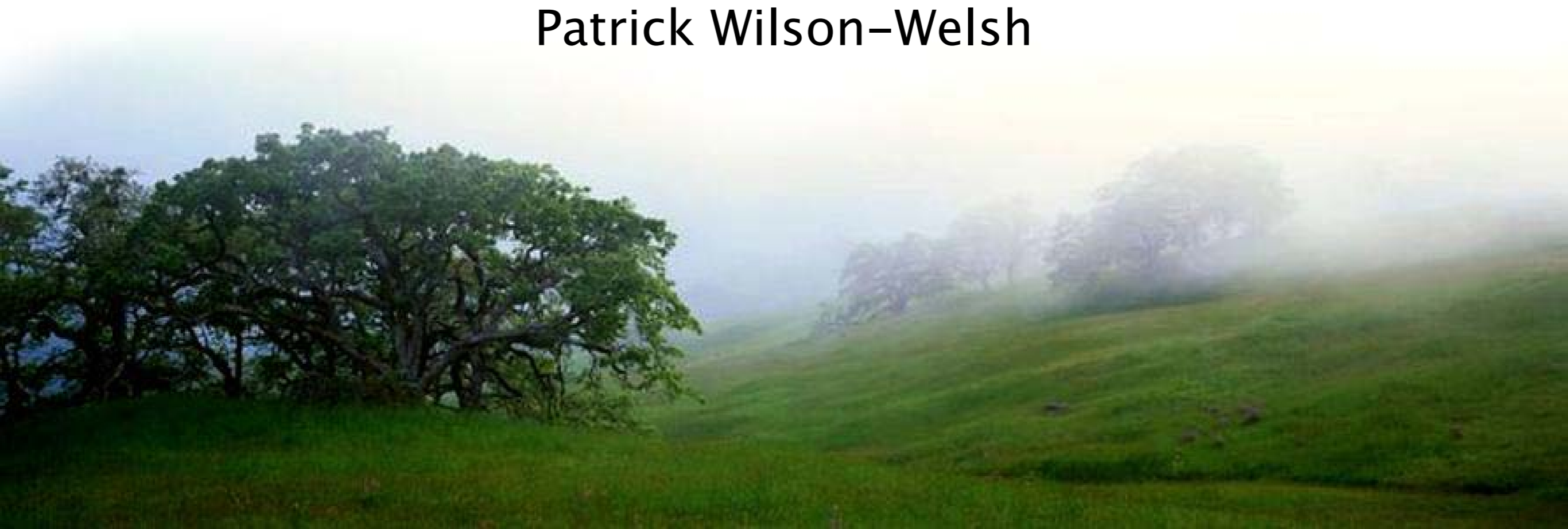


Se 1 RC Java

Advanced Patterns

Patrick Wilson-Welsh



Introduction

Patrick Wilson-Welsh

Senior Consultant

Pillar Technology Group

<http://pillartechnology.com>

pwelsh@pillartechnology.com

<http://patrickwilsonwelsh.com>

<http://coderetreat.ning.com/>

mobile: 248 565 6130

twitter: patrickwelsh

blog: patrickwilsonwelsh.com

[https://github.com/PillarTechnology/Sele
niumPatterns](https://github.com/PillarTechnology/Sele
niumPatterns)

I'm here to help ...



MISTAKES

IT COULD BE THAT THE PURPOSE OF YOUR LIFE IS
ONLY TO SERVE AS A WARNING TO OTHERS.

www.despair.com



Selenium is the leading open-source tool for automated web application testing.

Used well, it can do things other testing tools cannot.

Used poorly, it can cause enormous headaches.

Selenium IDE 1.0.10 *

Base URL:

Fast Slow

Test Case: **Untitled ***

Table Source

Command	Target	Value
open	/	
click	jumper	
click	link=Accounts	
waitForTextPresent	Padberg, Hammes and K...	
waitForPopUp		30000
clickAndWait	//div[@id='tabs']/ul/li[4...	
verifyTextPresent	CARLOS GIRALLDO	
clickAndWait	//div[@id='tabs']/ul/li[3	

Command:

Target:

Value:

Runs: 0

Failures: 0

Log Reference UI-Element Rollup

open(url)

Arguments:

- url - the URL to open; may be relative or absolute

Opens an URL in the test frame. This accepts both relative and absolute URLs. The "open" command waits for the page to load before proceeding, ie. the "AndWait" suffix is implicit. *Note:* The URL must be on the same domain as the runner HTML due to security restrictions in the browser (Same Origin Policy). If you need to open an URL on another domain, use the Selenium Server to start a new browser session on that domain.

Revenue | 234 leads | 19 opportunities | 8.0% conversion

Se IDE

**Se IDE: Adam,
Please Remove the Save Feature**

Don't use Se IDE except for prototyping

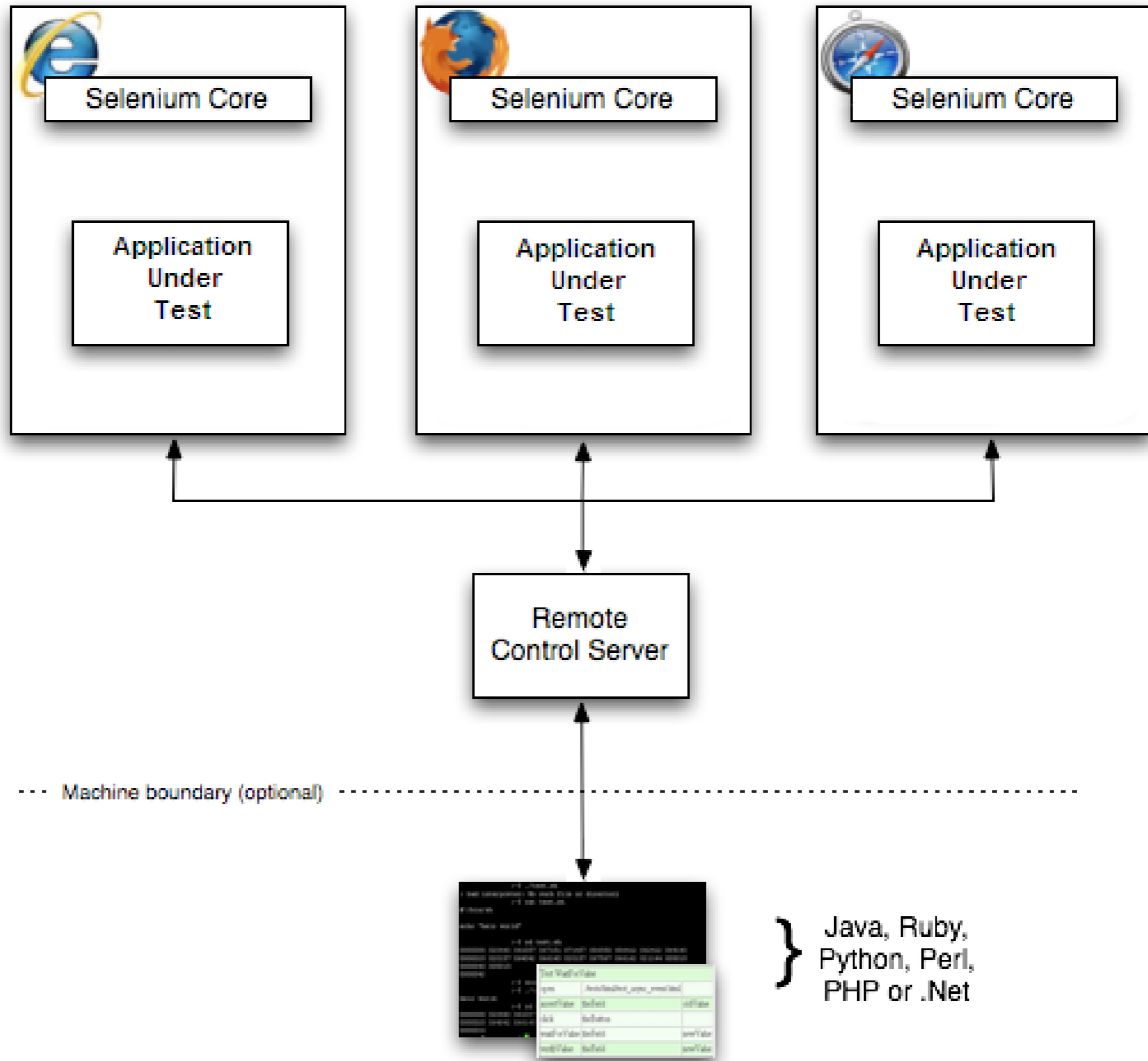
Hand-roll Object-Oriented Se RC tests

Selenium RC allows you to write Se tests in several languages, including Java.

It allows you to write decent OO Se test code as you would write OO production java. But its API is not inherently OO.

It allows you to write Se test code as if it was really bad Fortran. Migrating from Se IDE to Se RC: especially dangerous.

Se RC



Example 1: Bad-Ole Se 1 RC Multi-Page-Flow

Procedural code for verifying that the same links show up on each of several pages.

Even method extraction and custom assertions may not help as much as you need.

```
@Test
```

```
public void allPageNav() {
    selenium.open("http://demo.fatfreecrm.com/login");
    selenium.waitForPageToLoad("60000");

    selenium.type("css=input[id=authentication_username]", "seleniumpatterns");
    selenium.type("css=input[id=authentication_password]", "seleniumpatterns");
    selenium.click("css=input[id=authentication_submit]");
    selenium.waitForPageToLoad("60000");

    assertTrue(selenium.isElementPresent("css=div[id=welcome]"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] span[id='welcome_username']"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] a[id=jumper]"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] a[href='/profile']"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] a[href='/logout']"));

    selenium.click("css=div[id=tabs] a:contains('Tasks')");
    selenium.waitForPageToLoad("60000");

    assertTrue(selenium.isElementPresent("css=div[id=welcome]"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] span[id='welcome_username']"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] a[id=jumper]"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] a[href='/profile']"));
    assertTrue(selenium.isElementPresent("css=div[id=welcome] a[href='/logout']"));

    selenium.click("css=div[id=tabs] a:contains('Campaigns')");
    selenium.waitForPageToLoad("60000");
}
```

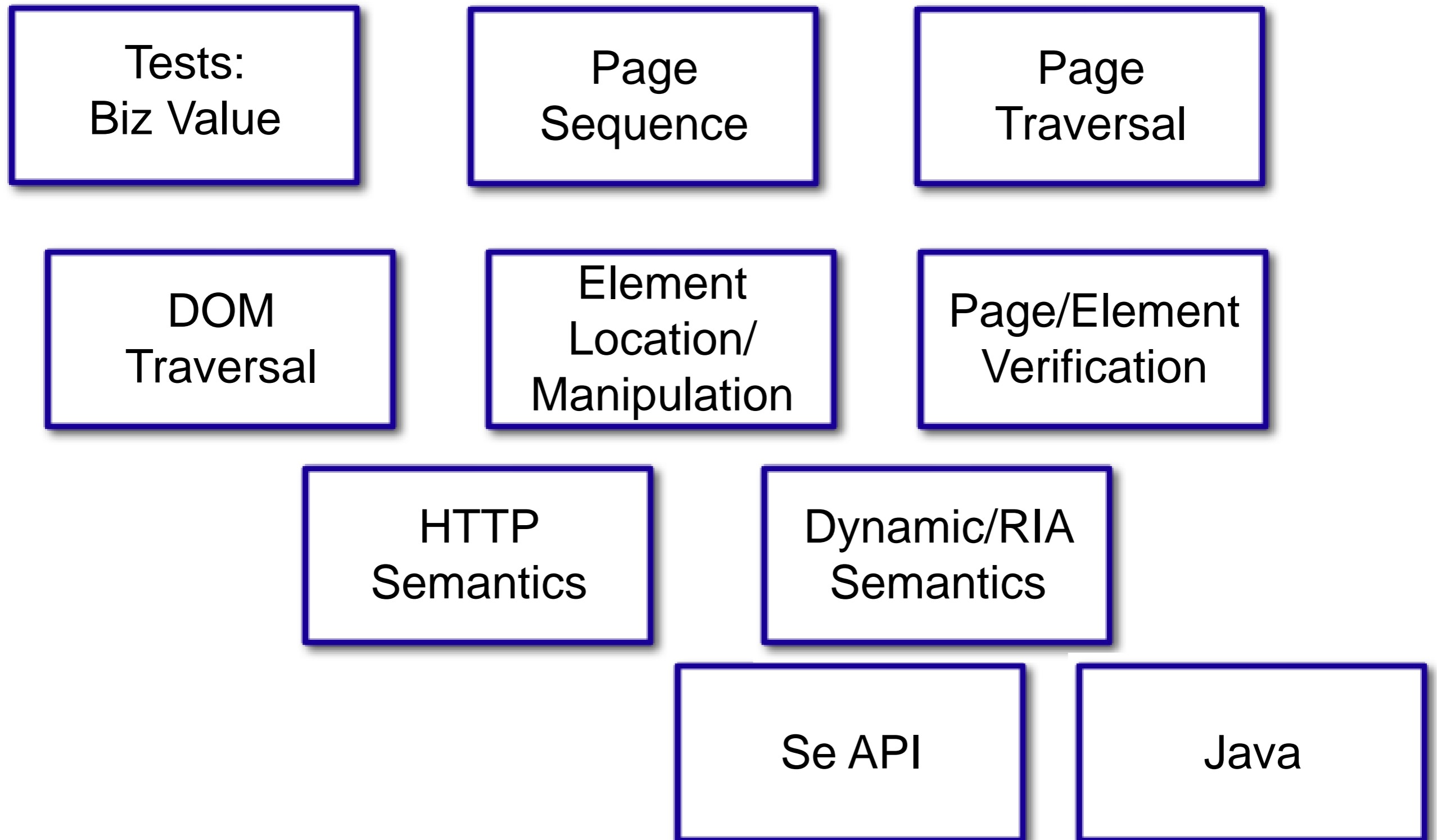
Domain-Specific Languages (DSLs)

How many different domain semantics are being mixed up in the above code? How many abstraction layers are coupled?

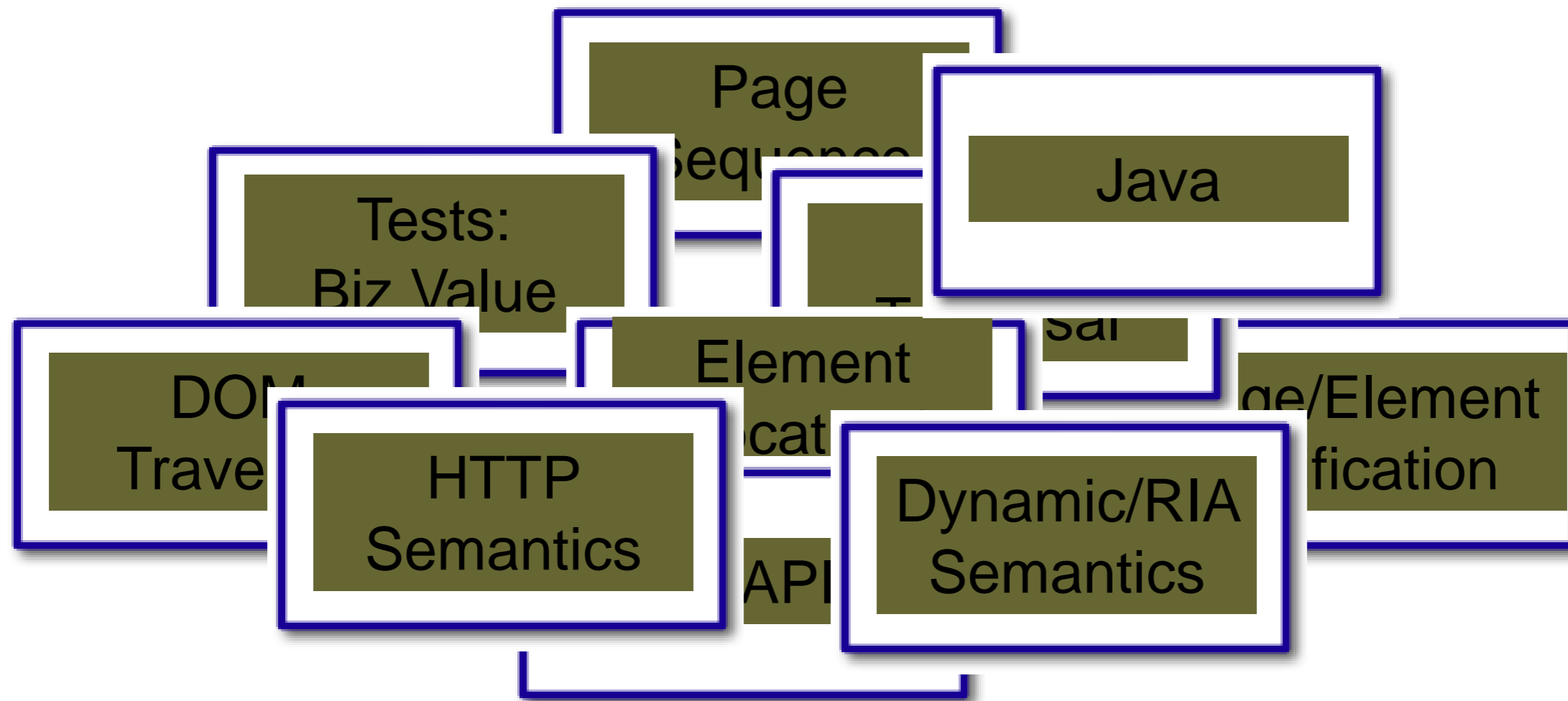
I'm abusing the term to mean, roughly, semantics or APIs or collections of modules that fit within certain domain boundaries.

I care much more about DSL boundaries than I do about their level of fluency (and I love natural-language fluency).

Separable Concerns, Layers, DSLs in Se Testing?



Don't Mix All The Food Coloring. You Get Brown.



Problems When you Mix the DSLs

- *Noisy tests*
- *Coupling/poor cohesion.*
- *Brittle tests*
- *Shotgun surgery*

Problems When you Mix the DSLs

- *Obscure page-flow and page verification.*
- *Messy state verification.*
- *Dynamic HTML/Ajax problems*
- *Inconsistent element locator strategies*
- *Very slow execution on some browsers*

A concern-separation principle to consider for Se testing:

DRY, reusable test framework.

Self-Verifying
“Page Objects”

Generic HTML
Element
Objects

Parameterized
“Link Objects”

Se Façade/
Decorator

src source folder: DRY reusability

src folder (or wherever) contains only reusable **domain-specific** and **domain-independent** classes.

src source folder: DRY reusability

Domain-specific pages, panes, and common components (nav menus, etc). These use the natural biz/page DSL.

Domain-independent framework (page elements/controls, Selenium and jQuery façade/decorator, locator strategy code)

A DSL-separating principle to consider for Se testing:

“Wet”, one-off test code.

```
public class FindAccountTest extends BaseWebTest {
    @SuppressWarnings("unchecked")
    @Test
    public void canFindExistingAccount() throws Exception {
        QuickFindBox<SmithamPage> quickFind = CommonComponents.topRightLinks.quickFind.clickToNewContainer();

        SmithamPage smithamAndSonsPage = quickFind.searchForAccount("Smitha", SmithamPage.class);
        assertTrue(smithamAndSonsPage.title.reads("Smitham and Sons"));
    }
}
```

test source folder: one-off, wet test code DSL: Page Flow as Business Domain

test folder contains behaviorally-organized test scenarios. Nearly all of them happy paths.

Test scenarios for page flow manipulation and state verification can include quite a bit of duplication. If you prefer, extract private helper methods like

“loginAndNavigateToSuchAndSuchPage()”

Specific patterns in the
selenium-rc-patterns eclipse project.

Pattern: Reusable Element Objects Framework

- Different kinds of page elements get matching classes; all extend BaseElement.
- All are completely generic; unrelated to biz domain.

Pattern: Reusable Selenium Singleton Facade

- All access to DefaultSelenium and SeleniumServer (Jetty HTTP proxy) are outside test code.
- All access to them is via singleton inside façade
- Façade decorates DefaultSelenium with other cool stuff you need

Pattern: Self-Verifying *Page Objects*: they Know When They Have Arrived



Pattern: Self-Verifying Page Objects

- Biz-domain-specific classes wrap the behavior of specific pages in your app, in page-specific ways.
- When you navigate to a page by clicking on a link, the matching page object gets launched, and waits until it is fully loaded by that object's definition (a CSS element selector).

Pattern: Encapsulated Link Semantics



Pattern: Encapsulated Link Semantics

- When you instantiate a DhtmlLink class, you parameterize it with the PageObject.class that you want to be launched when you click on that link.
- This pushes page flow semantics deep into the actual link flow, emulating actual app, and keeps page flow DRY.

Remember the good old days of every browser-resident change resulting from an HTTP Response to an HTTP Request? (Sigh.)

2 Galleria Pkwy
Atlanta, GA 30339



Directions Search nearby Save to... more

At this address:

- Canary Island Garlic & Herb Olive Oil -
- Cobb Galleria Centre - ★★★★★ 48 reviews
- Mother's Day Expo. 2011 -
- Outdoor Camping RV Super Show -

Guide to Atlanta, Georgia

Find Atlanta Attractions, Events, Activities & More At AJC.com
www.ajc.com/travel
Atlanta, GA

Ads

The map displays the area around 2 Galleria Pkwy in Atlanta, Georgia. It features a traffic overlay with a color-coded legend: red for slow traffic and green for fast traffic. Major roads shown include US-41, Cobb Pkwy SE, Spring Rd SE, and Cumberland Blvd. Landmarks such as the Cobb Galleria Centre, Galleria Mall Shopping Center, and Sheraton Suites Galleria are labeled. A street view inset shows the building at the destination. Navigation controls like a compass and a person icon are visible on the left side of the map.

Whole desktops, coming to browsers near you...

Principles to consider for testing dynamic stuff:

The rendered HTML is less and less your friend.

XPATH is not standard, thus not your friend.

The in-memory DOM is your friend.

CSS element location strategy is your friend.

jQuery is your friend.

Pattern: CSS Element locator strategies for DHTML/Ajax

- Use id or name attributes if you got 'em
- Use css as a second-resort locator
- Only use xpath as a last-resort element locator
- ***Actually, forget it. Never use xpath.***

Pattern: Using jQuery to Verify Dynamic Visibility

Consider using `usingSe` to *inject* jQuery (or some such js library) into a page, to get direct access to the DOM, not the rendered HTML, to verify dynamic changes.



Pattern: Using jQuery to Verify Dynamic Visibility



There will increasingly be behaviors for which jQuery, being cross-browser in a fairly standard way, is a better choice for DOM state verification than older techniques

Pattern: Common Page Element Singletons

Fat Free CRM
Welcome, seleniumpatterns! [Quick find](#) | [Profile](#) | [Logout](#)

Dashboard
Tasks
Campaigns
Leads
Accounts
Contacts
Opportunities

Search campaigns

<input checked="" type="checkbox"/> Planned	21
<input checked="" type="checkbox"/> Started	23
<input checked="" type="checkbox"/> Completed	22
<input checked="" type="checkbox"/> On Hold	8
<input checked="" type="checkbox"/> Called Off	9
<input type="checkbox"/> Other	0
Total Campaigns	83

Campaigns [▶ Create Campaign](#) | [▶ Options](#)

Started **Mitch Test Campaign Name** started about 17 hours ago (was supposed to finish on Mar 22, 0011)
 \$0 in revenue | 2 leads | 1 opportunity | 50.0% conversion

Completed **The ultimate driving machine** completed on Apr 24, 2011
 Target: \$47,000 in revenue | 259 leads | 13.0% conversion
 Actual: \$798,000 in revenue | 250 leads | 9 opportunities | 3.0% conversion

Started **Think Outside the Bun** started 5 days ago, finishes in about 1 month
 Target: \$656,000 in revenue | 181 leads | 8.0% conversion
 Actual: \$0 in revenue | 1 lead | 0 opportunities

Started **Don't leave home without it** started 16 days ago (was supposed to finish on Mar 6, 2011)
 Target: \$776,000 in revenue | 266 leads | 8.0% conversion
 Actual: \$328,000 in revenue | 223 leads | 9 opportunities | 4.0% conversion

Planned **All the new shades fit to print** ...

Recent Items

Account:	Smitham and Sons
Account:	Padberg, Hammes ...

Pattern: Common Page Element Singletons

Fat Free CRM Welcome, seleniumpatterns! [Quick find](#) | [Profile](#) | [Logout](#)

[Dashboard](#) | **[Tasks](#)** | [Campaigns](#) | [Leads](#) | [Accounts](#) | [Contacts](#) | [Opportunities](#)

Pending | Assigned | Completed

<input type="checkbox"/> Overdue	0
<input type="checkbox"/> As Soon As Possible	0
<input type="checkbox"/> Today	0
<input type="checkbox"/> Tomorrow	0
<input type="checkbox"/> This Week	0
<input type="checkbox"/> Next Week	0
<input type="checkbox"/> Sometime Later	0
Total Pending Tasks	0

Recent Items

Account: [Smitham and Sons](#)

Account: [Padberg, Hammes ...](#)

Tasks

[▶ Create Task](#)

You don't have any pending tasks. Feel free to [create new task](#).

Pattern: Common Page Element Singletons




Fat Free CRM Welcome, seleniumpatterns! [Quick find](#) | [Profile](#) | [Logout](#)

[Dashboard](#)
[Tasks](#)
[Campaigns](#)
[Leads](#)
[Accounts](#)
[Contacts](#)
[Opportunities](#)

Recent Items

There are no recent items to display yet.

Recent Activity [▶ Options](#)

	user1238@mail.nu deleted contact CARLOS GIRALLDO	Mar 21 at 1:24pm
<hr/>		
	sample@gmail.com completed task Llamar cliente Mexico Arturo	Mar 21 at 1:15pm
<hr/>		
	sample@gmail.com created task Llamar cliente Mexico Arturo	Mar 21 at 1:10pm

Pattern: Common Page Element Singletons

If all pages (or groups of pages) in an app all share certain common elements, consider hanging a static singleton off a BasePage object for all pages, or for that group of pages.

How *hard* is it to regression test an entire enterprise web app using *any* web-app-GUI-black-box testing tool?



too hard: avoid

Number #1 Reason Selenium RC, through the web app GUI tests are so brittle and expensive to maintain?

Because you are writing
too many of them.

Let me put that another way.

You are committing too high a percentage of total test-automation and programming resources to *that specific kind* of automated testing.

“Driving on the ice in in blizzard in Michigan.”

Avoid it whenever you can.

Know how to do it well.

What is Selenium *not good at testing*?

- Every functional app path (JUnit instead)
- Discrete app behaviors (JUnit instead)
- Javascript behaviors (Jasmine instead)
- All view layer behaviors
- Browser-vendor-neutral behavior (HTMLUnit)
- Automated Acceptance Testing (BDD Tools)

What is Selenium *good at testing*?

The browser-resident app behaviors you cannot test *any other way*. It is your tool of *last resort*.

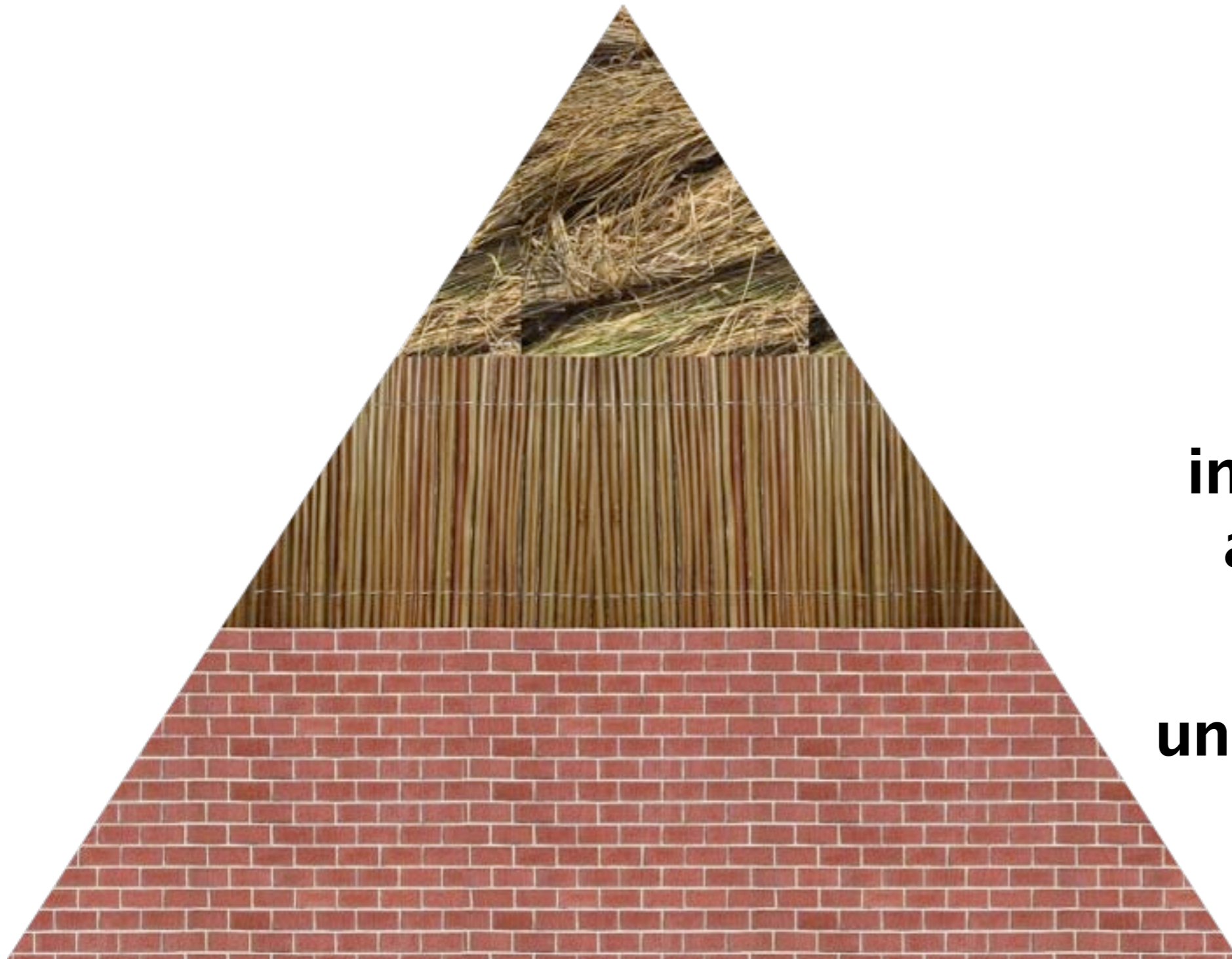
- Testing a few full-stack happy paths
- Browser-compatibility testing
- Tricky view behaviors
- Automated functional demos
- Demo BDD (with Cuke or JBehave on top)
- Policing load-time thresholds

If you are committing too high a percentage of total test-automation and programming resources to Se testing...

...have a plan for outgrowing that pattern.
Actually, have a fairy tale:



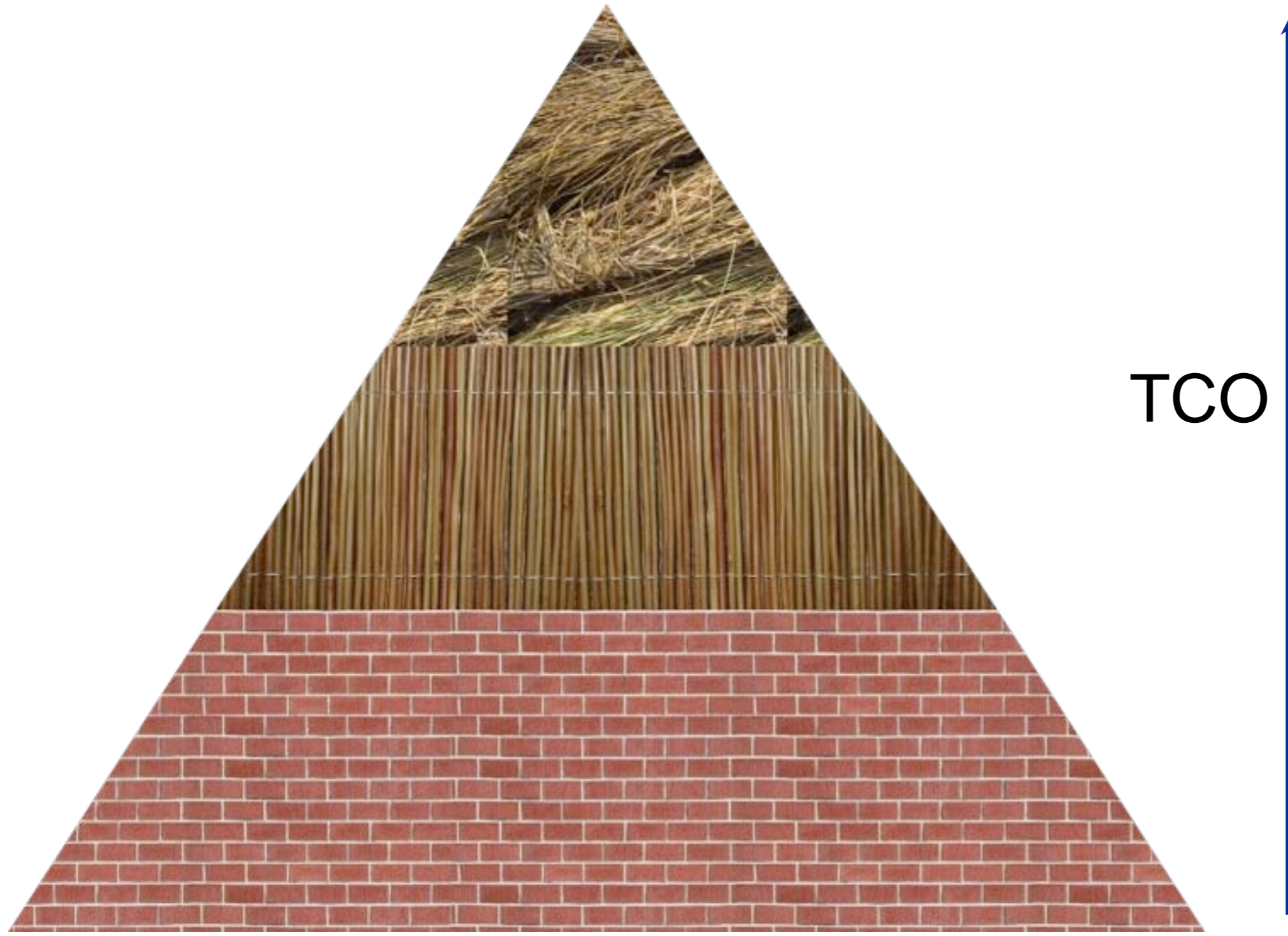
3 kinds of automated tests



GUI

**end-2-end,
integration, story,
acceptance, BDD**

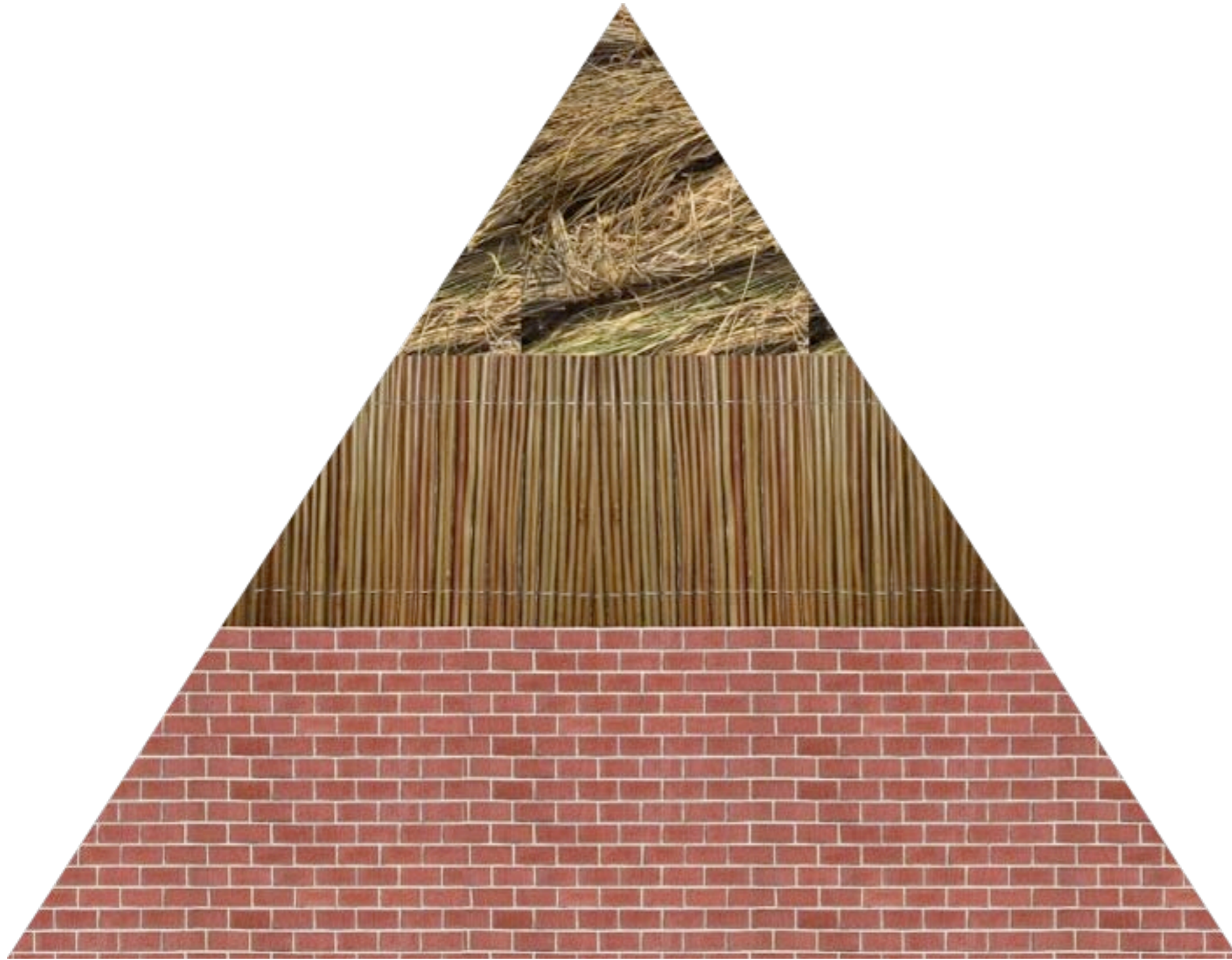
unit/micro/isolation
n



TCO

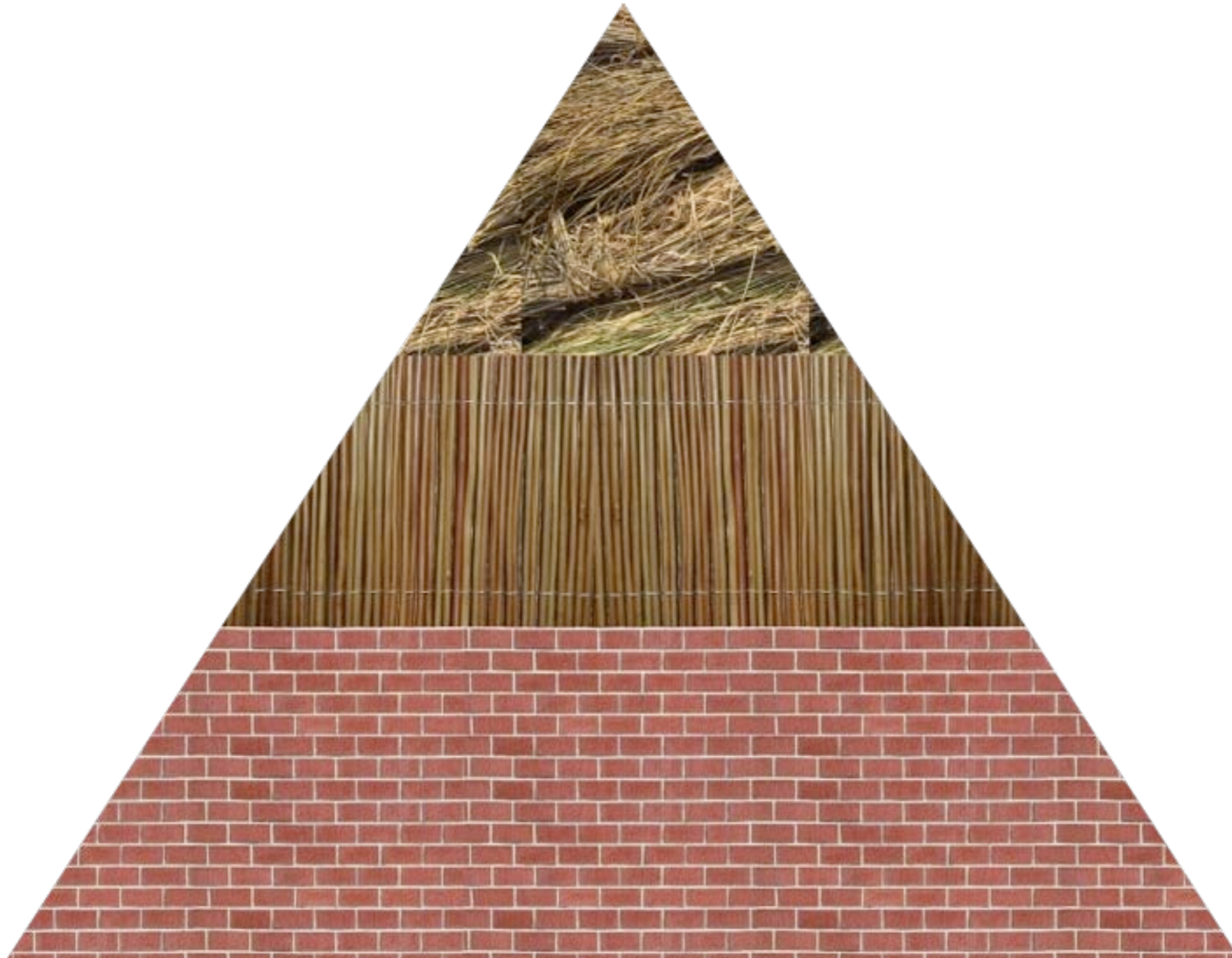
% of automated testing work





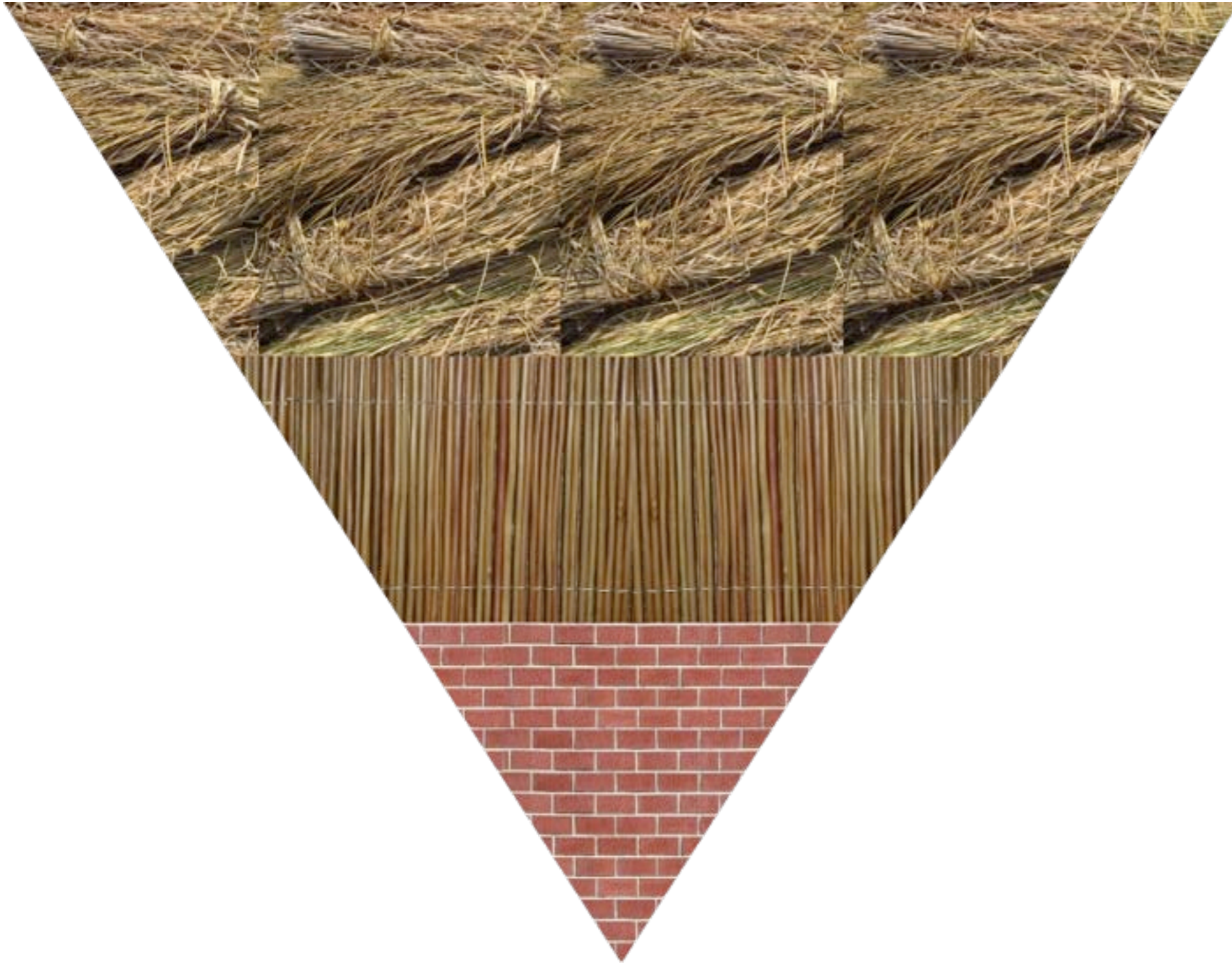
good:
least
investment/ROI

good:
most
investment/ROI

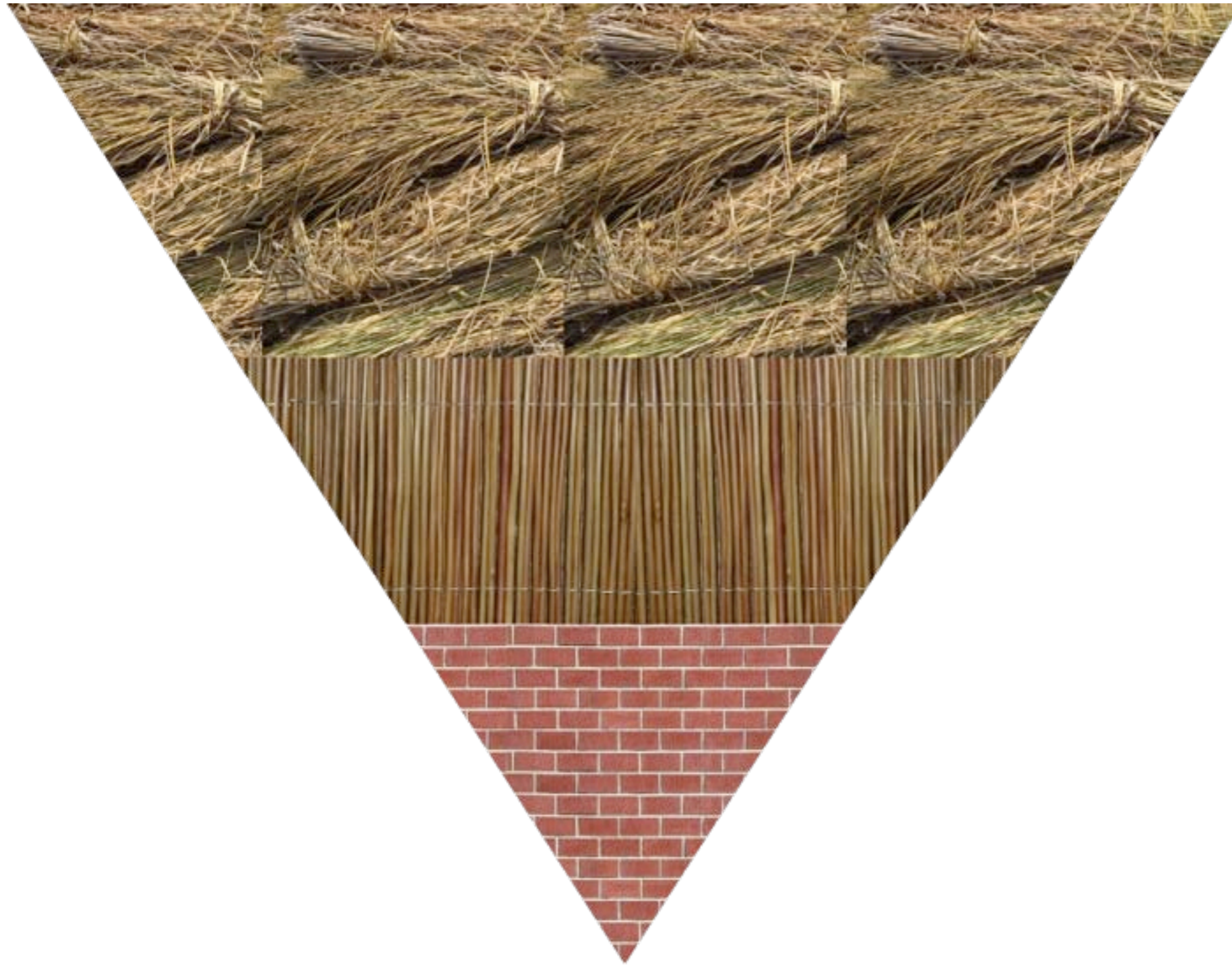


least rework & waste; lowest TCO

But that's not your triangle, yet, perhaps.



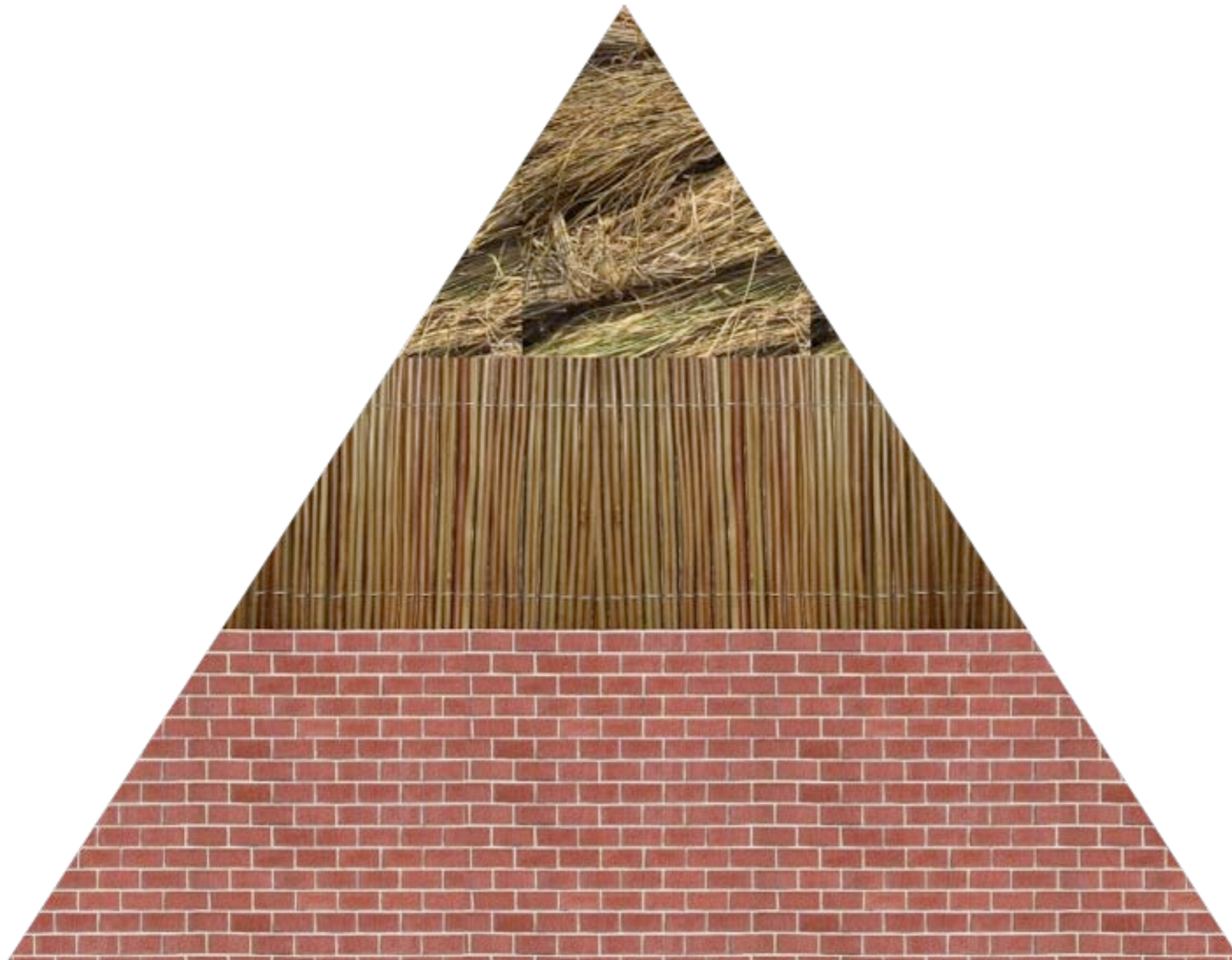
we often start here



good:
easy to learn

bad:
hard to learn

for good reason!



bad:
high TCO,
low ROI

good:
low TCO,
high ROI

But again, the price

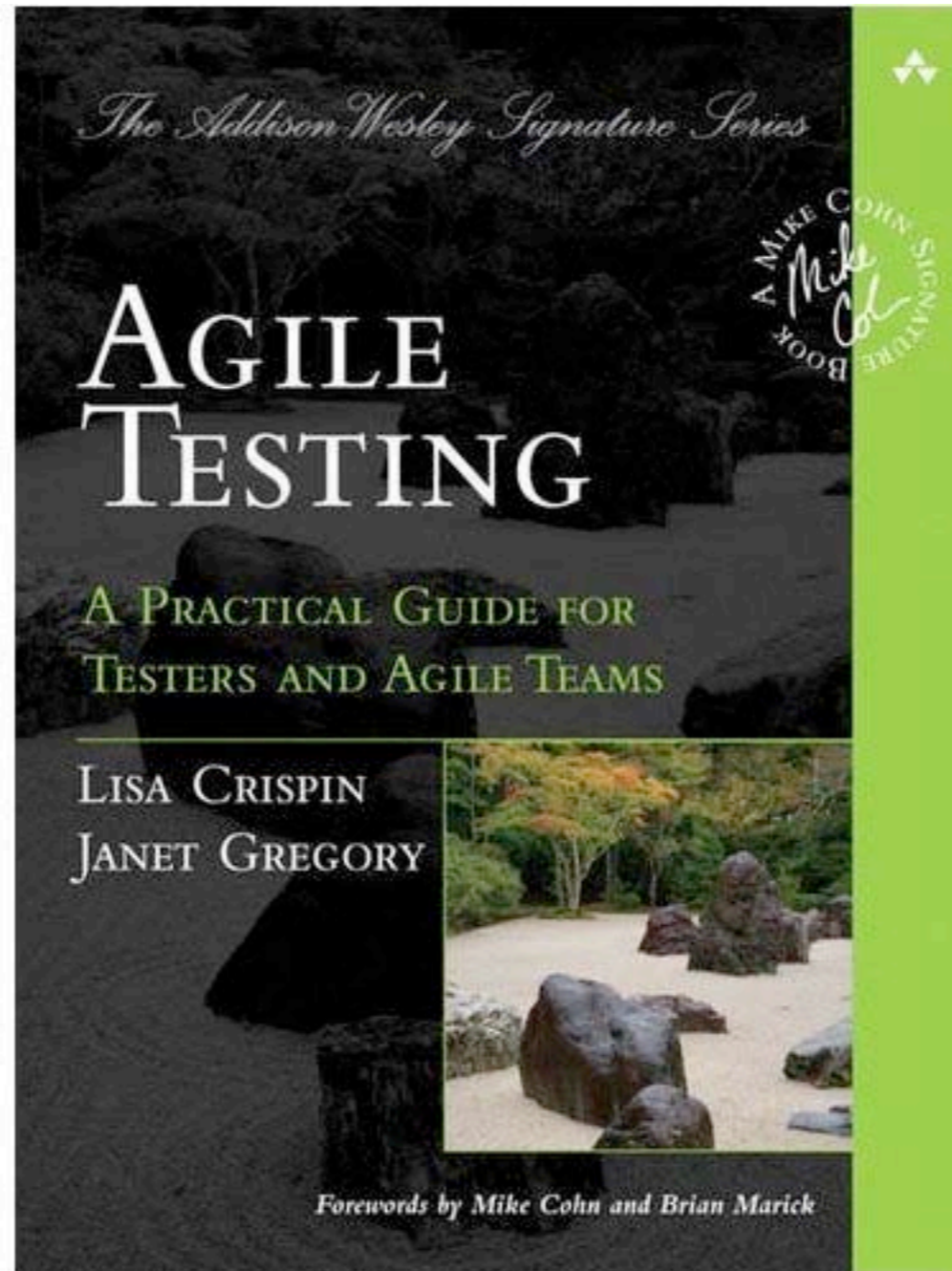
Summary

Use Se less.

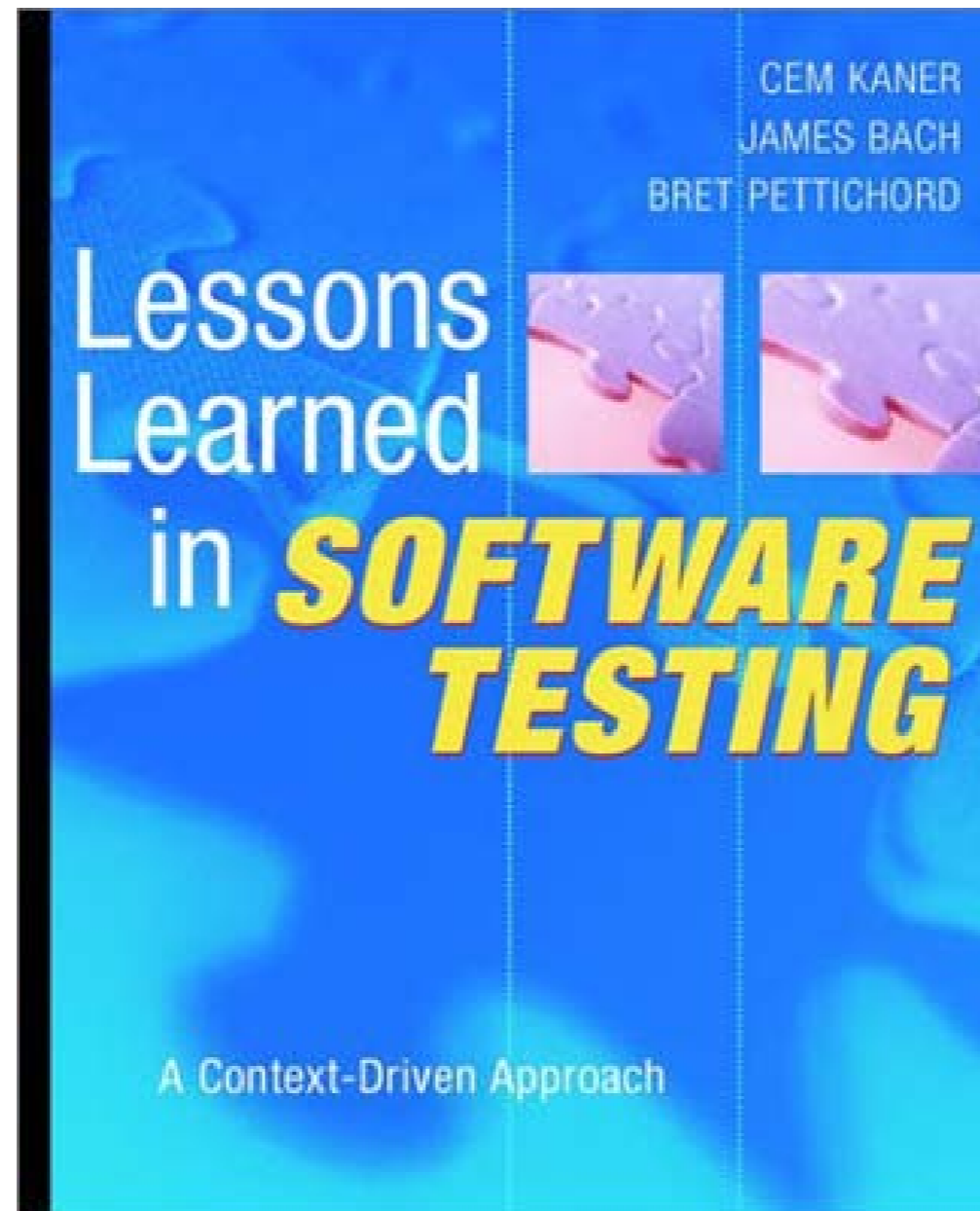
Use Se only for what it is good at.
Treat Se RC code with OO respect.

DRY framework; "wet" tests

Watch out for Se 2; it is a game-changer.



Get this book...



...and this book.

Q/A