Automated Testing in the Fracture Registry

Austin Bingham

austin@sixty-north.com

22.06.2016

Table of Contents

What is automated testing?

…the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes. – Wikipedia

What is it used for?

- Ensure expected behavior
- Detect regressions
- Guide development efforts (e.g. TDD)
- Save time and money

How are we using it?

Two main approaches

- End-to-end testing of the registry
- Low-level/unit testing of "complex" details

Unit testing

Tests isolate and exercise details of the overall implementation.

Example: Test functionality of Angular date-time editor directive.

End-to-end (E2E) testing

Tests execute the site just like a user and verify that it responds as expected.

Example: Test that users have to assign Gustilo-Anderson rating to all fractures.

Use-case driven tests (behavior-driven design)

E2E tests are intended to capture important stories or use-cases in the registry.

Written in "natural language" using Cucumber.

Pragmatic, not dogmatic!

Coverage is not (even close to) 100%.

Testing primarily supports development effort right now, and especially change management.

Tests are created when and if we feel they're worth the effort. This is by definition based on judgement.

Leveraged

We don't test the backend at all because we rely on MRS's tests.

Technical overview of fracture registry

- Research-oriented registry of bone fractures and treatments
- Based on MRS
- MVC
- Entity Framework
- Razor
- Angular
- Bootstrap

Unit testing of front-end components

- Jasmine: BDD-style JavaScript testing library
- Karma: JavaScript test runner

Jasmine

A behavior-driven development framework for JavaScript.

It provides functionality for:

- Structuring and organizing tests
- Defining fixtures
- Expressing assertions

Jasmine example: Top-level configuration

```
// Sources/Client.Web.UI.tests/unittests/DateTimeEditor-tests.js

describe('DateTimeEditor directive', function() {
    var $compile, $scope;

    beforeEach(module('DateTimeEditor'));

. . .
```

Jasmine example: Set up environment before each test

```
beforeEach(inject(
    ['$compile', '$rootScope', function($c, $rootScope) {
     $compile = $c;
     $scope = $rootScope.$new();
     $scope.date = new Date(1975, 0, 19, 17, 12, 34, 5);
     $scope.disabled = function() { return false; }

     var html = '<date-time-editor model="date" disabled="disabled()"></date-time-editor>'
     $scope.element = $compile(html)($scope);
     $scope.$digest();
    }]
));
```

Jasmine example: An individual test

```
it('sets the fields to empty on null dates', function() {
    var dateInput = $scope.element.find('[name=date-input]');
    var timeInput = $scope.element.find('[name=time-input]');

    $scope.date = null;
    $scope.$digest();
    expect(dateInput.val()).toEqual("");
    expect(timeInput.val()).toEqual("");
});
```

Karma

Created by AngularJS team to meet their testing needs.

You specify:

- test code
- code under test
- testing framework (e.g. Jasmine).

Executes tests in a browser.

Karma example: Configuration file

```
// Source/Client.Web.UI.tests/karma.conf.js
module.exports = function(config) {
    config.set({
        basePath: '..',
        frameworks: ['jasmine'],
        browsers: ['Chrome'],
        browserNoActivityTimeout: 1000000000,
        files: [
            // Core libraries
            'Client.Web.UI/Core/Scripts/Bootstrap/bootstrap.js',
            'Client.Web.UI/Scripts/angular.min.js',
            . . .

            // Testing infrastructure stuff
            'Client.Web.UI/Scripts/angular-mocks.js',

            // CODE UNDER TEST: date-time-editor
            'Client.Web.UI/LocalScripts/Directives/DateTimeEditor-directive.js',
```

Karma example: Executing karma

```
$ karma start karma.conf.js
. . .
15 06 2016 15:28:13.894:INFO [launcher]: Starting browser Chrome
15 06 2016 15:28:16.194:INFO [Chrome 50.0.2661 (Windows 8.1 0.0.0)]: Connected on socket /
Chrome 50.0.2661 (Windows 8.1 0.0.0): Executed 27 of 27 SUCCESS (0.399 secs / 0.358 secs)
```

This launches chrome and runs the tests therein.

It also monitors files for changes, re-executing tests as needed.

End-to-end testing

- Selenium WebDriver
- Protractor
- PhantomJS
- Cucumber

Selenium WebDriver

WebDriver allows you to automate browsers.

1. Start server
2. Server launches browser and listens for commands
3. Test runner sends commands to server

```
C:\Users\mrsdev> webdriver-manager start
```

Just leave this running all the time.

Protractor

An end-to-end- test framework for AngularJS apps.

Takes care of many of the details associated with Angular.

Protractor example: Selenium and browser configuration

```
exports.config = {
    seleniumAddress:
    (process.env.SELENIUM_URL || 'http://localhost:4444/wd/hub'),

    capabilities: {
        'browserName':
        (process.env.TEST_BROWSER_NAME || 'phantomjs'),
        'version':
        (process.env.TEST_BROWSER_VERSION || 'ANY'),
        'phantomjs.binary.path':
        (process.env.PHANTOMJS_BINARY_PATH || require('phantomjs-prebuilt').path)
    },

    . . .
```

Protractor example: Parameters and preparation configuration

```
allScriptsTimeout: 31000,
getPageTimeout: 30000,

// This can be controlled with the '--baseUrl' arguments to protractor as well.
baseUrl: 'https://mrsdev.helsemn.no/Frakturregister/',

onPrepare: function() {
    browser.manage().window().setSize(1600, 1000);
    // TODO: Should we do login/patient selection here?
},

rootElement: '[ng-app]',
```

Protractor example: Cucumber configuration

```
framework: 'custom',

frameworkPath: require.resolve('protractor-cucumber-framework'),

// Spec patterns are relative to this directory.
specs: [
    'tests/*.feature'
],

cucumberOpts: {
    require: ['tests/config.js',
              'tests/stepDefinitions/mrs_common_steps.js',
              'tests/stepDefinitions/consultation_form_steps.js',
              'tests/stepDefinitions/fracture_form_steps.js',
              'tests/stepDefinitions/incident_form_steps.js',
              'tests/stepDefinitions/procedure_form_steps.js'],
    // tags: ['@dev'],
    format: 'pretty'
```

Protractor example: Page objects

```
function EpifyseMetafyseDialog() {
    this.dialog = element(by.name('epifyse-metafyse-dialog'));
};

EpifyseMetafyseDialog.prototype = Object.create({}, {
    epifyseButton: { get: function() {
        return this.dialog.element(by.buttonText('Epifyse'));
    }},
    metafyseButton: { get: function() {
        return this.dialog.element(by.buttonText('Metafyse'));
    }}
});

module.exports = EpifyseMetafyseDialog;
```

Protractor example: Test steps

```
this.When('I select "$choice" for epifyse/metafyse', function (choice) {
    var dialog = new EpifyseMetafyseDialog();
    switch (choice.toLowerCase()) {
    case 'metafyse':
        return dialog.metafyseButton.click();
    case 'epifyse':
        return dialog.epifyseButton.click();
    default:
        throw 'epifyse-metafyse choice must be epifyse or metafyse';
    }
});
```

Protractor example: Assertions

```
this.Then("AO Code is \"$code\"", function (expected, next) {
    var page = new FracturePage();
    page.aoCode.get().then(function (ao) {
        expect(ao).to.equal(expected);
        next();
    });
});
```

PhantomJS

A headless, WebKit-based browser.

Good for running E2E tests in the background.

Also good for CI using headless machines.

Cucumber

"Natural language" test definitions using gherkin syntax.

Allow non-programmer domain experts to write tests. In principle...

Tests are tied via regular expressions to executable code.

Cucumber example: Feature files

```
Feature: Incident form rules
  The incident form should enforce certain rules and constraints
  so that it doesn't generate invalid incidents.

  Background:
    Given I create a new incident

  Scenario: Showing skadested V1 suboptions
    When I select skadested "V1"
    Then the V1 fremkomstmiddel options are displayed
    And the N fremkomstmiddel options are not displayed
```

Cucumber example: Mapping gherkin to javascript

```
this.When(/^I create a new incident$/, function(next) {
    var that = this;
    var page = new IncidentPage();
    page.create(this.patientGuid).then(function () {
        return page.formId;
    }).then(function(guid) {
        that.incidentFormGuid = guid;
        next();
    });
});
```

Cucumber example: Passing parameters to steps

```
this.When("I select skadested \"$code\"", function(code, next) {
    var page  = new IncidentPage();
    page.skadested.set(code).then(function() {
        next();
    });
});
```

Results

How much effort is involved?

The main effort was learning the testing ecosystem.

Writing tests can be time consuming.

It has been a small fraction of overall development effort.

Is it worth the effort?

YES!

These tests have paid for themselves several times over.

I can make large changes to the registry with confidence.

How about the use of cucumber?

This was a mixed result.

Domain experts are not involved in writing these tests as envisioned.

The extra level of abstraction does entail extra work and maintenance.

However, the constraints have led to well-structured, reusable tests.

On balance, I think the approach is effective. And it may prove even more useful in future maintenance.

Future work

- Integrate testing with continuous integration
- Find ways to speed up tests
- Polish and improve existing tests

References

Technology	
Karma	karma-runner.github.io
Jasmine	jasmine.github.io
Selenium	seleniumhq.org
Protractor	protractortest.org
PhantomJS	phantomjs.org
Cucumber	cucumber.io
Misc	
Unit testing for Angular	docs.angularjs.org/guide/unit-testing
The importance of testability	speakerdeck.com/abingham/the-primacy-of-testability
This presentation	github.com/abingham/fracture-registry-testing-presentation