

Software Quality and Test Strategies for Ruby and Rails Applications

- Bhavin Javia



May 29th, 2011

About Me

Ex ThoughtWorker

Agile/RoR Consultant - Ennova, Australia

Founder - Ennova, India

bhavinjava 

@bhavinjava 

bhavin@ennova.in 

Co-Author



Dr Adrian Smith

Managing Director - Technology

Agile Coach

@

ENNOVA
Engineering Innovation.

Content

Software Quality

Test Strategy

Ruby/Rails Testing Ecosystem

Quality and Testing for Rails Projects

Software Quality

Importance of Quality

*“Quality is the most important factor for deriving
long term value from a software product”*

- Jim Highsmith (Co-Author of Agile Manifesto)

*”Continuous attention to quality can help
maintain velocity, maintain the ability to deliver,
without it the team will slow down”*

- Robert C. Martin (Author of Clean Code)

Why worry ?

Needs to be built in (implicitly)

Needs to be defined (to prioritize vs cost/scope/time)

Agile project delivery relies on continuous attention to quality

Begins at the code level

```
code_quality != "App Quality"
```

Internal Quality

vs

External Quality

External Quality

(Quality as perceived by users)

interface design

performance

defects

user experience

Internal Quality

(Quality of the implementation and system architecture)

duplication

complexity

coupling

test coverage



“Technical Debt may be costing you more than you imagined!”

- Jim Highsmith

The trade-off

External Quality

May be traded
against project
objectives

Internal Quality

Should NOT be traded
against project
objectives

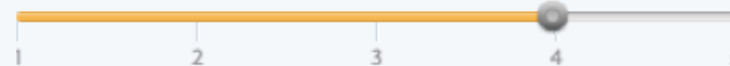
PROJECT SUCCESS CALCULATOR



Deliver all planned scope



Meet all External Quality requirements



Meet all Internal Quality requirements



Edit Slider Labels

PROJECT SUCCESS CALCULATOR



Deliver all planned scope



Meet all External Quality requirements



Meet all Internal Quality requirements



Edit Slider Labels

Managing Quality

Managing Quality

Set Goals

"Max 3 clicks to any page"

Measure

"Page X takes 5 clicks"

Prioritize

"Make page X reachable in ≤ 3 clicks"

How to achieve Quality ?

TEST TEST TEST

What/When/How to Test ?

" Test Strategy "

Test Strategy Template

Test Strategy Template

Not a Waterfall style document

A template to structure the discussion

Prompts for things to consider

Flexible format

Purpose

Create a shared understanding of

approach

tools

targets

timing

of test activities

Guiding Principles

Shared Responsibility

Everyone is Responsible for Testing and Quality

Test Automation

All types of tests should be automated (except exploratory tests)

Data Management

Production data must be obfuscated before use

Test Management

Tests, documents and data treated as production code

Quality and Test Objectives

Correctness

Integrity

Maintainability

Availability

Interoperability

Performance

Quality and Test Objectives

Correctness

Features and functions work as intended

e.g. 0 Critical Defects

Integrity

Prevent unauthorized access or information loss

e.g. All access via HTTPS, Encrypted passwords

Maintainability

Easy to add new features or fix defects

e.g. Code Complexity < 8, Unit Test Coverage > 80%

...

Quality and Test Objectives

Availability

Planned uptime percentage

e.g. 99.99 % availability

Interoperability

Ease of information exchange with other systems

e.g. Published & versioned API, Supports - IE 8, FF 3.5, Chrome 11 etc

Performance

Responsiveness & Scalability of the system

e.g. Apdex Score > 0.9, Response Time < 200ms

Quality and Test Objectives

Target
Measure
Prioritize

Test Scope

Systems and features in and out of scope

Test both business processes and the technical solution

Specify regions and sub-regions included in testing

Identify interfaces with other systems

System / feature / integration / region / environment / dependency / service

In Scope

Things to be tested with automation

Things to be tested manually

Out of Scope

Things we can't test e.g. Third Party systems

Test Approach

Identify the test types

Identify the tools

Include the timing of execution

Test Types

Unit

Functional

Integration

Acceptance

Performance

Data Conversion

Test Approach

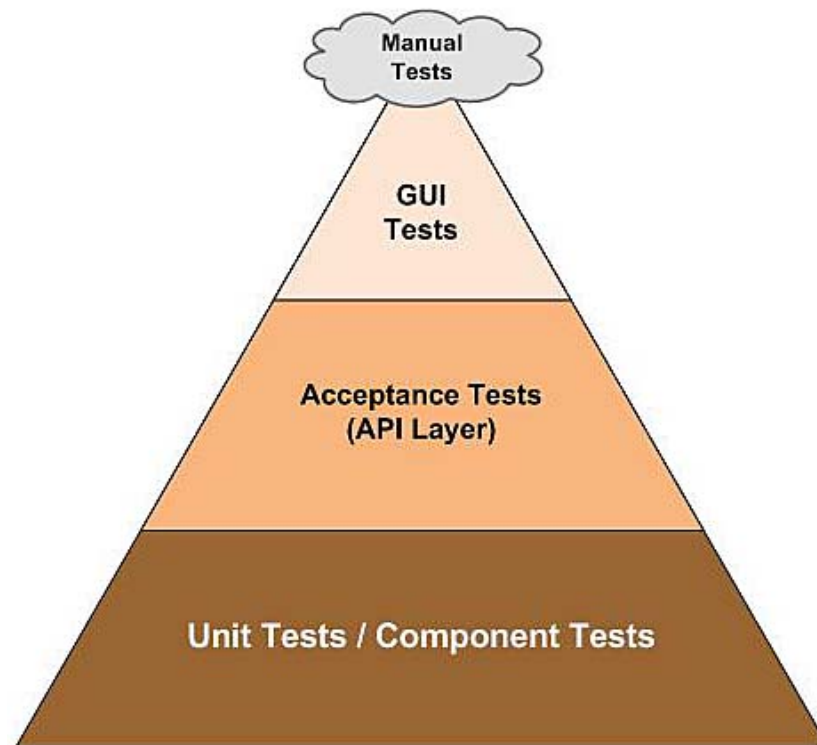
Identify Tools

e.g. rspec, cucumber etc

Decide Timing of Execution

e.g. dev machine, CI server, after each commit etc

Test Automation Pyramid



Test Preparation

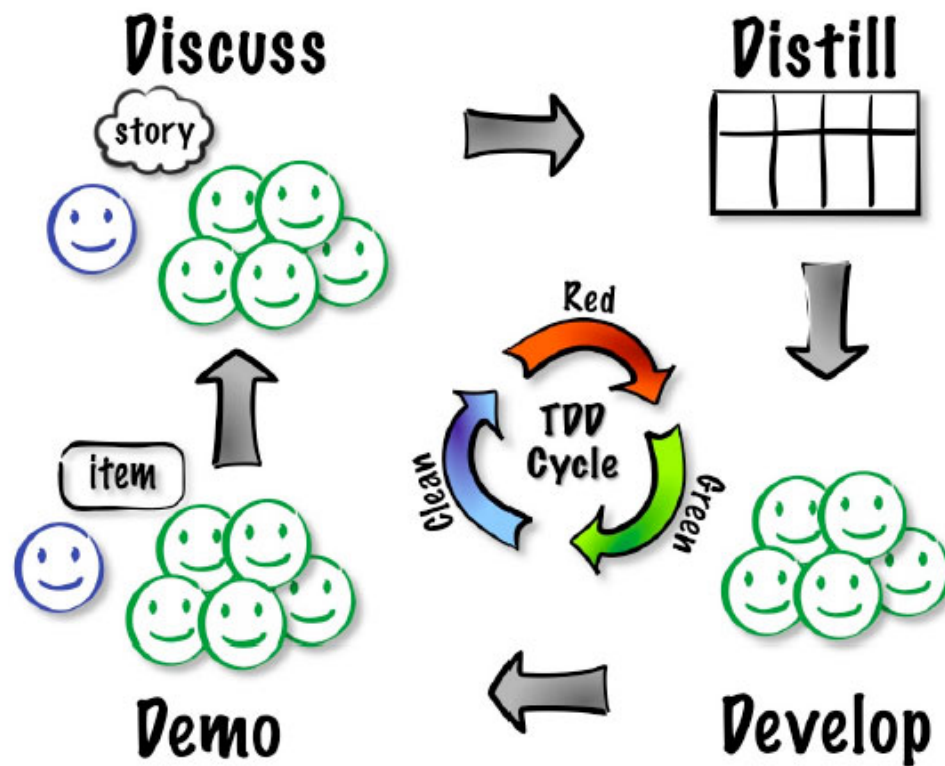
Use requirements captured in user stories

Include acceptance criteria in user story

Decide approach to prepare and execute tests

Example

Acceptance Test Driven Development (ATDD) Cycle



Acceptance Test

Feature: Create user accounts

In order to use the system

As a user

I want to login and manage my profile

Background:

Given I am the registered user "Malcolm Reynolds"

Scenario: Sign in with username/password

Given I am on the login page

When I fill in "Username" with "malcolm.reynolds"

And I fill in "Password" with "serenity"

And I press "Sign In"

Then I should be signed in as "Malcolm Reynolds"

Environments

Development

Unit, Functional, Acceptance tests

Integration

Continuous Integration - Unit, Functional, Acceptance tests

Code Analysis/Metrics

Staging

For exploratory testing

Demos to customers

Production

Smoke/Sanity tests & Monitoring

Test Execution

steps in preparation for deployment/release

Build the system

Populate test/reference data

Execute automated tests

Generate test report/code metrics

Test Data Management

System and user acceptance tests

Use subset of production data

Performance/volume/stress test

Use full size production files

Generate large volumes of data

begin

database.**load!** "Sensitive Production Data"

ensure

database.obfuscate! "Sensitive Data"

end

Defect Management

Why defects ?

```
:missing_tests => "Defects"
```

Defect Management

Defects only raised when not fixed immediately

Capture conditions and severity

Capture steps to reproduce

Critical

Major

Minor

Trivial

Defect LifeCycle

Identify

Prioritize

Assign Severity

Analyze

Write test to reproduce

Resolve

Fix code to pass the test

Verify

Run all tests, verify manually

Close

Test Strategy Example

Ennova Test Strategy

listthings.com

listthings

My notes

Ennova Test Strategy

Retrospective 1-10-10

Retro 28-2-11

bhavin.java@ennova.com.au

Log out

Principles :

Everyone must test
Automate as much as possible
Always obfuscate data
Treat Tests as Prod Code

Integrity:

Secure public instances with HTTPS

User passwords are encrypted

Passwords not transmitted in plain text over the internet

Instance/infrastructure settings not exposed via the application

Quality Objectives :

Correctness:

100% completion of planned features
Critical defects = 0
High defects = 0
Medium defects < 5
Low defects < 10

Interoperability:

Supported browsers:
IE version >= 8.0
Firefox version >= 3.5
Safari version >= 5.0
Chrome version >= 11.0

Versioned API for iPhone app

Maintainability:

Common logic extracted out into reusable modules

Generic utilities extracted out into gems

Unit, Functional, Integration tests cover 80 % of the application features (not necessarily code)

Performance:

Apdex Score (T 0.5) > 0.9
Response Time < 200ms
Throughput > 100 rpm

(measured with NewRelic)

Availability:

99.99% up-time
(measured through NewRelic)

Test Scope:

In scope (Manual) :

Google Sketchup
iPhone app

In scope (Automated) :

All unit testable code
All browser based features
All API features
All custom gems

Out of scope:

Third Party Systems
Kanbanery, Aconex etc

Test Types & Tools:

Data Conversion:

Import/Export via CSV
Features + Manual testing

Performance:

Monitoring: NewRelic

Benchmarking: TBD ??

Unit :
Model specs (rs)

Functional:
Controller specs

No view specs

Integration:
Features (Cucumber)

Acceptance:
Green build + M

Principles and Objectives

Principles :

Everyone must test
Automate as much as possible
Always obfuscate data
Treat Tests as Prod Code

Integrity:

Secure public instances with HTTPS

User passwords are encrypted

Passwords not transmitted in plain text over the internet

Instance/infrastructure settings not exposed via the application

Quality Objectives :

Correctness:

100% completion of planned features
Critical defects = 0
High defects = 0
Medium defects < 5
Low defects < 10

Interoperability:

Supported browsers:
IE version ≥ 8.0
Firefox version ≥ 3.5
Safari version ≥ 5.0
Chrome version ≥ 11.0

Versioned API for iPhone app

Maintainability:

Common logic extracted out into reusable modules

Generic utilities extracted out into gems

Unit, Functional, Integration tests cover 80 % of the application features (not necessarily code)

Performance:

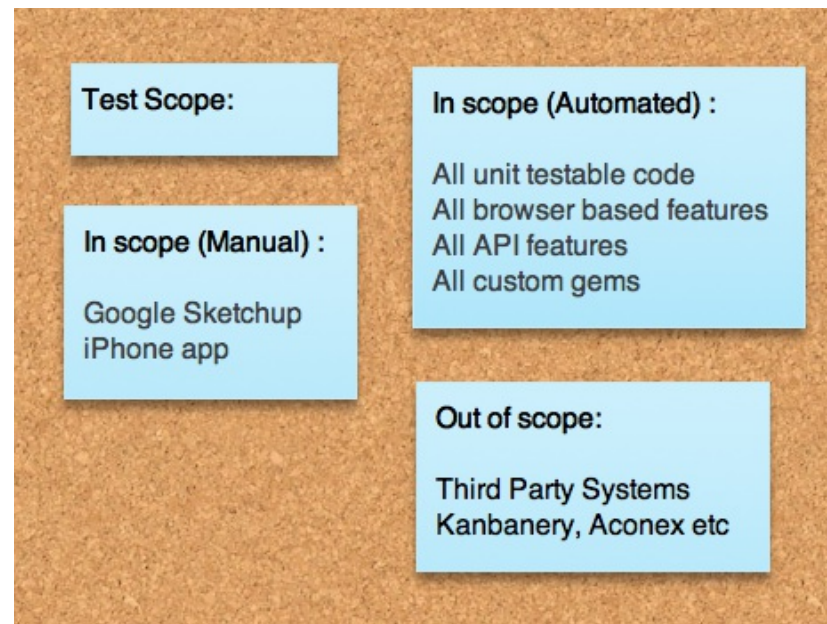
Apdex Score (T 0.5) > 0.9
Response Time $< 200\text{ms}$
Throughput $> 100 \text{ rpm}$

(measured with NewRelic)

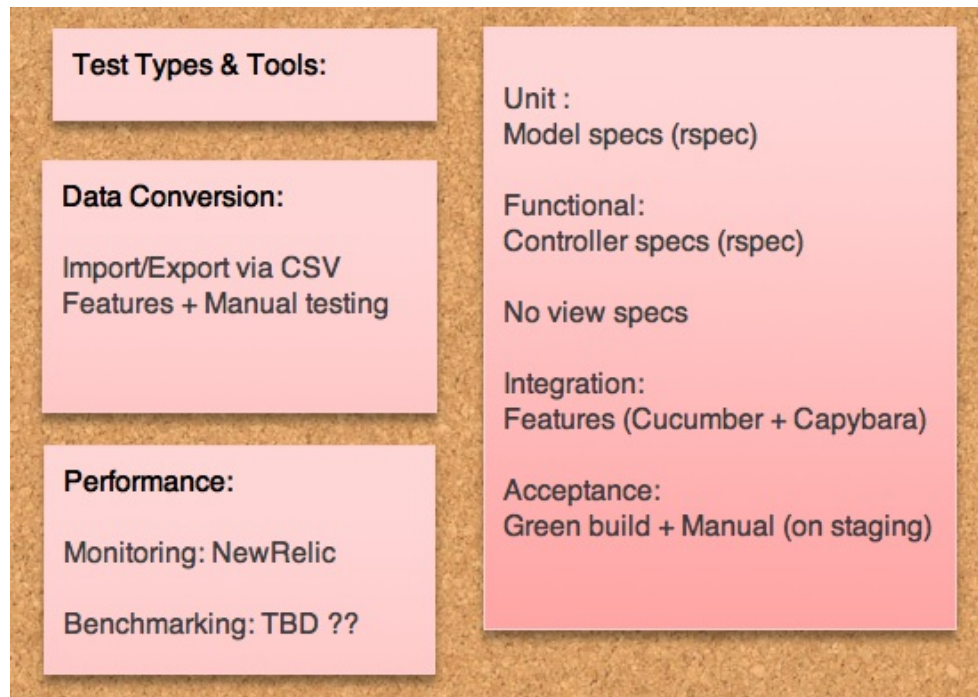
Availability:

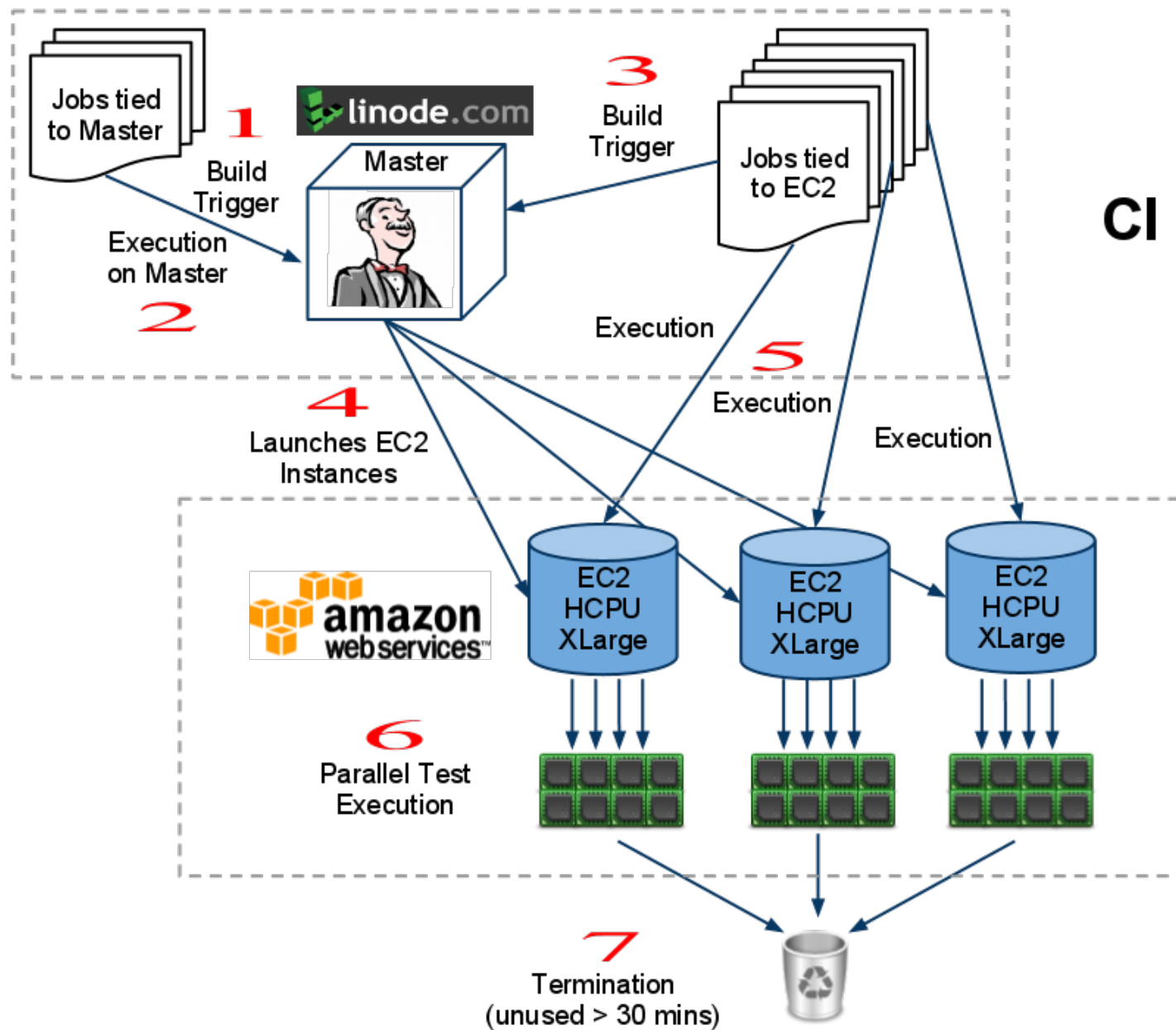
99.99% up-time
(measured through NewRelic)

Test Scope



Test Approach





Ruby/Rails Testing Ecosystem

What's different
in
Ruby/Rails ?



Advantages

Testing built right-in

Passionate developers

Vibrant open source community

Excellent tools support

So what are our options ?

The Ruby Toolbox

Latest News

Categories



Know your options!

Ruby developers can choose from a variety of tools

The Ruby Toolbox gives you an overview of these tools, sorted by the amount of watchers and forks in the repository on GitHub so you can find out easily which are the most common ones in the Ruby community.

ActiveRecord Default Values

`default_value_for` and `active_record_defaults`

ActiveRecord Encryption

`attr_encrypted`, `strongbox`, `sentry`, `Lockbox`, and more

ActiveRecord Enumerations

`enumerated_attribute`, `enumerate_it`, `enumerate_by`, and more

ActiveRecord Index Assist

`rails_indexes` and `ambitious_query_indexer`

There are ...

23 Testing Frameworks

8 Javascript Testing Frameworks

8 Browser Testing Frameworks

8 Distributed Testing Frameworks

6 Object Mocking Frameworks

7 Web Mocking Frameworks

2 Continuous Testing Frameworks

7 Continuous Integration Frameworks

12 Code Metrics Frameworks

and many more ...

```
raise "OMG !!"
```

What to Test ?

Anything that could possibly break

Public interface of each layer

Whole application stack

Depends on what else you're testing

What NOT to test ?

Cost of test $> 10 * \text{Cost of breakage (very rare)}$

Internal implementation

Framework code

You're doing it wrong if

Fixing tests all the time

Tests that always fail together

Tests that never fail

What to Test ?

Models

Controllers

Views

Helpers

Libraries

APIs

Testing Options

Models

Test::Unit (Ruby 1.8, built-in)

MiniTest (Ruby 1.9, built-in, replaces Test::Unit)

Unit Tests (ActiveSupport::TestCase)

Model Specs (rspec)

Controllers

Functional Tests (built-in)

Controller Specs (rspec)

Testing Options

Views

`assert_select` (built-in)

View Specs (rspec)

Helpers

`ActionView::TestCase`

Helper Specs (rspec)

Testing Options

Routes

assert_routing (built-in)

Routing specs (rspec)

Mailers

Unit Tests (built-in)

Functional Tests (built-in)

Testing Options

Workflow

Integration Tests (built-in)

Request Specs (rspec)

BDD

Cucumber + Capybara

Cucumber + Rspec

Testing Options

Performance

Performance Tests (built-in)

NewRelic

Testing Helpers

Data Setup

Fixtures (built-in)

factory_girl

Machinist

Testing Helpers

Mocks/Stubs

Rspec mocks

Mocha

Flexmock

API / Web Services

VCR

Fakeweb

Webmock

Testing Helpers

Speed

NuIDB

Autotest/Infinity Test/Watchr

Spork

parallel_tests

TLB

CI

cijoe

cruisecontrol.rb/goldberg

integrity

Quality/Metrics

Test Coverage

rcov

SimpleCov

Code Analysis

metric_fu

Metrical

Rails Best Practices

Which is the Best ?

Even the God(s) Can't Say !!

Which is the Best for Me ?

IMHO

```
WHATS_BEST = {  
  :newbie => "Start with built-ins",  
  :intermediate => "Checkout alternatives",  
  :expert => "Venture into the obscure"  
}
```

```
WHATS_BEST[:god] = "Write your own !"
```


References

<http://ennova.com.au/blog>

<http://www.jimhighsmith.com>

<http://www.mountangoatsoftware.com/tools>

<http://www.informit.com/articles>

<http://www.agileacademy.com.au>

<http://railsrx.com>

<http://www.bootspring.com/blog>

Questions ?

<http://sqats-rubyconfindia2011.heroku.com>

bhavinjava 

@bhavinjava 

bhavin@ennova.in 