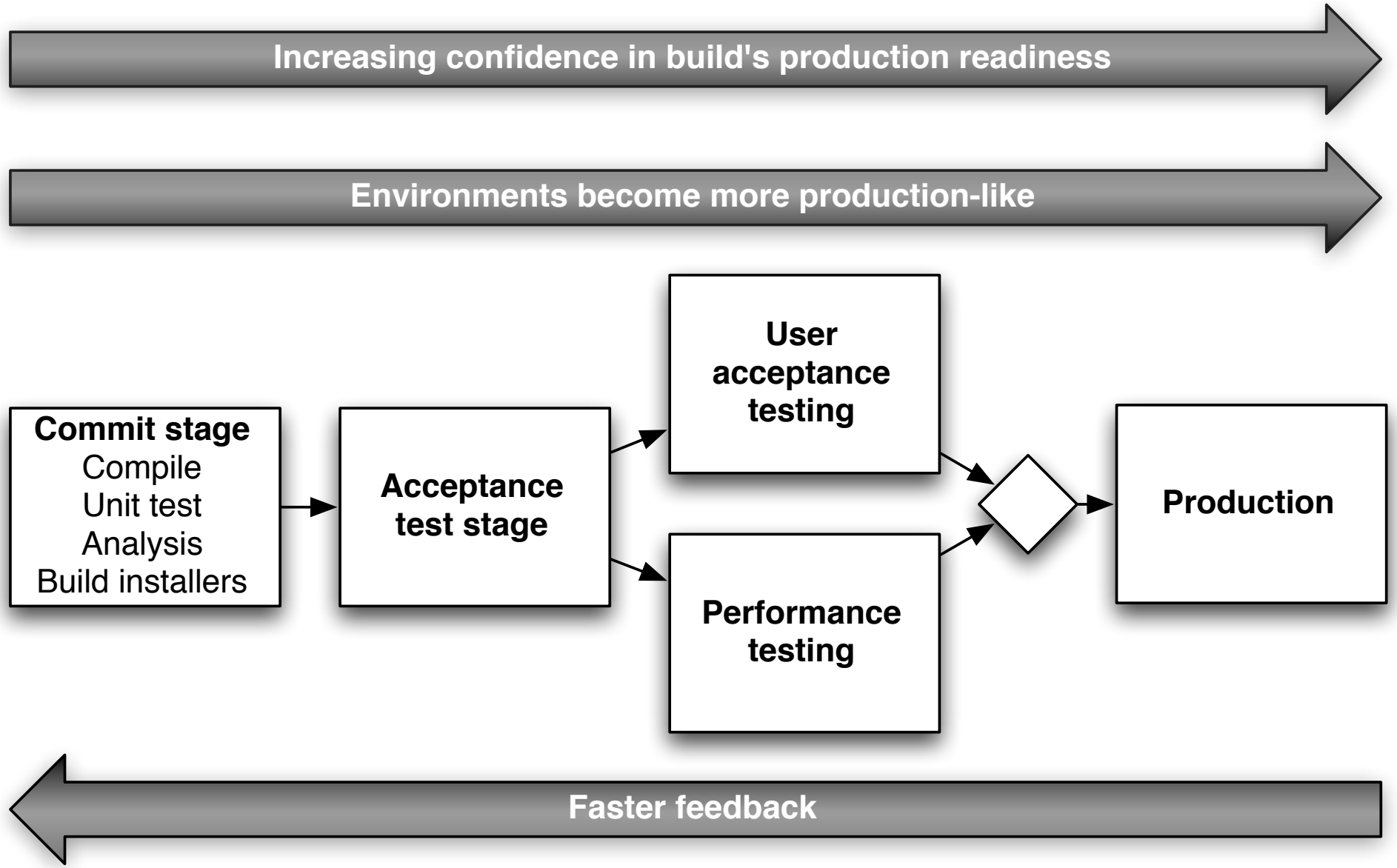
keep your application releasable






Source code
Env & app config
Version control
Env & app config
UAT
Configure environment
Deploy binaries
Smoke test
Testers
Self-service deployments
Developers
See code metrics and test failures
Commit stage
Compile
Commit tests
Assemble
Code analysis
Acceptance stage
Configure environment
Deploy binaries
Smoke test
Acceptance tests
Capacity stage
Configure environment
Deploy binaries
Smoke test
Run capacity tests
Operations
perform push-button releases
Production
Configure environment
Deploy binaries
Smoke test
reports
binaries
metadata
binaries
reports
metadata
binaries
reports
metadata
Artifact repository

“production-like”

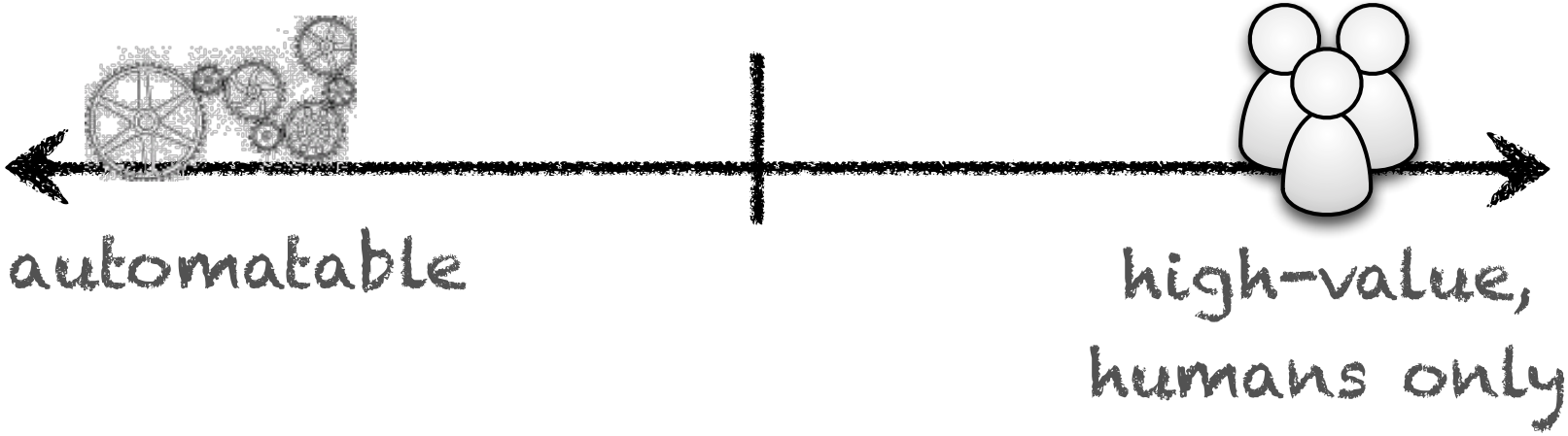


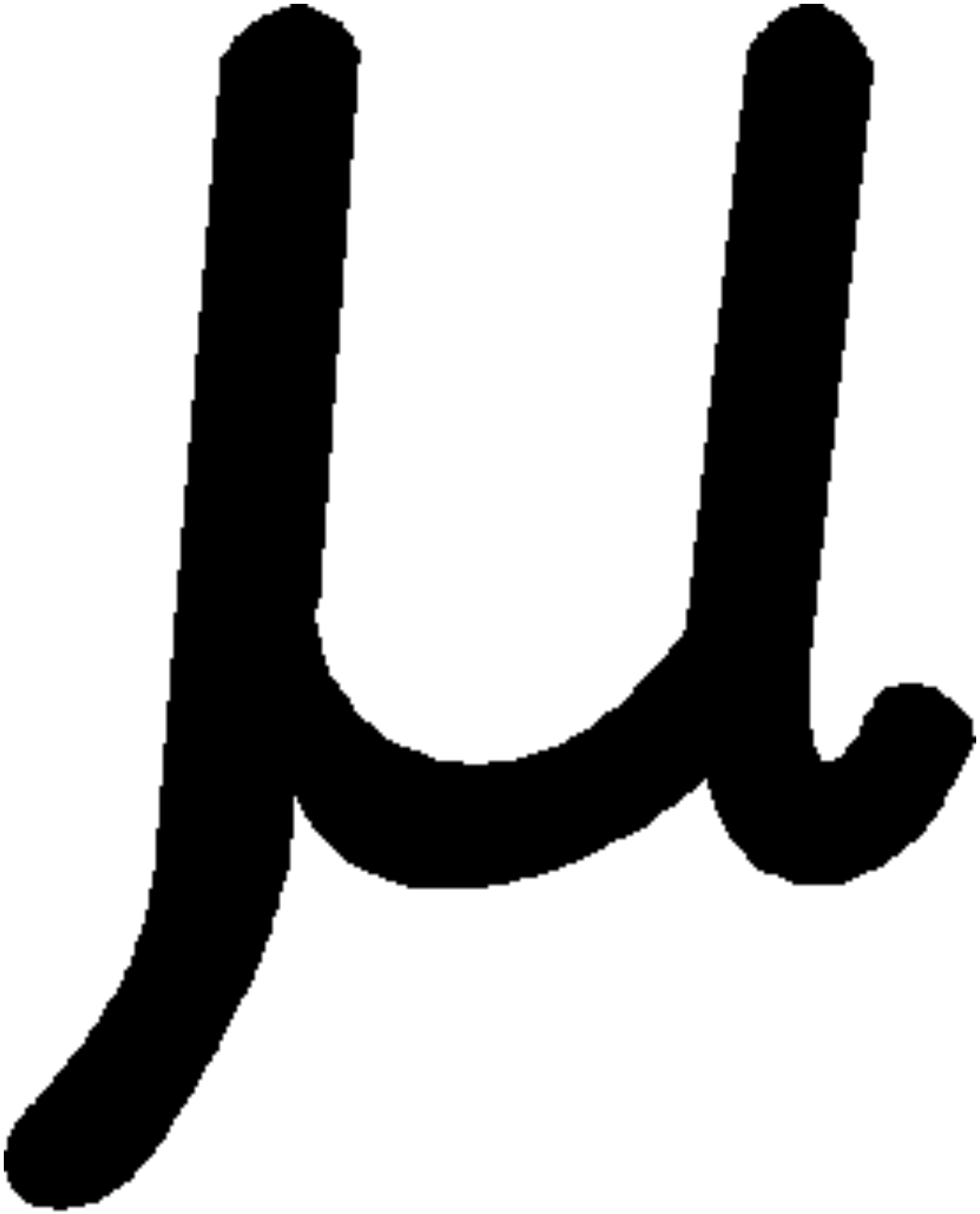

principles of
software delivery

principles

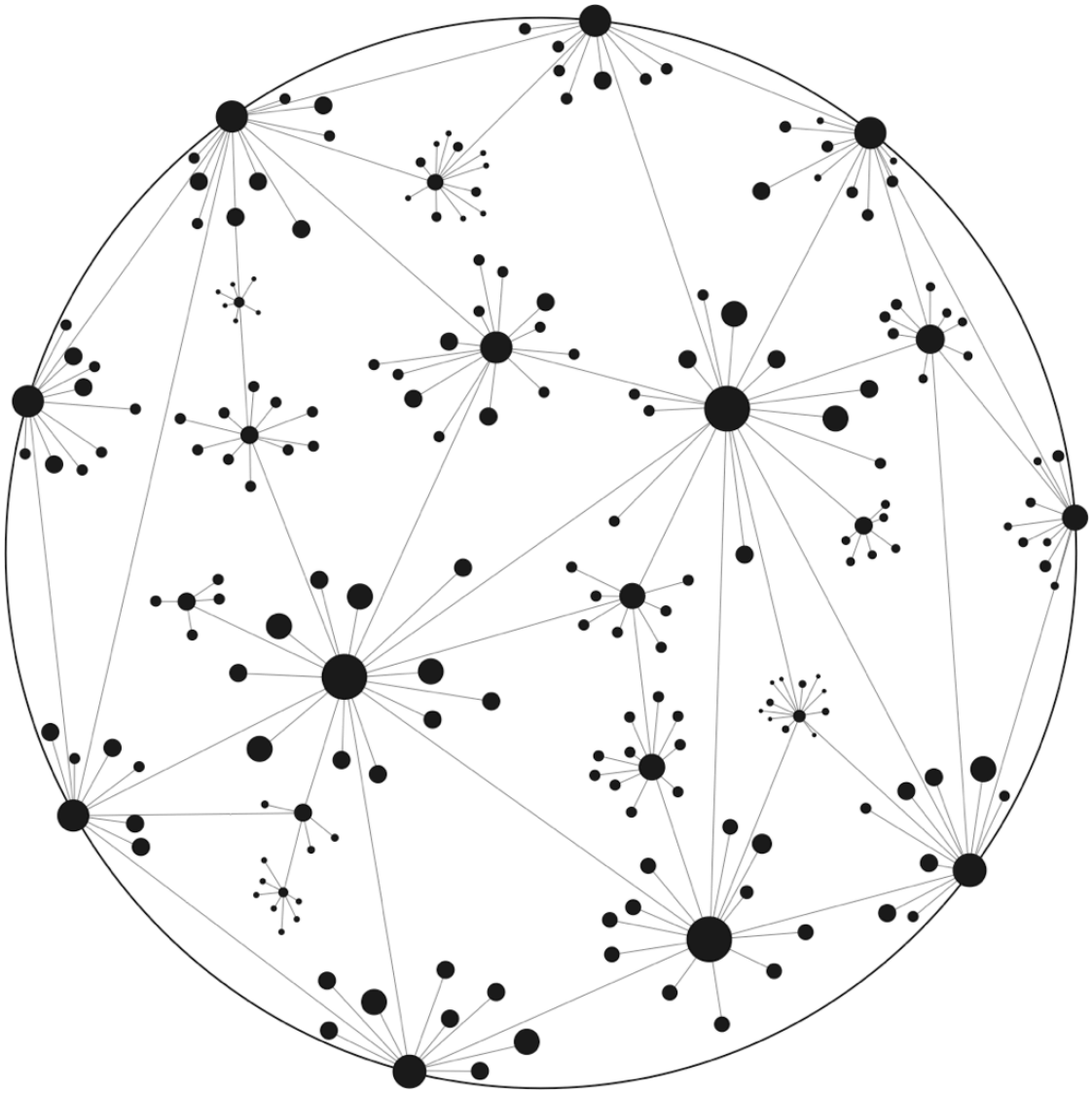
automate!

everything needed to build, test, deploy, & release

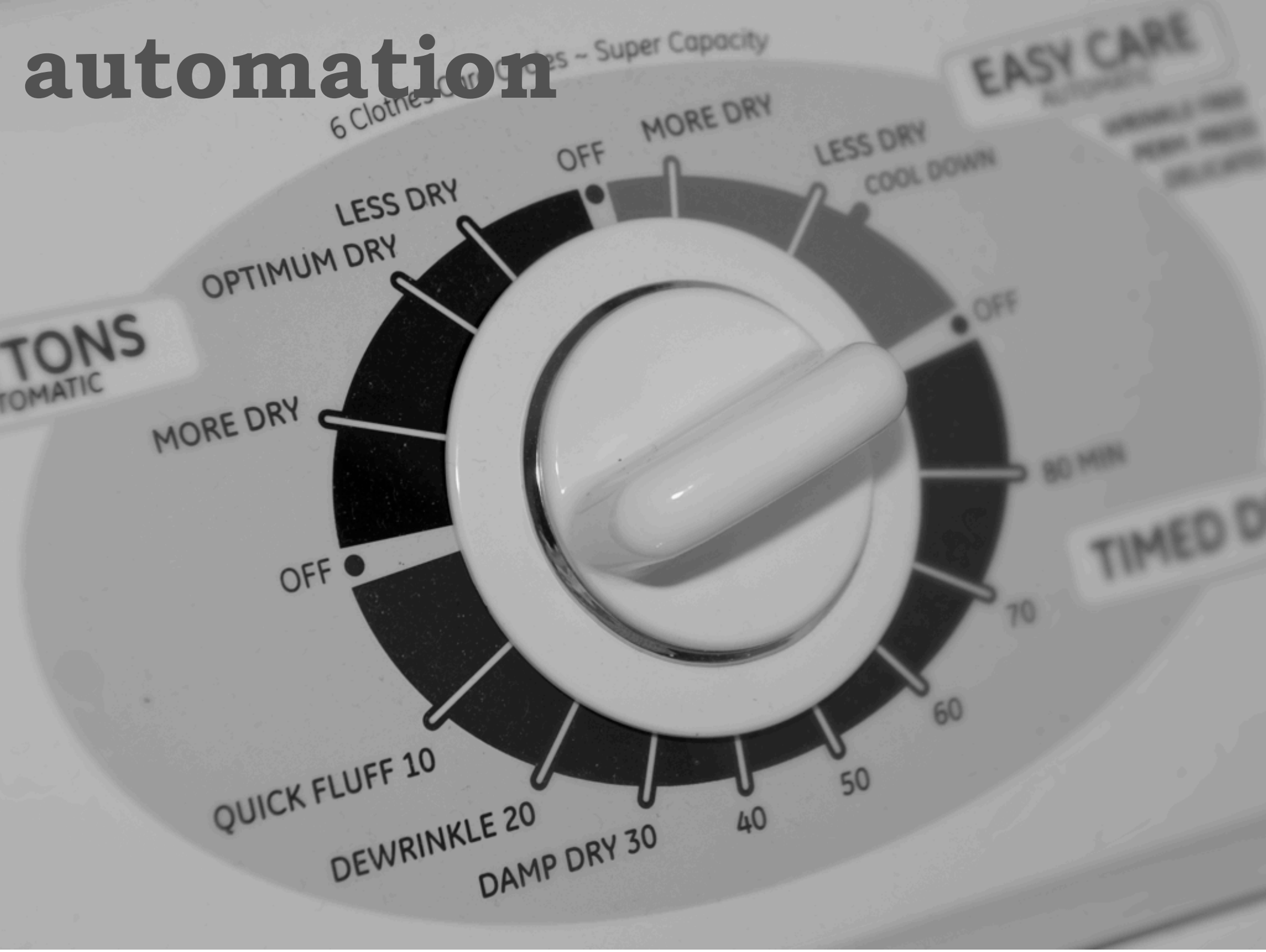




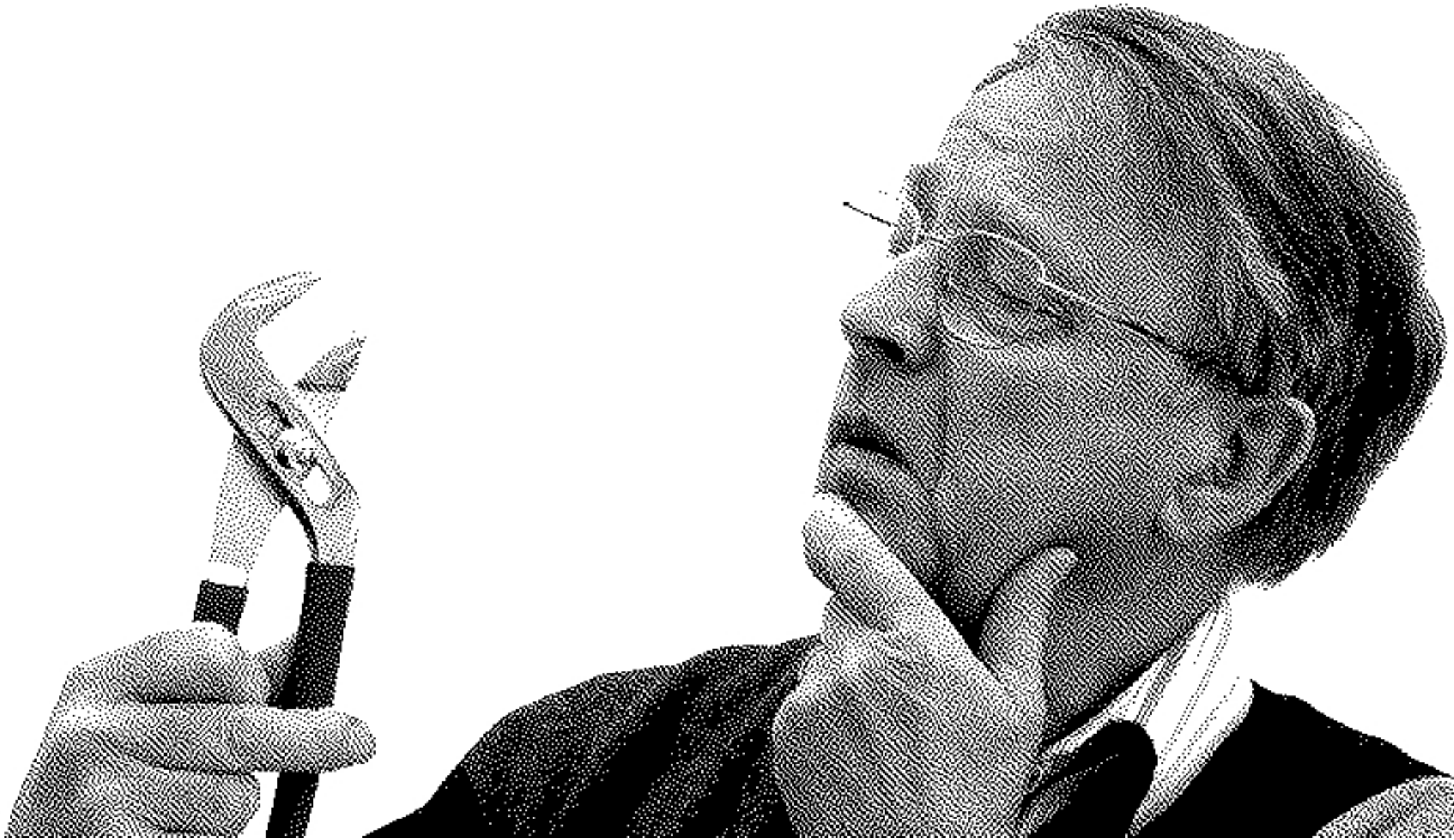
holistic engineering



automation


EASY CARE
OFF
MORE DRY
LESS DRY
COOL DOWN
OFF
80 MIN
70
60
50
40
DAMP DRY 30
DEWRINKLE 20
QUICK FLUFF 10
OFF
MORE DRY
OPTIMUM DRY
LESS DRY

subverting other tools



cucumber-puppet

```
Feature: General catalog policy
  In order to ensure applicability of a host's catalog
  As a manifest developer
  I want all catalogs to obey some general rules

  Scenario Outline: Compile and verify catalog
    Given a node specified by "features/yaml/<hostname>.example.com.yaml"
    When I compile its catalog
    Then compilation should succeed
    And all resource dependencies should resolve

    Examples:
      | hostname  |
      | localhost |
```

build your own tools

timebox

don’t give up

bring the pain forward

pursue good ideas

controlled

reserved time?


don’t shave yaks!

promote development

operations

- versioning machine configurations
- “machines are compiled output”

QA

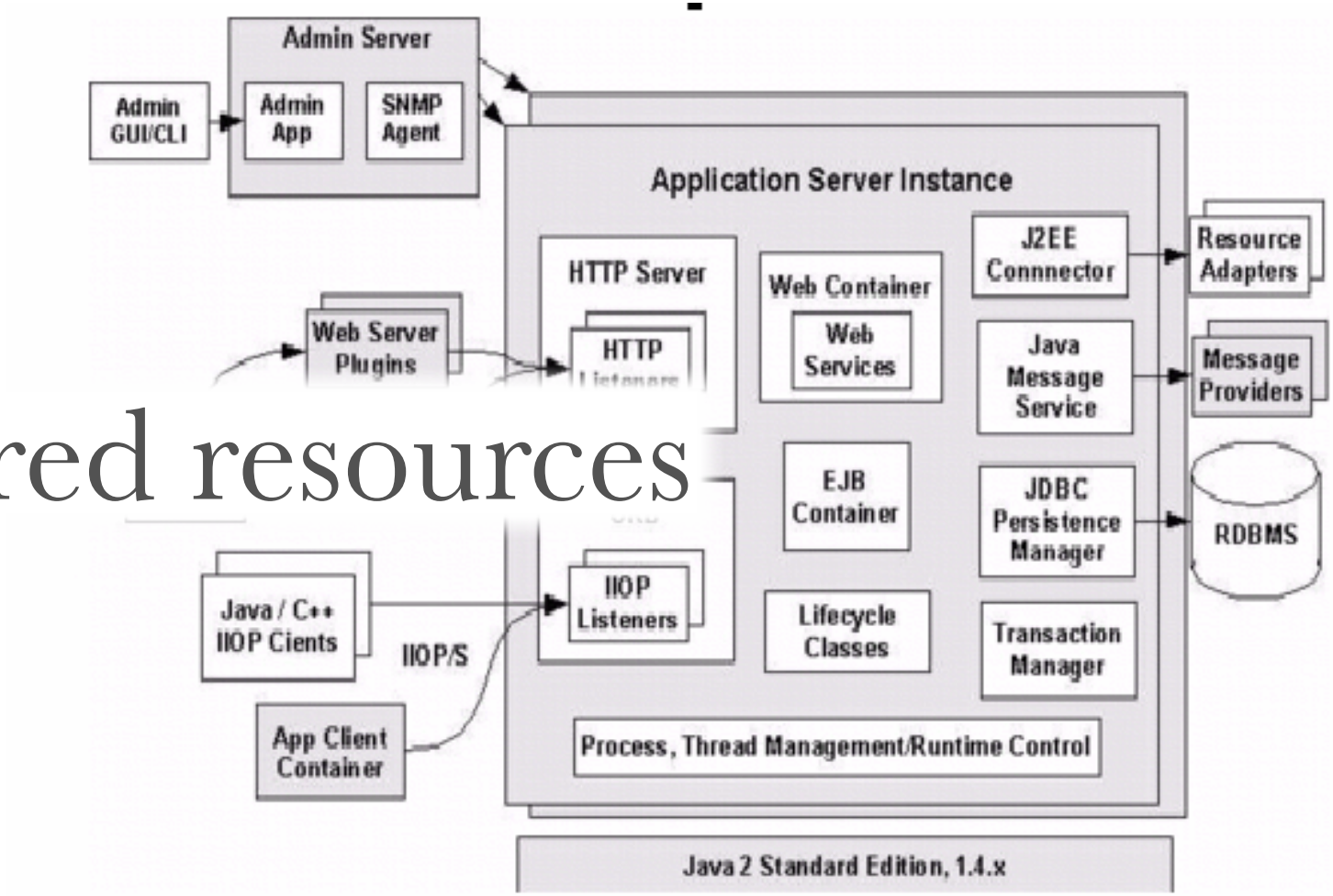
- automate regression testing
- separate what’s automatable from what’s

architectural trends

Yesterday's best practice is tomorrow's anti-pattern.

trends…

-



shared resources

…and reasons

shared resources

application servers

industrial strength database servers

layered architecture

data-level integration

coupling!

~~shared resources~~

prevents isolated changes

encourages big-bang deployments

makes upgrades difficult

prevents change


evolutionary
architecture &
emergent design

traditional role

product

dynamic

static

perceived concrete vision

agile role

product

dynamic

static

consequences of < time up front:

good feel for current events

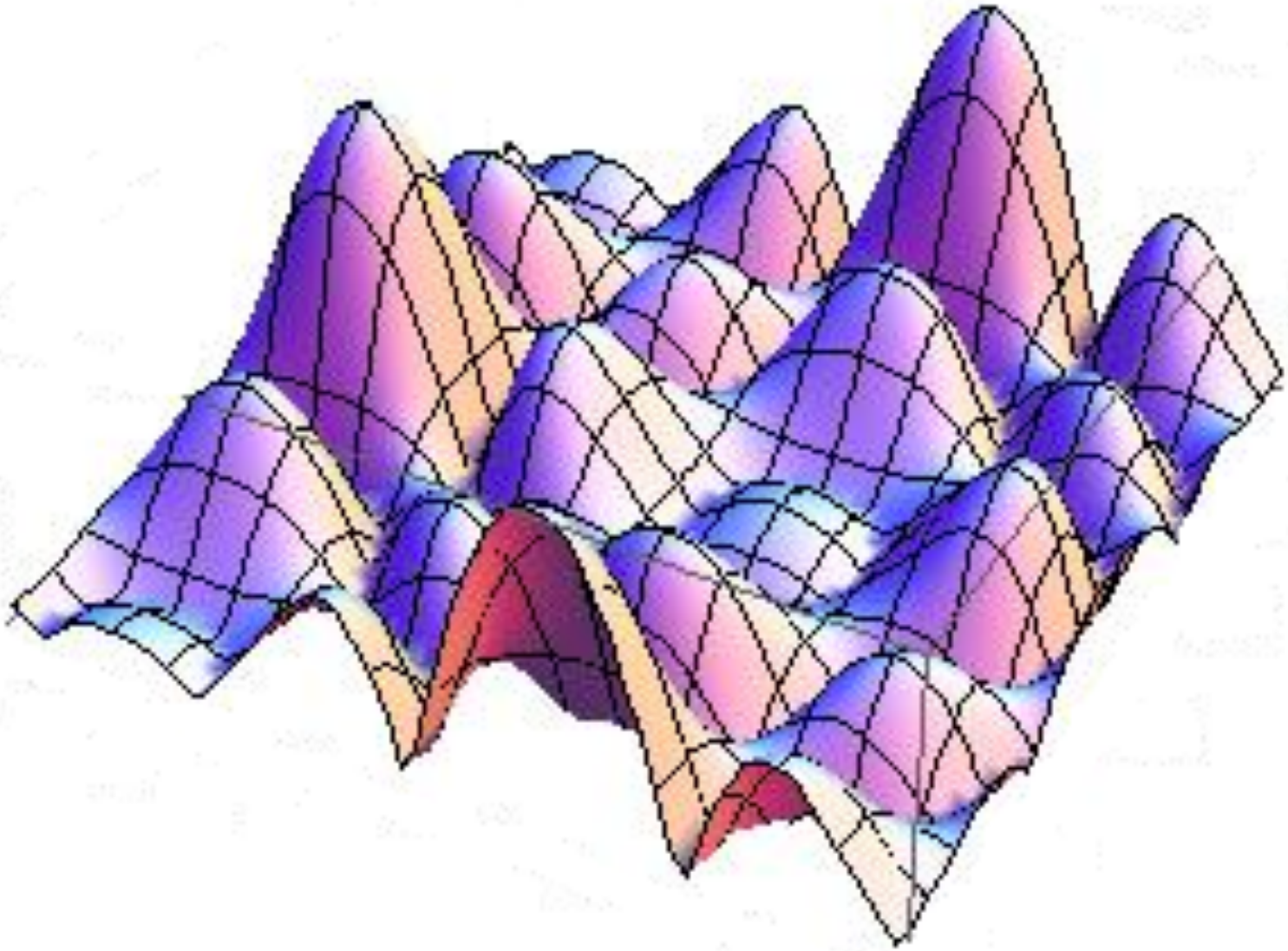
hands-on

macro <=> micro

fewer boat anchors

pivotable

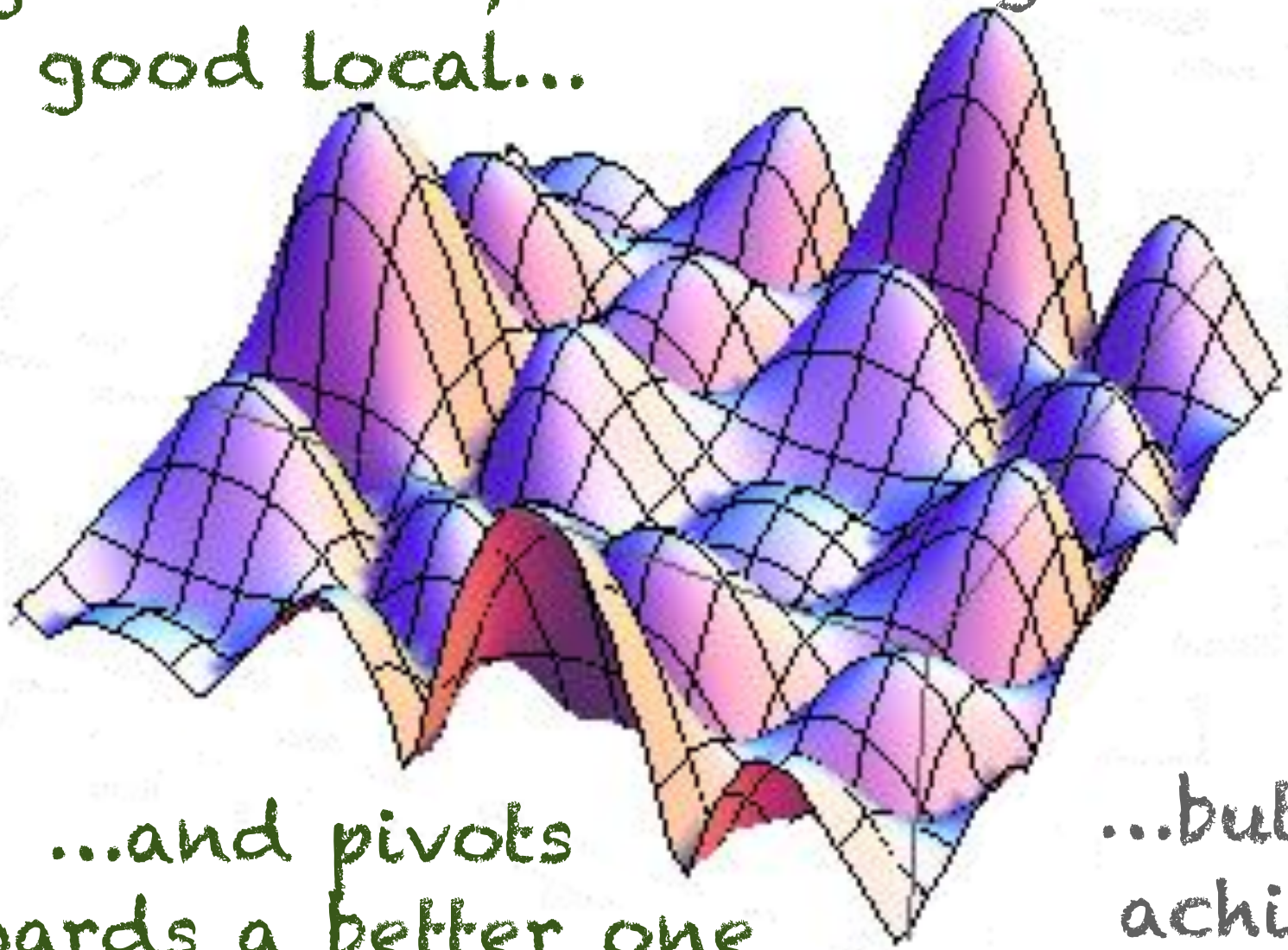
global vs local maxima



global vs local maxima

traditional shoots for global...

agile shoots for a good local...

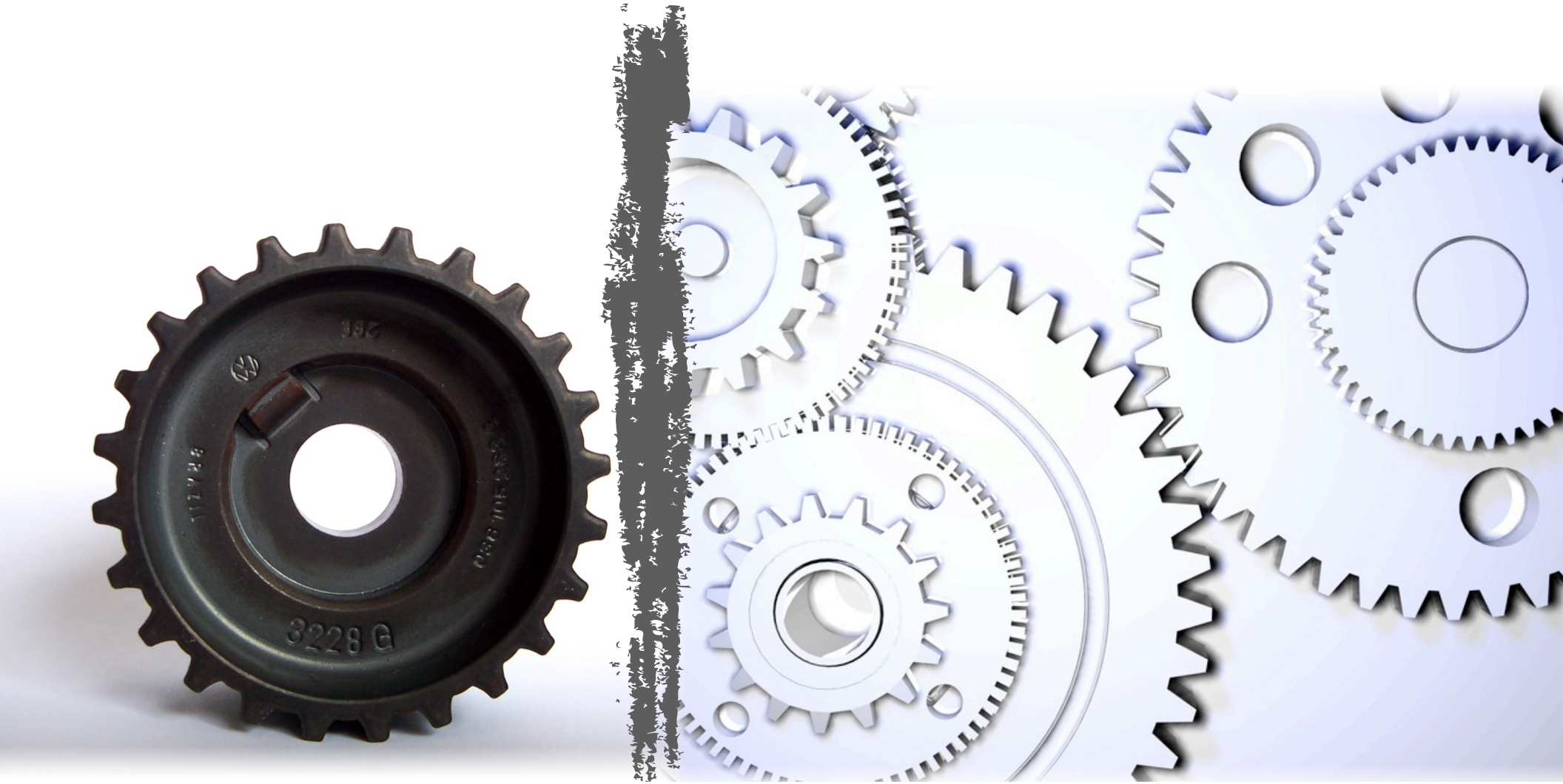


...and pivots towards a better one

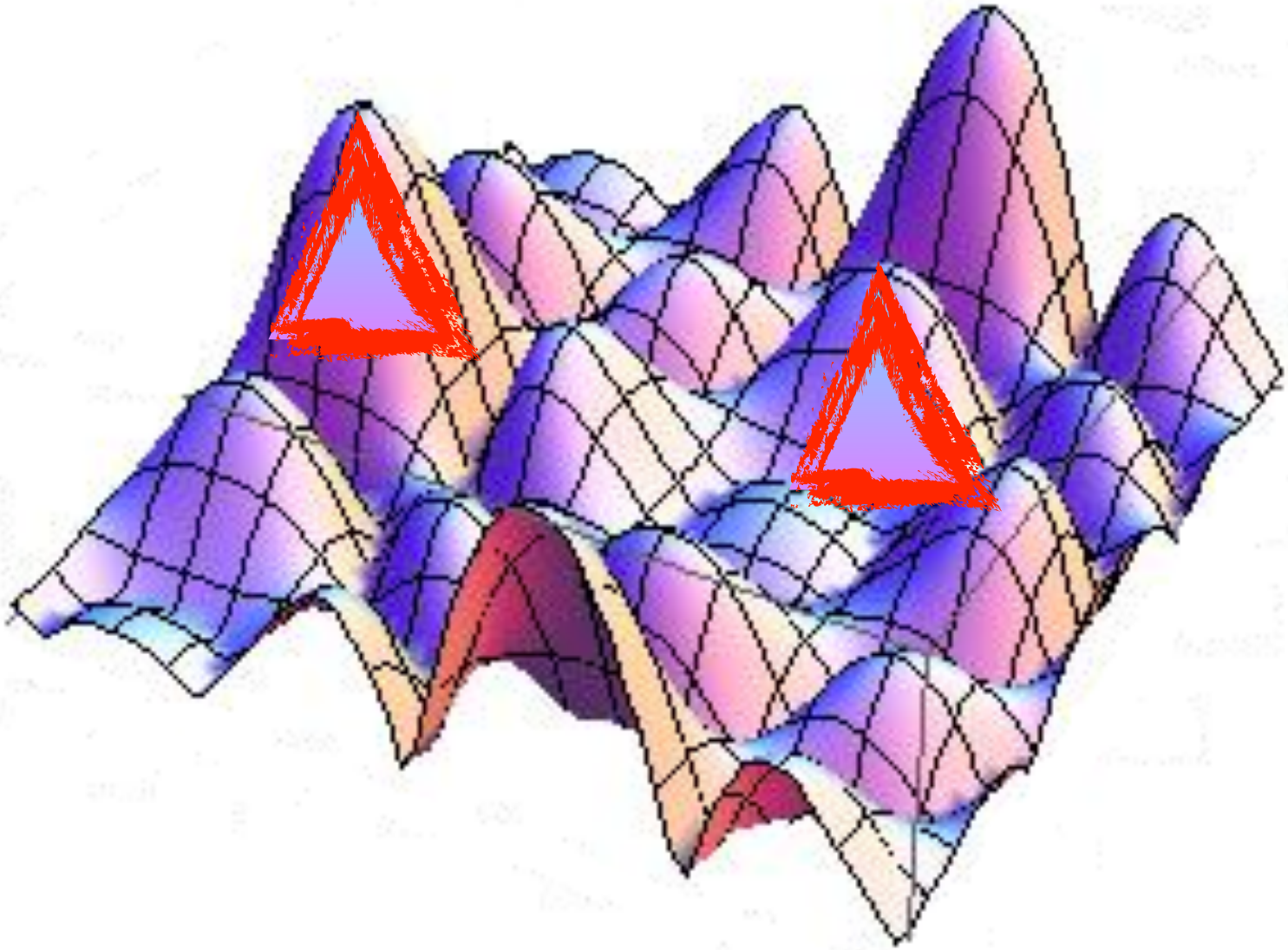
...but rarely achieves it

Minimum
Viable Product

Minimum *Feasible*
Product



pivot friendly architecture

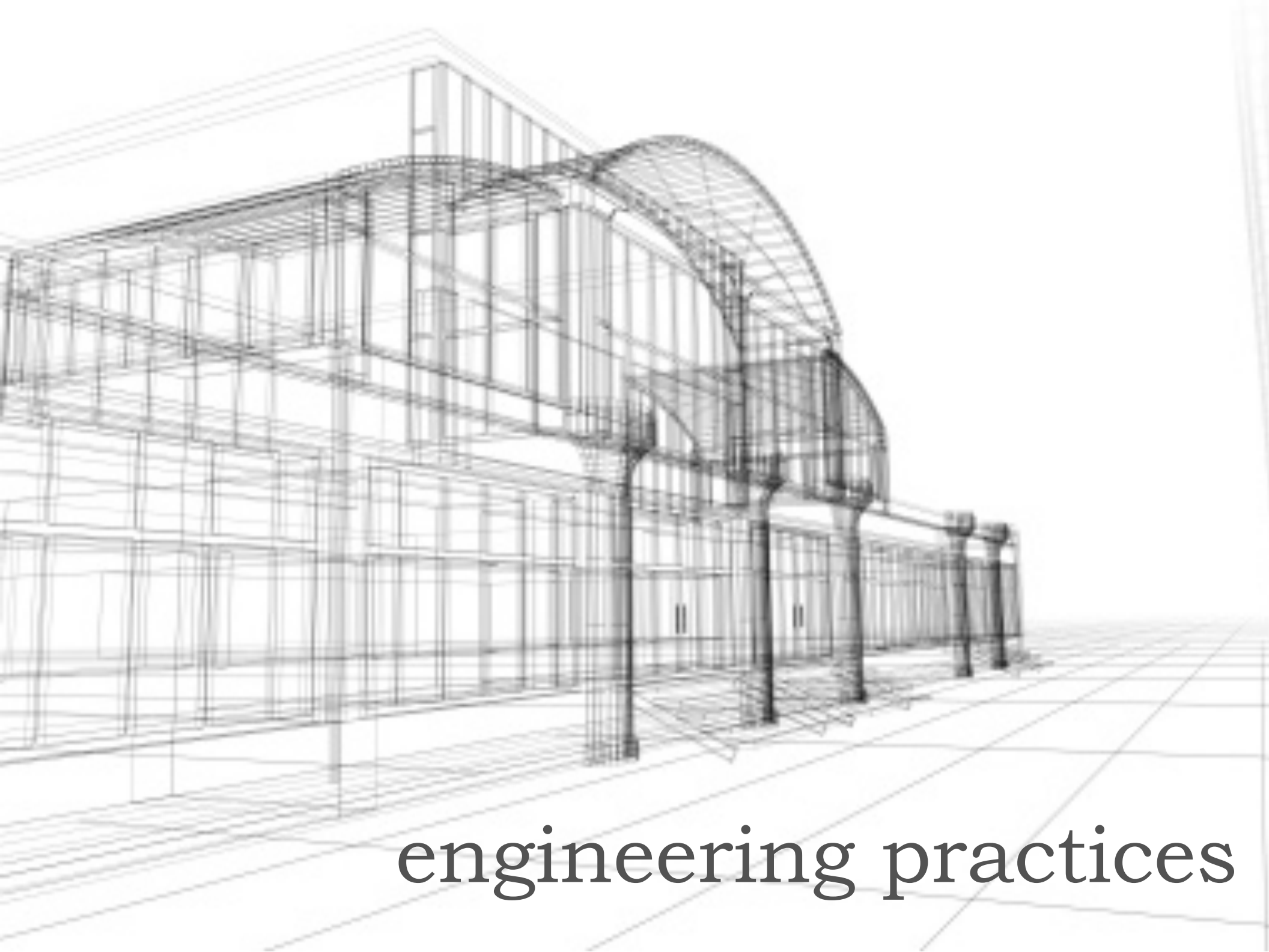


pivot-friendly architecture

designed to allow choices late in the development process

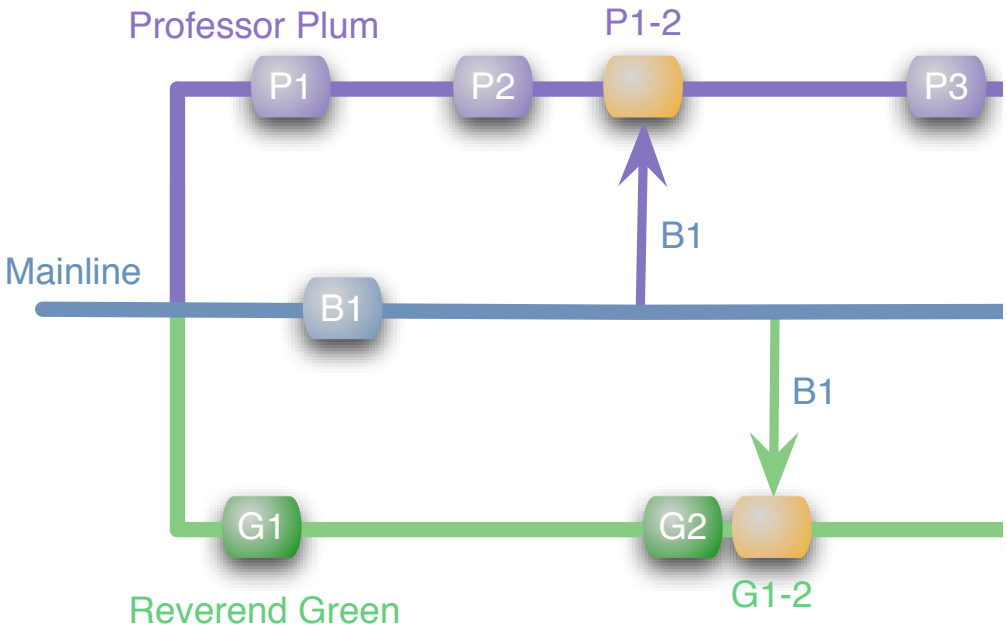
more architecture / design work than MVP

all layers (including user experience) must create tight feedback loops

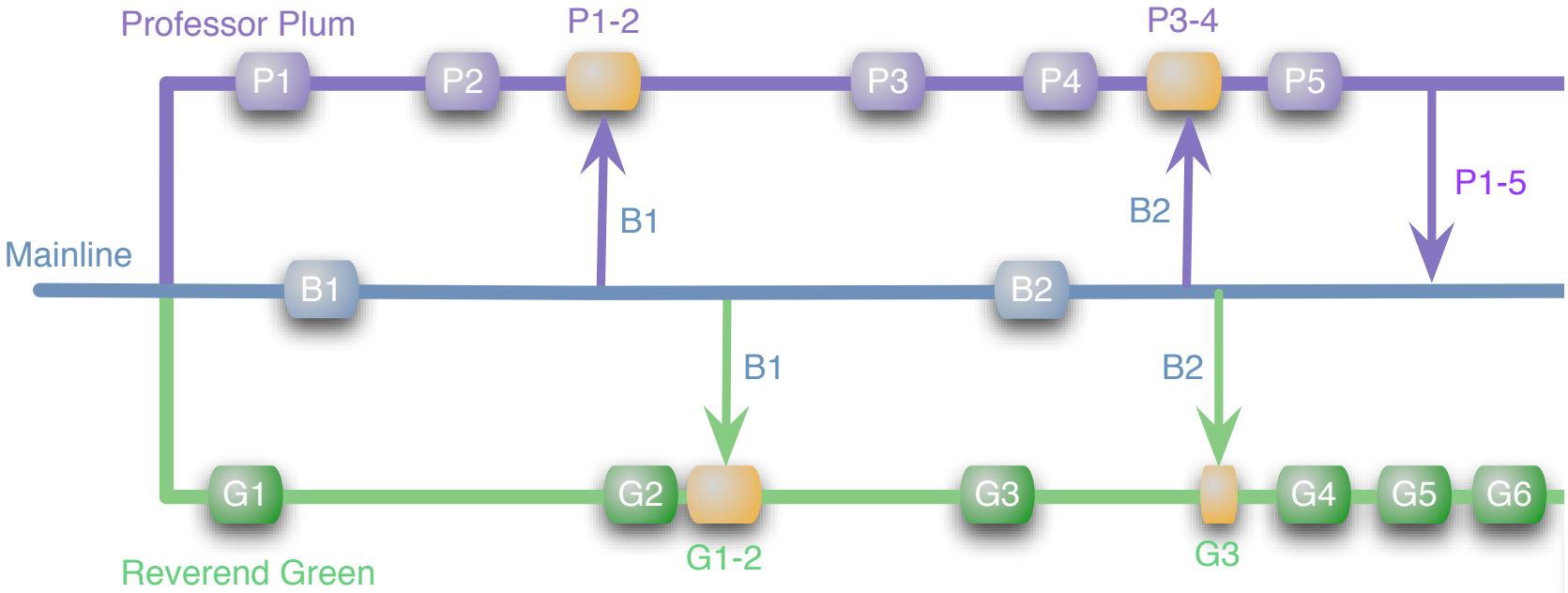


engineering practices


hide new functionality
until it is finished

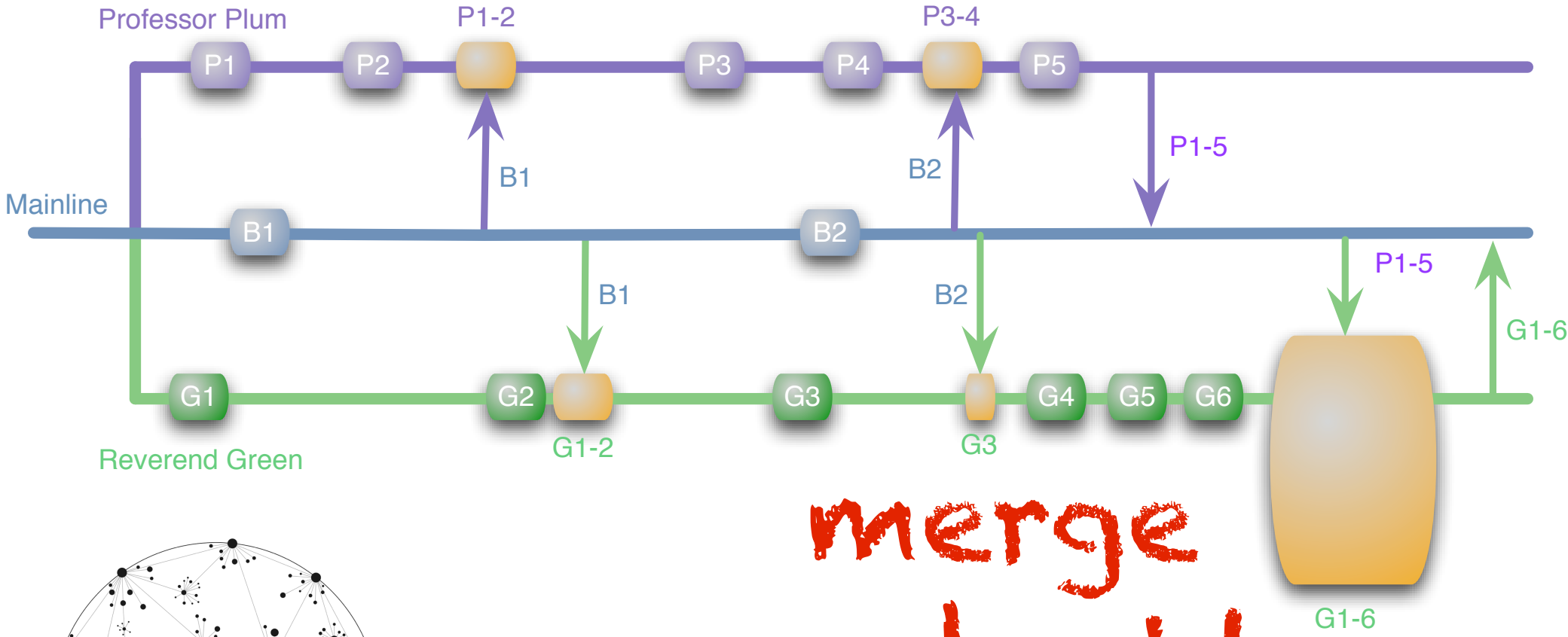


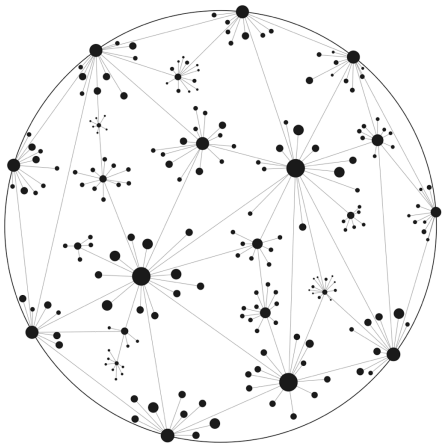
Feature Branching



Feature Branching

copy/paste reuse !!

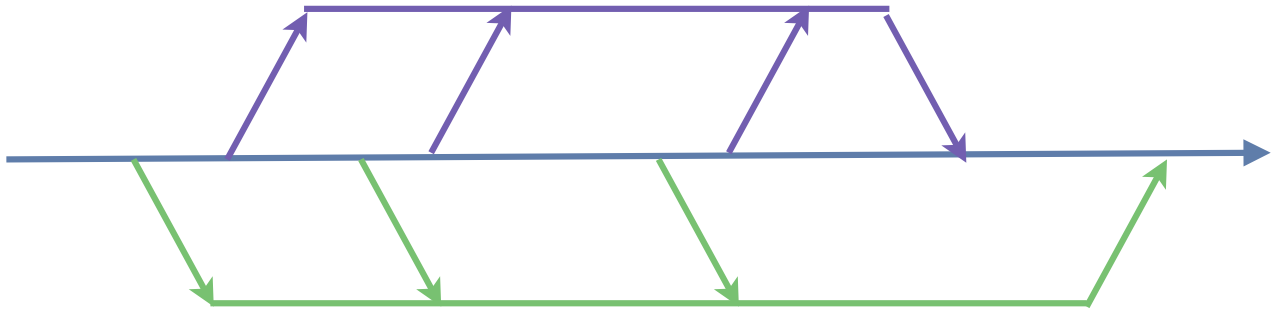




merge ambush!

Feature Branching

Feature Branch



Continuous Integration

Feature Toggles

Config File

```
[featureToggles]
wobblyFoobars: true
flightyForkHandles: false
```

some.jsp

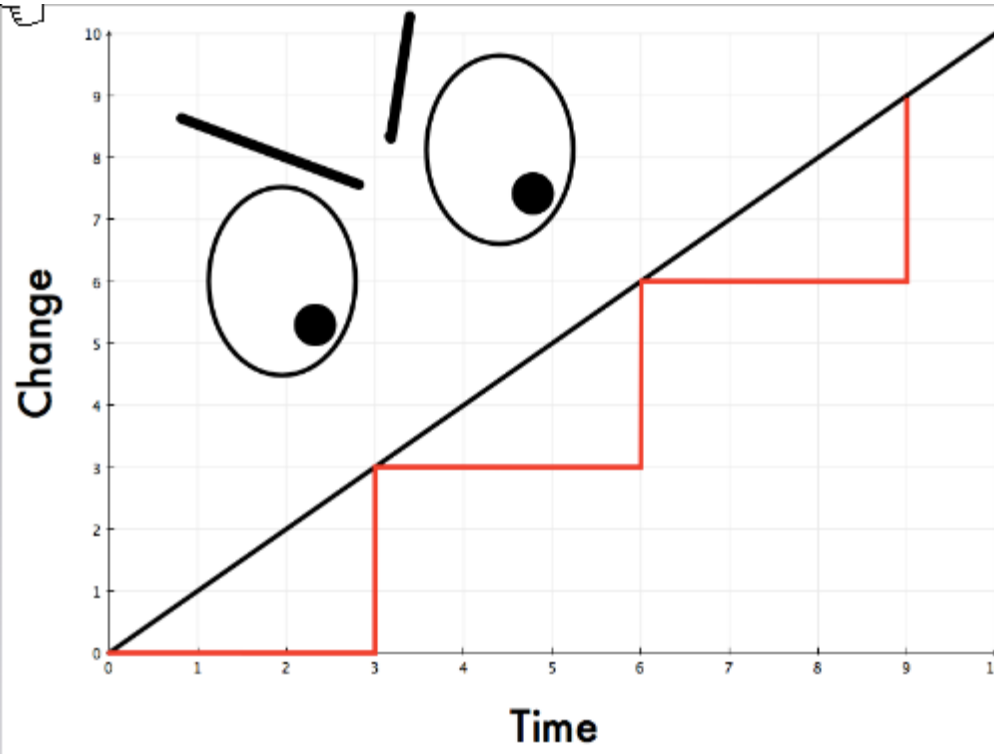
```
<toggle name=wobblyFoobars>
  . various UI elements
</toggle>
```

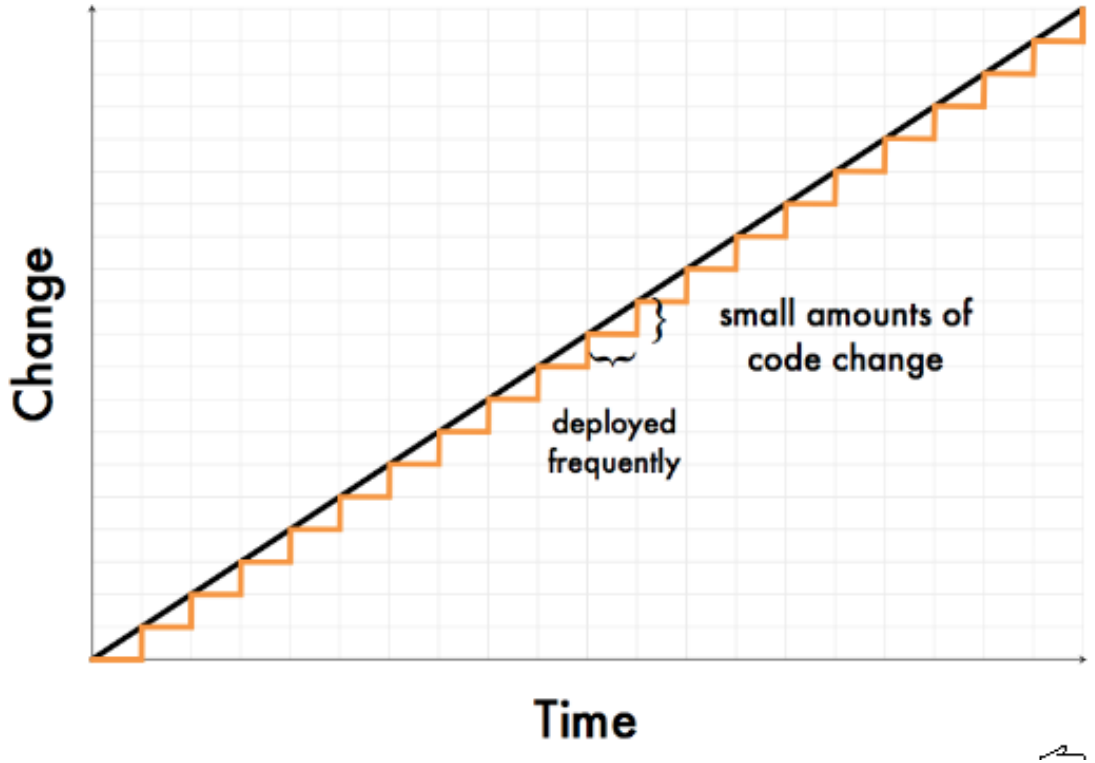
other.java

```
forkHandle = (featureConfig.isOn(‘flightlyForkHandles)) ?
                new FlightyForkHander(aCandle) :
                new ForkHandler(aCandle)
```

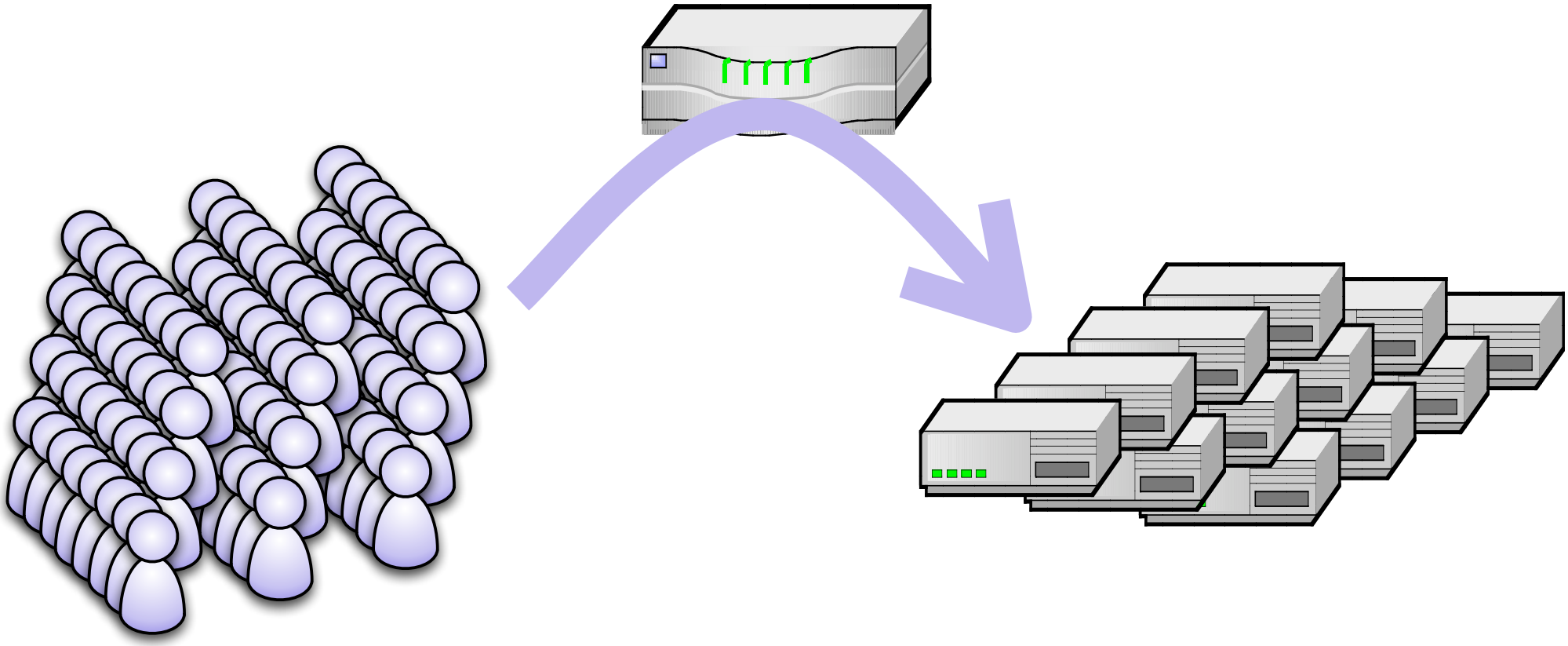

Togglz - Features flag for Java
www.togglz.org
Reader
Feature Flags for the Java plattform
MAIN
Home
Downloads
Source Code
Forums
Issue Tracker
stackoverflow.com
Continuous Integration
License
REFERENCE
What's new?
Getting Started
Javadocs 2.0.0.Final
Javadocs 1.1.0.Final
Javadocs 1.0.0.Final
Updating Notes
DOCUMENTATION
Togglz
What is it about?
Togglz is an implementation of the Feature Toggles pattern for Java. Feature Toggles are a very common agile development practices in the context of continuous deployment and delivery. The basic idea is to associate a toggle with each new feature you are working on. This allows you to enable or disable these features at application runtime, even for individual users.
Want to learn more? Have a look at an usage example or check the quickstart guide.
News
01-Jul-2013
Togglz 2.0.0.Final released
I'm very happy to announce the release of Togglz 2.0.0.Final. This new version is the result of many months of hard work. Many core concepts of Togglz have been revised to provide much more flexibility.
The most noteworthy change in Togglz 2.0.0.Final is the new extendible feature activation mechanism that allows to implement custom strategies for activating features. Beside that there are many other updates.

Make all changes incrementally as a series of small changes, each of which is releasable.

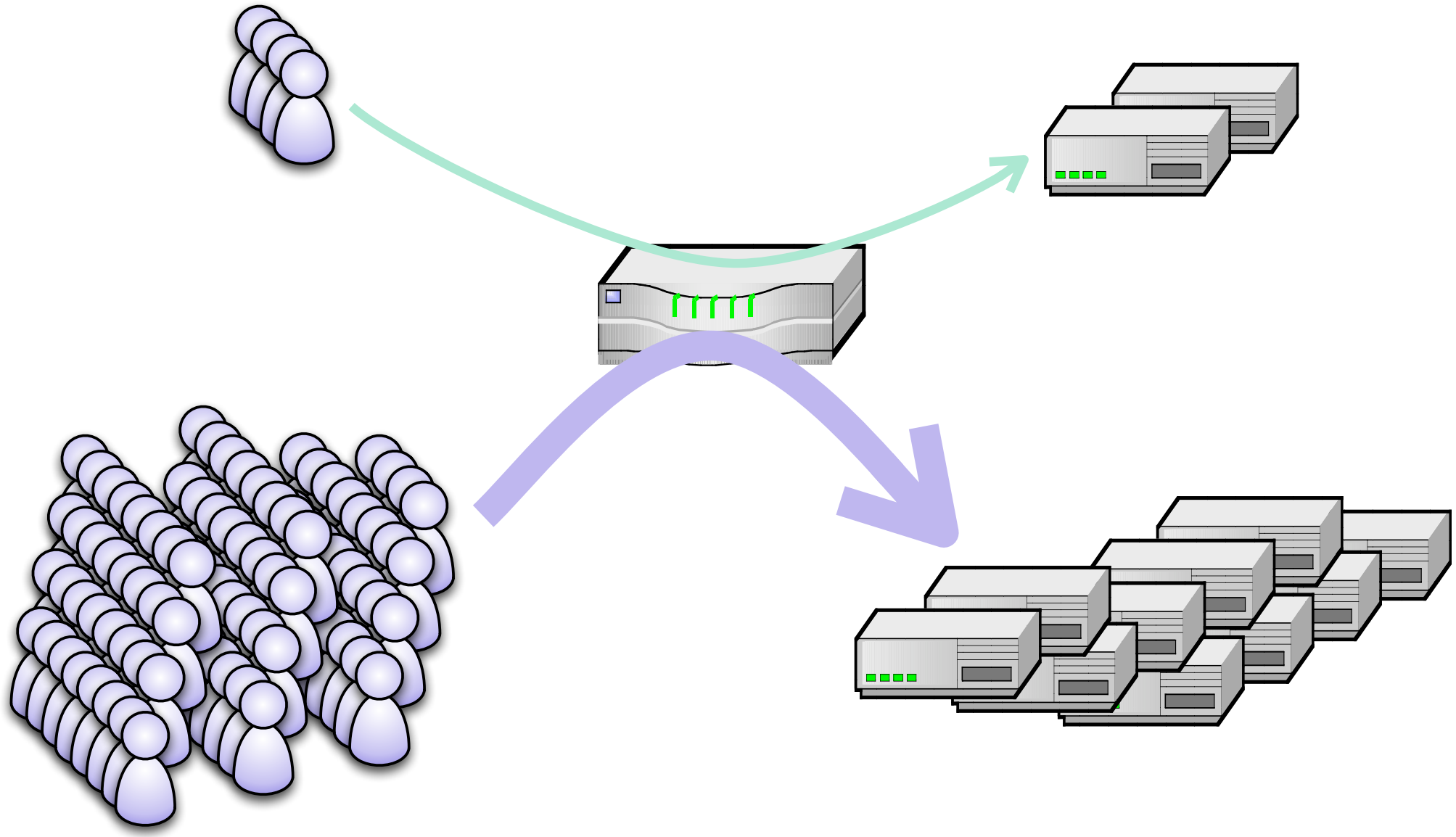




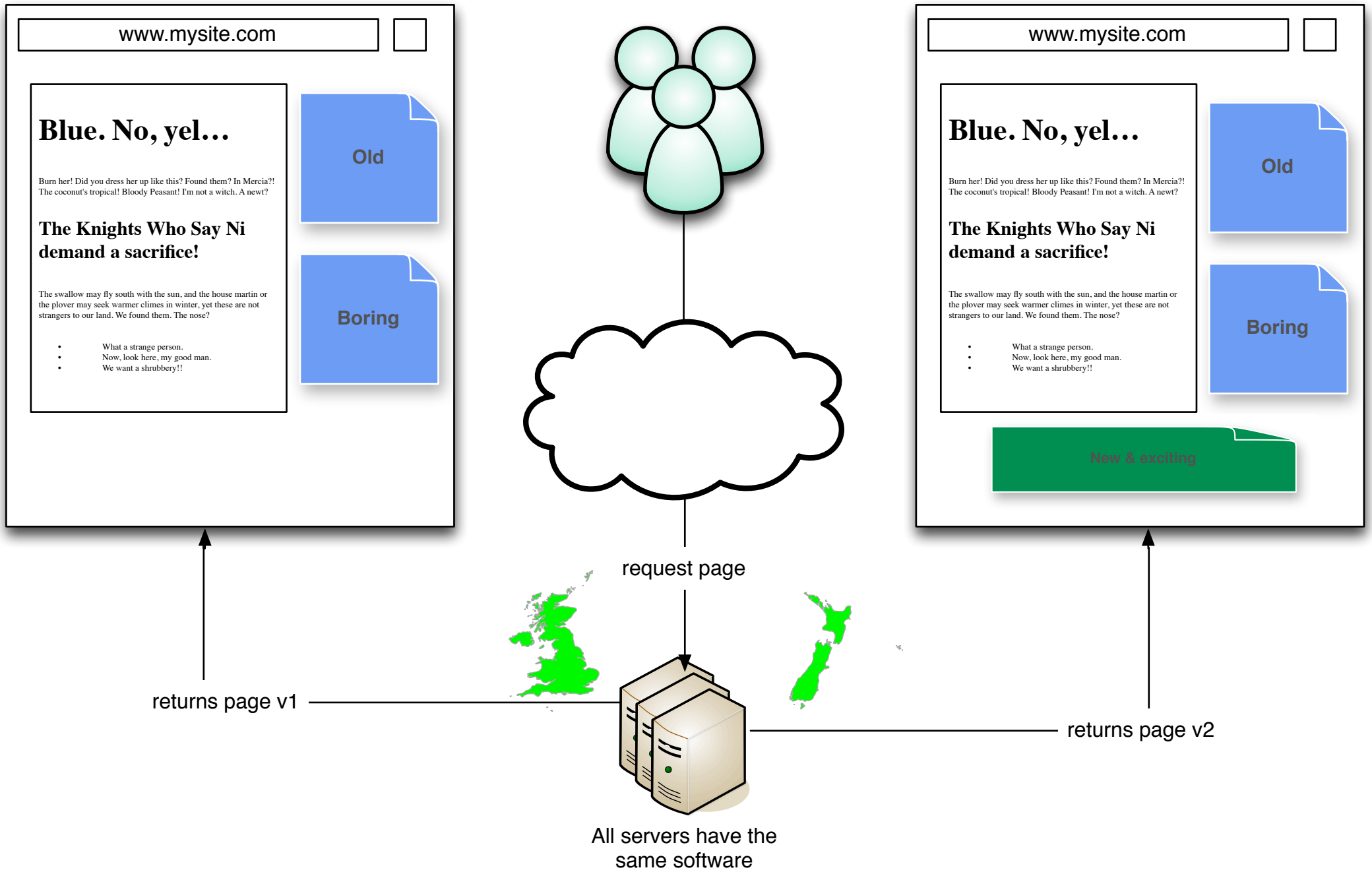
canary releasing

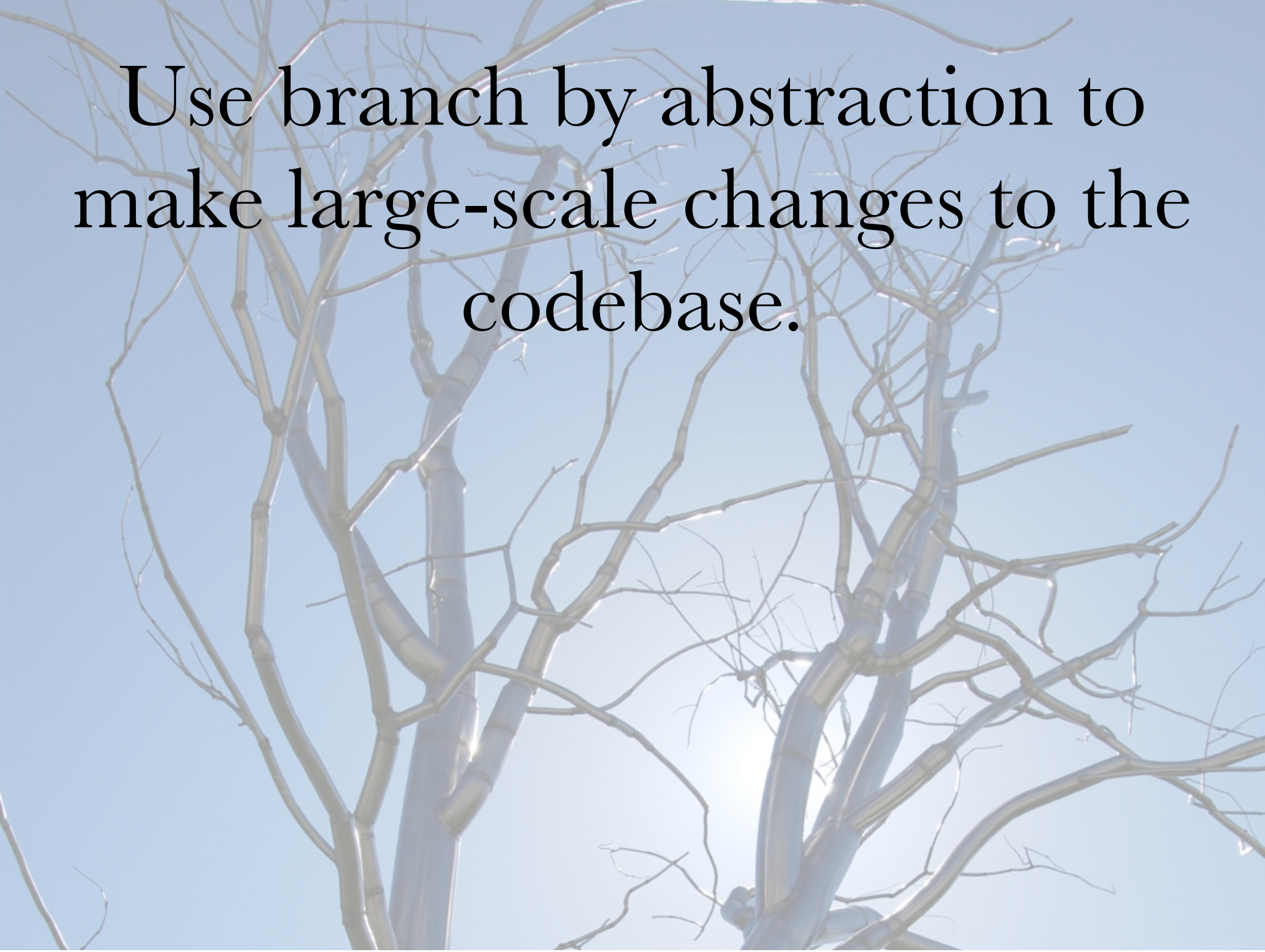


canary releasing

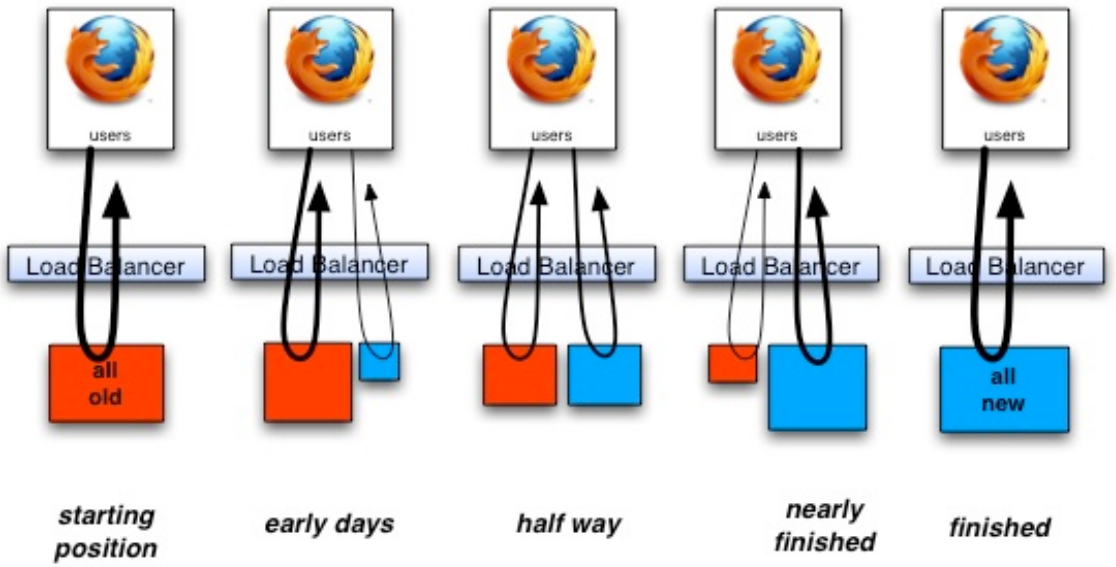


dark launching




Use branch by abstraction to make large-scale changes to the codebase.

“strangler” pattern

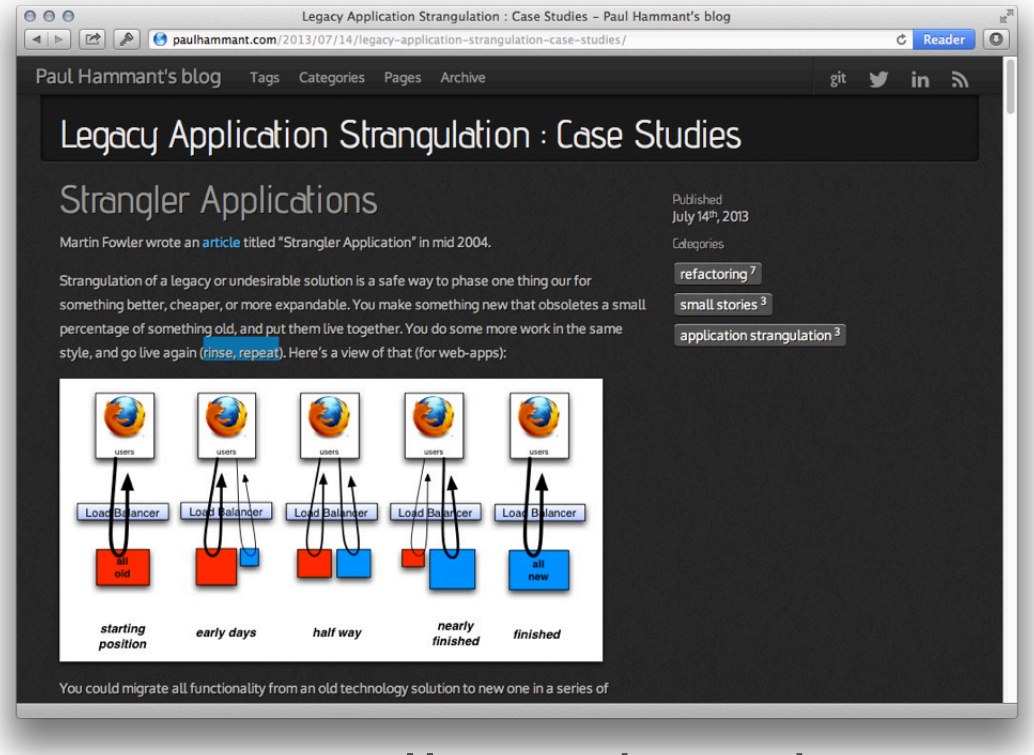


make something new that obsoletes a small percentage of something old

put them live together

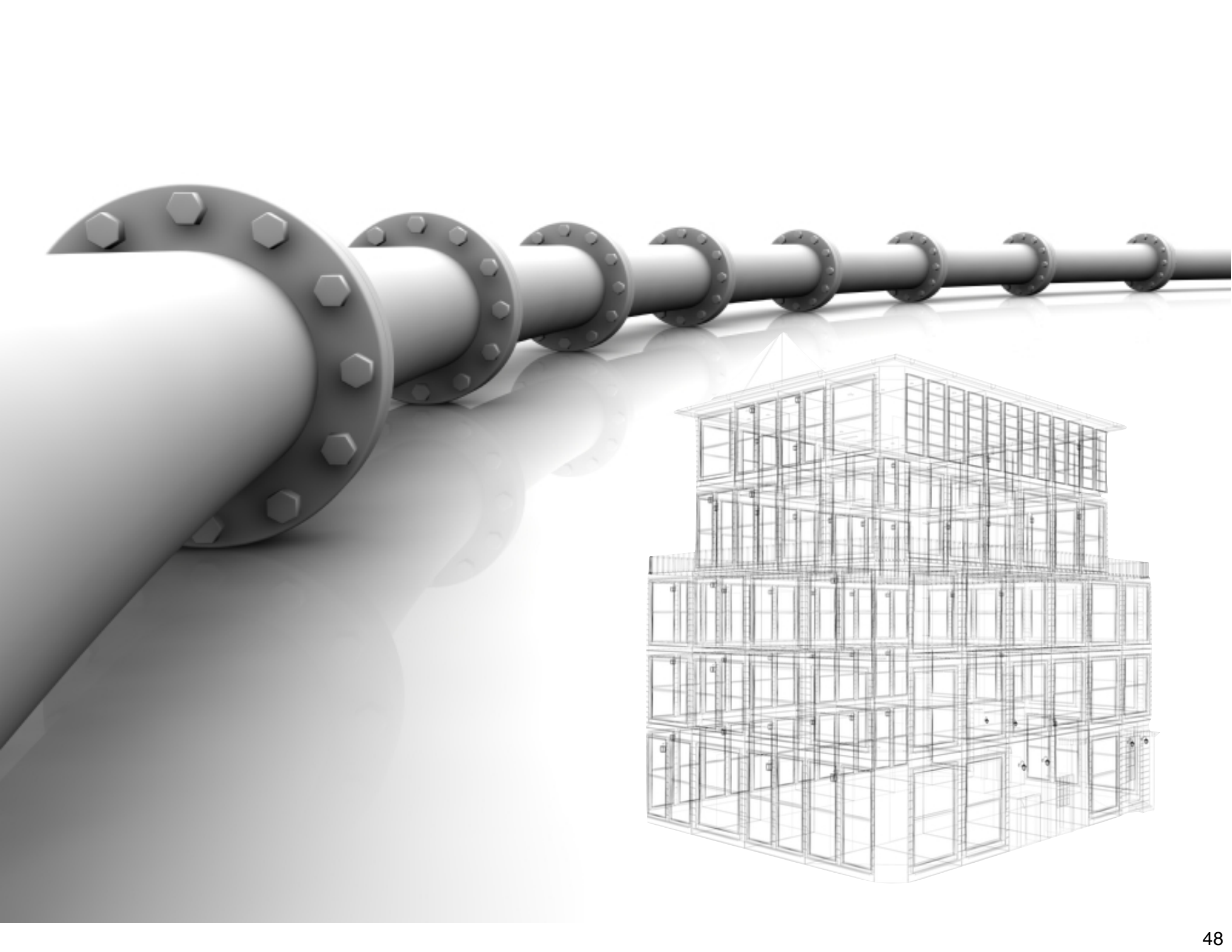
rinse, repeat

“strangler” pattern



“We never told the users that they must use the new system. Nor did we remove access to the old system. We relied on making the system so compelling that there was no reason to use the old. This also meant that we stayed focused on the users real requirements”

paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/



?'s

Mark Richards

Independent Consultant

Hands-on Enterprise / Integration Architect
Published Author / Conference Speaker

http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9

Published Books:

Java Message Service, 2nd Edition
97 Things Every Software Architect Should Know
Java Transaction Design Strategies

Neal Ford

Director / Software Architect /
Meme Wrangler

ThoughtWorks®

2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1 40 4242 9929 Twitter: @neal4d
E: nford@thoughtworks.com W: thoughtworks.com