

The Business Case for Automated Testing with Behat

Scott Falconer



PACIFIC NORTHWEST
DRUPAL SUMMIT

Scott Falconer

- Technical Architect - Acquia
- @ScottFalconer
- <http://drupal.org/user/52557>



PACIFIC NORTHWEST
DRUPAL SUMMIT

We'll cover:

- Why you should test
- How to test
- What to test

We'll also cover:

- Why you should *automate* your tests
- Tools you can use
 - Specifically Drupal + Behat

Why Test?

Because we want to release products that work.

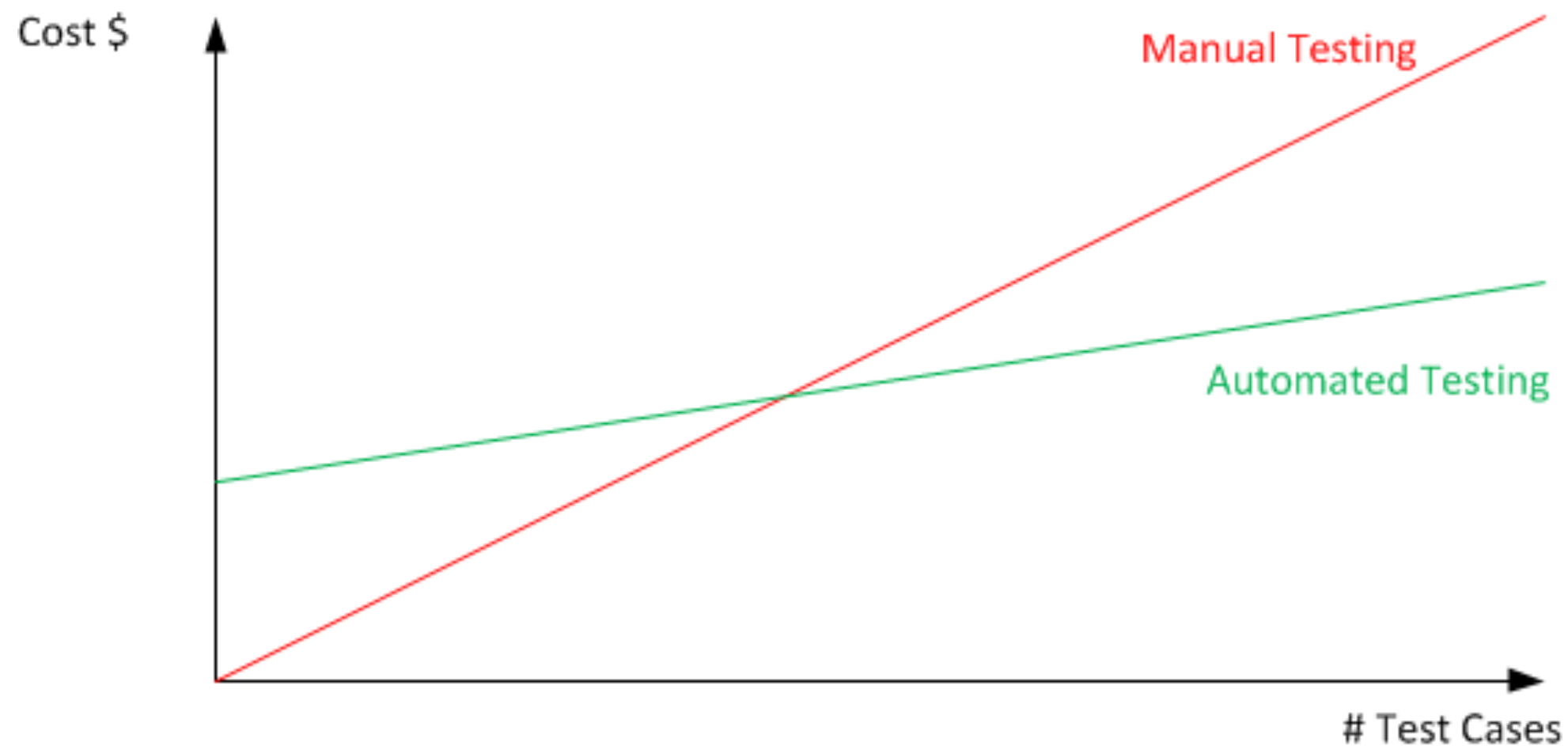
But

- Things break
- And when you fix it, something else breaks.
- And you don't know it's broken.



Sounds great, but...

Upfront investment



Developers may push back

- This project is very simple.
- Testing is too complicated.
- I need to make a small change and I'm in a hurry.
 - = VS ==

It forces discipline

Levels of testing

- Unit testing
 - Integration testing
- Acceptance testing
 - Functional testing
 - Regression testing

Test the full experience





Unit test vs. Integration test



Behavior-driven development

Managing software development with both business interests and technical insight.

Client Request


“Update the payment module to remove the expired credit card block and process the transaction.”

- payment.module?
- I can't find the expired credit card block.

Payment Module

Edit Payment Method

Payment method

 Visa

Name on card

Scott Falconer

Expiration date

02

2009

Expired


Cancel

Save

Credit Card Block

Edit Payment Method

Payment method

 Visa

Name on card

Scott Falconer

Expiration date

02 2009

Expired

We're sorry, you can't update an expired credit card online.

Cancel

Process the transaction



Ubiquitous Language

Communicate and think in the domain specific language of the business.

Common Language

- **Payment Module:** A section of the dashboard where a user can manage payment methods.
- **Error message:** A message indicating an action is unavailable.
- **Transaction:** Updating a user's information.
- **Processed payment:** When we charge a user's credit card.

User Story

As a member with an expired credit card I want to be able to update my credit card info so future payments go through.

Acceptance Criteria

- **Given** I am logged in as a user with a member role
- **When** I visit “/payment”
- **Then** I should see the Payment Module

Given

Put the system in a known state

When

Describe the key action the user performs

Then

Observe outcome.

And and But

- **Given** I am logged in as a user with a member role
- **And** my saved credit card is expired
- **When** I enter a new valid credit card
- **And** I press “Save”
- **Then** I should see “Thank you.”
- **But** not “Your payment has been processed”

Why this format works

- Given I am logged in as a user with a member role
- When I visit “/payment”
- Then I should see the Payment Module

```
1 Scenario: Ensure that the payment form is present.  
2   Given I am logged in as a user with a member role  
3   When I visit "/payment"  
4   Then I should see "Update Payment Info"  
5   And I should see a "field_payment[0][credit_card]" field  
6
```

Code example

```
1 Scenario: Update an expired credit card.  
2   Given I am logged in as a user with a member role  
3   And my stored credit card is expired  
4   When I visit "/payment"  
5   And I enter "1111 1111 1111 1111" for "Credit Card"  
6   And I enter "2/17" for "Expiration Date"  
7   And I press "Save"  
8   Then I should see "You credit card info has been updated"  
9   And I should not see "Your payment has been processed"
```

And and But

- **Given** I am logged in as a user with a member role
- **And** my saved credit card is expired
- **When** I enter a new valid credit card
- **And** I press “Save”
- **Then** I should see “Thank you.”
- **But** not “Your payment has been processed”

It's magic

It's Behat (and Gherkin)

- Behat is a php framework for **autotesting your business expectations.**
- It helps you to build the right system in the first place by facilitating and enriching requirements communication.
- It's a testing framework **and** a communication tool

Gherkin

- Business Readable, Domain Specific Language

Behat / Gherkin

- Features
- Scenarios
- Steps

Features

A file conventionally consisting of a single feature containing a list of scenarios.

Scenarios

A list of steps.

Steps

Given, When, Then

```
1 Scenario: Ensure that the payment form is present.
```

```
2     Given I am logged in as a user with a member role
```

```
3     When I visit "/payment"
```

```
4     Then I should see "Update Payment Info"
```

```
5     And I should see a "field_payment[0][credit_card]" field
```

```
6
```

Step Definitions

Then I should see a "field_payment[0][credit_card]" field

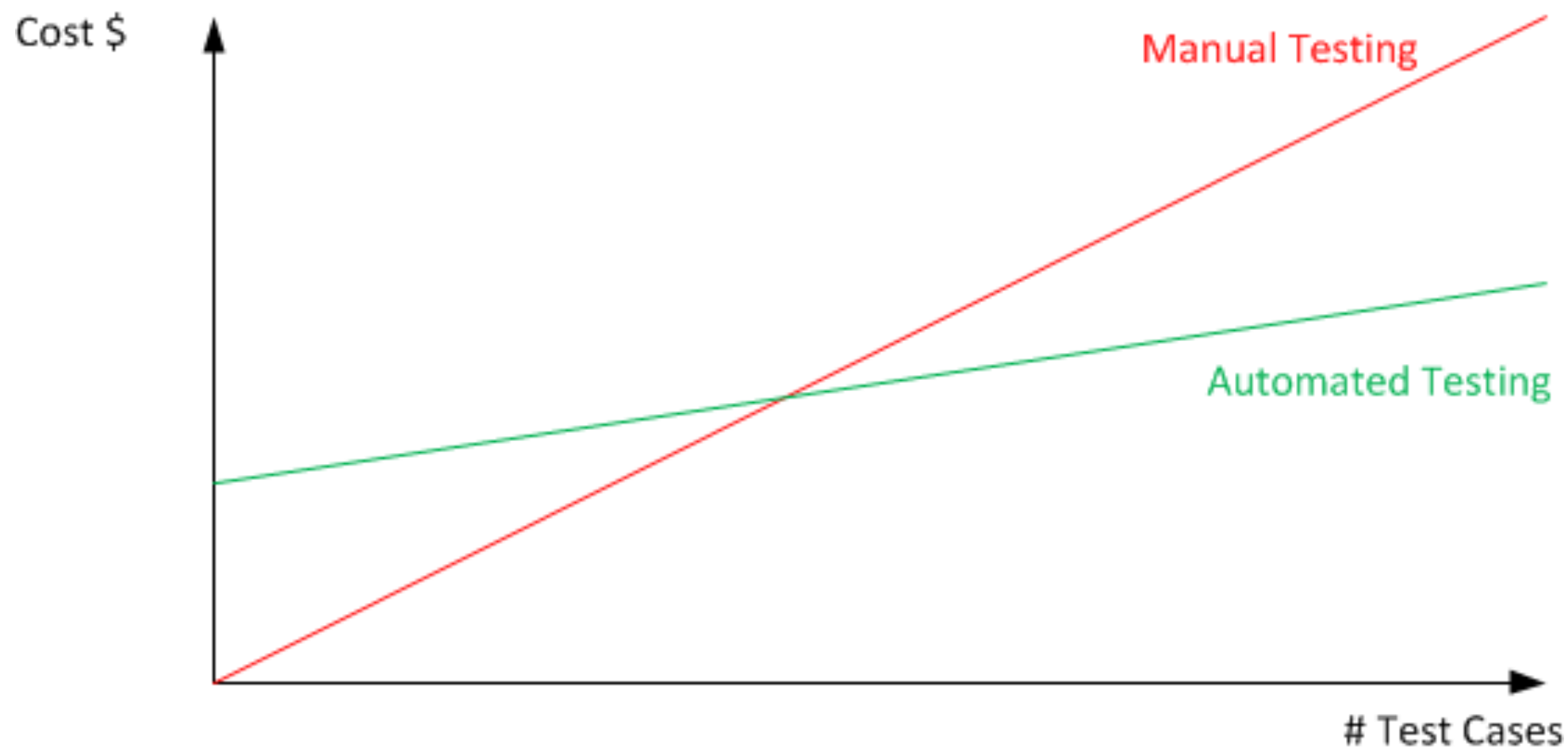
Step Definitions

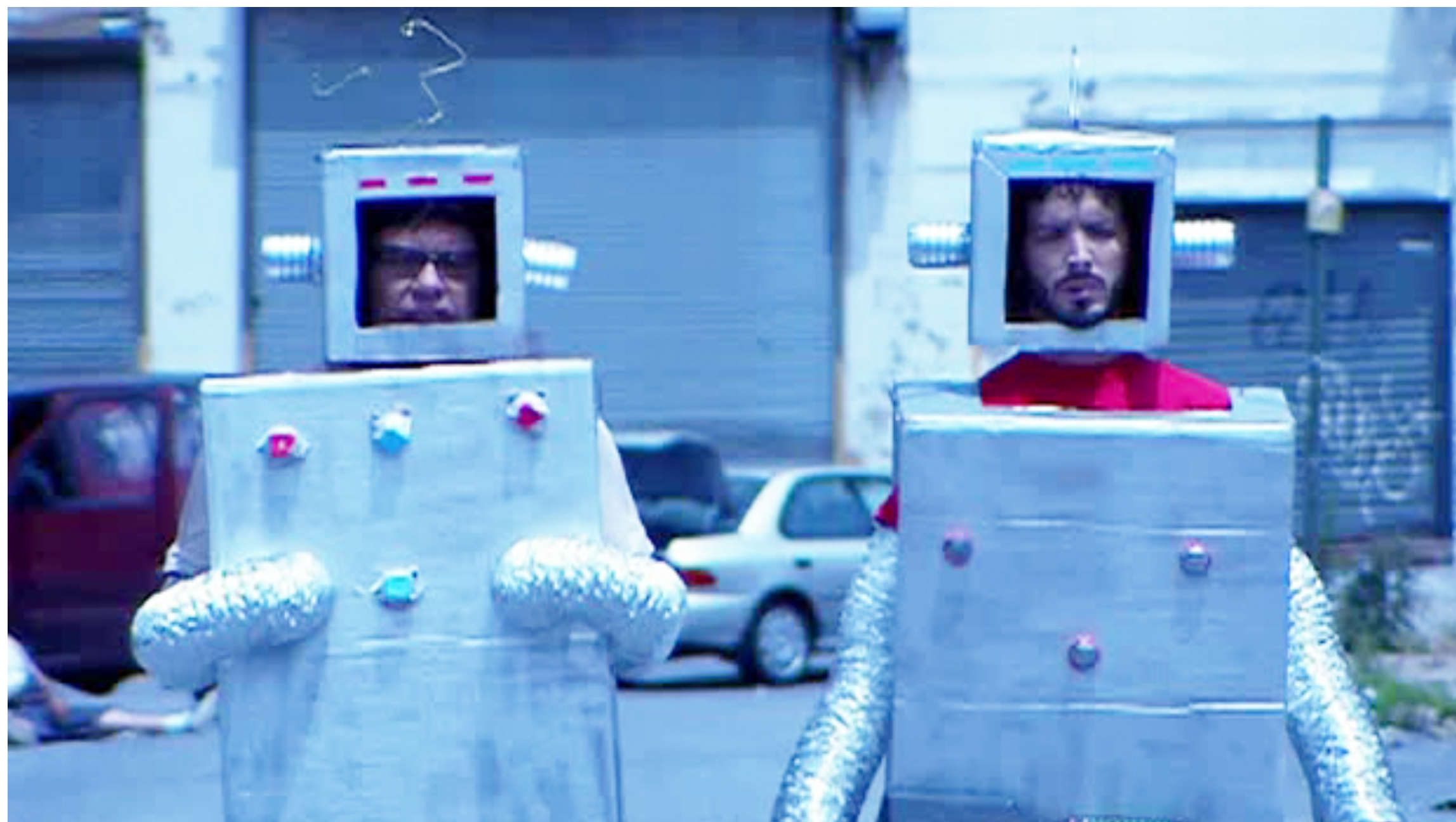
```
/**
 * Assets that a field exists.
 *
 * @param string $locator
 *     The field ID, name, label, or value.
 *
 * @Then I should see a :locator field
 */
public function assertFieldExists($locator) {
    $this->assertSession()->fieldExists($locator);
}
```


What to test

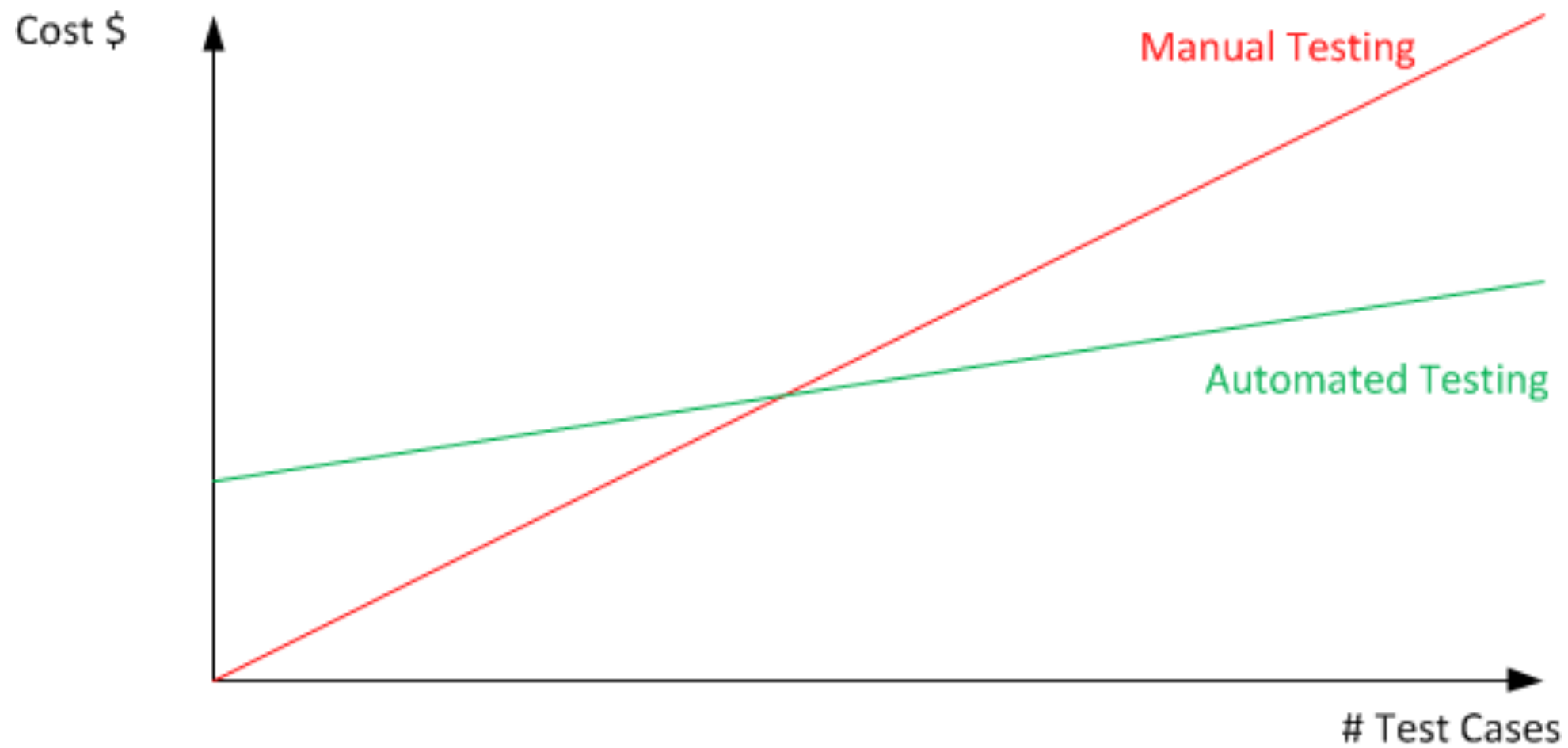
- Test what's important
- Tests have costs
 - \$\$\$ for environment to run the test
 - Time to execute
- Do you need the feature?

Why you should automate





Enforced testing



Automation side benefits

- Validate security releases
- Enforce commit messages
- Validate coding standards
- *Don't let developers use comic sans*

How this looks in practice

- PR -> Passing Test -> Good to merge.
- PR -> Everything broke -> Back to developer



Where it (hopefully) gets you

- Clearly designed code that does exactly what it's supposed to do.
- A regression test suite that can be automatically executed.
- Living documentation of your code that will evolve (or die) as your code does.
- Confidence that your product will work today and tomorrow.

Tools: behat.org

- Documentation
- Code examples
- User guides



PACIFIC NORTHWEST
DRUPAL SUMMIT

Tools: BLT

- Provides a standard project template for Drupal based projects
- Provides tools that automate much of the setup and maintenance work for projects
- Documents and enforces Drupal standards and best practices via default configuration, **automated testing**, and continuous integration

<https://github.com/acquia/blt>

Tools: Drupal Extension

- Integration layer between Behat and Drupal.
- Provides step definitions for common testing scenarios specific to Drupal sites.

<https://www.drupal.org/project/drupalextension>

Tools: Lightning

- Drupal distribution to enable developers to create great authoring experiences
- Hundreds of Drupal specific automated tests
 - Great for examples for writing your own.

<https://www.drupal.org/project/lightning>



PACIFIC NORTHWEST
DRUPAL SUMMIT