Testing with easyb

Venkat Subramaniam
venkats@agiledeveloper.com
@venkat_s

Testing with easyb

Agile Development

Sustainability

Circle of Expectations and Circle of Relevance

Types of Tests

Behavior Driven Development

Functional Testing

Where does it fit in?

what's easyb?

Stories vs. specifications

Stories

Writing stories as executable documentation

What's Agile Development?

The essence of Agile Development...

What's Agile Development?

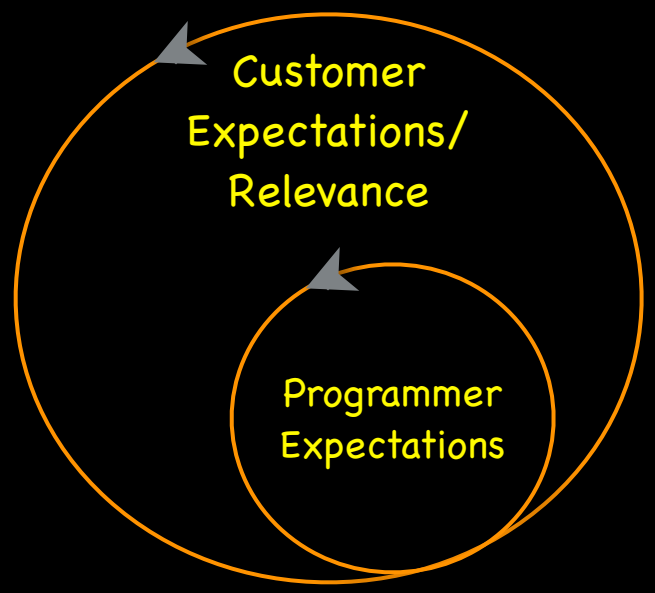
Feedback Driven Development

Sustainability

Rapid feedback is good, but the pace has to be sustainable

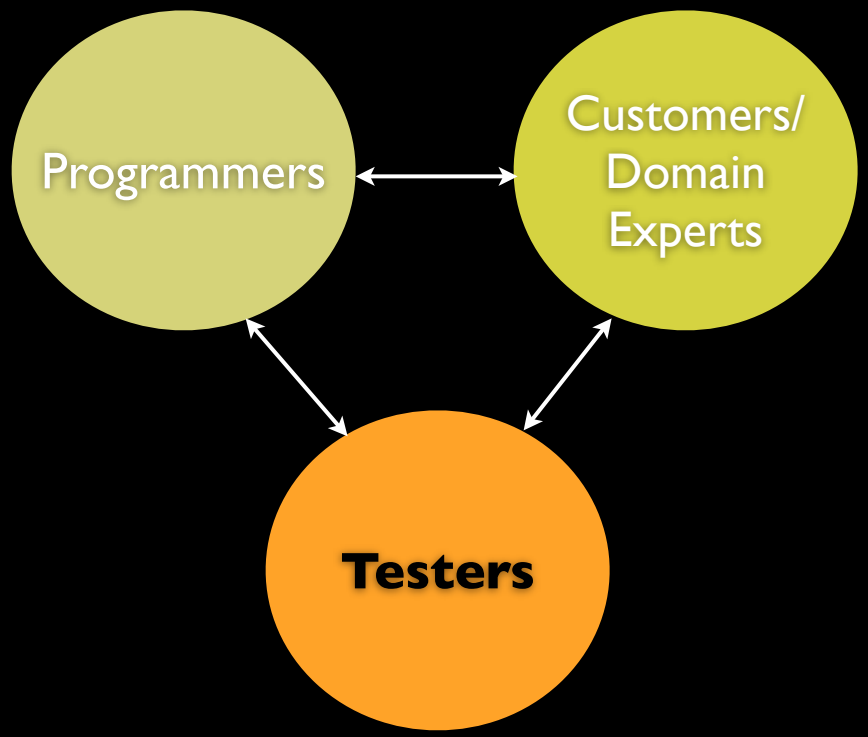
No point running fast in the wrong direction

Circles of Feedback

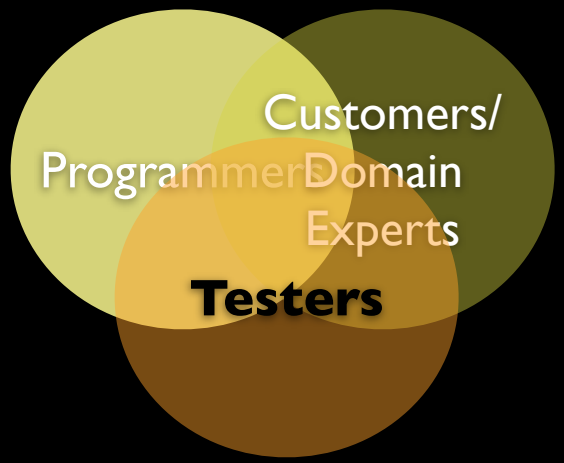


Team

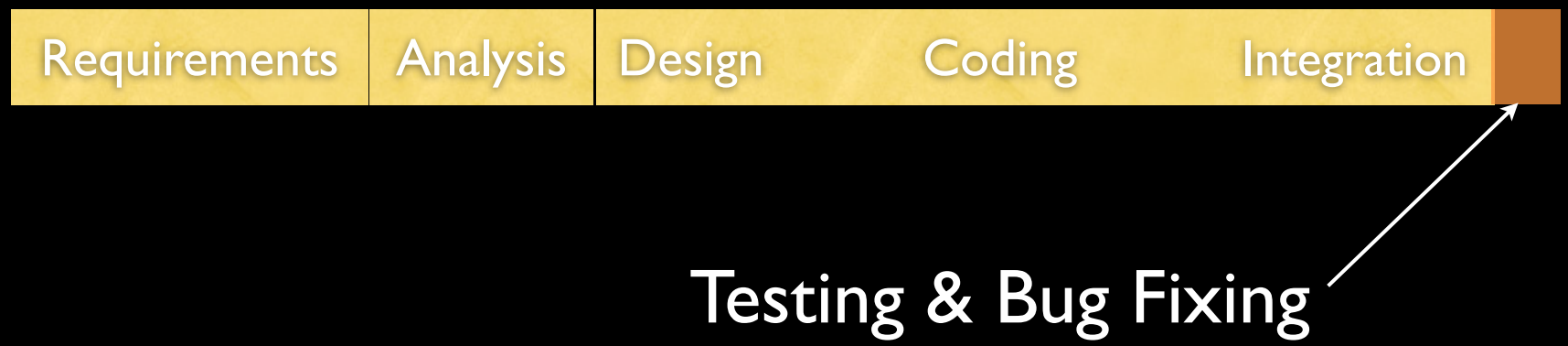
Traditional



Agile

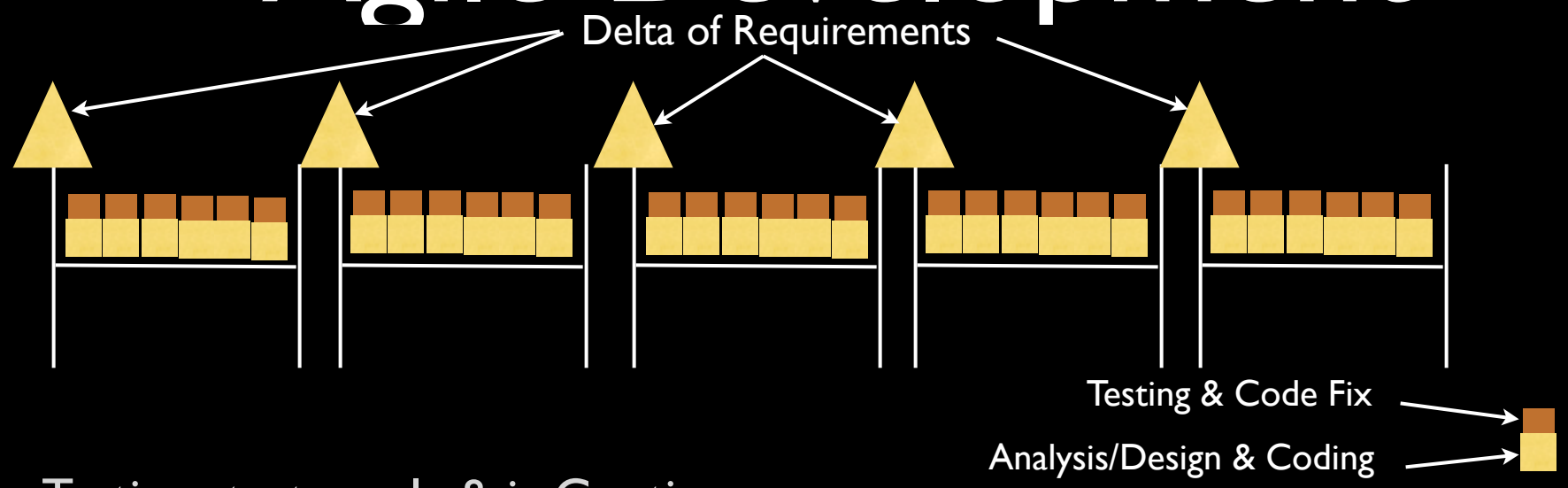


Traditional Testing



- Too late in the game
- Often pressure to release
- QA become defenders
- Often looked at as adversaries

Agile Development



- Testing starts early & is Continuous
  - Don't wait until end of iteration to test–test frequently and regularly
- Application is exercised constantly, no surprises later
- QA become support
- Not adversaries, become part of the team
  - Work with customer and programmers—co-located with them

Tenet Of Testing

As a tester, your responsibility is to author tests, not to run them!

Why Automate Tests?

"Error rate in manual testing is comparable to the bug rate in the code being tested."—Boriz Beizer.

Types of Tests

White-box testing

Black-box testing

Unit testing

Functional testing

Behavior Driven Design

It is a TDD approach

It is a ubiquitous language

It is an executable documentation

It promotes communication

Helps develop common vocabulary and metaphor

Help you to get the "words" right

Can be used by programmers, testers, business analysts, domain experts, and customers.

Behavior and Story

You can use BDD to express Stories and Behaviors

Story Framework and Spec Framework

Stories correspond to User Stories—to express behavior at application level

Spec or Behavior correspond to expectations at class level—to express behavior at service/component level

These can help express requirements that can be specified, understood, and negotiated by developers, testers, business analysts, and business customers.

Behavior

Each behavior is expressed as a test/*exercise* method

It tells what the object *should* do

Notice the keyword "should"—that's a main focus in BDD—the *shoulds* and the *shouldn't*

Building Stories

You may define user stories as a series of acceptance criteria as scenarios

It has the givens, events, and outcomes

That is

Given some initial condition(s),

When event(s) occurs,

Then ensure some outcome(s)

Functional Testing

Focused on what the application should do for the user

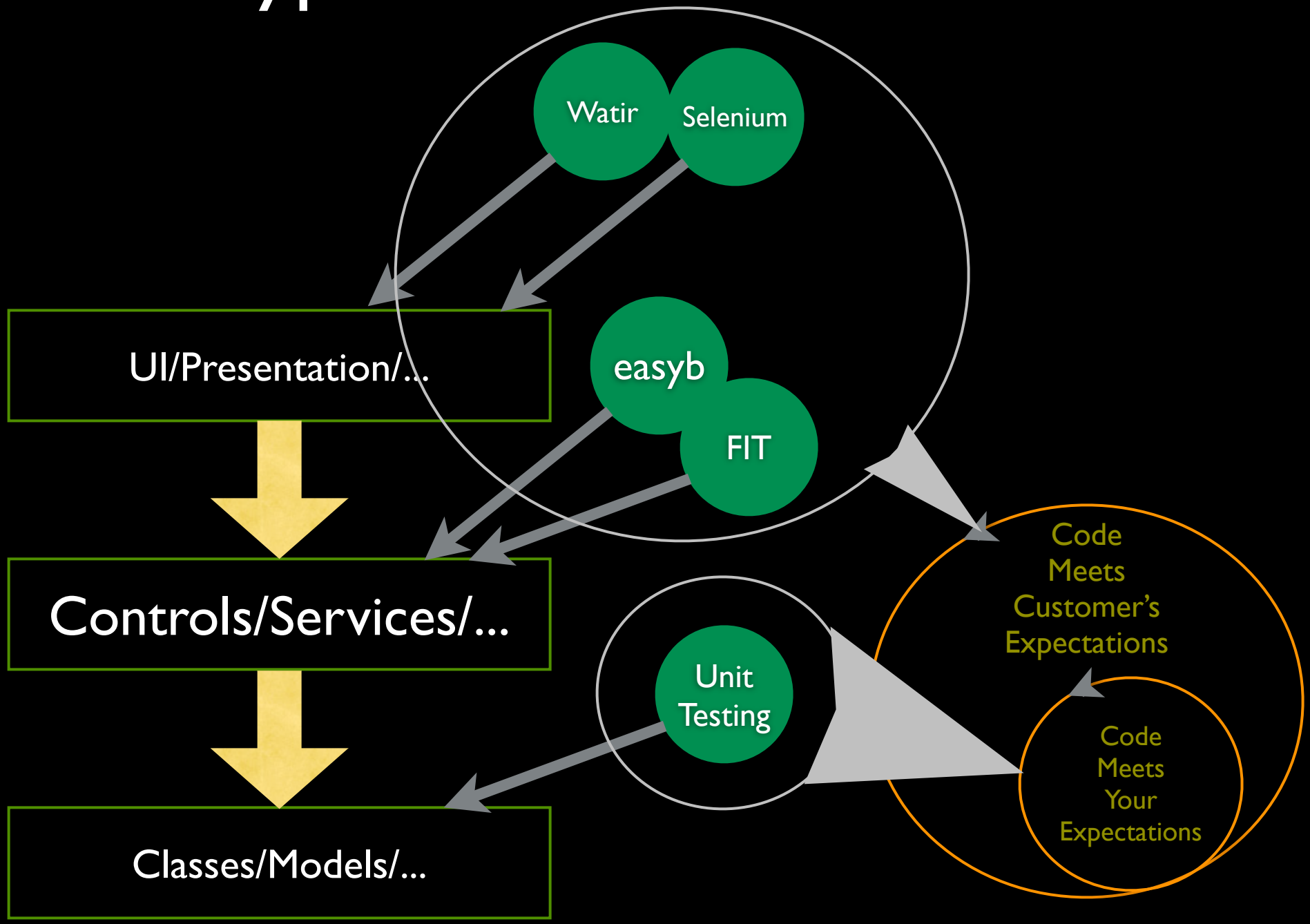
Features oriented

Often lead by customers and testers

Coarse grain

User stories can be expressed as executable documentation

Types of Tests and Levels



easyb

It is a Behavior-Driven Design Tool

Started by Andy Glover

Express Story and Spec using Groovy Based Domain Specific Language (DSL)

Highly expressive

Can be used for Java and Groovy applications

Story Framework and Spec Framework

Story Example

file:money.story

```
scenario 'deposit money', {
  given 'account 12345'
  when 'deposit $50'
  then 'balance of account 12345 goes up by $50'
}
```

Unintegrated or Pending Story

Specifications

file: purchaseSoda.specification

```
vendingMachine = VendingMachine.instance

it "should dispense a can of Pepsi", {
   cans = vendingMachine.cans
   vendingMachine.purchaseSoda "Pepsi", 100
   vendingMachine.cans.shouldEqual cans - 1
}

it "should fail if you ask for Coke", {
  cans = vendingMachine.cans

  ensureThrows IllegalArgumentException, {
    vendingMachine.purchaseSoda "Coke", 100
  }
  vendingMachine.cans.shouldEqual cans
}
```

Running easyb

```
alias easyb="java -classpath
/opt/groovy/easyb-0.9.8/easyb-0.9.8.jar:/opt/groovy/easyb-0.9.8/lib/commons-cl
i-1.2.jar:/opt/groovy/easyb-0.9.8/lib/groovy-all-1.7.5.jar:.
org.easyb.BehaviorRunner"
```

You can now simply run easyb storyfile.story

Personas

Personas help us communicate and relate to specific type of users and situations

For example, Jane may be rich customer, Bob may be saving hard so he can buy a new car

Creating Stories as Tests

transactions.story

```
scenario 'Bob deposits money'
```

Terminal — bash — 80×24

```
> easyb transactions.story
Running transactions story (transactions.story)
Scenarios run: 1, Failures: 0, Pending: 1, Time elapsed: 0.512 sec

1 total behavior ran with no failures
>
```

Creating Scenarios

```
scenario 'Bob deposits money'

scenario 'Jane deposits money'
```

Terminal — bash — 80×24

```
> easyb transactions.story
Running transactions story (transactions.story)
Scenarios run: 2, Failures: 0, Pending: 2, Time elapsed: 0.51 sec

2 total behaviors ran with no failures
>
```

Detailing Story

```
scenario 'Bob deposits money', {
  given 'Account for Bob'
  when 'deposit $50'
  then 'balance goes up by $50'
}

scenario 'Jane deposits money'
```

```
Terminal — bash — 80×24
> easyb transactions.story
Running transactions story (transactions.story)
Scenarios run: 2, Failures: 0, Pending: 2, Time elapsed: 0.611 sec

2 total behaviors ran with no failures
>
```

Integrating the Test

```
scenario 'Bob deposits money', {
  given 'Account for Bob', {
    account = 123456
    service = AccountService.instance
    balance = service.getBalance(account)
  }
  when 'deposit $50', {
    service.deposit(account, 50)
  }
  then 'balance goes up by $50', {
    newBalance = service.getBalance(account)
    newBalance.shouldBe balance + 50
  }
}

scenario 'Jane deposits money'
```

Integrating the Test

```
> easyb transactions.story
Running transactions story (transactions.story)
FAILURE Scenarios run: 2, Failures: 1, Pending: 1, Time elapsed: 0.603 sec

        scenario "Bob deposits money"
        step GIVEN "Account for Bob" -- No such property: AccountService for cla
ss: transactions
        scenario "Bob deposits money"
        step WHEN "deposit $50" -- No such property: service for class: transact
ions
        scenario "Bob deposits money"
        step THEN "balance goes up by $50" -- No such property: service for clas
s: transactions
2 total behaviors ran with 1 failure
>
```

Integrating the Test

```
> easyb transactions.story
Running transactions story (transactions.story)
Scenarios run: 2, Failures: 0, Pending: 1, Time elapsed: 0.714 sec

2 total behaviors ran with no failures
>
```

After Implementing AccountService

The and clause

```
scenario 'Jane deposits money', {
  given 'Account for Jane'
  when 'deposits $5000'
  then 'balance goes up by $50'
  and
  then 'notify homeland security'
}
```

Verifying

shouldBe, shouldntBe, equalTo, ...

ensure(expression1) { expression2 }

ensure(currentBalance) { oldBalance + 50 }

Grouping Methods

before and after

before_each and after_each

shared_behavior and it_behaves_as

Narrative

```
description "..."

narrative 'customer deposits money', {
  as_a 'checking account customer'
  i_want 'to deposit money'
  so_that 'I can save money for a new car'
}
```

Using -txtstory option

```
> easyb transactions.story -txtstory
Running transactions story (transactions.story)
Scenarios run: 2, Failures: 0, Pending: 1, Time elapsed: 0.657 sec

2 total behaviors ran with no failures
> ls easyb-story-report.txt
easyb-story-report.txt
>
```

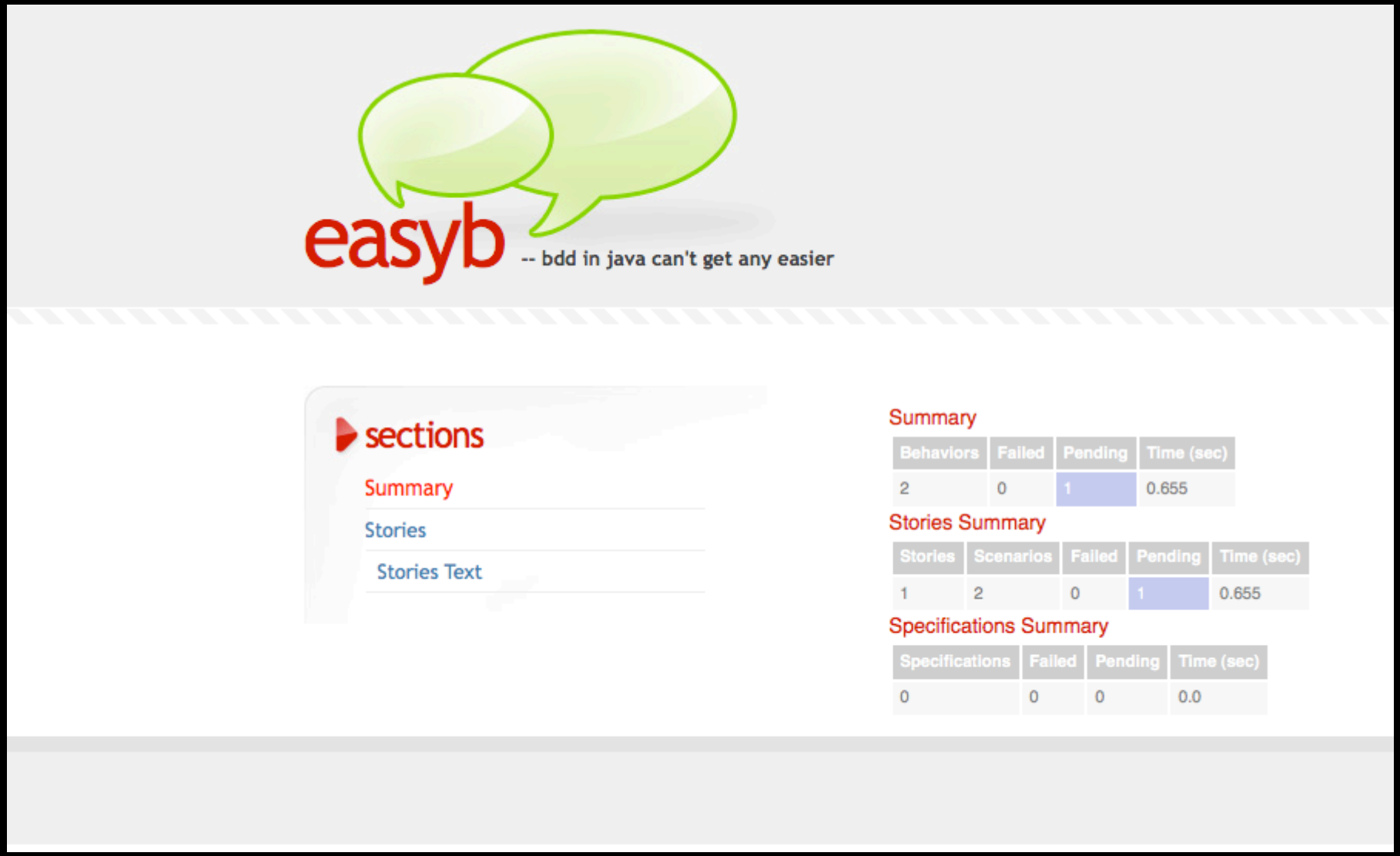
```
2 scenarios (including 1 pending) executed successfully. (including 1 pending
behavior)

  Story: transactions

    scenario Bob deposits money
      given Account for Bob
      when deposit $50
      then balance goes up by $50

    scenario Jane deposits money [PENDING]
      given Account for Jane
      when deposits $5000
      then balance goes up by $50 [PENDING]
      then notify homeland security [PENDING]
```

Using -html option



-prettyprint option

```
> easyb transactions.story -prettyprint
Running transactions story (transactions.story)
Scenarios run: 2, Failures: 0, Pending: 1, Time elapsed: 0.652 sec

2 total behaviors ran with no failures
2 scenarios (including 1 pending) executed successfully.
  Story: transactions


    scenario Bob deposits money
      given Account for Bob
      when deposit $50
      then balance goes up by $50

    scenario Jane deposits money [PENDING]
      given Account for Jane
      when deposits $5000
      then balance goes up by $50 [PENDING]
      then notify homeland security [PENDING]

>
```

Thank you!

http://www.agiledeveloper.com

Venkat Subramaniam
venkats@agiledeveloper.com
twitter: venkat_s