The Risks We Take

Ruby in System Testing

Stephan Kämper

Better living through automation...
...because life is too short for manual testing.

Overview

- [ ] An Issue with large Projects & Context
- [ ] A solution: Automated System Tests
- [ ] A framework I am building to solve this: ATFW

Ruby? In large Companies?

Yes!

DANGER!

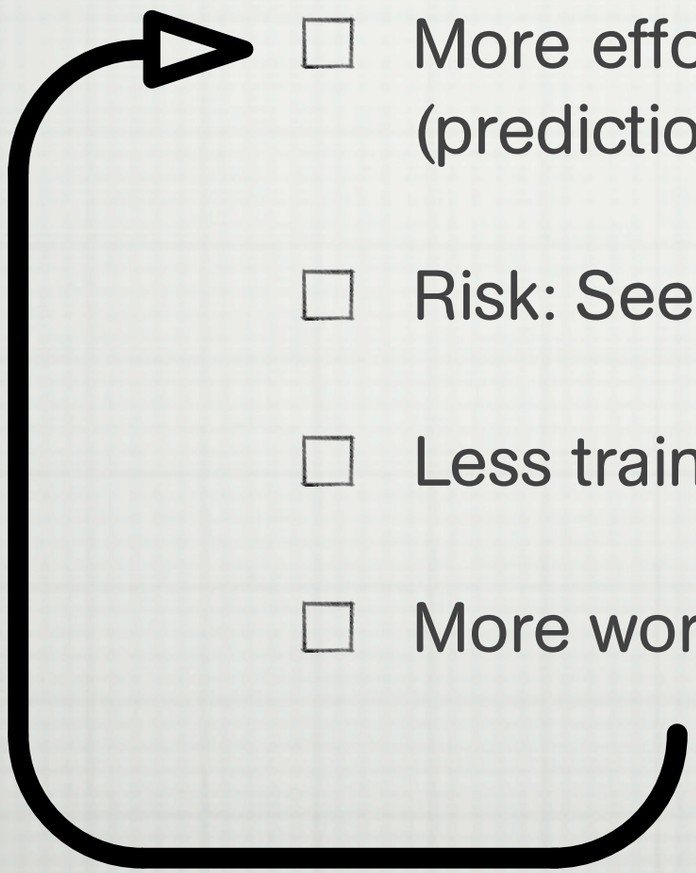
Credit: Jason Taellious
On Flickr: dreamsjung

The Problem...

- ☐ Risk or not? ➝ Context
  - ☐ What‘s the Environment?
  - ☐ Anything/Nothing left to lose?
  - ☐ Perception & Experience

How Things Develop

- Projects grow (size/duration) → More PROCESS



- More effort put in problem forecast
  (prediction of risks & mitigation)
- Risk: Seen everywhere, avoided at (nearly) all cost
- Less training of actually failing
- More worries to fail in the future

The Effect...

FRUSTRATION

STAGNATION

DEMOTIVATION

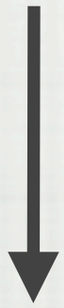
A Word about PROCESS

- Software development is **not** industrial production
  - Building bridges (nowadays)
  - Comparable to the automobile industry

- It is more like manufacturing and craftsmanship
  ➙ Philip‘s presentation today

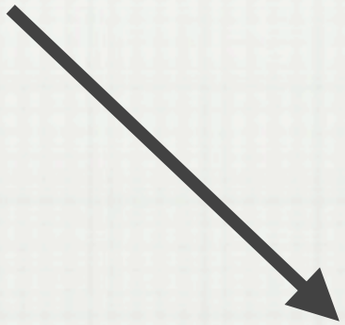
A Solution: FAIL

FAIL



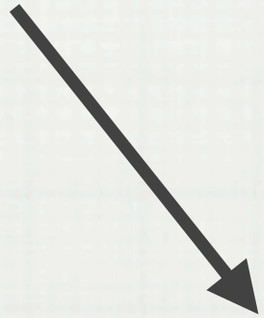
Get used to failure – think test 1st development

OFTEN



Only automated tests can run often enough

IN A SECURE ENVIRONMENT



Have a test environment very similar to production

Failing Is Good For You

- ☐ Test environment **very** similar to production
  Same hardware, software versions, configuration
  Hint: Automated deployment

- ☐ Train regularly ➙ Have the tests always ready to run

- ☐ In a secure environment ➙ Won‘t hurt yourself (or others) badly

- ☐ Think TDD/BDD/Test first … on a system test level

Context Matters

- ☐ Project:
  - ☐ What‘s at stake (convenience, money, health)?
    How much of it is at risk?
- ☐ Development
  - ☐ What‘s recently created is easier, faster, less expensive to fix
- ☐ Automation ➙ Have a fast way to test your systems

My Way To Deal With System Tests: An Automated Test Framework, written in Ruby

I apologise for the project name ‚aatfrawir‘, let‘s call it ATFW for the moment.

ATFW

- ☐ Enable reproducible test execution
- ☐ Shortly after installing/deploying new versions
- ☐ Write test cases in Ruby
- ☐ Execute against (essentially) any interface (you'll implement)
- ☐ Provide simple summary/report of results
- ☐ Keep log of previous results to track history

By All Means

- [ ] Log all the details about...
  - [ ] Configuration of the test environment
  - [ ] Date & Time of execution
  - [ ] In case the version is available from the tested system: LOG IT
  - [ ] What you attempt to do
  - [ ] What the result is
  - [ ] And of course if that‘s what you expected

How to Write Test Cases

- ☐ Focus on business & system level (not unit tests, technical details)
  - ☐ Adress the important functionality
  - ☐ Reduce risk of not testing functionality
- ☐ Easy to understand for
  - ☐ Developers
  - ☐ Test Automators
  - ☐ Business people

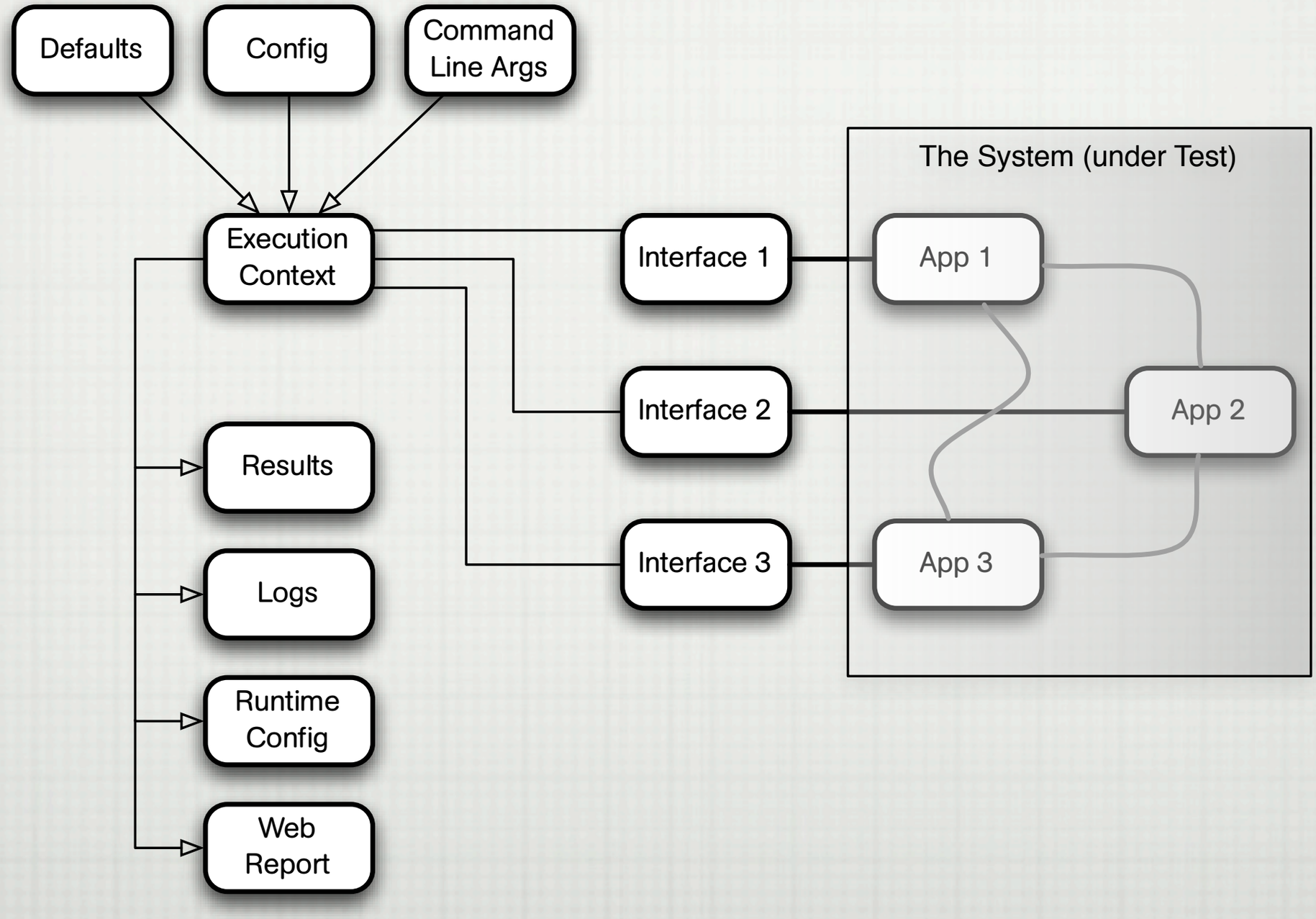
An Example Test Case

```
test_cycle 'create_and_change_account' do
  acc_name = 'such_and_so'
  initial_amt = 345.43

  test_case 'create_new_accout' do
    interface 'nabob'
    create_new_accout :name => 'foobar', :password => 'secret'
    assert_find_on_page :name => /Account Name/, :content => 'foobar'
  end

  test_case 'change_account' do
    interface 'nabob'
    login :system => config.system, :user => 'faa', :password => 'bar'
    find_account :name => acc_name
    assert_no_error
    assert_find_on_page :name => /Account Name/, :content => acc_name
    assert_find_on_page :name => /Amount/, :content => initial_amt
    log_off
  end
end
```

Design – Not Rocket Science



ATFW.state?

- (Currently) hosted at www.gitcentral.com as aatfrawir
  Open Source & free (as in free beer)
- Not yet ready for public consumption
- Will get a liberal license…
  - …but I need to find out about which one to use.

Right now it doesn‘t completely

Need to apply some further tweaking

My Questions

- [ ] Someone available…
  - [ ] ➙ to discuss OSS licensing later today?
  - [ ] ➙ to find way to encapsulate execution?

A Context Problem

```
test_cycle 'reload_interfaces' do
   test_case 'use_interface_1' do
      interface 'one'
      # Do something with interface one
   end                  # Automatic clean up here
   test_case 'use_infertace_2' do
      interface 'two'
      # Two something else here
   end                # Automatic clean up for interface 2
   test_case 'interface_1_again' do
      interface 'one'                  # Not loaded again
   end
   # Clean up code for interface two called
end
```

The Short Story

- Context matters
  - For the project:
    - Environment – Value@Stake? – Experience & Training
    - Context of project tasks of team members
  - For test execution
    - Everything that may influence test result
      (including execution date and time – you wouldn‘t believe)

Learn to fail, learn how to fail, train to fail- and you less likely will.

Apply some
& the perspective could be...

...quite good, indeed

Thanks & Questions

- Contact me
  - SK@stephankaemper.de or phvalue@mac.com
  - www.stephankaemper.de
  - phvalues.wordpress.com

Better living through automation...
...because life is too short for manual testing.