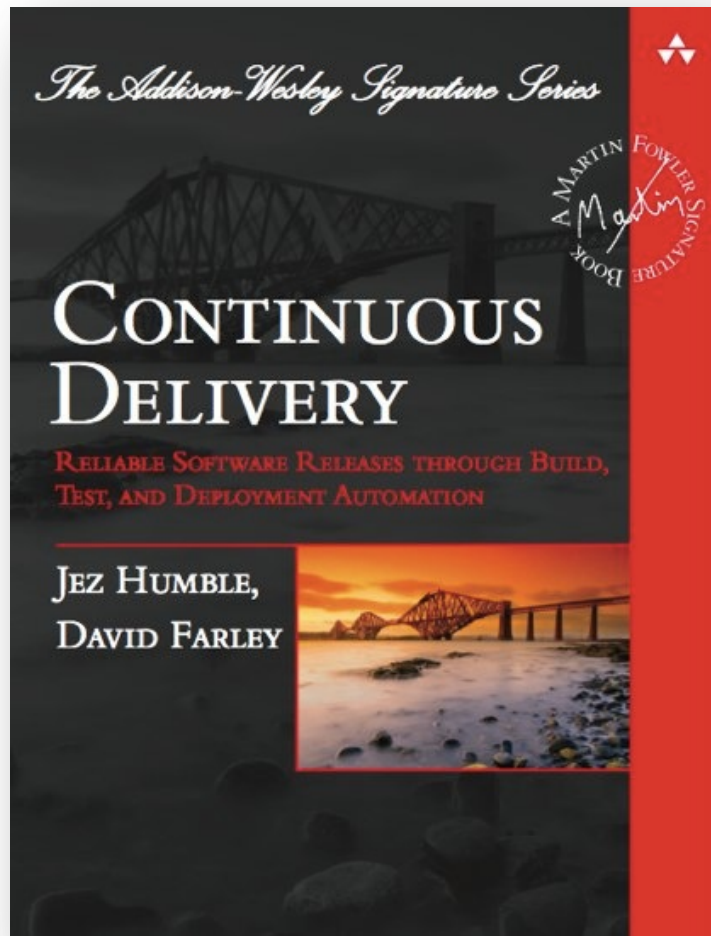


# Continuous Delivery Workshop

Tests, Synergistic  
Practices, &  
Deployment

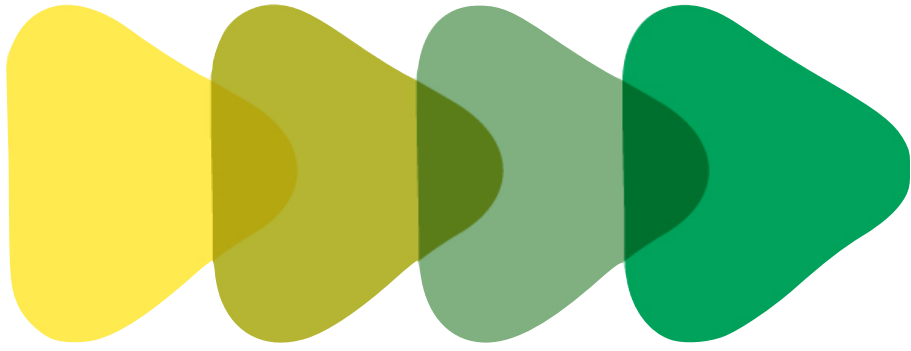


ThoughtWorks®

**NEAL FORD**

*Director / Software Architect / Meme Wrangler*

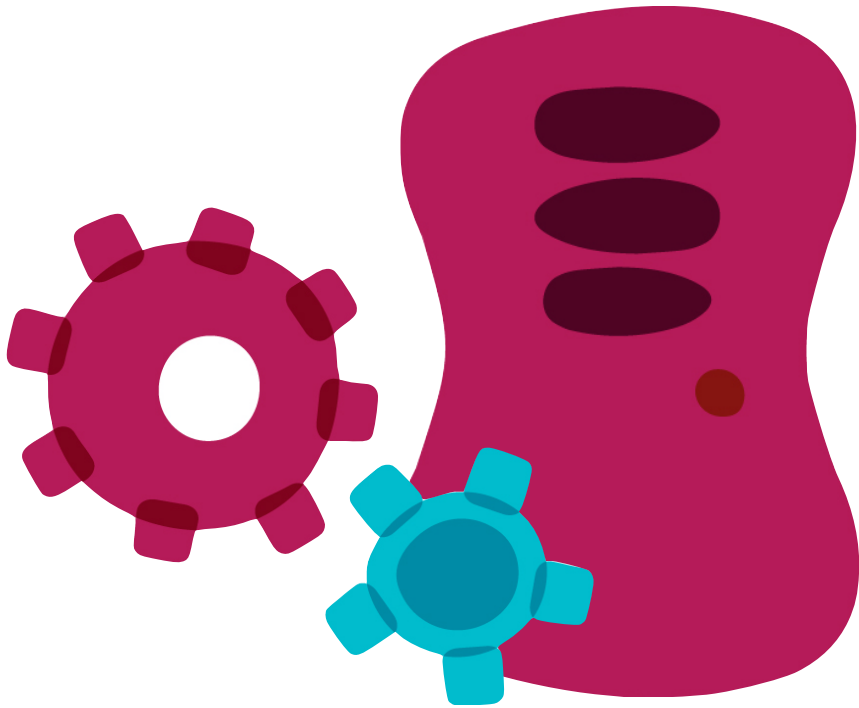
*Workshop materials created by Jez Humble, Martin Fowler, Tom Sulston, & Neal Ford*



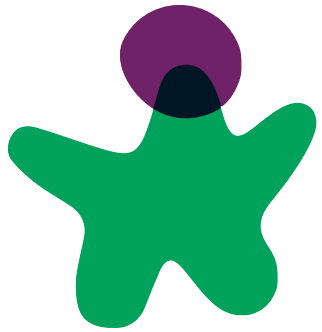
deployment pipelines



tests, synergistic practices,  
incremental deployment



data & infrastructure



effort



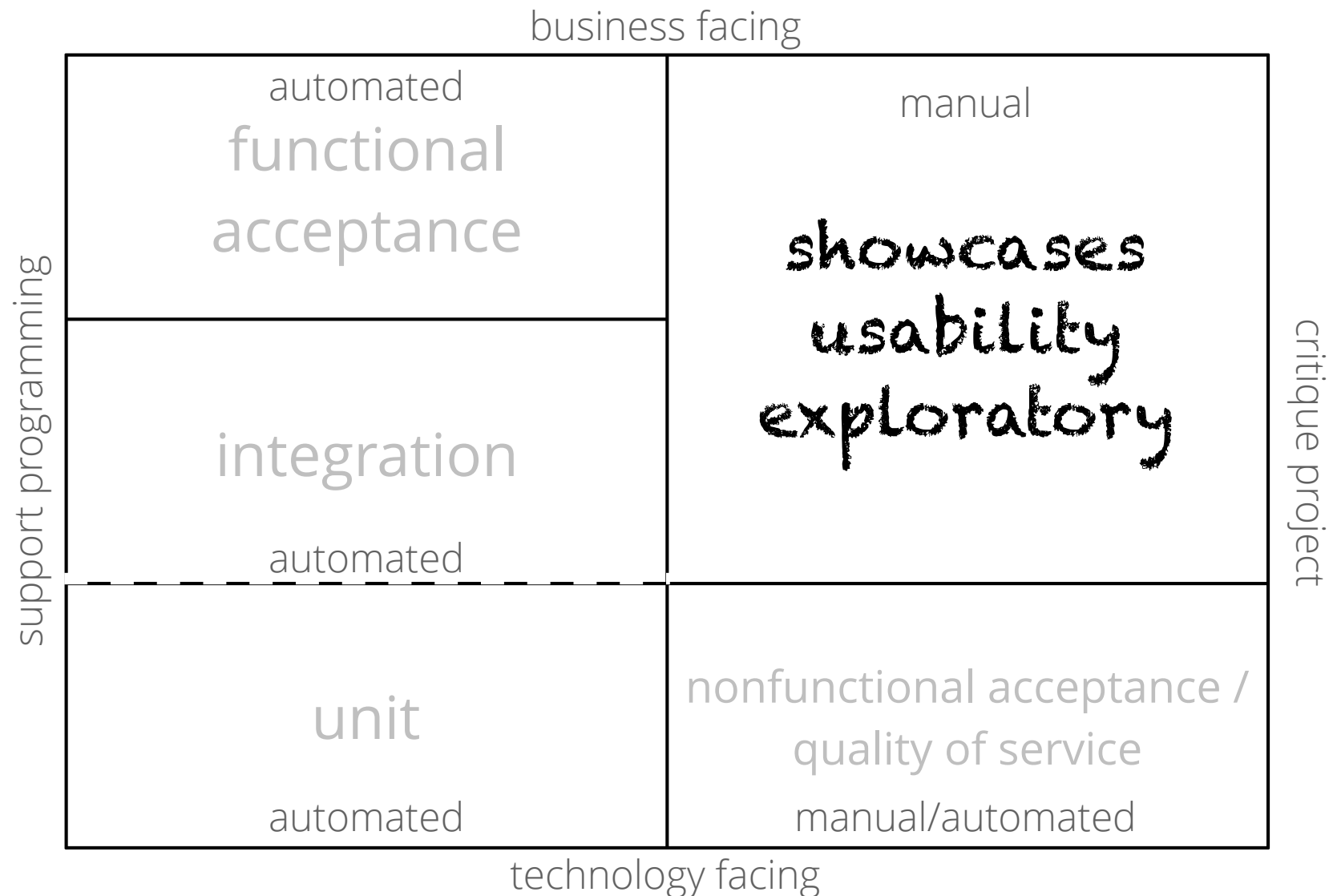
audience



feedback

Testing

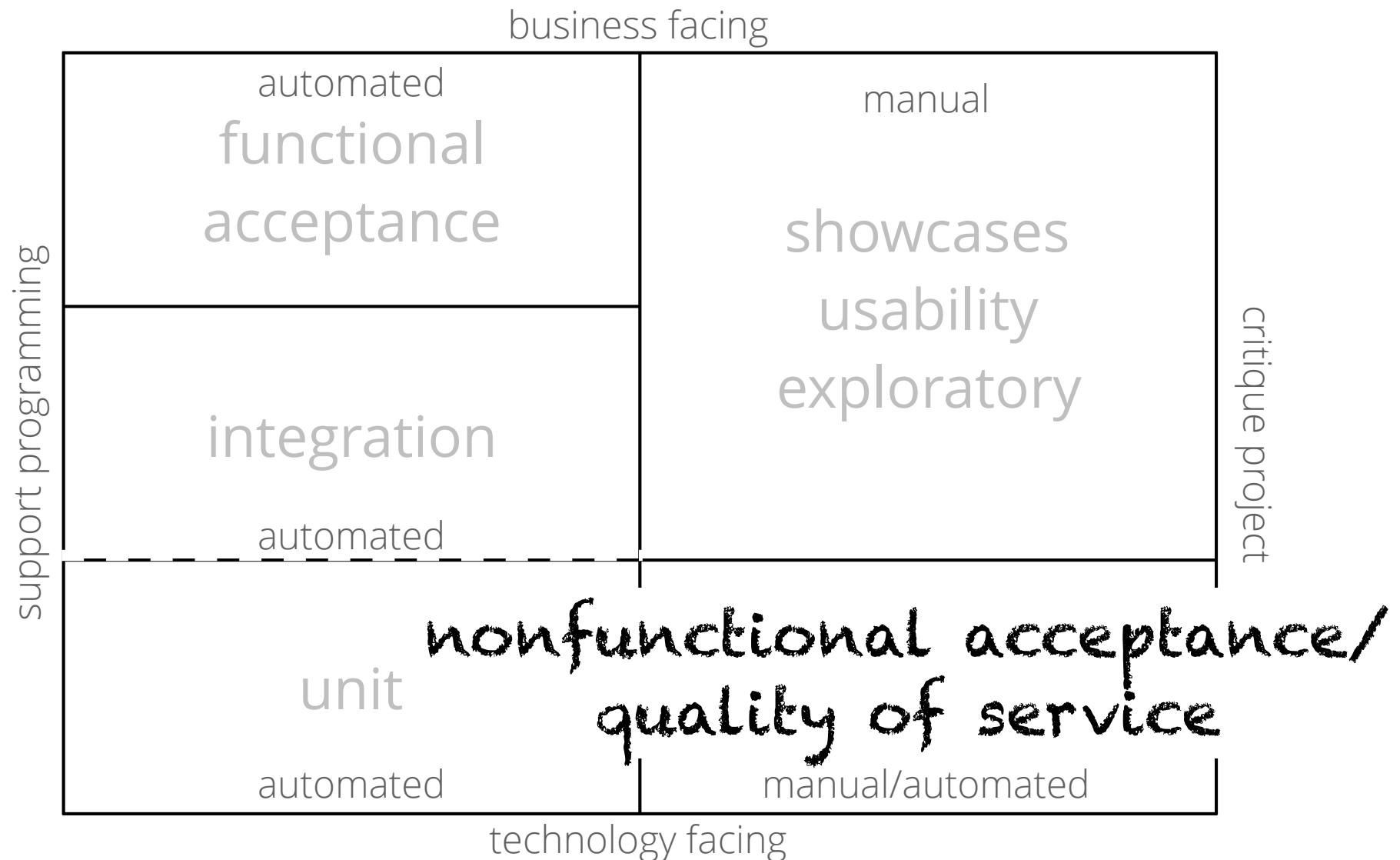
# Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

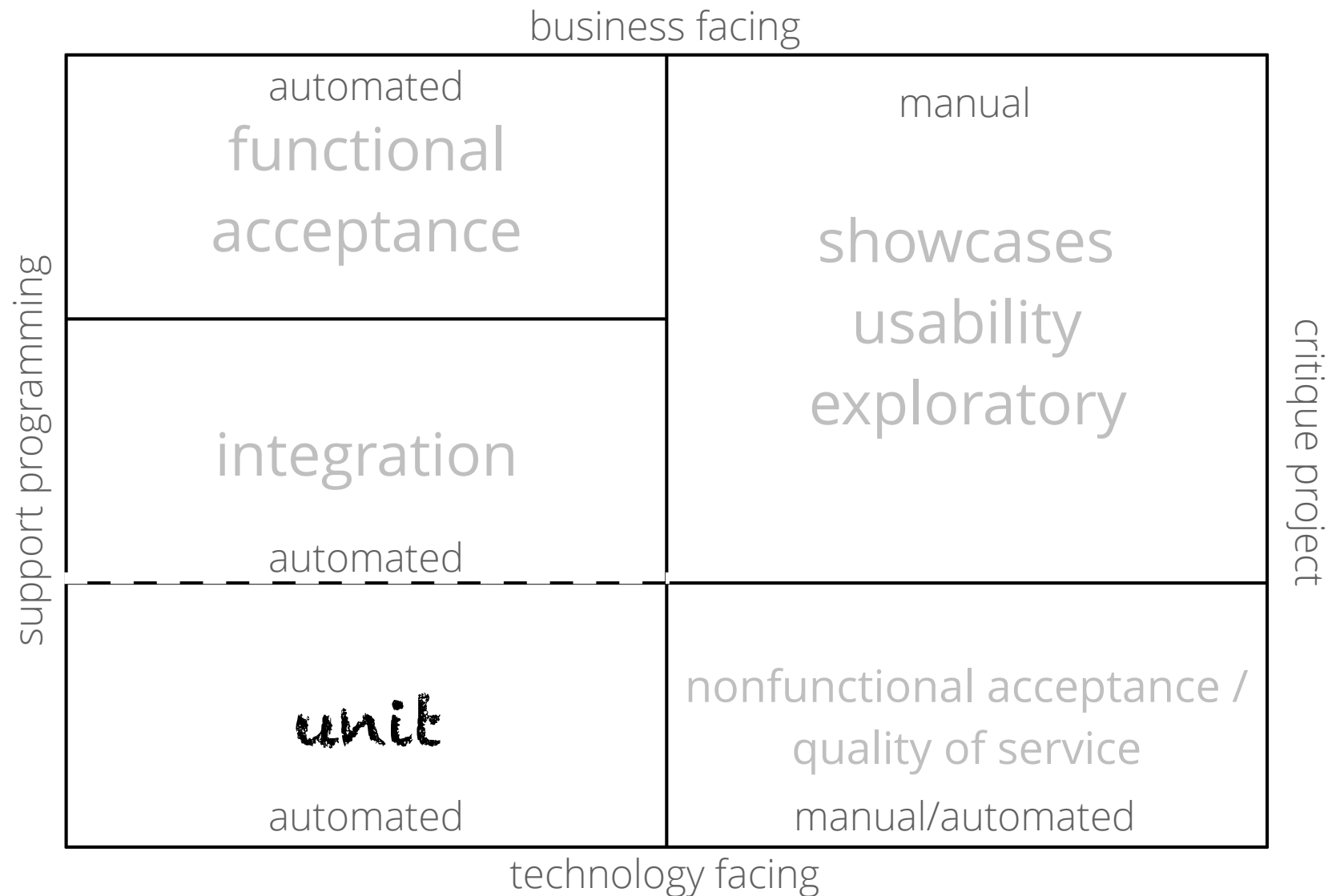


# Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

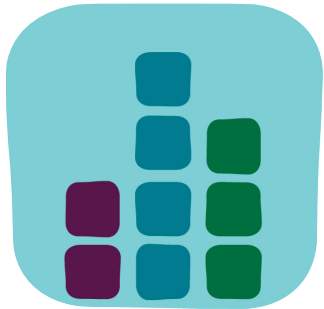
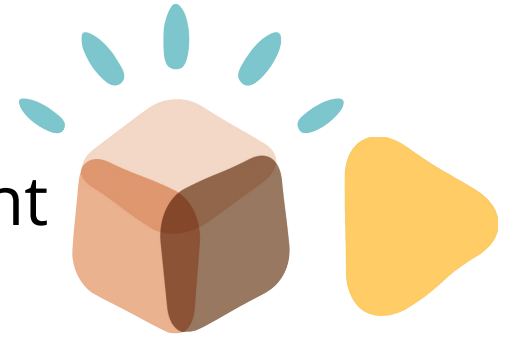
# Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

# unit Testing

prefer test-driven to test-after development

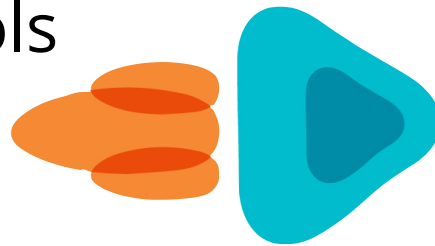


prefer pragmatism over dogmatic metrics

optimize for the target audience



use sharp tools



how much time?

# Common Anti-pattern

mixed unit/  
functional tests



# Fast/Slow Tests

mixed unit/  
functional tests

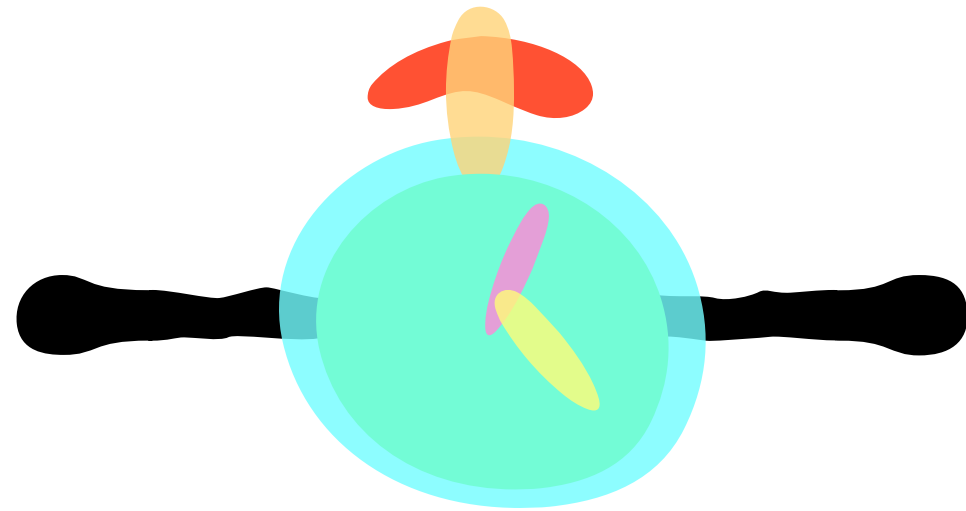


functional  
unit

A thick, black, wavy horizontal line that spans the width of the text. It has a slightly irregular, hand-drawn appearance.

# Fast/Slow Tests Redux

mixed unit/  
functional tests





zsh



ruby

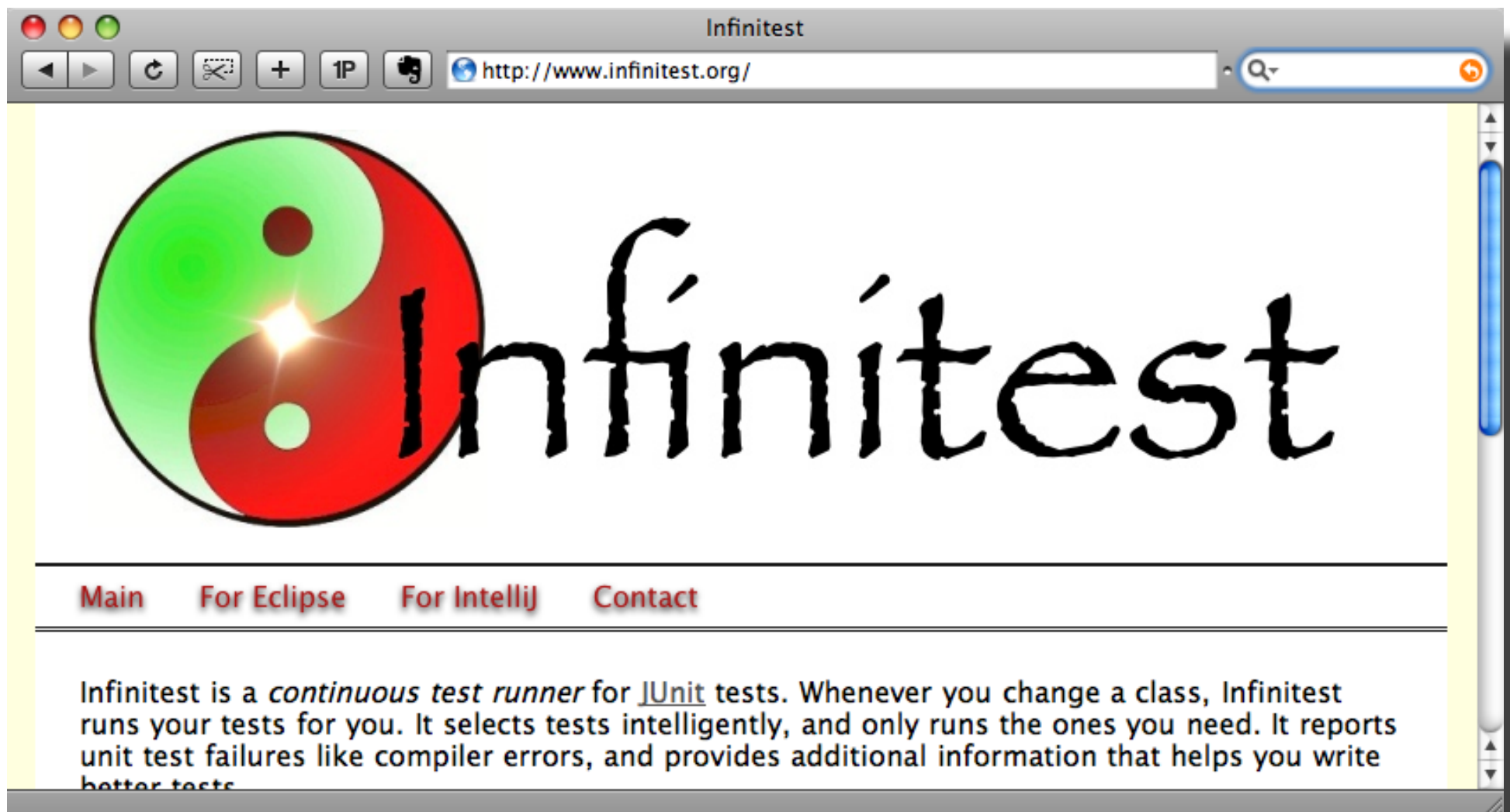
.....  
Finished in 41.353372 seconds.

8996 tests, 12746 assertions, 0 failures, 0 errors

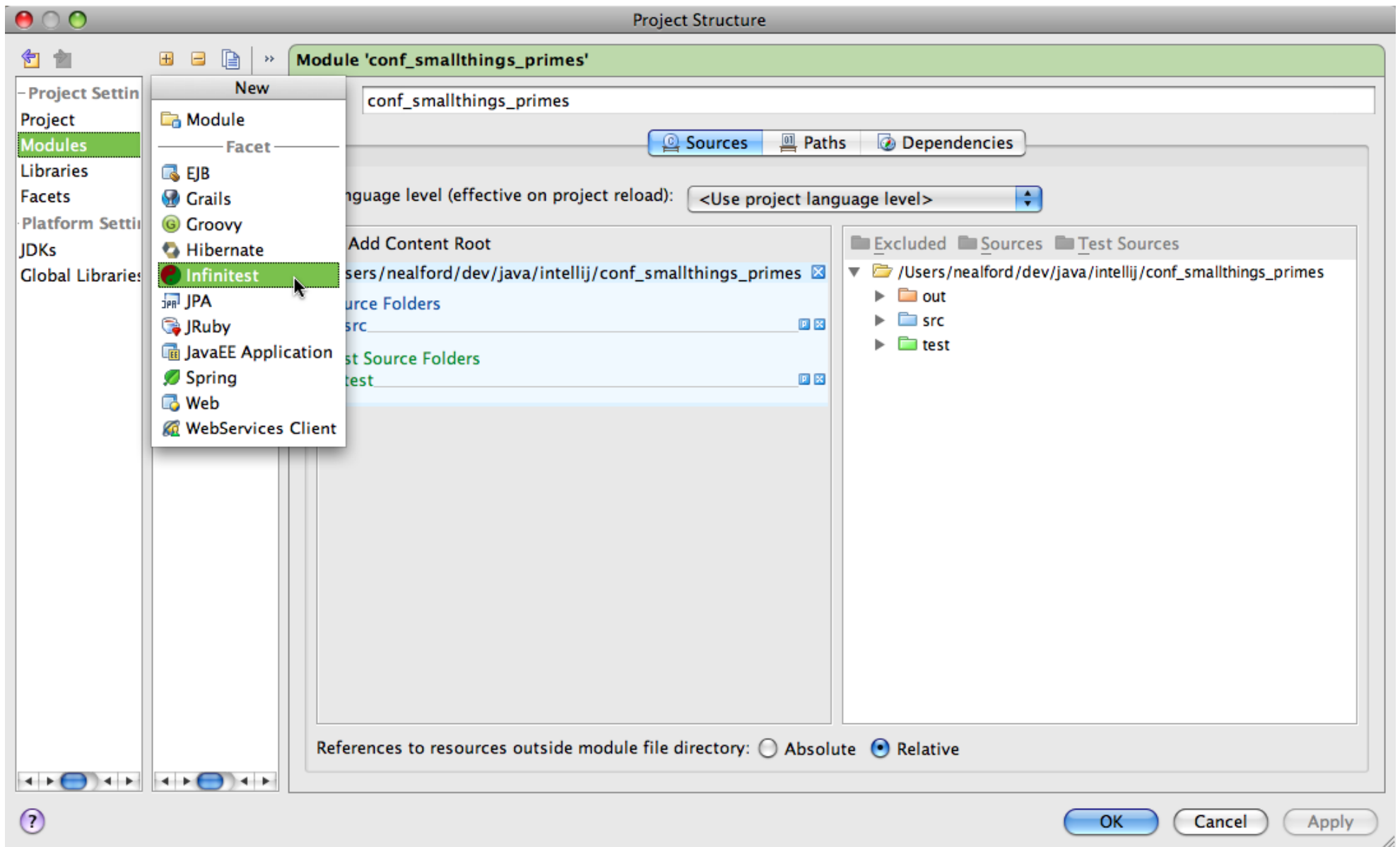
.....  
Finished in 248.573856 seconds.

3964 tests, 8288 assertions, 0 failures, 0 errors





[code.google.com/p/infinitest/](http://code.google.com/p/infinitest/)



conf\_smallthings\_primes - [/Users/nealford/dev/java/intellij/conf\_smallthings\_primes] - [conf\_smallthings\_primes] - .../tes...

FactorsFinderTest

Project /Users/nealford/dev/j... View as: Project

conf\_smallthings\_primes

- conf\_smallthings\_primes (/Users/nealford/dev/ja
  - src
  - test
    - com.nealford.smallthings.primes
      - FactorsFinderTest
- conf\_smallthings\_primes.iml
- conf\_smallthings\_primes.ipr
- conf\_smallthings\_primes.iws

Libraries

Finder.java x FactorsFinder.java x FactorsFinderTest.java x

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

```
+import ...

@SuppressWarnings({"unchecked"})
public class FactorsFinderTest {

    @Test public void is_factor_the_old_way() {
        FactorsFinder ff = new FactorsFinder(10);
        assertTrue(ff.isFactor(1));
    }

    @Test public void is_factor() {
        FactorsFinder f = new FactorsFinder(25);
        assertFalse(f.isFactor(4));
        assertTrue(f.isFactor(5));
        assertTrue(f.isFactor(12));
    }

    @Test public void factors_for_the_old_way() {
        FactorsFinder f = new FactorsFinder(28);
        Set<Integer> expected =
            new HashSet<Integer>(Arrays.asList(1, 28, 2,
            assertEquals(expected, f.factors());
    }
}
```

Infinittest\_conf\_smallthings\_primes Infinittest

Results Logging

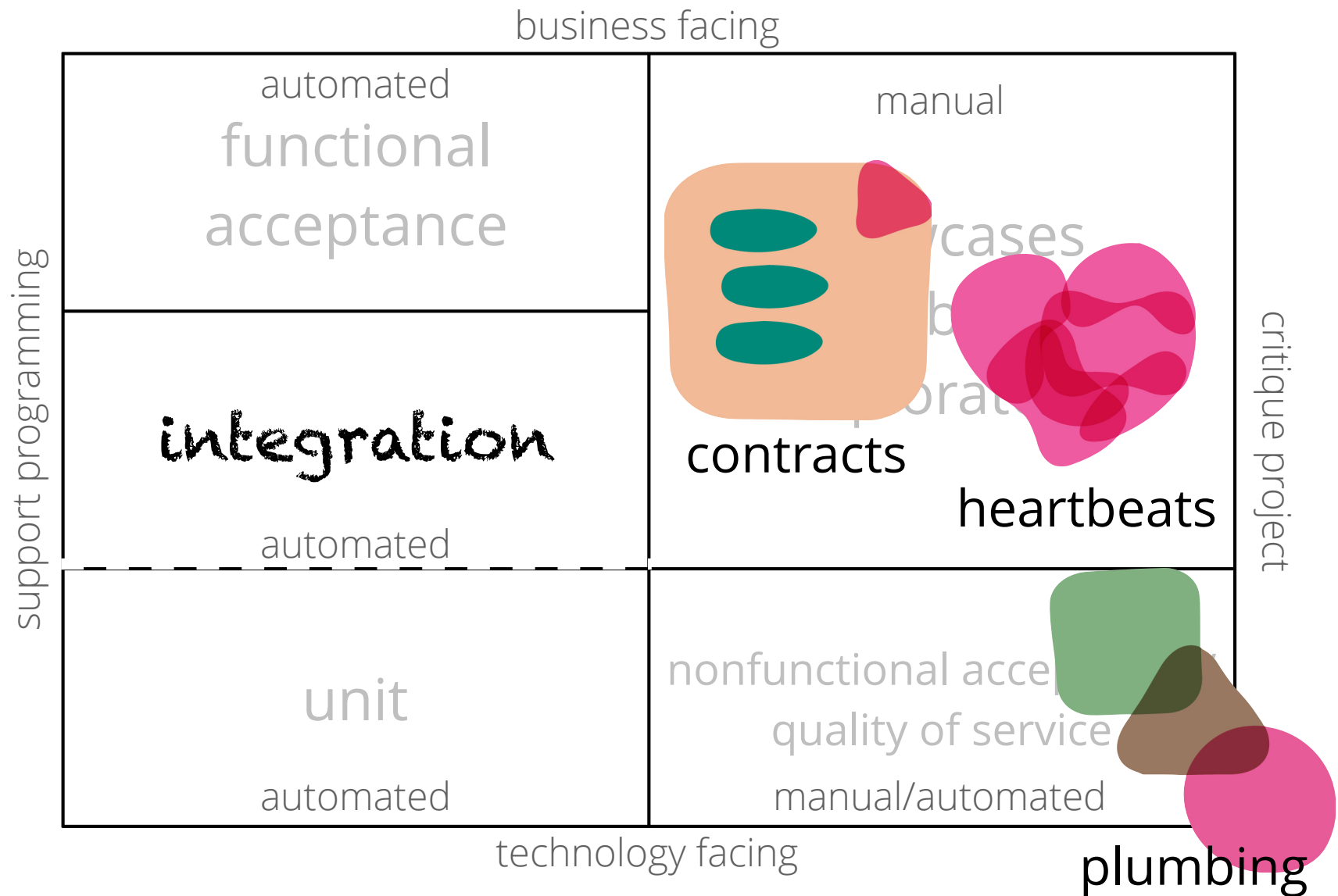
com.nealford.smallthings.primes.FactorsFinderTest:28 - AssertionError()  
com.nealford.smallthings.primes.FactorsFinderTest.is\_factor

Ran 1 Tests - Failures Detected

6: TODO Infinittest\_conf\_smallthings\_primes Web Preview

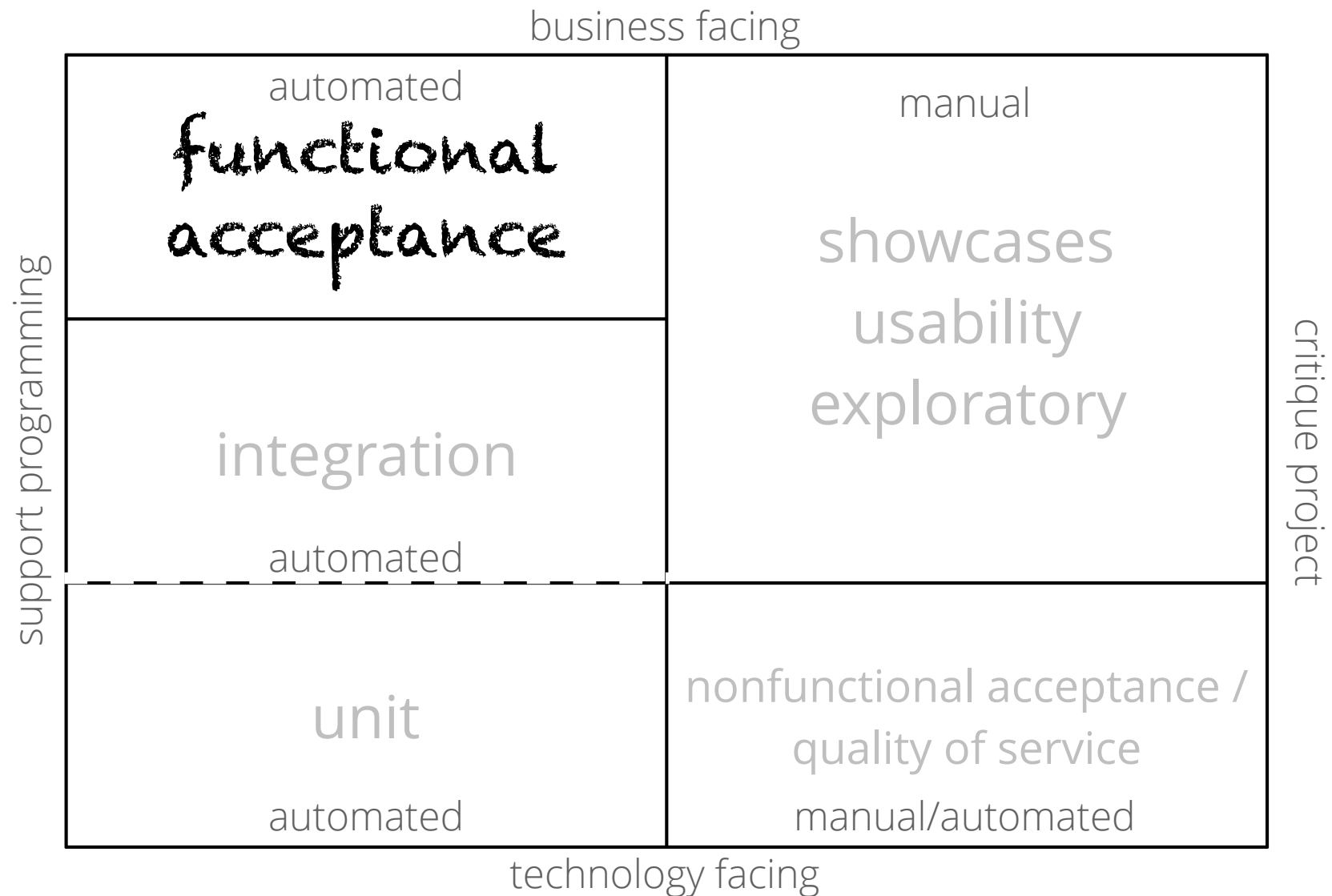
Compilation completed successfully 28:36 Insert MacRoman Default 81M of 101M

# Testing Quadrants



source: Brian Marrick, *Continuous Delivery* (Humble/Farley), with modifications

# Testing Quadrants



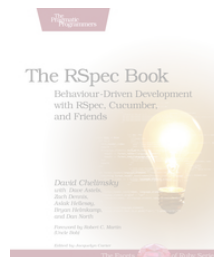
# evolution of functional testing



extension of  
unit tests



BA/SME



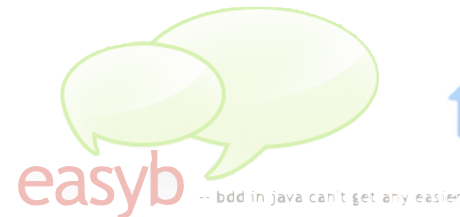
wiki



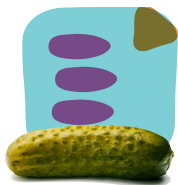
fixture



developer



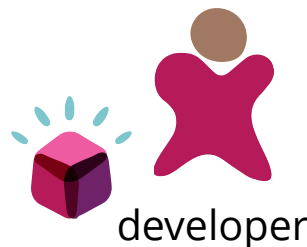
spock  
the enterprise ready specification framework



- ▶ auto-generation of fixtures
- ▶ cross-platform for step definitions



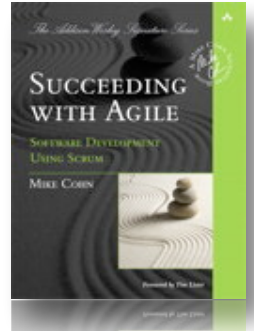
BA/SME



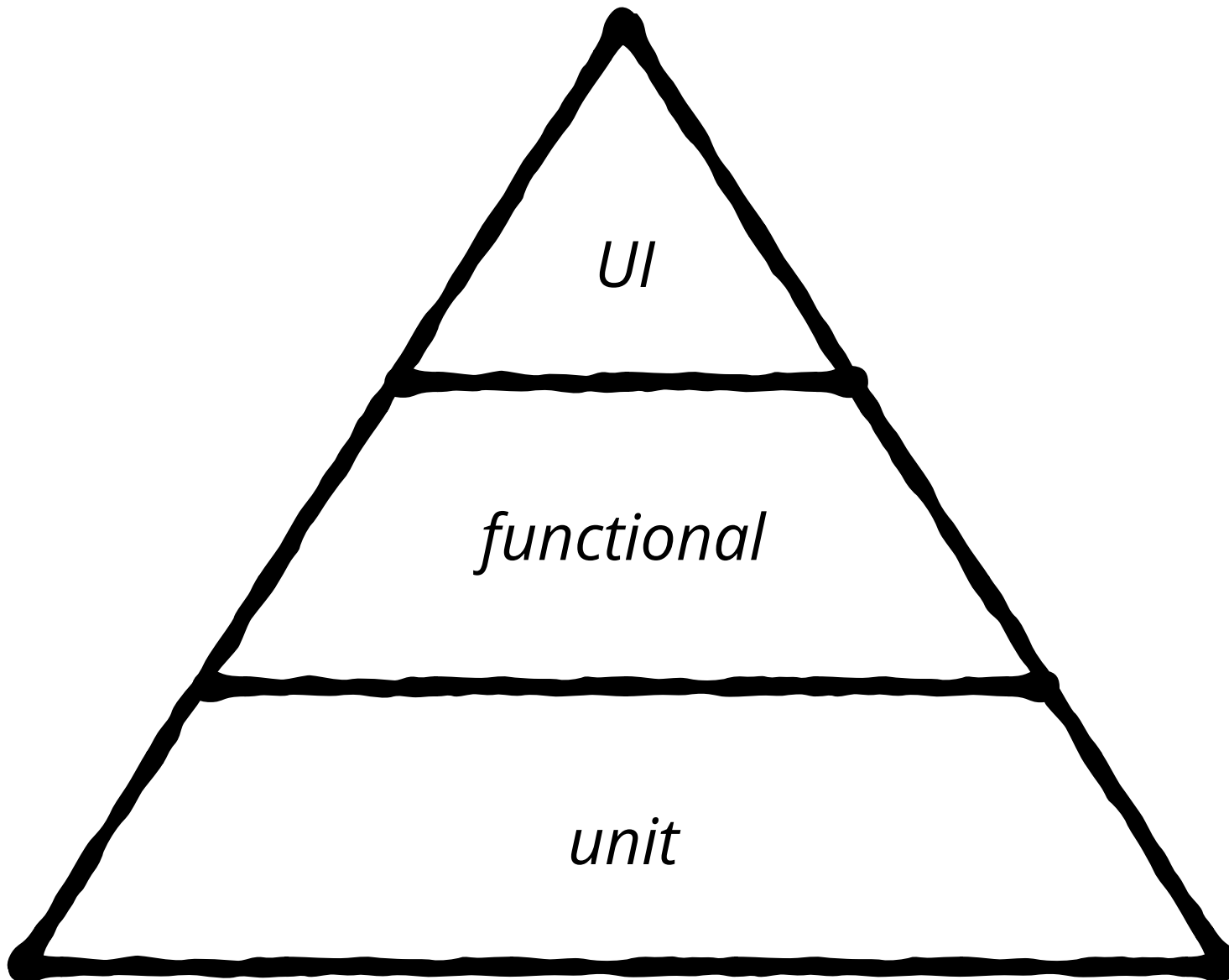
developer

Cucumber

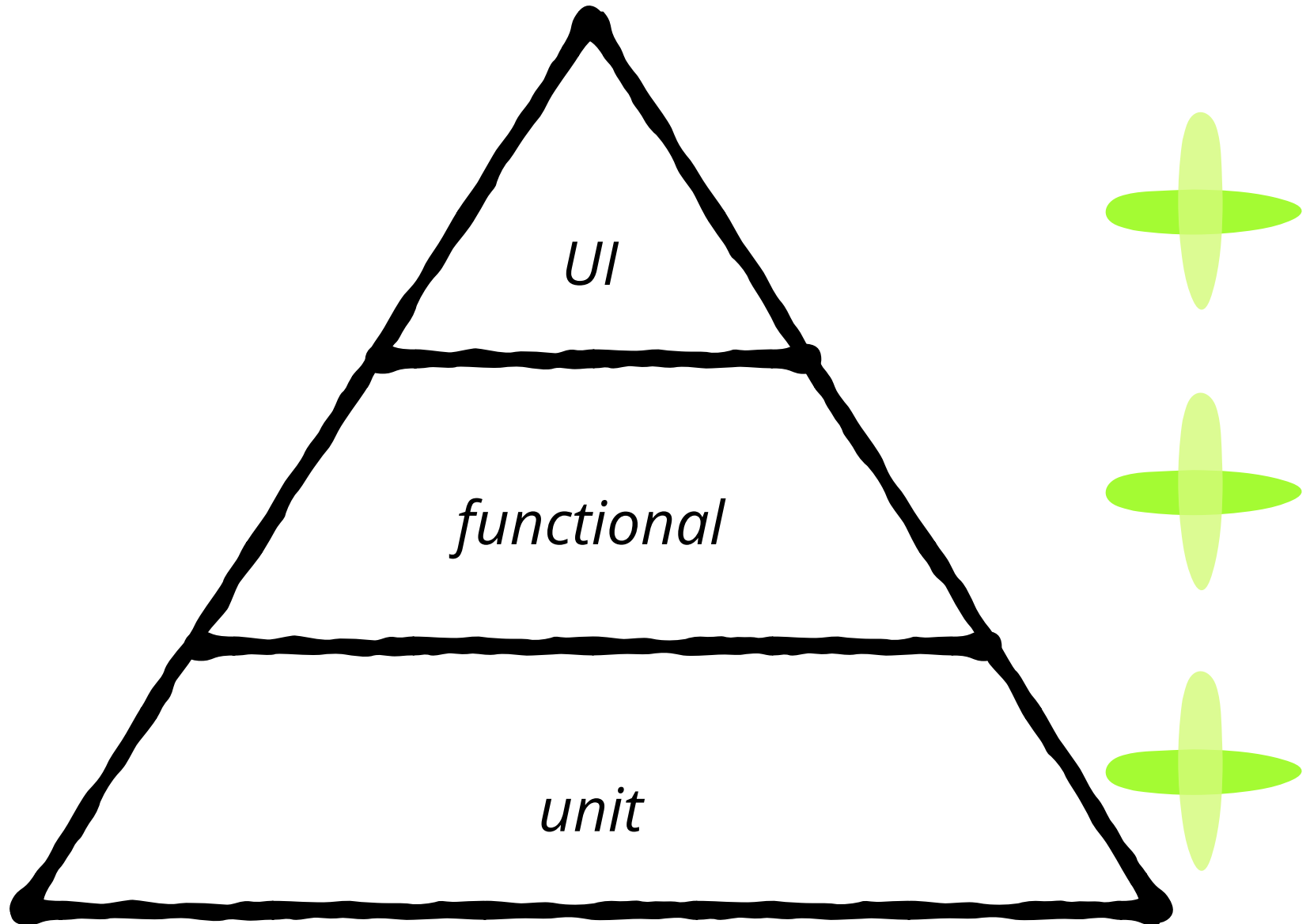
# Testing Ratios



*Succeeding with Agile*  
(Mike Cohn)

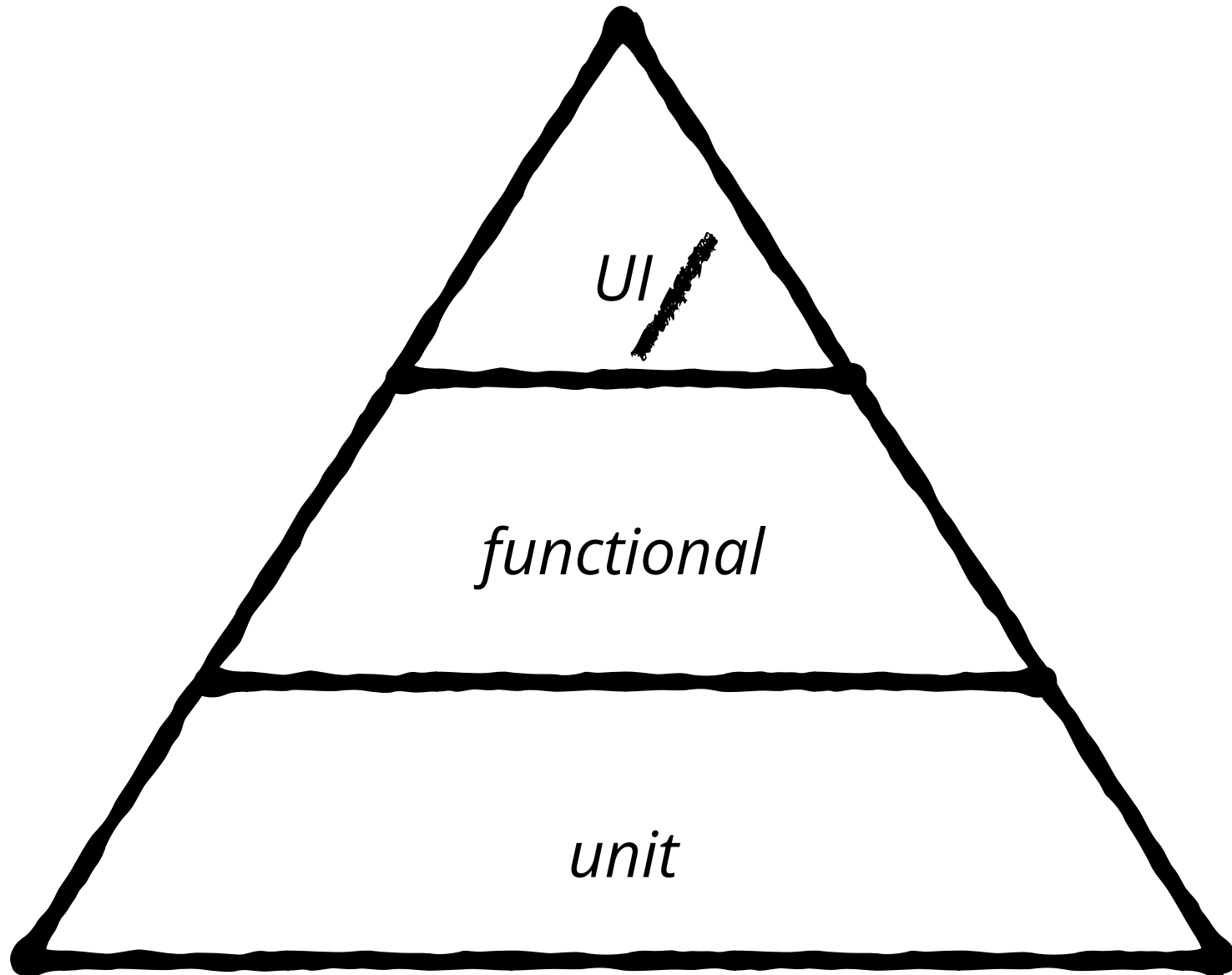


# Flavors of Functional Tests

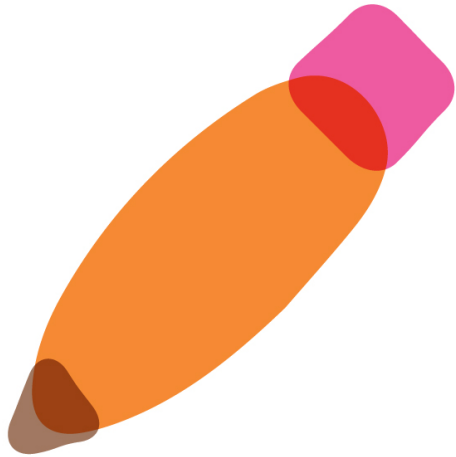




# Flavors of Functional Tests



# To Cuke or not to Cuke...



BDD  $\neq$

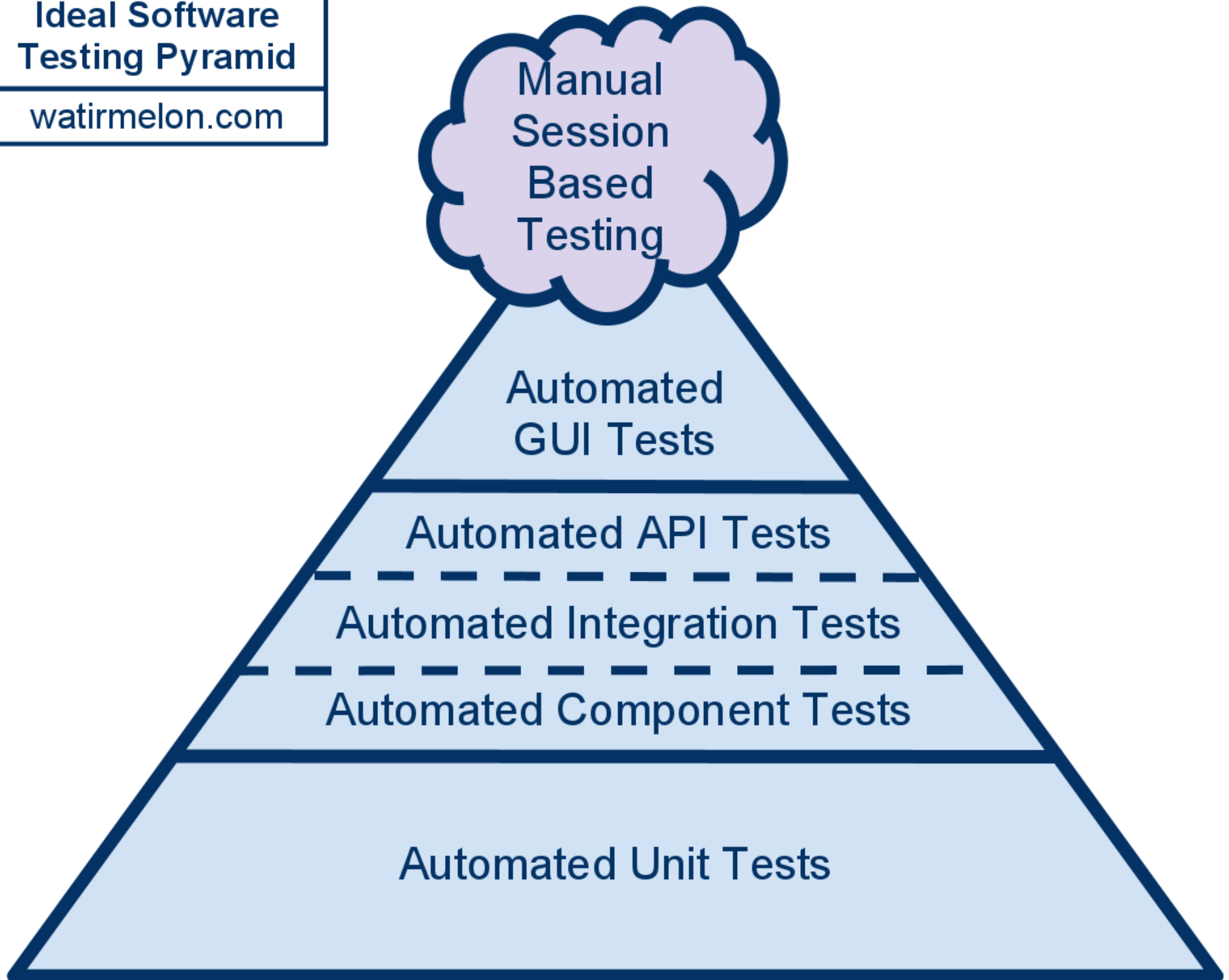


*use when it provides useful  
feedback to the target audience*

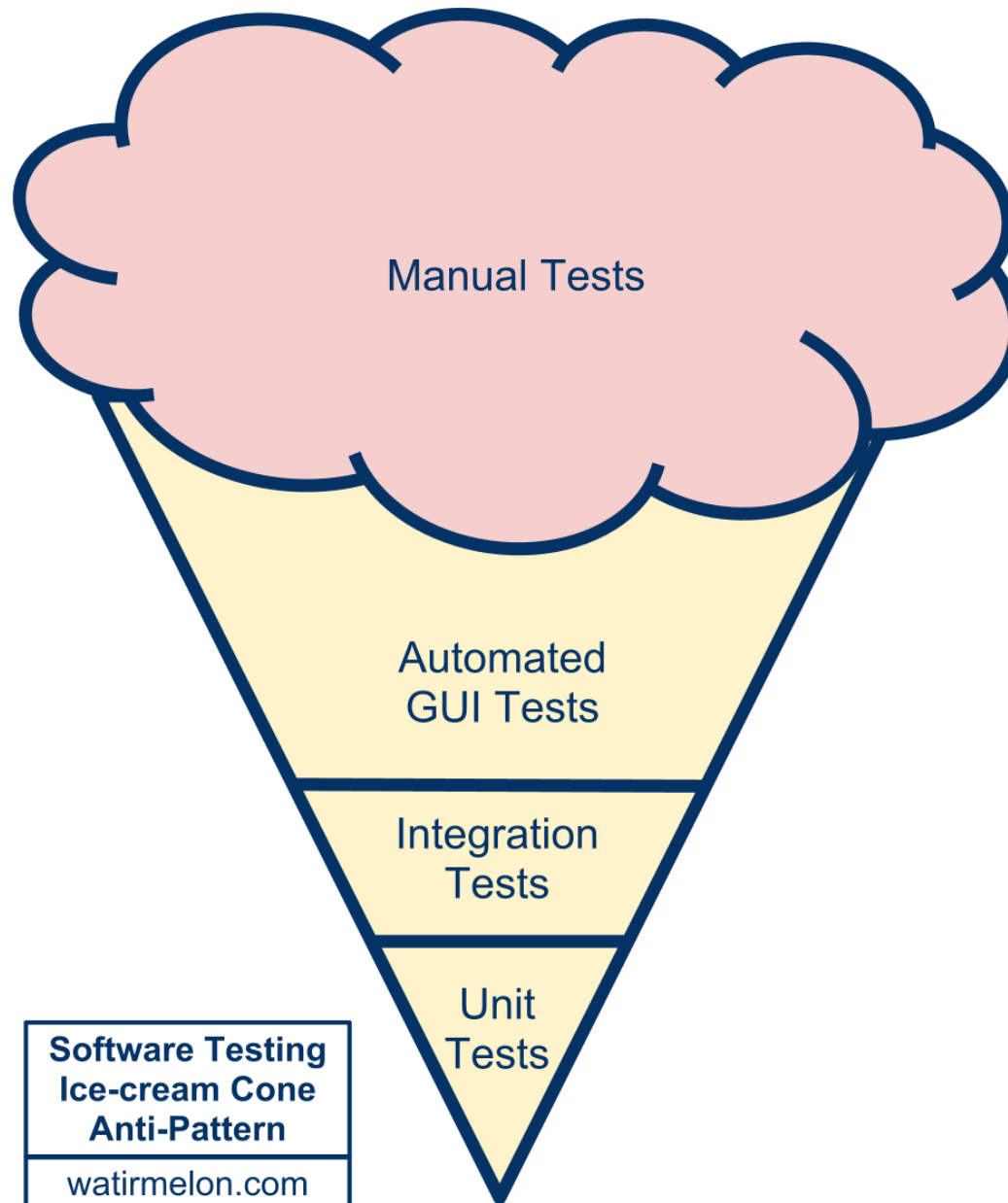


# Ideal Software Testing Pyramid

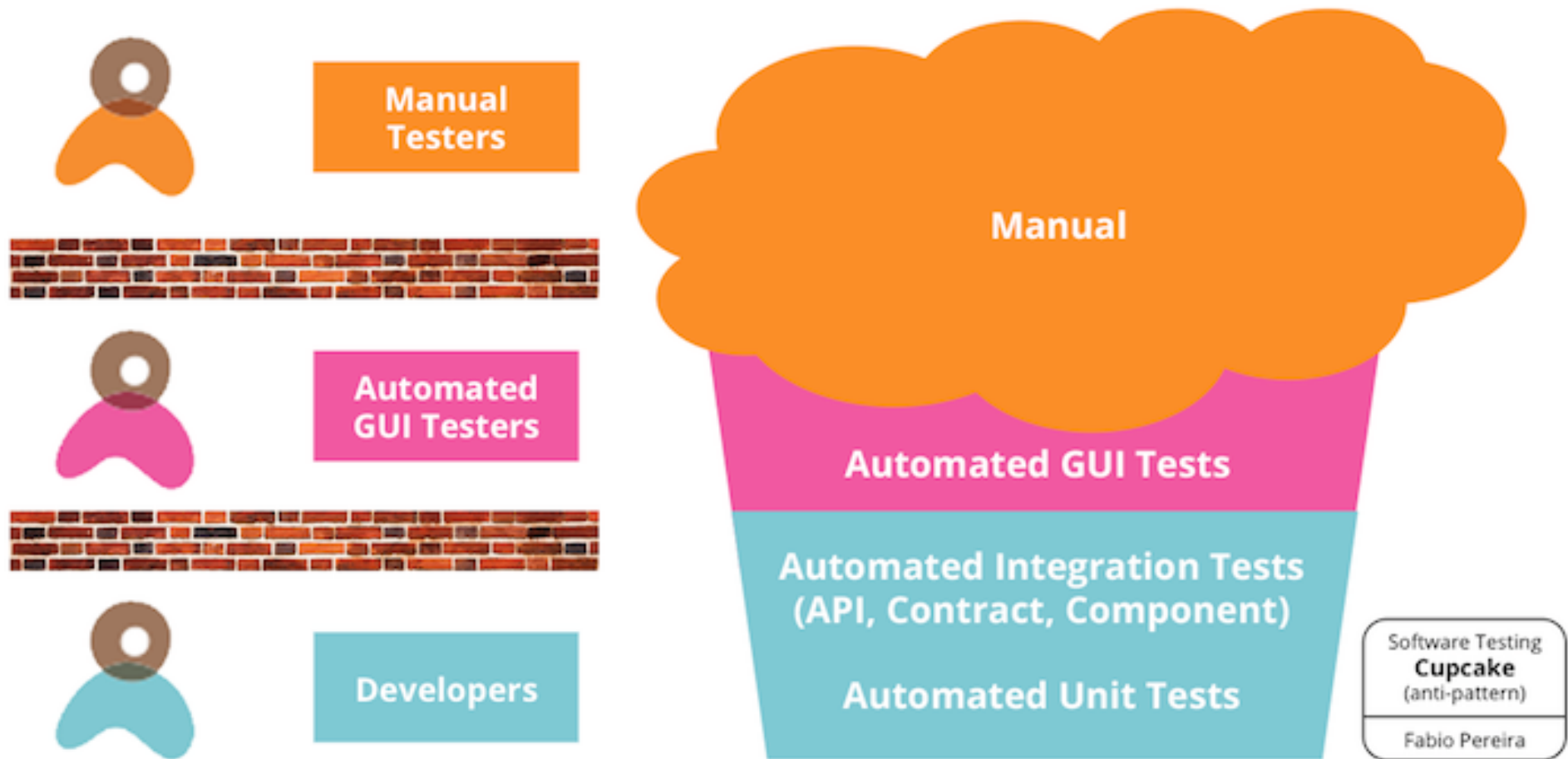
watirmelon.com



# Anti-pattern: Ice-cream Cone



# Cupcake Anti-pattern



# Avoiding Cupcakes

collaborate

work in sync

cross-role pair programming

story kickoff

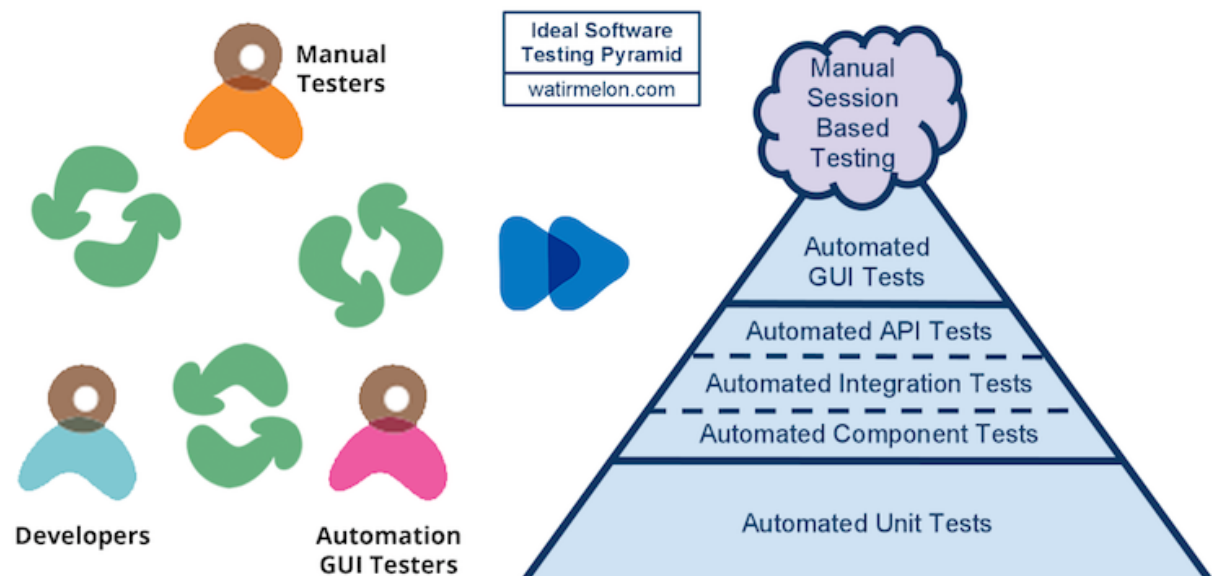
test at the lowest level

merge teams

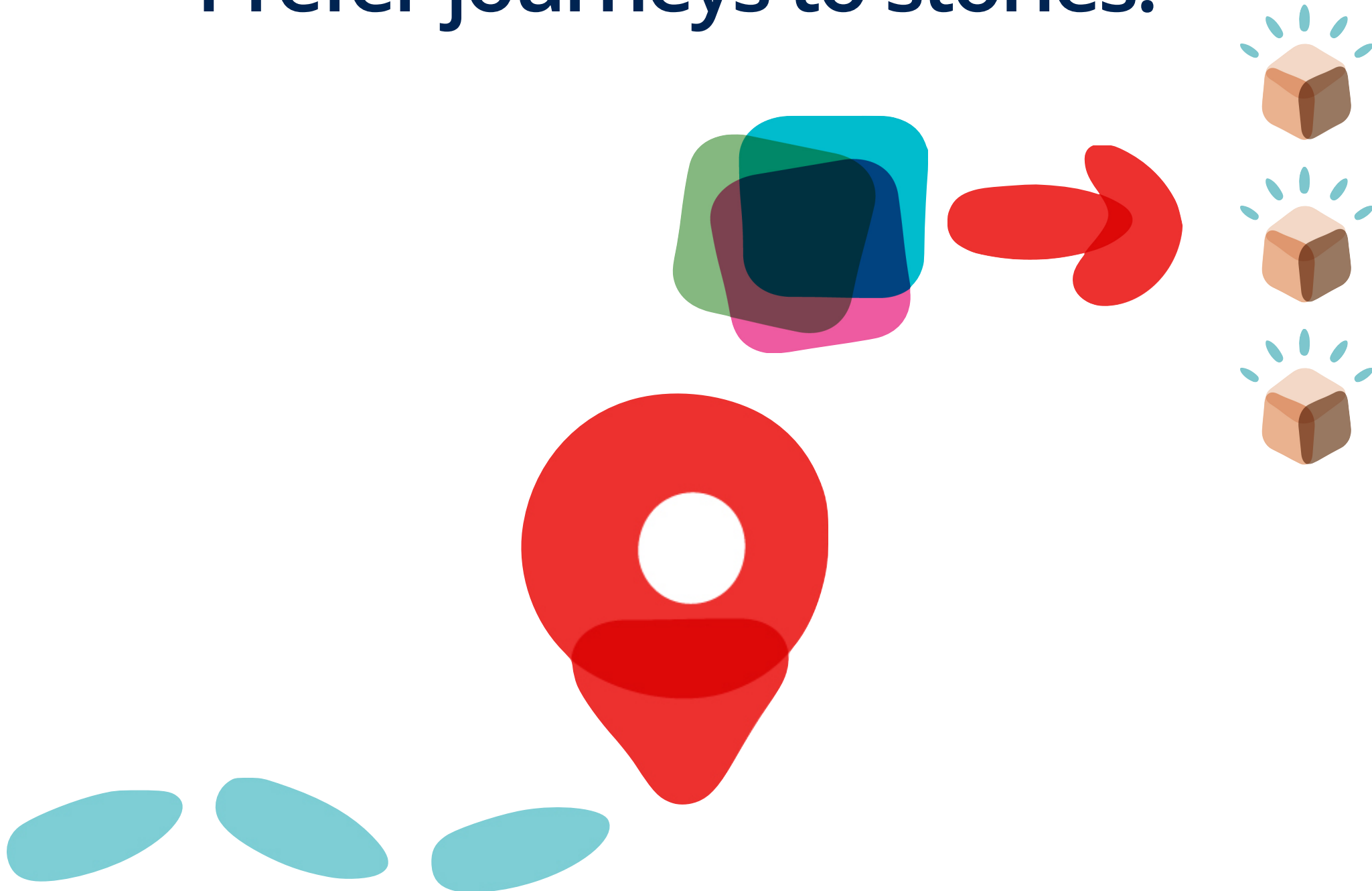
when possible

agree on goals

and metrics



# Prefer journeys to stories.



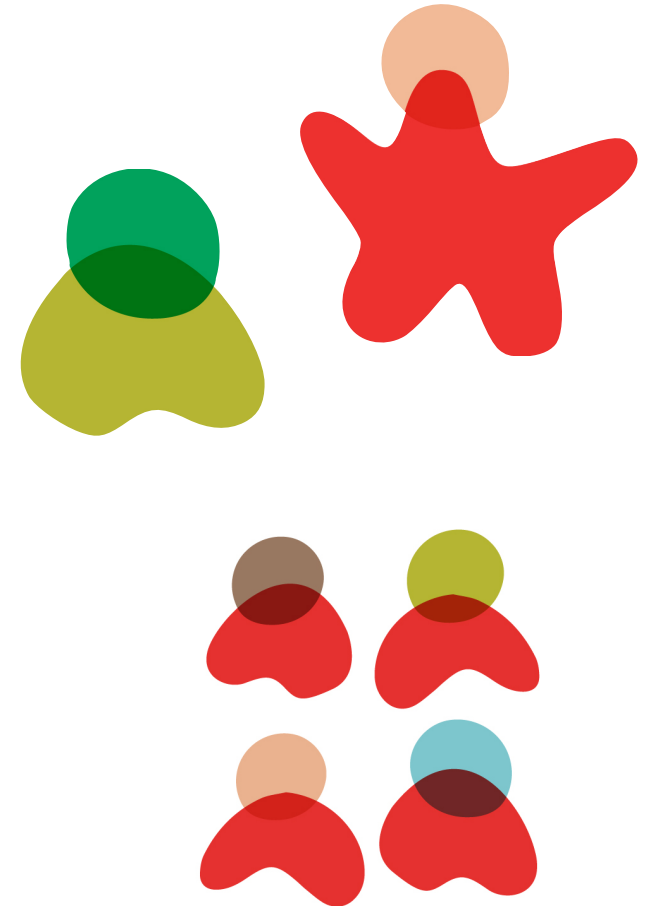
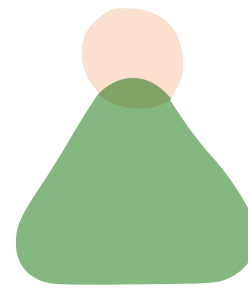


Continuous  
Integration

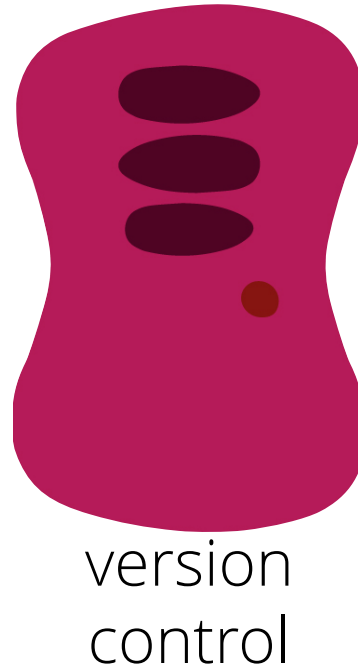




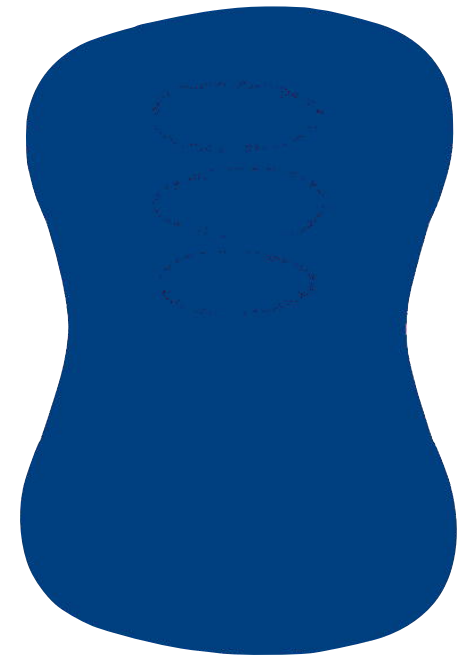
machinery

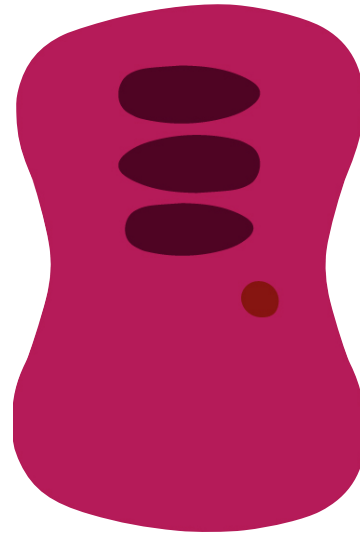
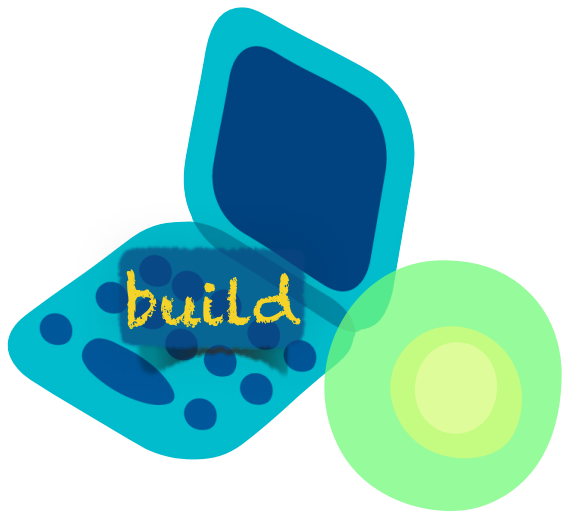


Continuous  
Integration



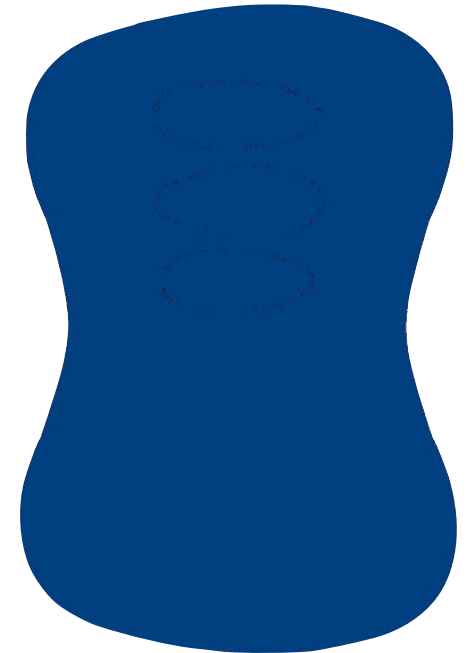
continuous integration  
server

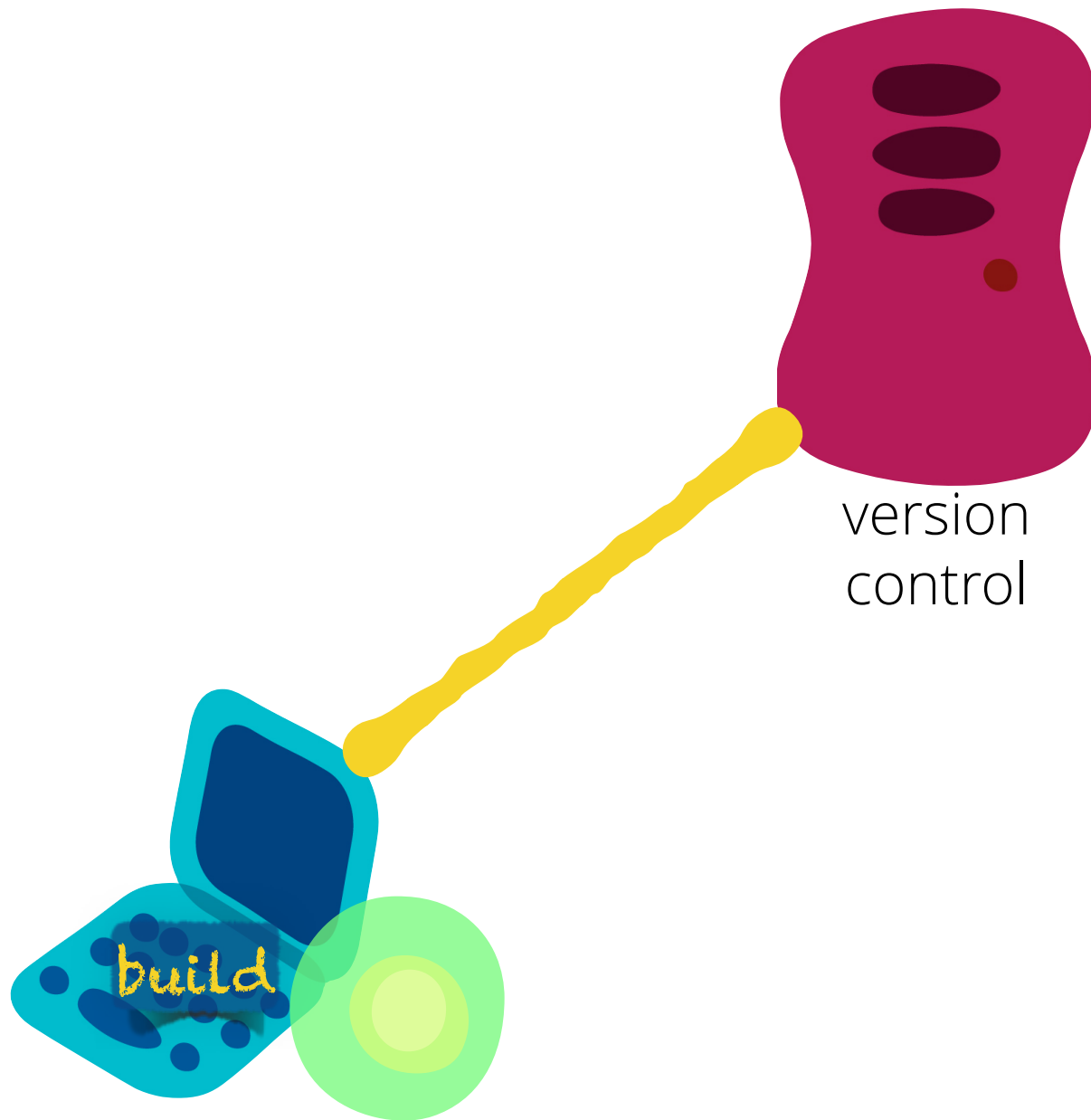




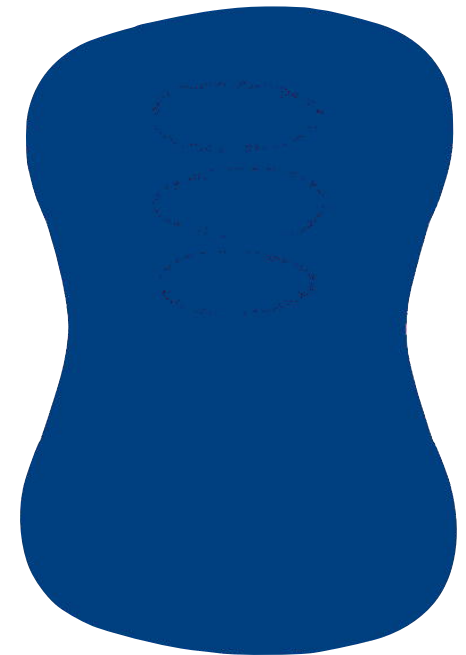
version  
control

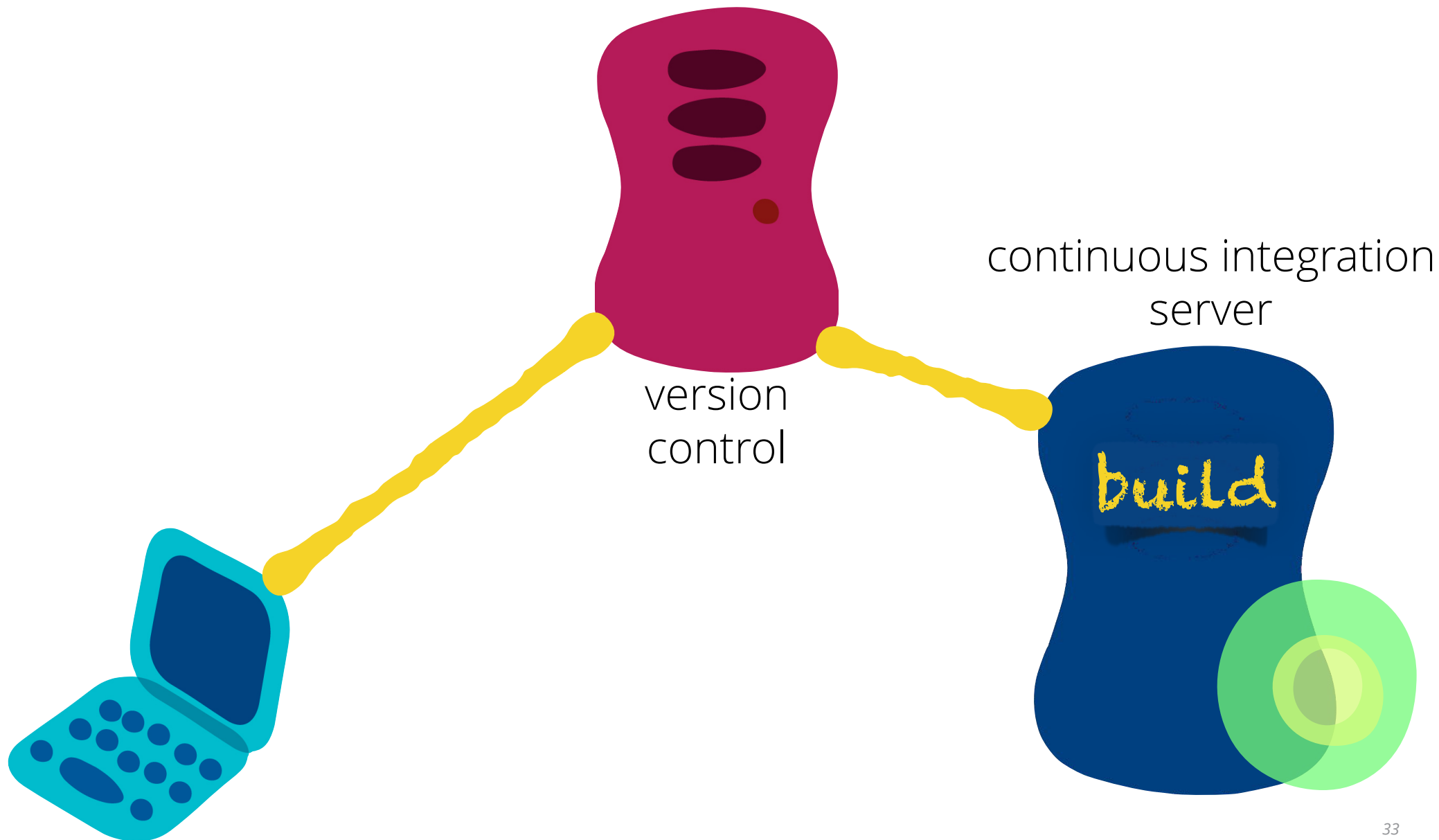
continuous integration  
server





continuous integration  
server



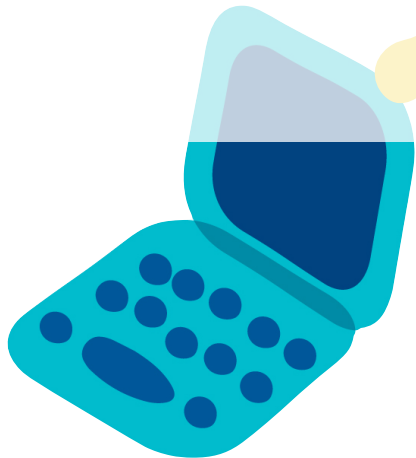


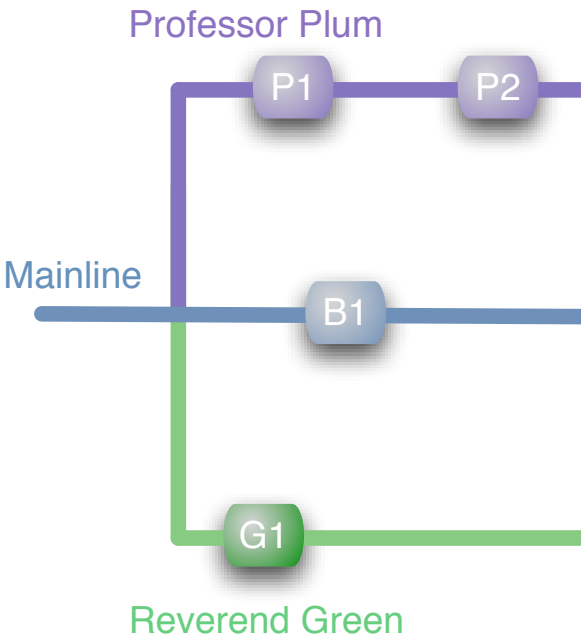
everyone commits  
to trunk at least  
once a day

continuous integration  
server

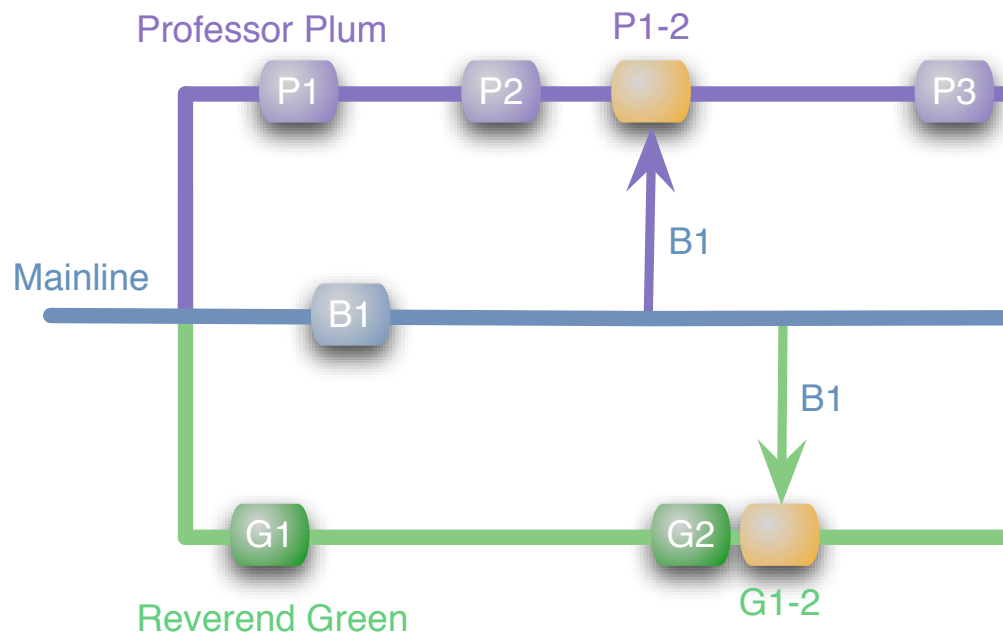
version  
control

build



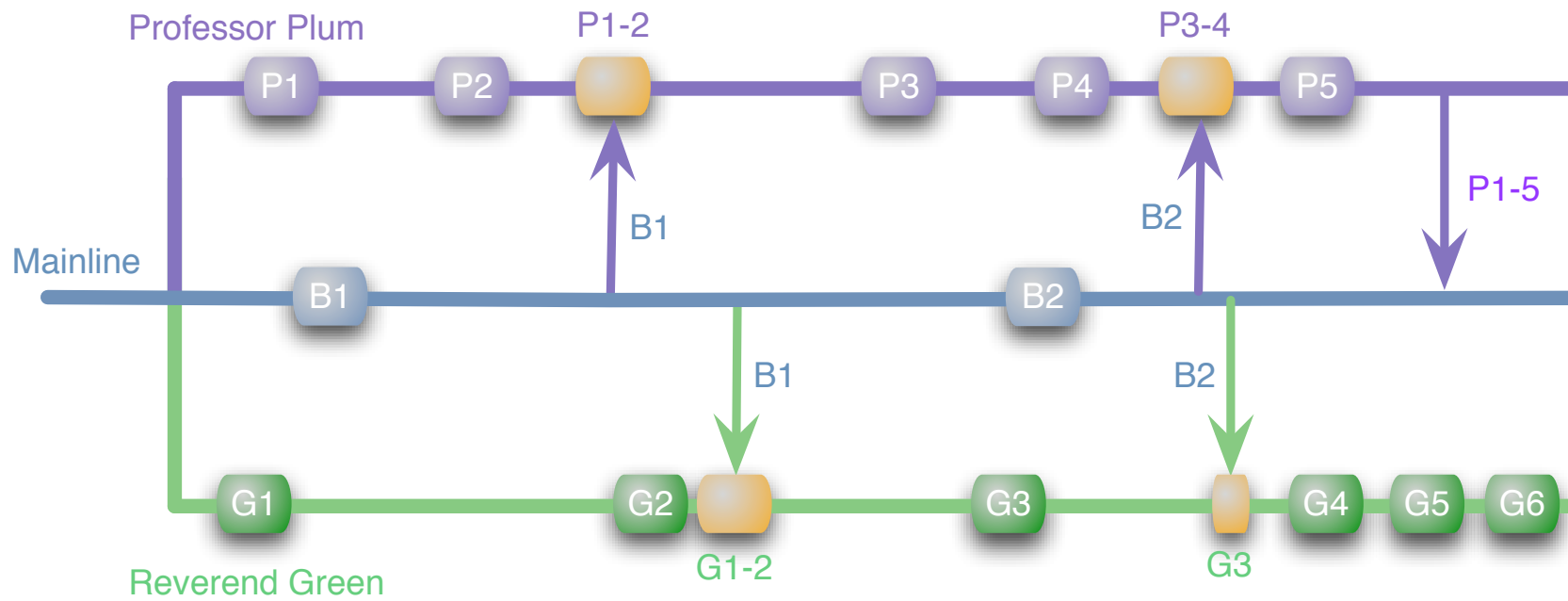


# Feature Branching



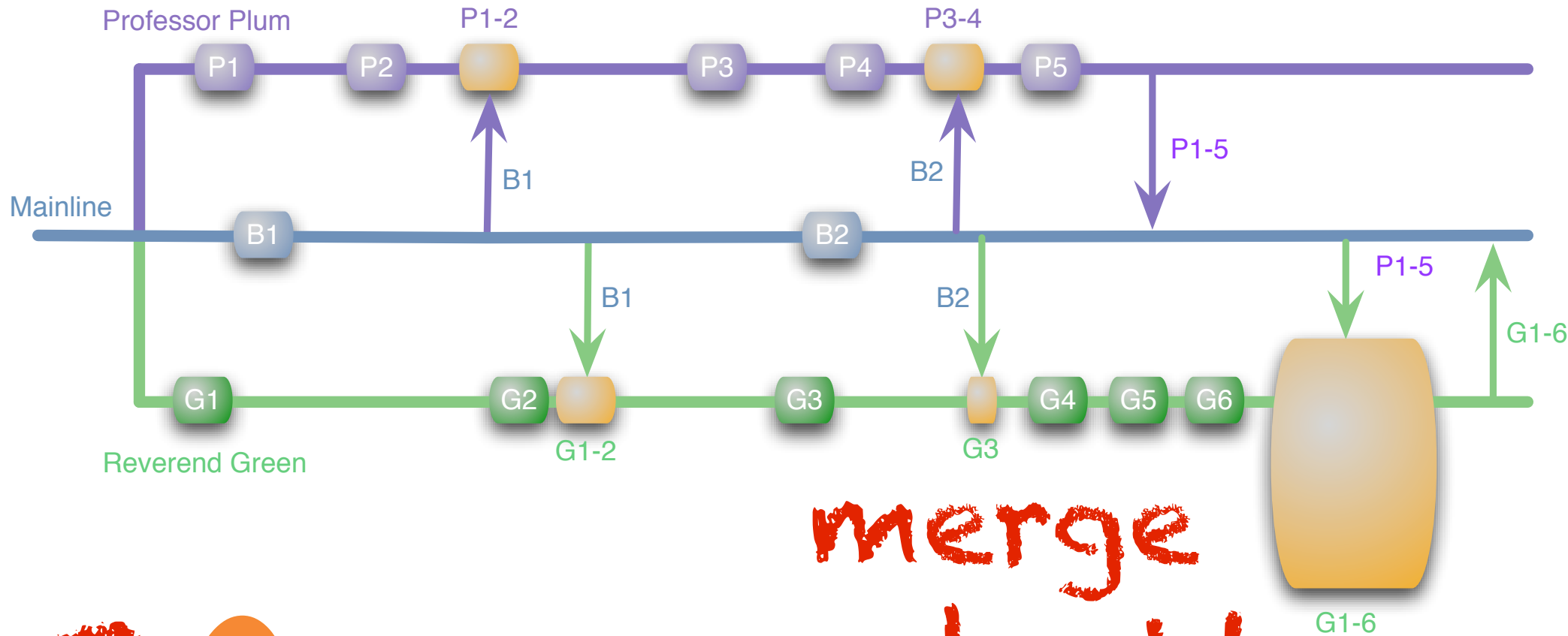
## Feature Branching





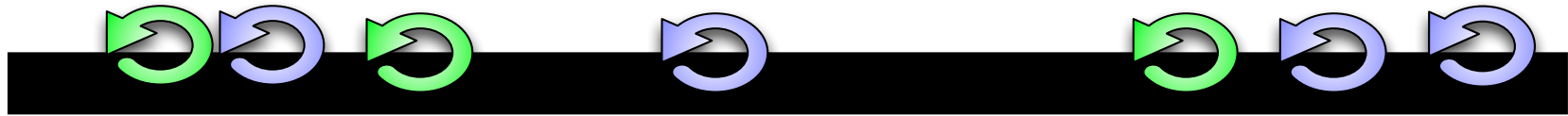
# Feature Branching

copy/paste  
reuse !!

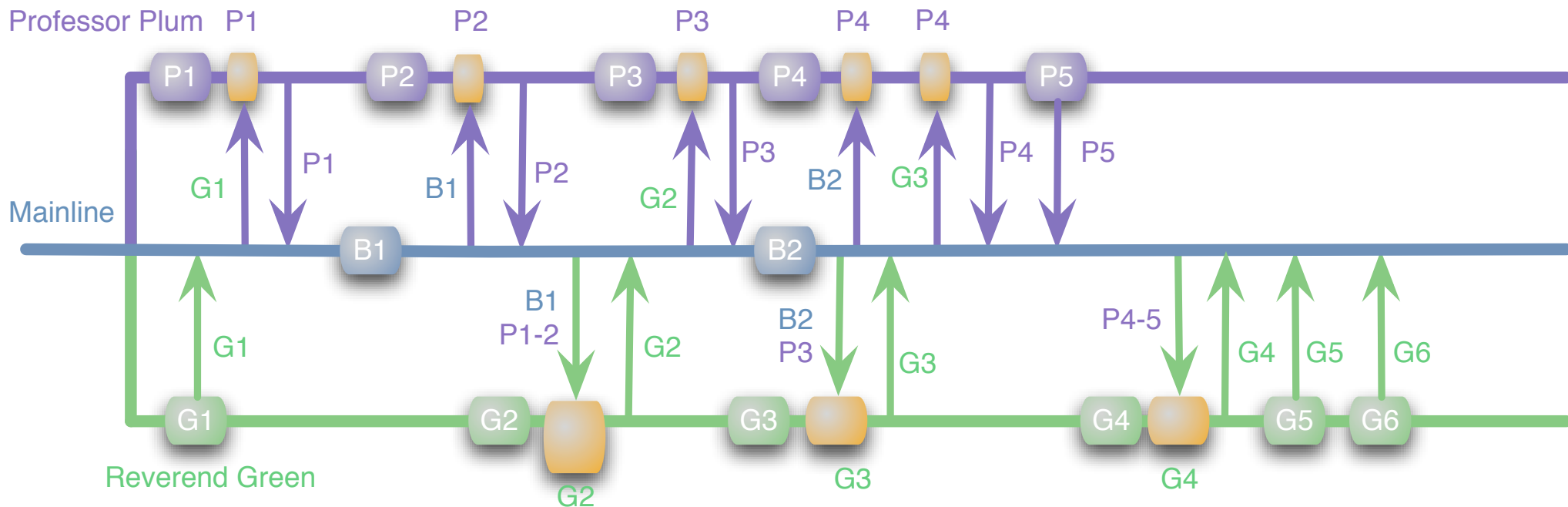


merge  
ambush!

Feature Branching



# trunk-based development



Continuous Integration removes the pain...

# What is Trunk Based Development?

## What it is...

It is a branching model for software development. Historically, it has also been called "mainline" (see later).

It requires much more concentration and rigor, than making a branch (on the shared source-control server) to suit a whim. Though you could do it without Continuous Integration (CI), as many open source projects do, for enterprise development you have to have CI linked to the trunk, enforcing multiple aspects of "that commit was good".

In this article, I'm saying nothing about what developers do on their own workstations by way of 'local' branching to suit their hour by hour activities. This is all about the shared repo, where multiple developers integrate/merge their daily work for the greater good :)

Published  
April 5<sup>th</sup>, 2013

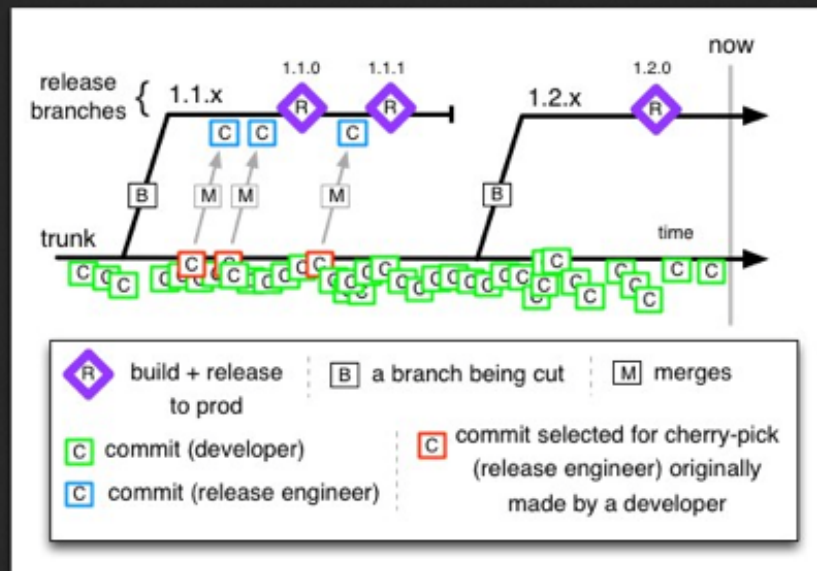
Categories

Source Control <sup>38</sup>

Branch by Abstraction, etc <sup>14</sup>

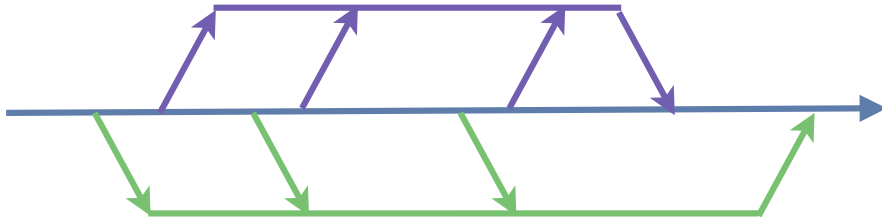
Trunk Based Development <sup>22</sup>

DevOPS <sup>10</sup>



Trunk Based Development (TBD) is where all developers (for a particular deployable unit) commit to one shared branch

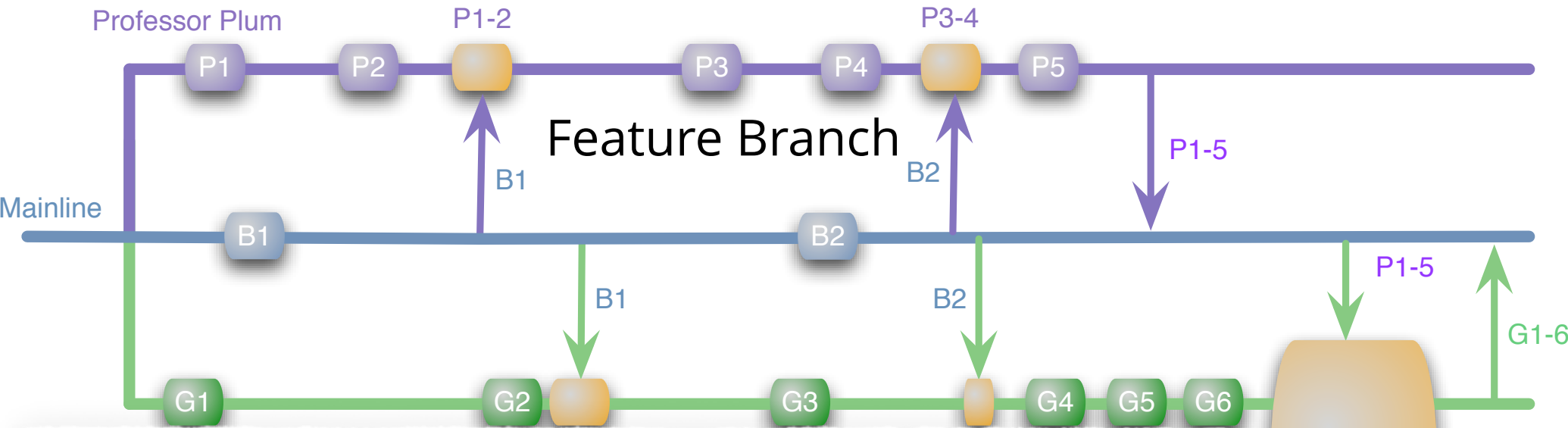
## Feature Branching



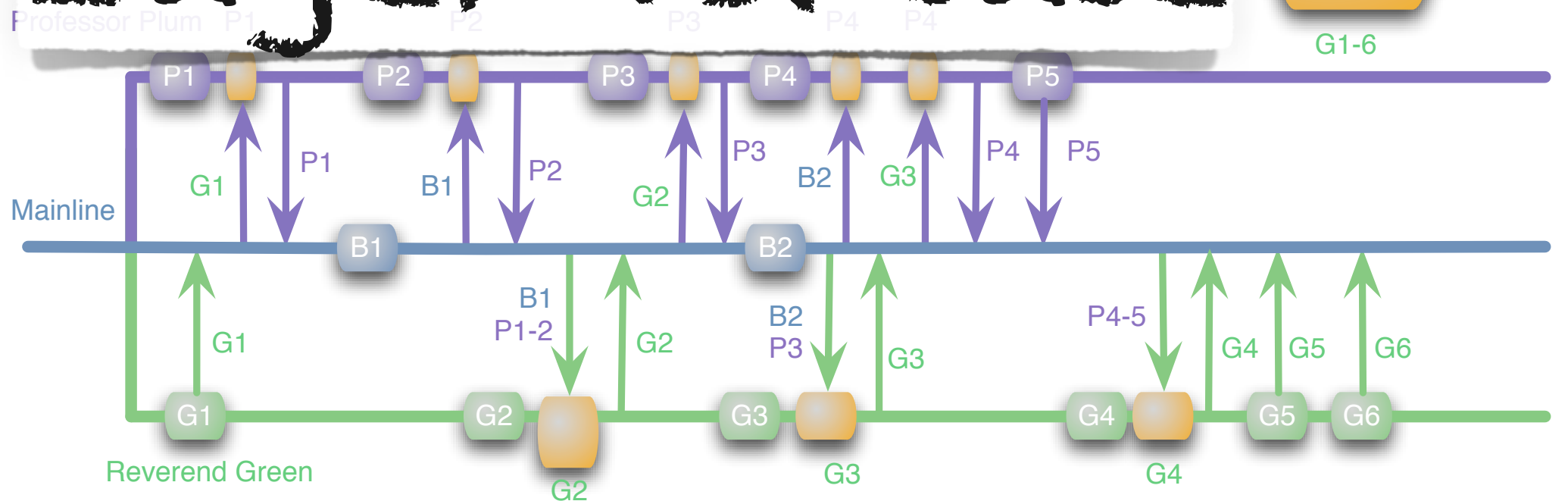
## *Big Scary Merge*



## Continuous Integration

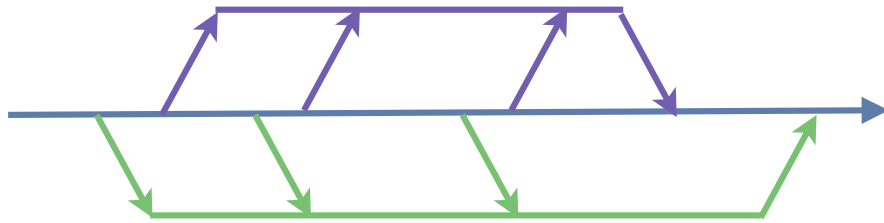



# Eager vs. Late



Continuous Integration

## Feature Branching



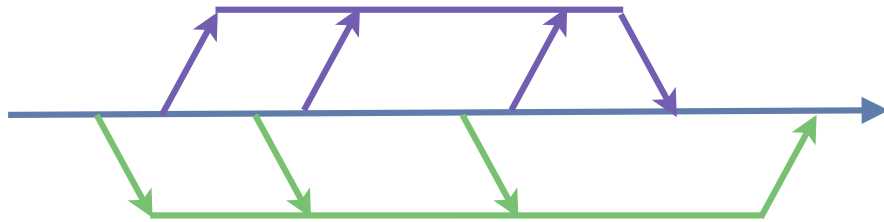
*Big Scary Merge* 1 

*Untrusted  
Contributors*



Continuous Integration

Feature Branching



*Big Scary Merge*

*Untrusted  
Contributors*

*Cherry Picking*



Continuous Integration



## Config File

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

some.jsp

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

# feature toggles

other.java

```
forkHandle = (featureConfig.isOn('flightyForkHandles')) ?
    new FlightyForkHandler(aCandle) :
    new ForkHandler(aCandle)
```



[www.togglz.org](http://www.togglz.org)



```
public enum MyFeatures implements Feature {
```

```
    @EnabledByDefault
```

```
    @Label("First Feature")
```

```
    FEATURE_ONE,
```

```
    @Label("Second Feature")
```

```
    FEATURE_TWO;
```

```
    public boolean isActive() {
```

```
        return FeatureContext.getFeatureManager().isActive(this);
```

```
    }
```

```
}
```

```
public void someBusinessMethod() {
```

```
    if( MyFeatures.FEATURE_ONE.isActive() ) {
```

```
        // do new exciting stuff here
```

```
    }
```

```
    [...]
```

```
}
```



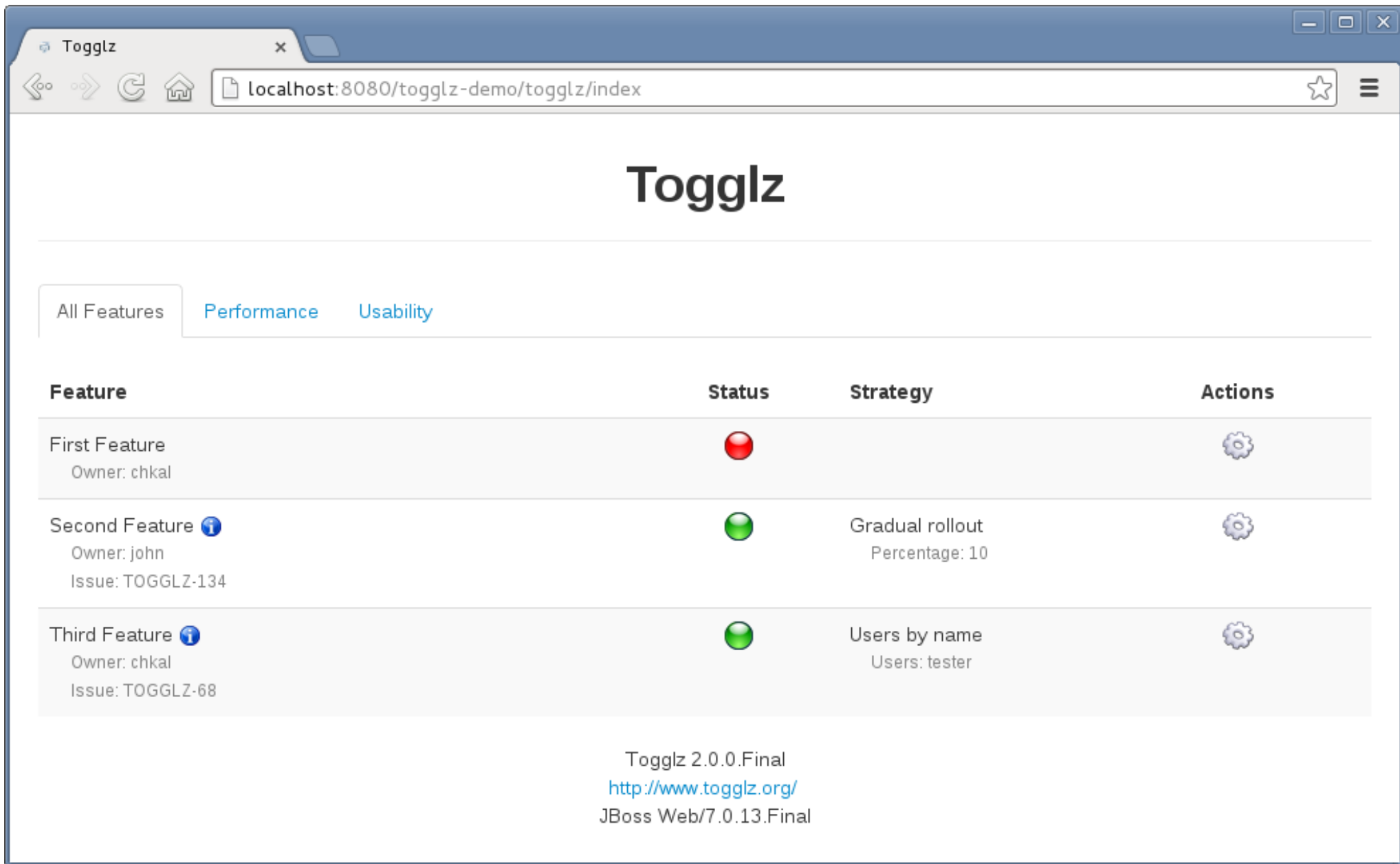
```
@ApplicationScoped
public class DemoConfiguration implements TogglzConfig {

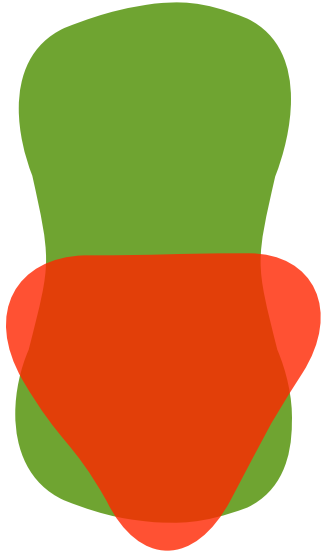
    public Class<? extends Feature> getFeatureClass() {
        return MyFeatures.class;
    }

    public StateRepository getStateRepository() {
        return new FileBasedStateRepository(new File("/tmp/features.properties"));
    }

    public UserProvider getUserProvider() {
        return new ServletUserProvider();
    }

}
```

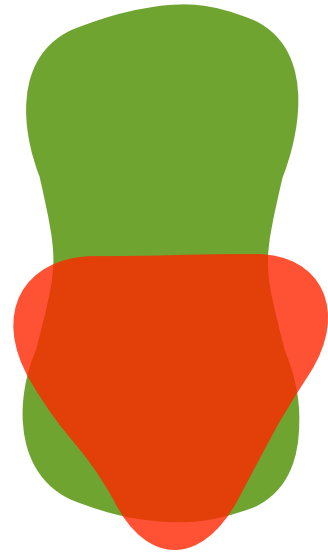




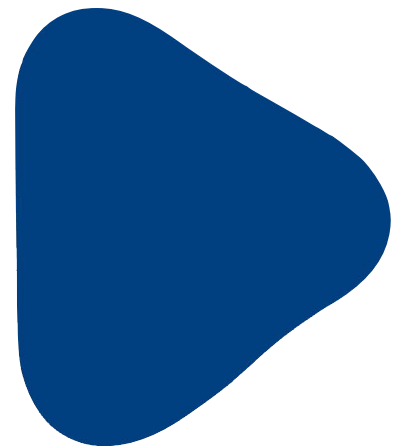
removed as soon as  
feature decision is resolved

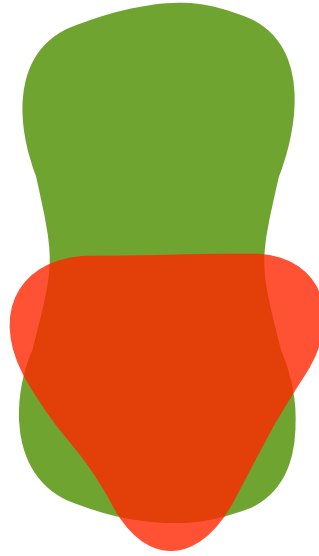
Feature toggles are purposeful  
technical debt added to support  
engineering practices like  
Continuous Delivery.





build-time vs. run-time



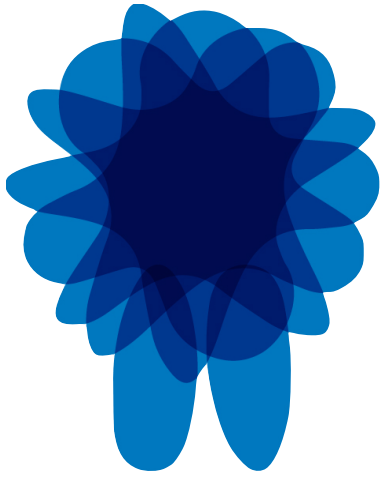


# taxonomy

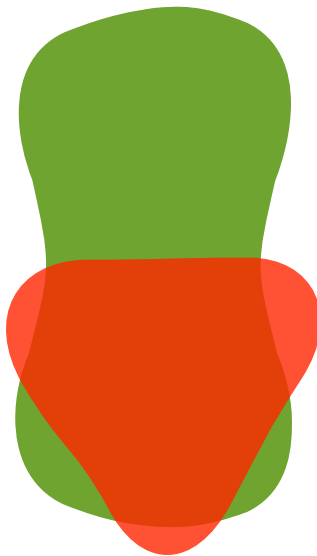
```
@FeatureGroup
@Label("Performance Improvements")
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Performance {
    // no content
}
```

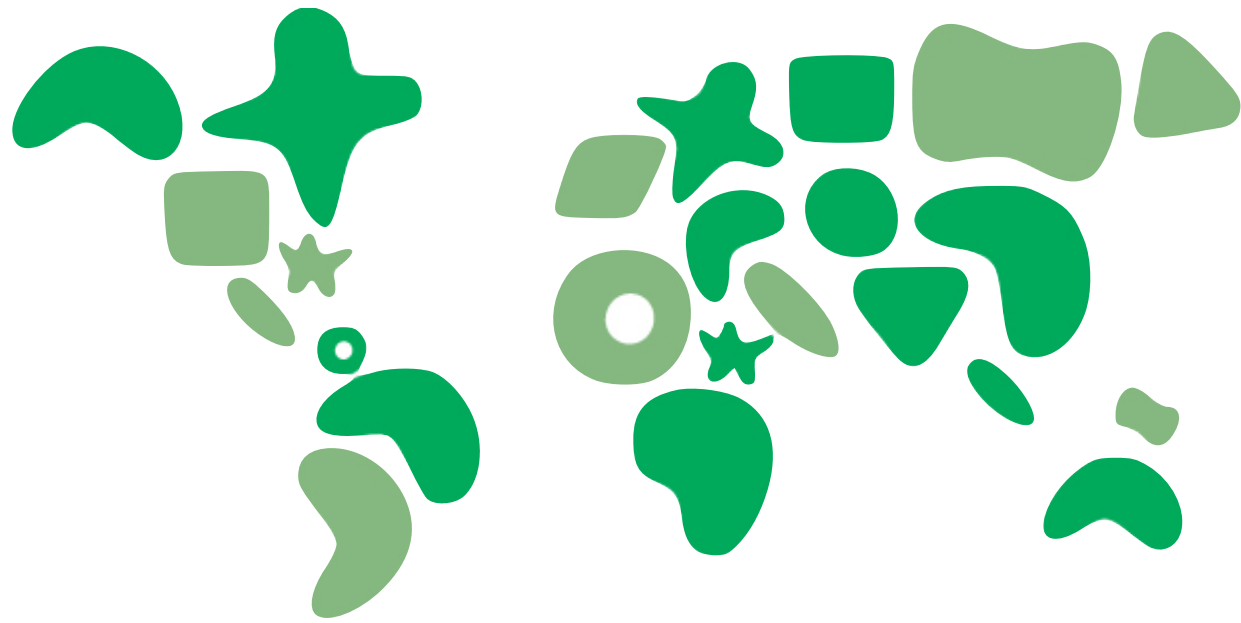




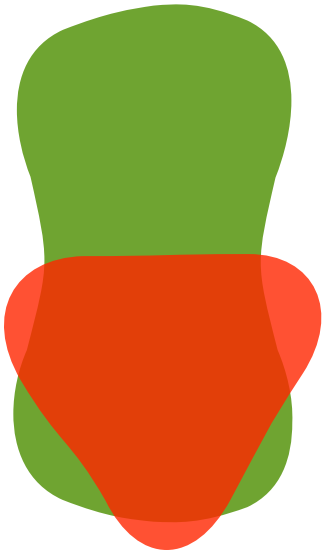


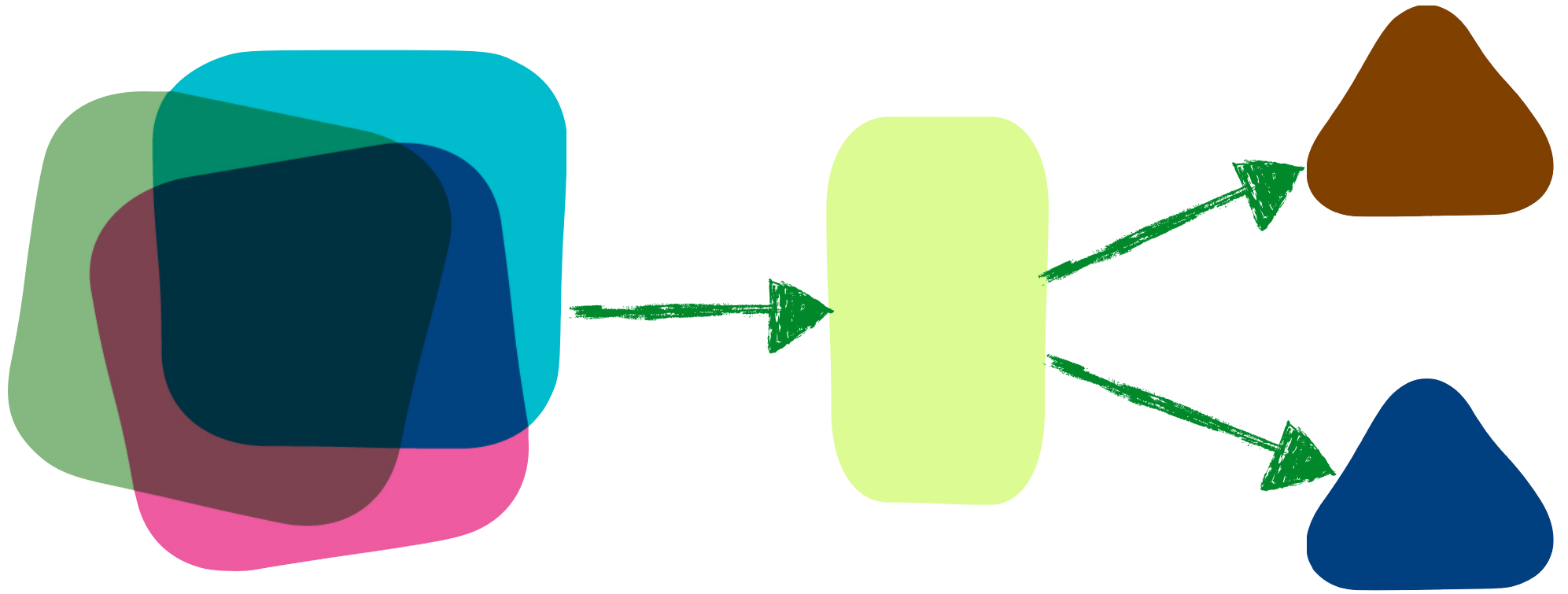
works on all platforms &  
technology stacks



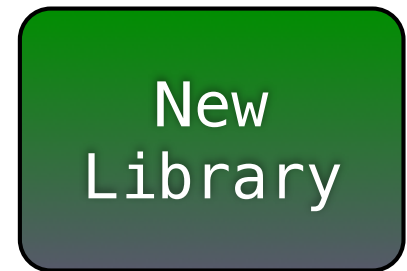
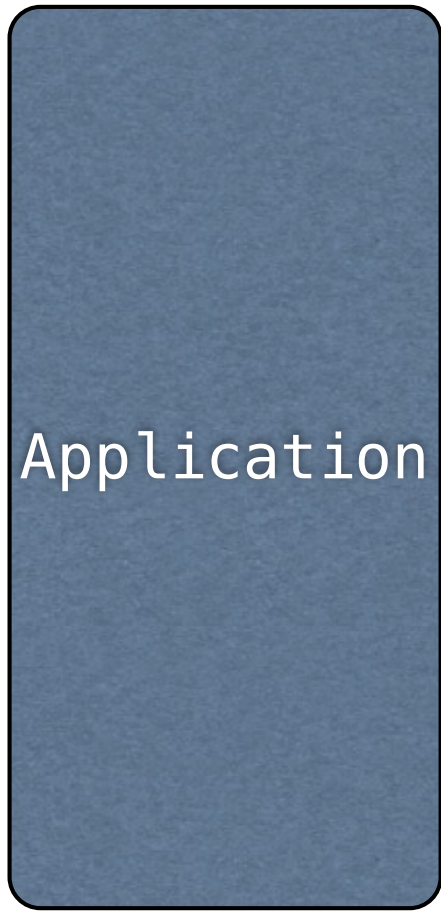


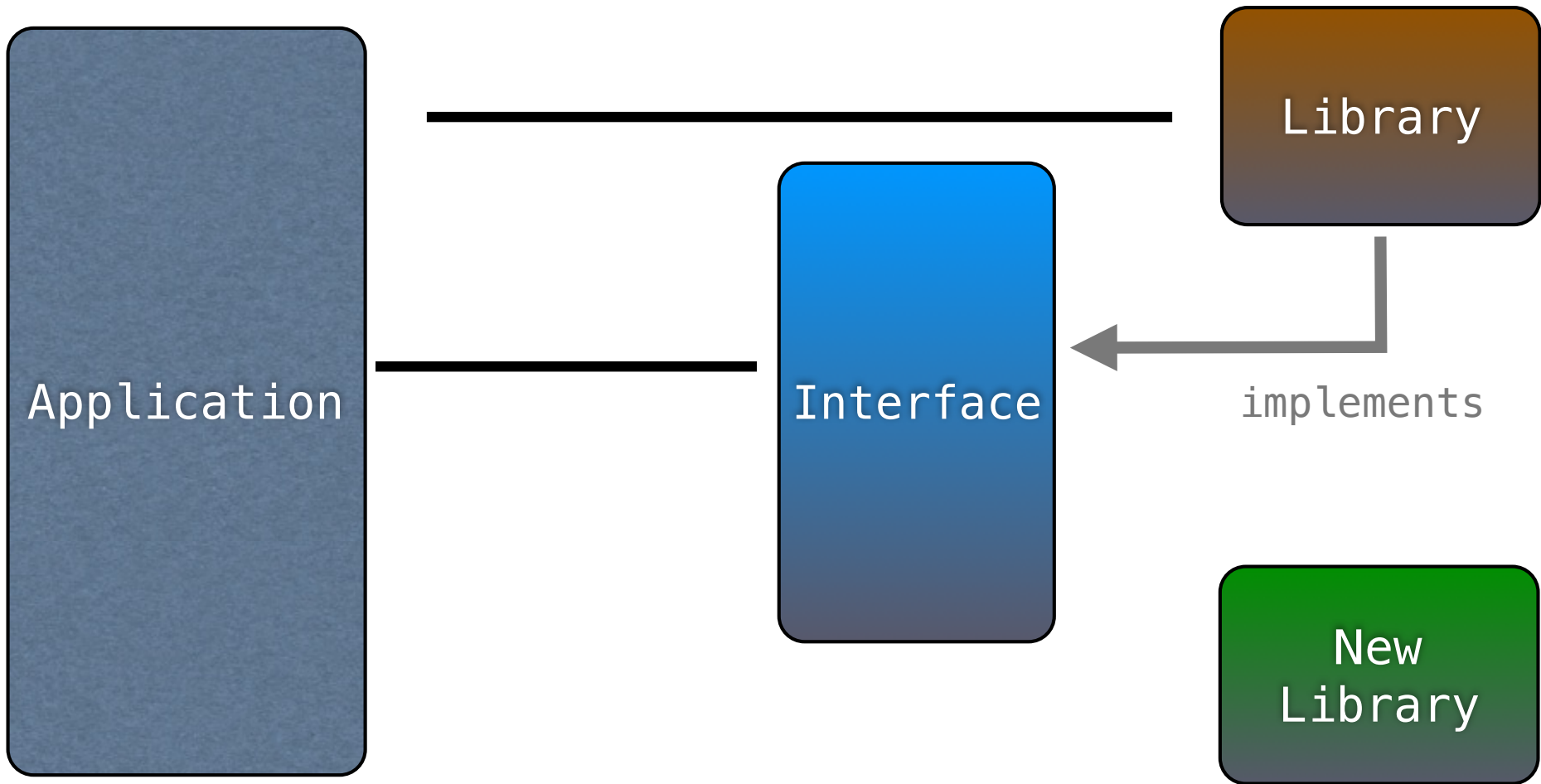
ubiquitous

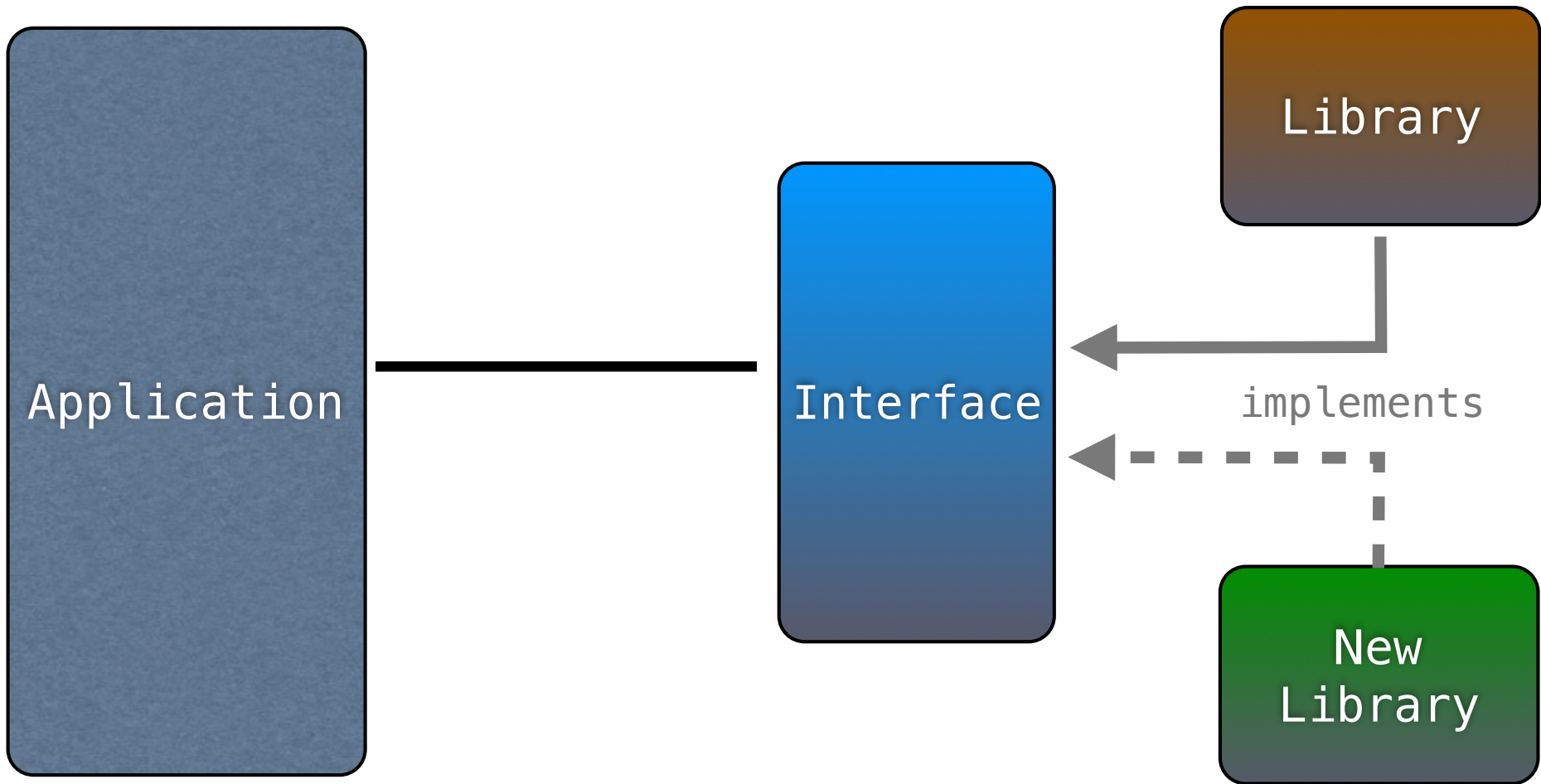


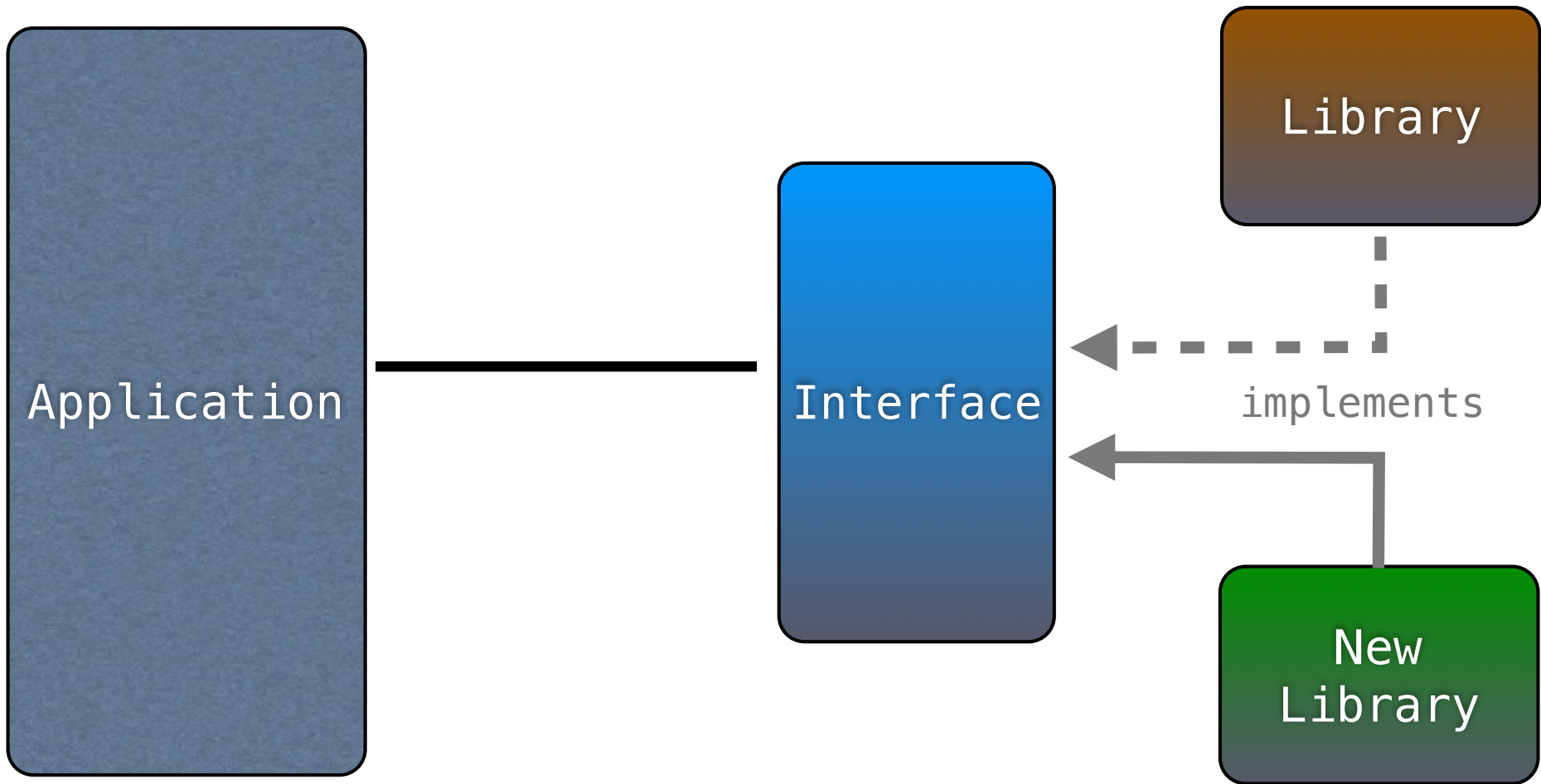


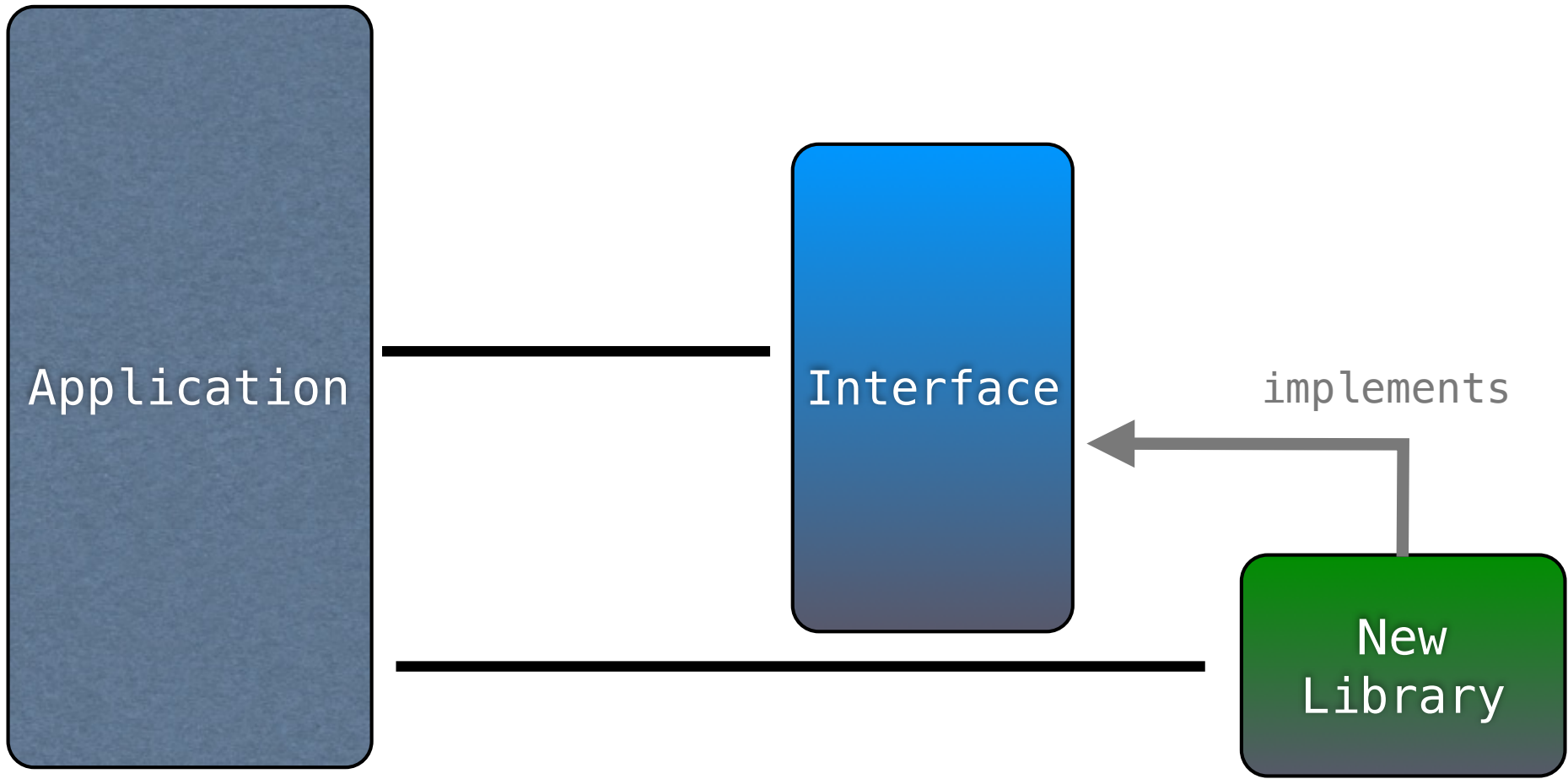
Branch by  
Abstraction





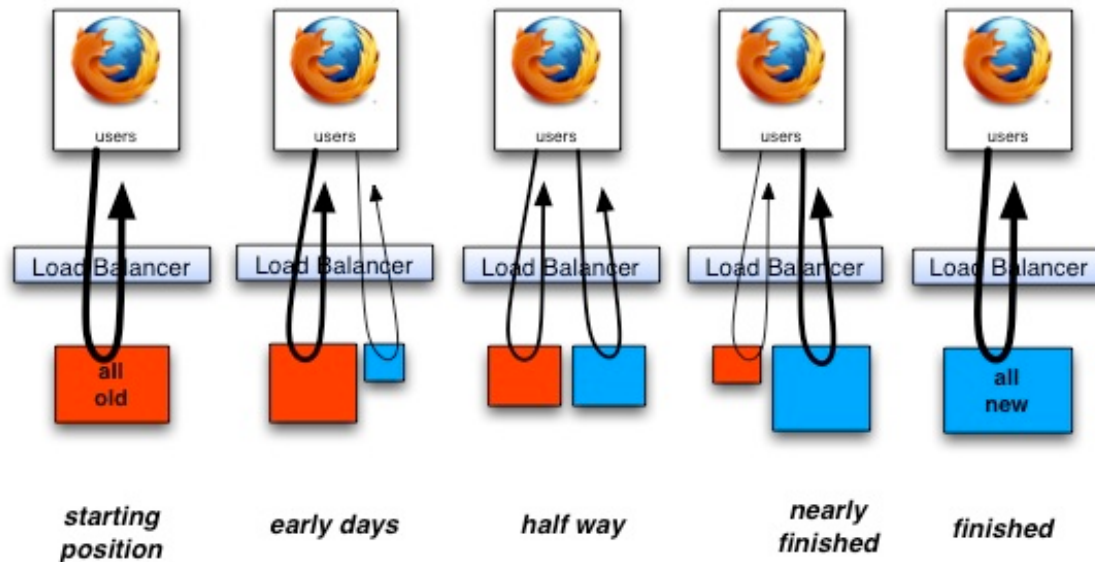








# "Strangler" Pattern



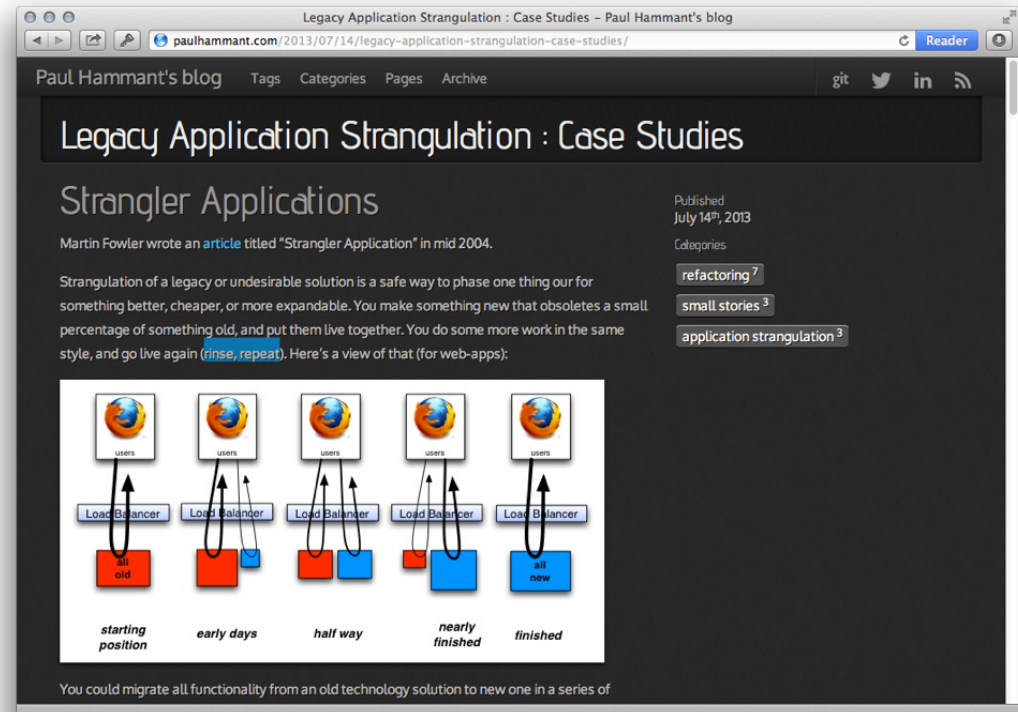
make something new that obsoletes a small percentage of something old

put them live together

rinse, repeat

# "Strangler" Pattern

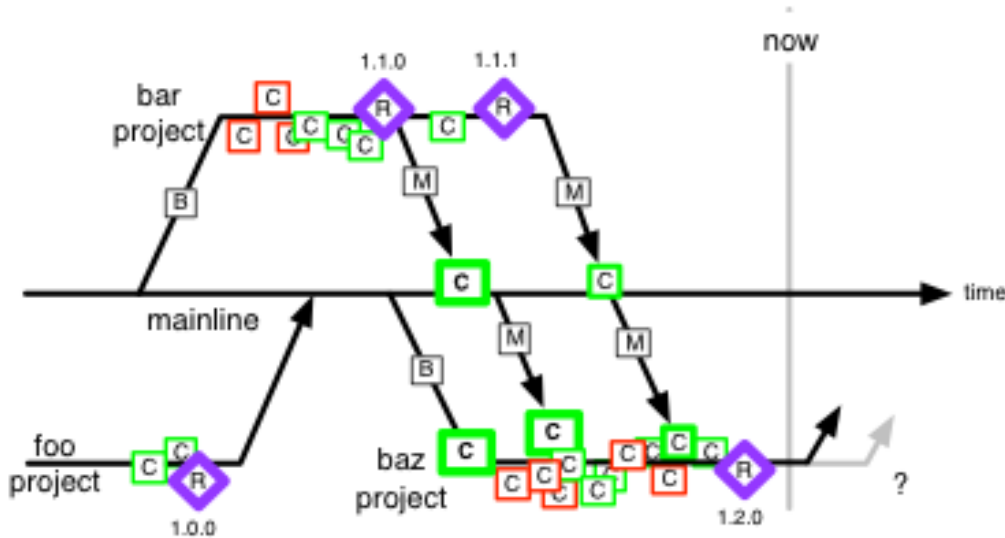
"We never told the users that they must use the new system. Nor did we remove access to the old system. We relied on making the system so compelling that there was no reason to use the old. This also meant that we stayed focused on the users real requirements"



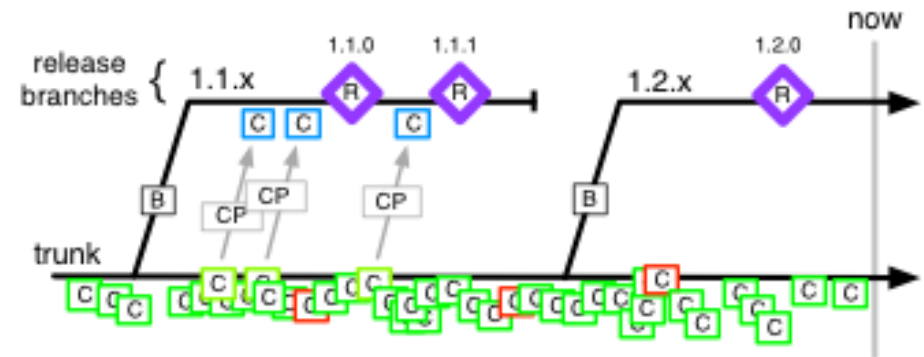
[paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/](http://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/)

# Release branches are OK...

Mainline

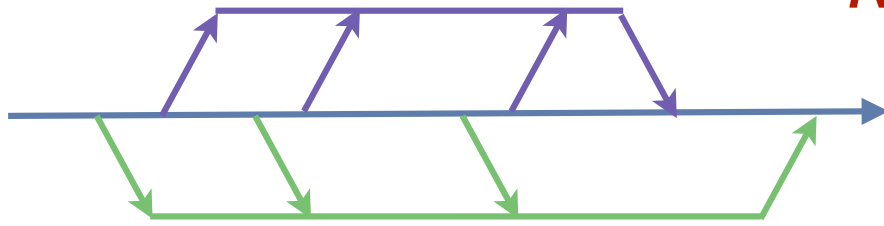


Trunk Based Development



Long-lived branches damage continuous integration.

Feature Branching



*No Refactoring*



*Big Scary Merge*



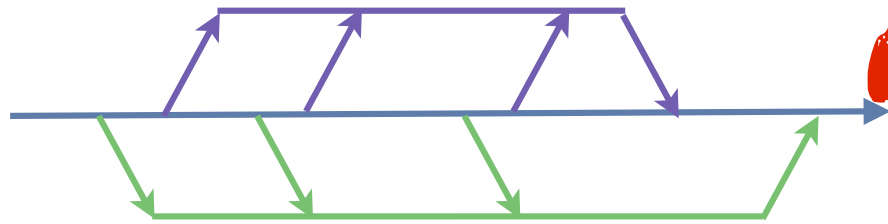
*Untrusted  
Contributors*

*Cherry Picking*



Continuous Integration

Feature Branching



*No Refactoring*

2



*No combined features*

3



*Big Scary Merge*

1



*Untrusted Contributors*

*Cherry Picking*

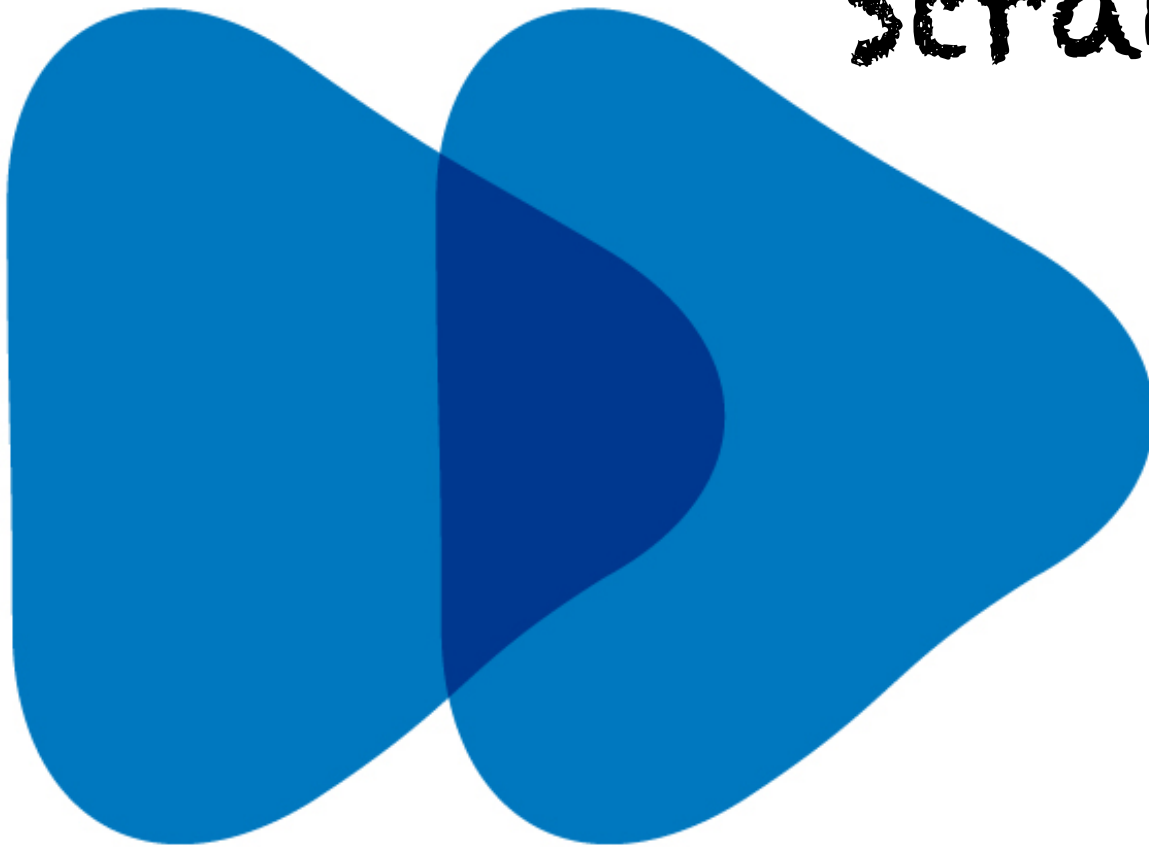


Continuous Integration

meta-work is  
more interesting  
than work

make sure your practices support your  
goals (and don't become goals unto  
themselves)

# Release Strategies



blue-green deployments

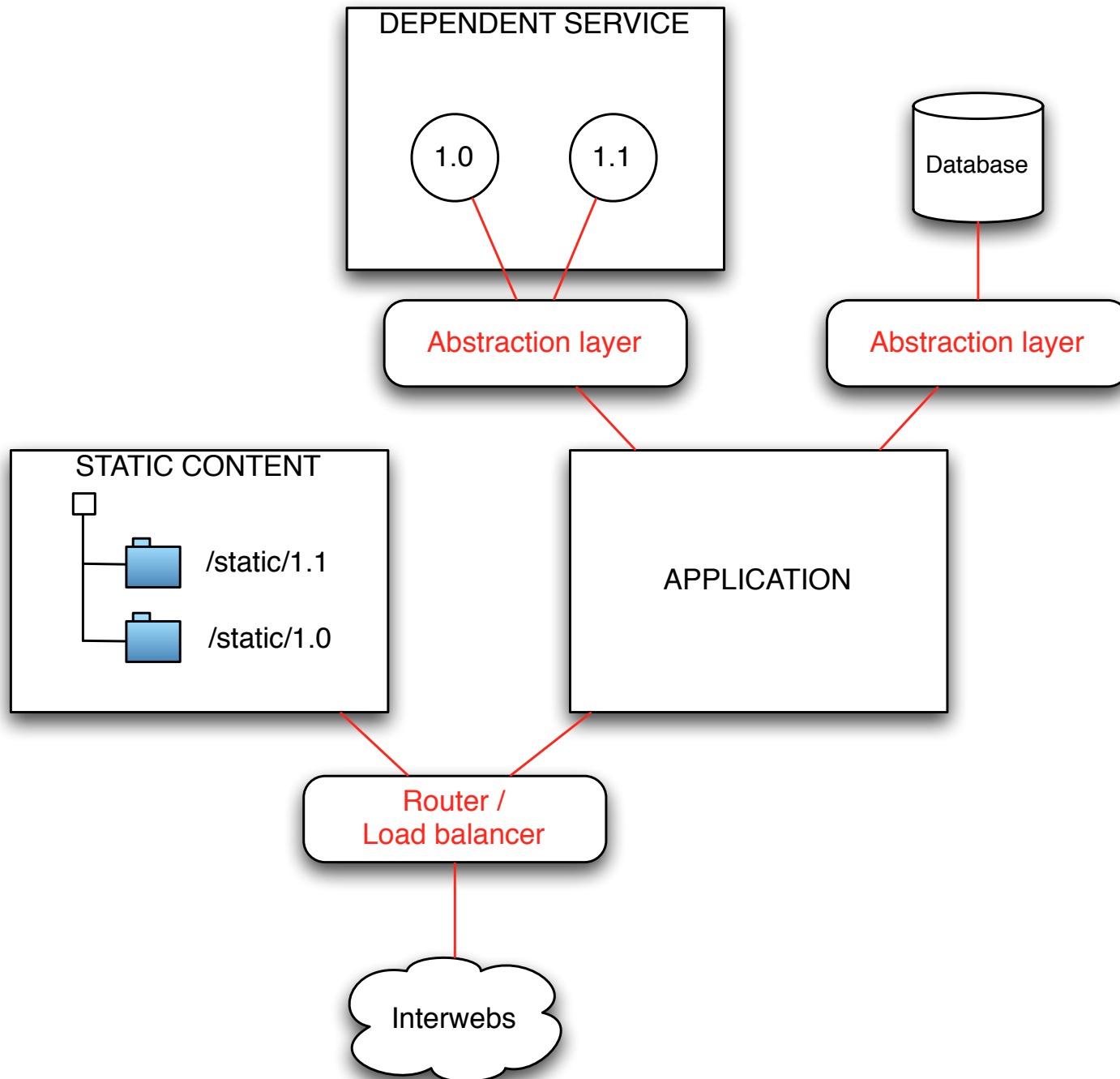
canary releases

# Incremental Release Strategies

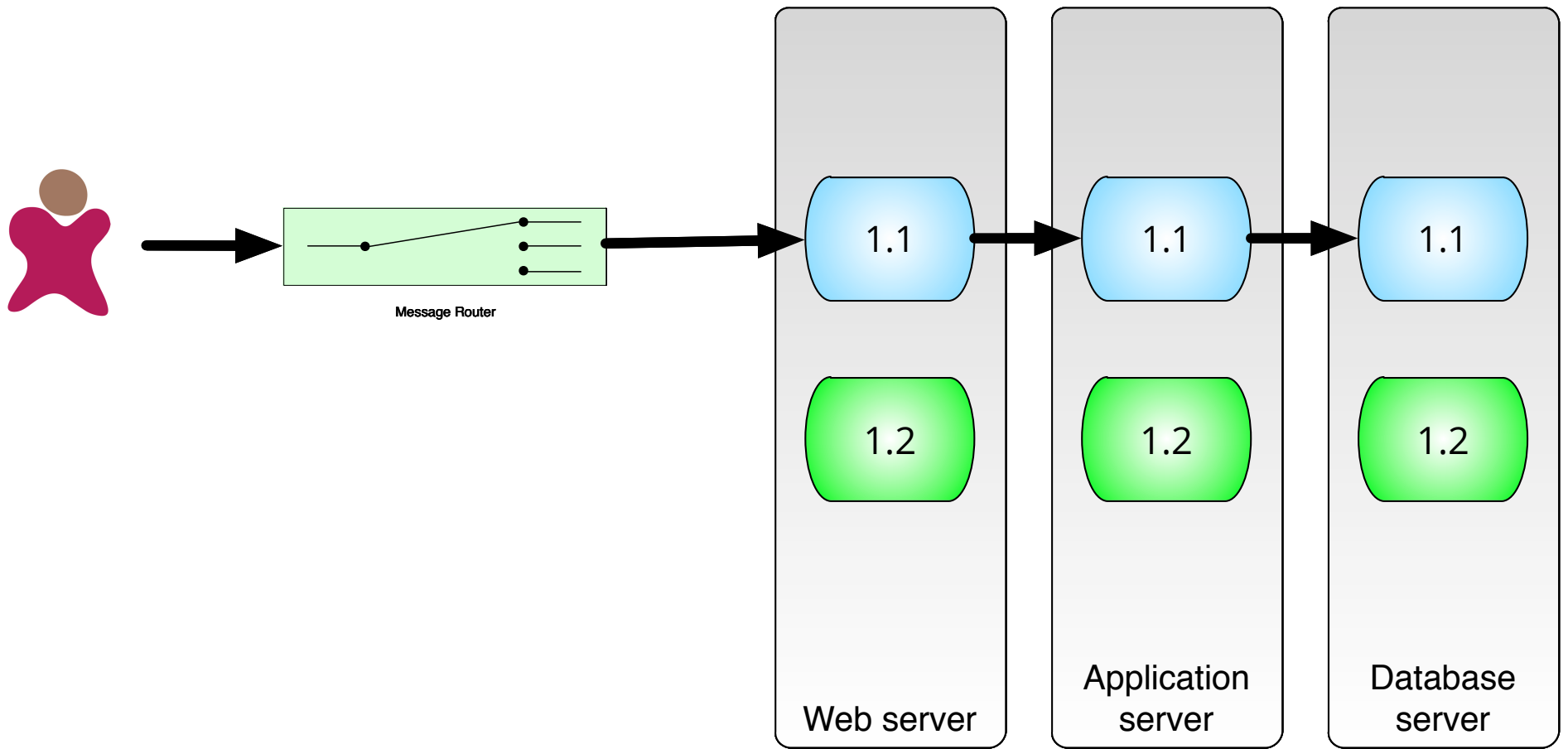
dark launching

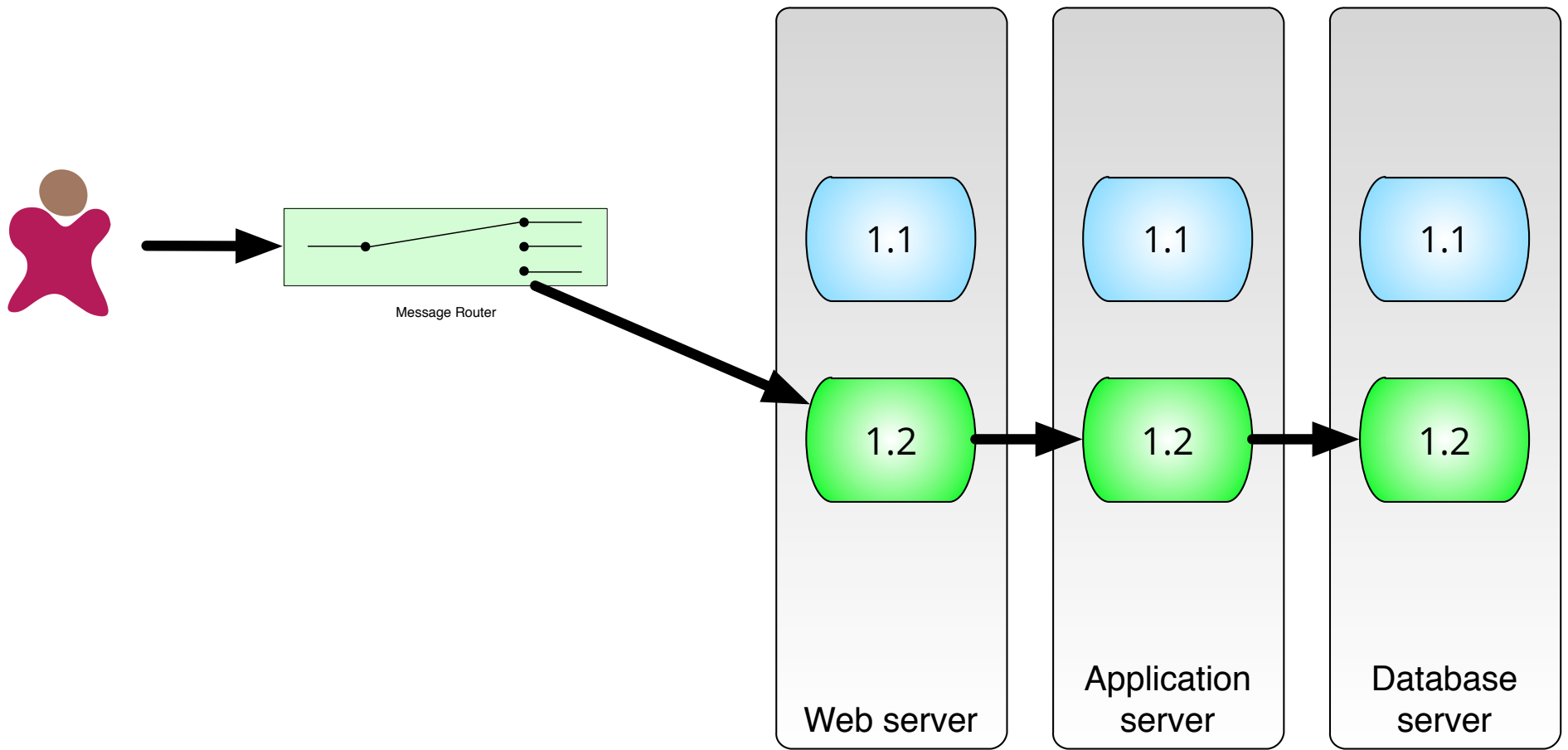
production immune system





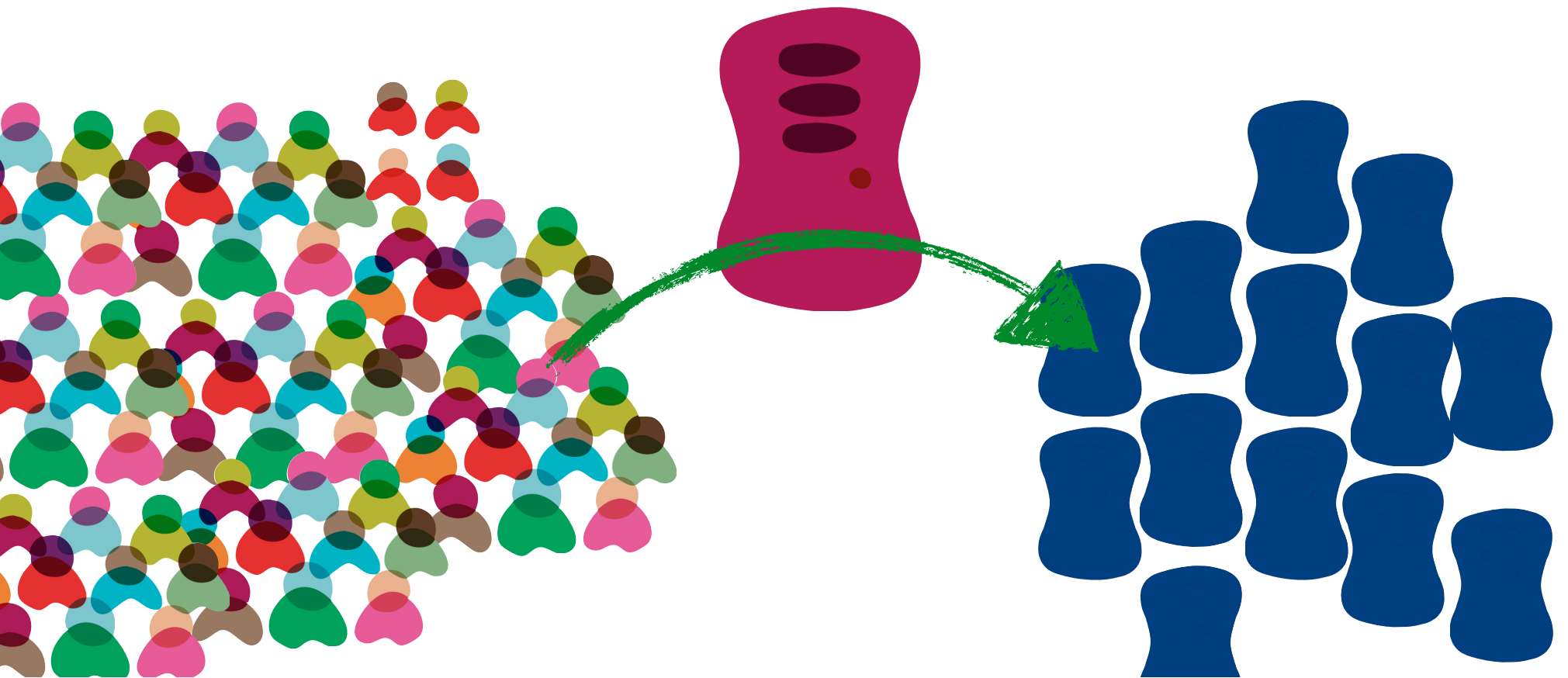
blue-green  
deployments



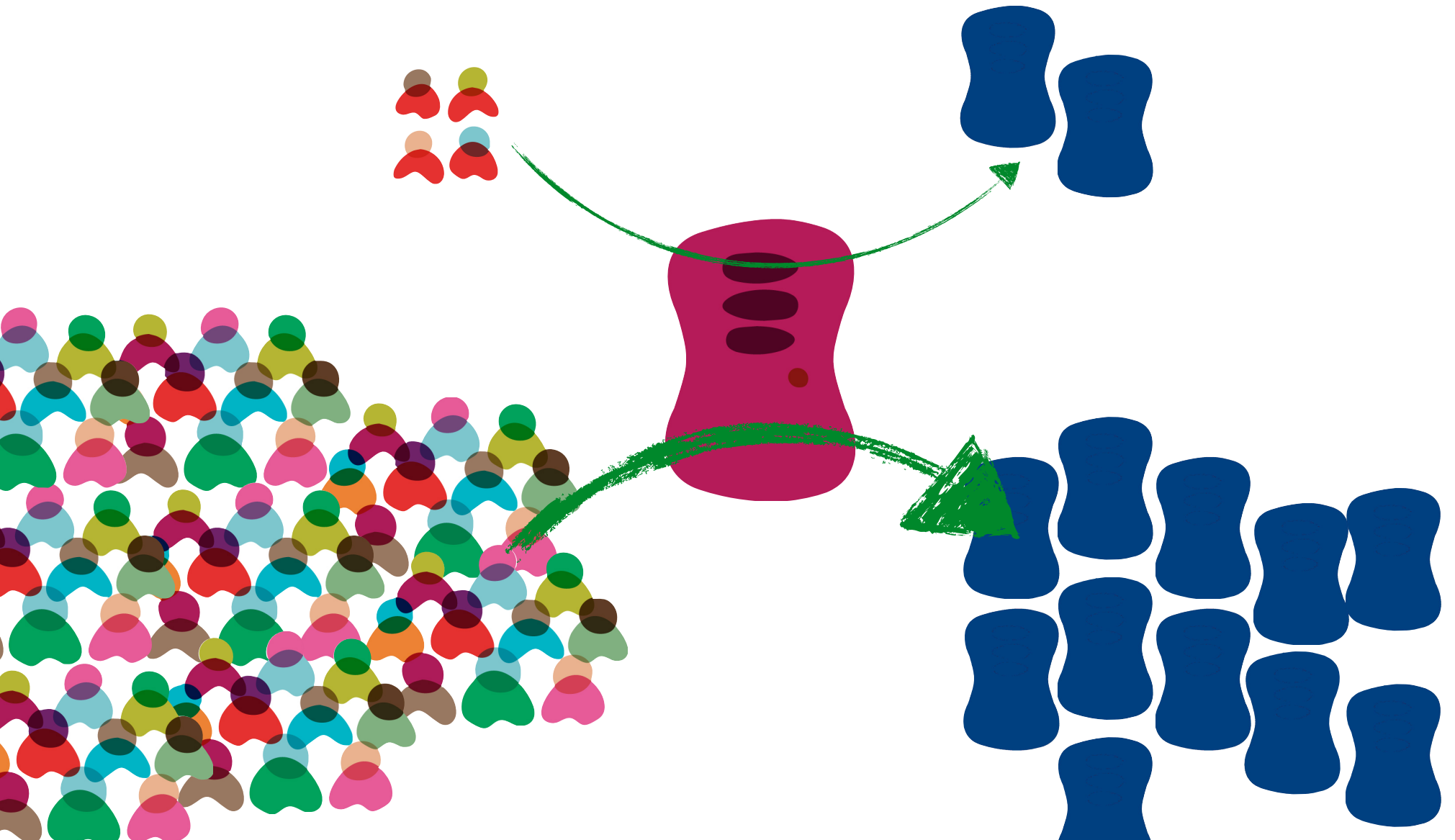


Canary Releasing

# Canary Releasing

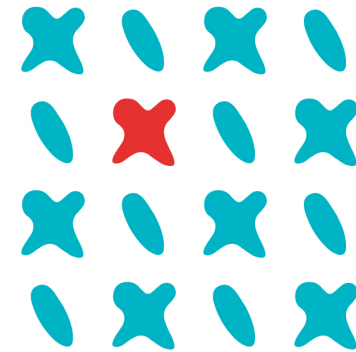


# Canary Releasing



# Canary Releasing

reduce risk of release

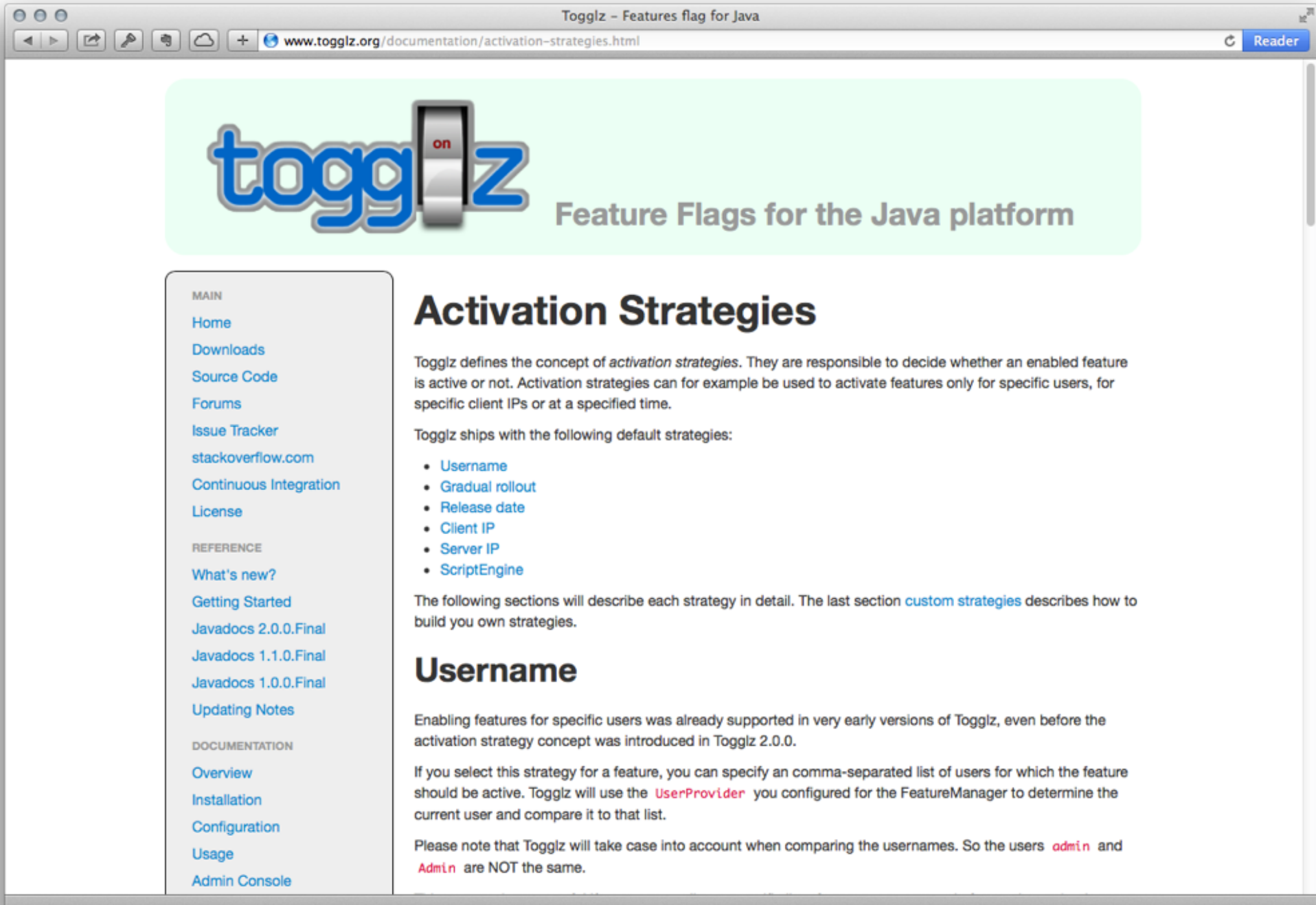


multi-variant testing



performance testing





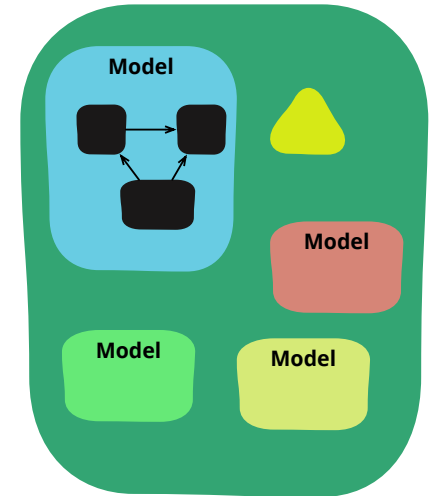
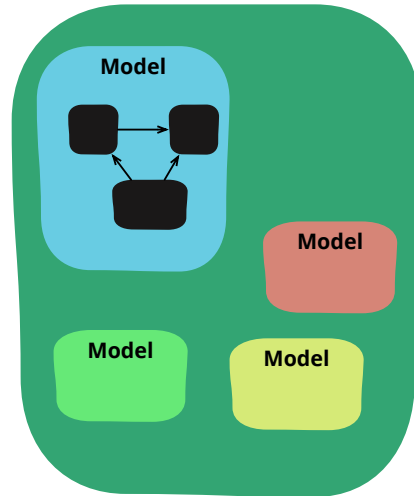
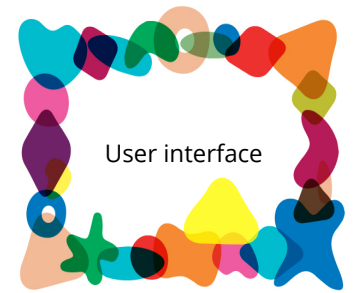
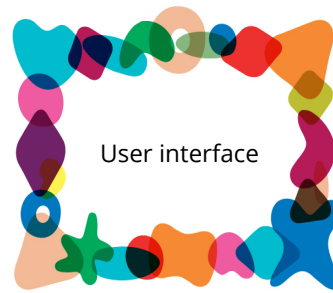
[www.togglz.org](http://www.togglz.org)

A microscopic image showing a dense cluster of immune cells on the left and more scattered cells on the right. The cells have blue nuclei and green or red cytoplasm, with some showing bright yellow or orange spots. The background is black.

# immune system

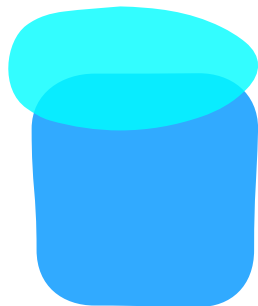
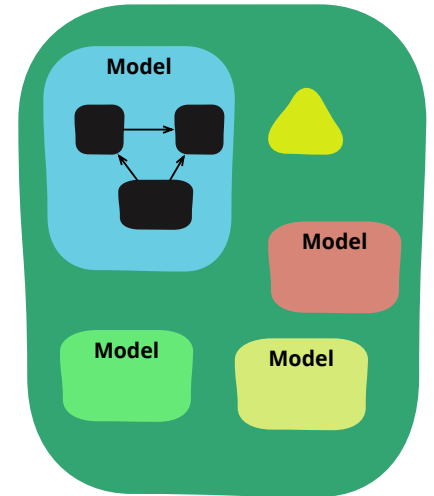
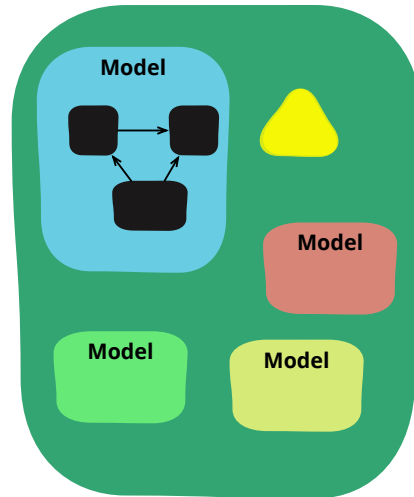
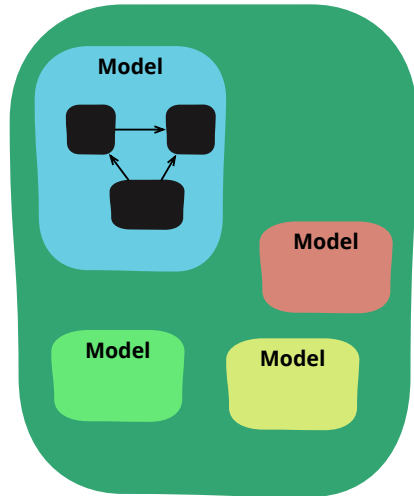
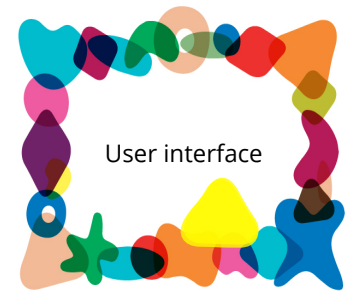
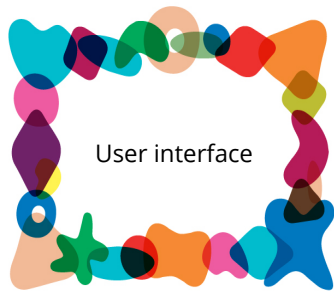
What if a developer accidentally  
replaced the "*Buy*" button with  
"*spacer.gif*" ?

Dark Launching

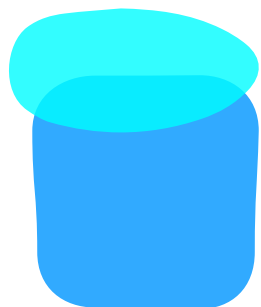
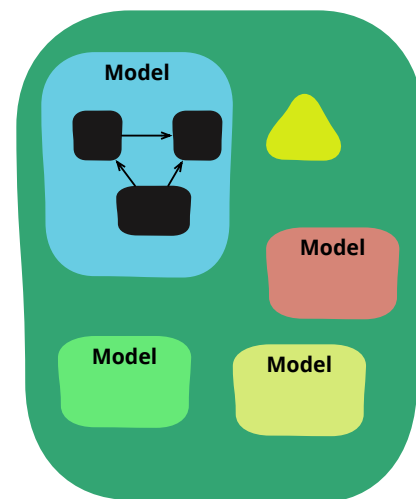
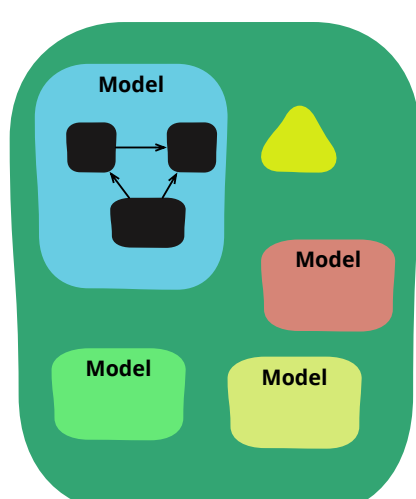
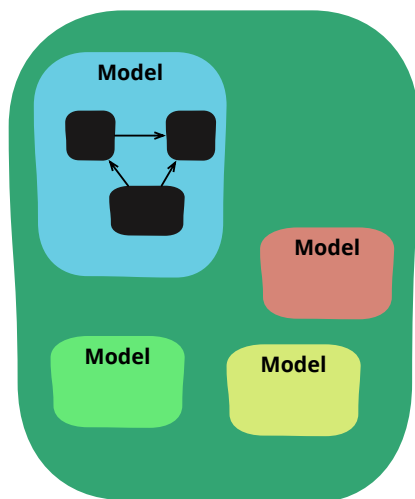
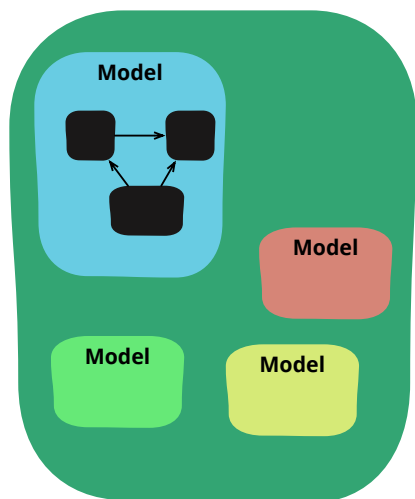
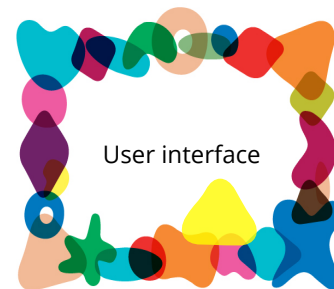


*timeline*

Dark Launching

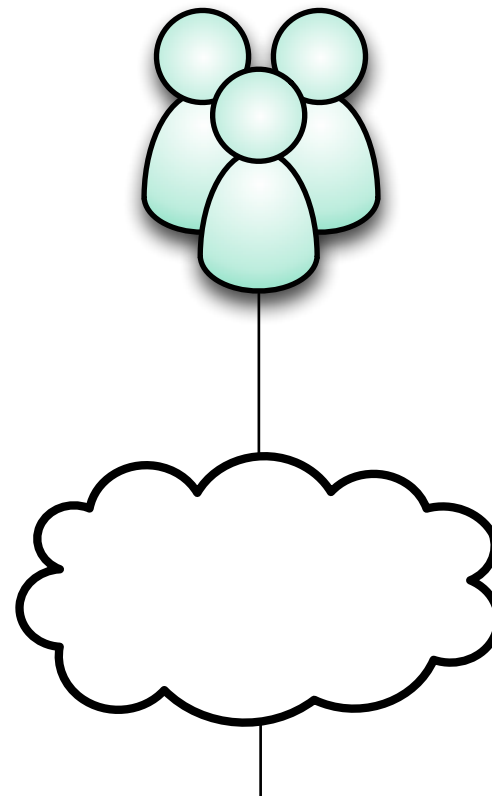
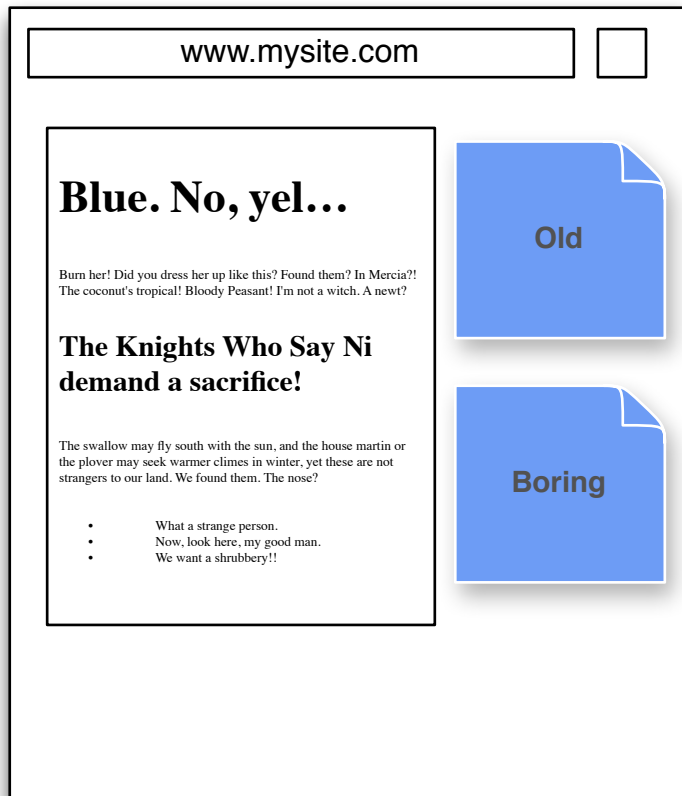


*timeline*



*timeline*





returns page v1

request page

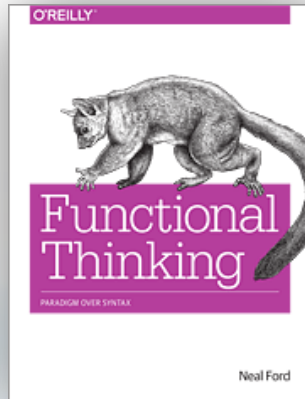
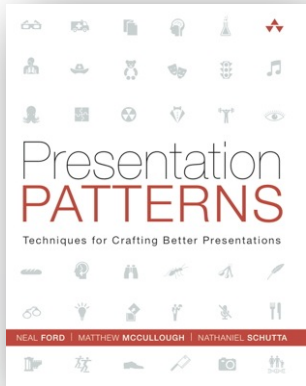
returns page v2

All servers have the same software

Emergency  
Fixes?



Remediation?



[nealford.com](http://nealford.com)



@neal4d

ThoughtWorks®

**NEAL FORD**

*Director / Software Architect / Meme Wrangler*

