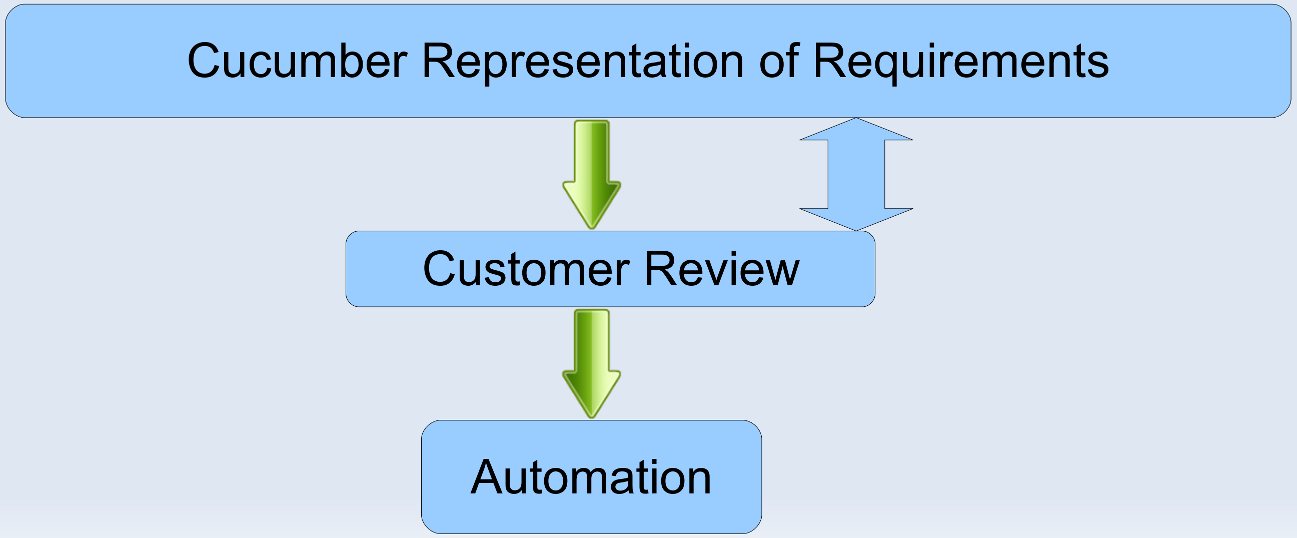
Cucumber-Rails

Integration Testing Tool for Rails 3

About Cucumber...

- Acceptance Test Driven Development
- Collaborative effort between the customer and the delivery team



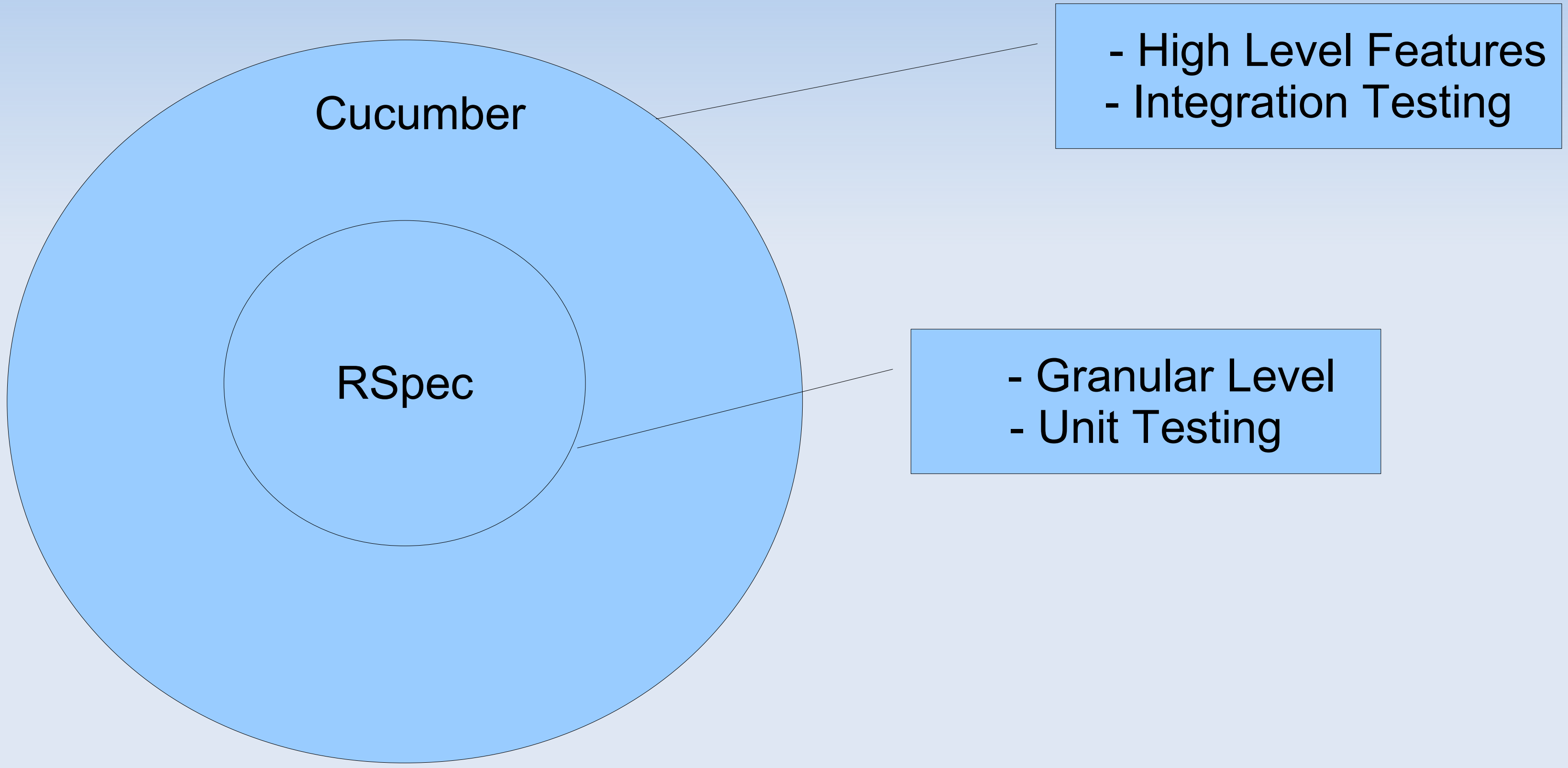
Why Cucumber ?? (Advantages)

- Clarity of Requirements
- Communication Medium used viz Gherkin syntax
- Self explanatory notes for 3rd person.
- Communication gap is overcomed.

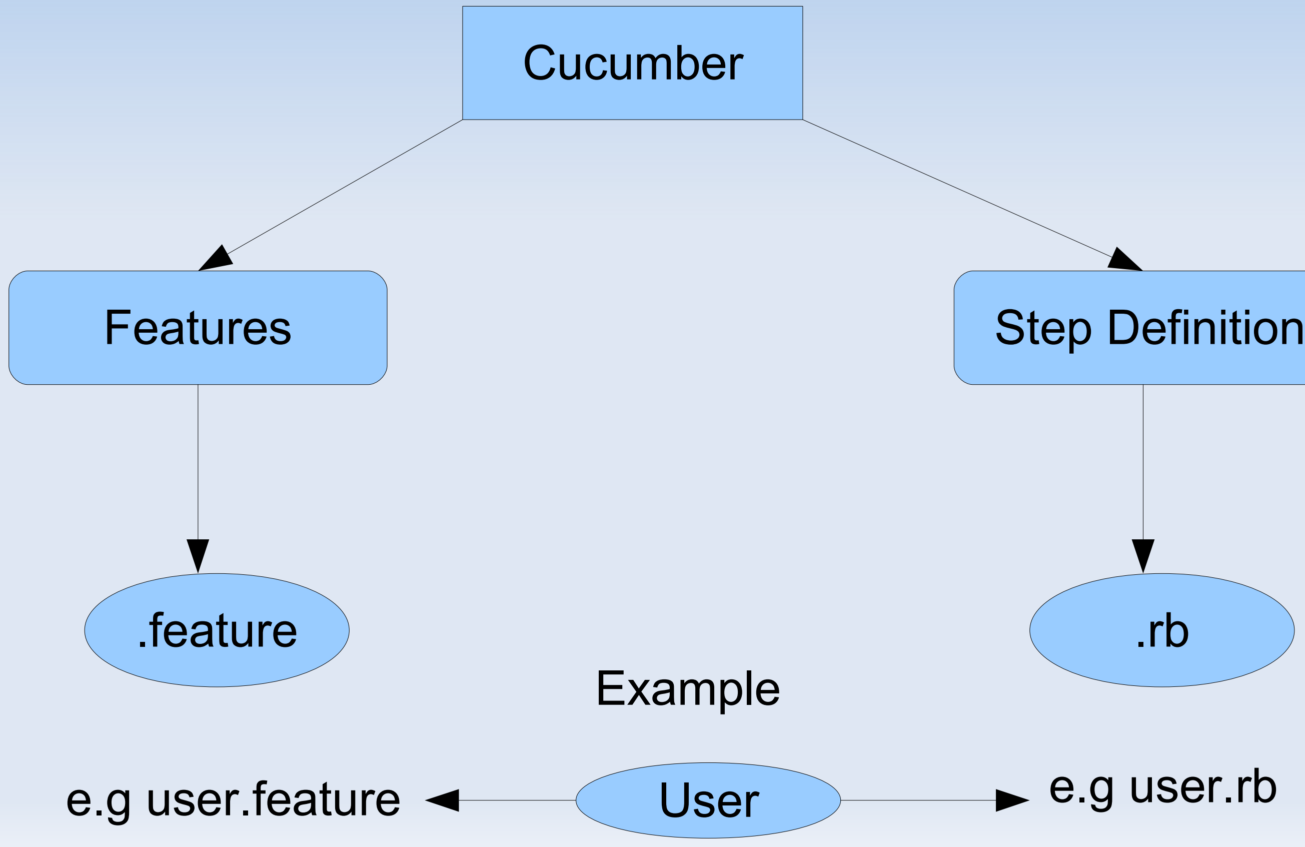
Cucumber Advantages (Cont...)

- Can be used with other programming languages as well.
- Works with Ruby, Java, .NET, Flex or web applications written in any language. It has been translated to over 40 spoken languages.

Relation between Rspec & Cucumber !



Main Elements of Cucumber !!



Features...

Example:

```
Feature: pay bill on-line
        - In order to reduce the time I spend paying bills
        - As a bank customer with a checking account
        - I want to pay my bills on-line

Scenario: pay a bill
        Given checking account with $50
        And a payee named XYZ
        And a bill for $37
        When I pay the bill to XYZ
        Then I should have $13 remaining in my checking account
        And the payment of $37 to XYZ should be listed in Recent Payments
```

Step Definitions...

```
Given /^checking account with \$(\d+)$/ do |bill_amount|
    # Ruby code here
End

Given /^a payee named (.*)$/ do |payee_name|
    # Ruby code here
End

Given /^a bill for \$(\d+)$/ do |bill_amount|
    # Ruby code here
end
```

```
Given checking account with $50
And a payee named XYZ
And a bill for $37
```

Step Definitions (Continued...)

```
When I pay the bill to XYZ
Then I should have $13 remaining in my checking account
And the payment of $37 to XYZ should be listed in Recent Payments
```

```
When /^I pay the bill to (.*)$/ do |payee_name|
        # Ruby code here
End


Then /^I should have \$(\d+) remaining in my checking account$/ do |bill_amount|
        # Ruby code here
End


Then /^the payment of \$(\d+) to (.*) should be listed in Recent Payments$/
do |amount,payee_name|
        # Ruby code here
end
```

Cucumber with Rails 3...

- gem 'cucumber-rails'

```
group :test do
  gem 'cucumber-rails'
end
```

- Bootstrap Cucumber

```
rails g cucumber:install
```

- config/cucumber.yml

- script/cucumber

- features/step_definitions

- features/support

- features/support/env.rb

- lib/tasks/cucumber.rake

”Background” feature

```
Feature: User Authorization
  As an account owner
  I should be able to view all the reports for my account

Background:
    Given the user "xyz" is logged-in
    And has the role as "account owner"

Scenario: User "xyz" view his reports
    When the user "xyz" access the reports page
    Then user "xyz" should see all his reports
```

Scenario Outlines

```
Feature: User Authorization
  As an account owner
  I should be able to view all the reports & employees for my account
Background:
    Given the user "xyz" is logged-in
    And has the role as "account owner"
Scenario Outline: User "xyz" view his reports & employees
    When the account owner <user> access the <page> page
    Then account owner <user> should see <count> <page>
Examples:
 |user|page|count|
  |abc|reports|500|
  |xyz|reports|700|
  |xyz|employees|20|
```

Tags...

- To organize your features and schenario's

```
@signin
Feature: Signin

@login
 Scenario: Login
   Given Credentails...
   When logged in
   Then...

@fyp
Scenario: Forgot your password
   Given registered email
   When clicked on forgot your password
   Then email should be delivered
```

```
@signup
 Feature: Signup

@signup @resourcename
 Scenario: Signup
  Given Information...
  When Signed up
   Then...
```

Execution...

- All Features

  > cucumber

  OR

  > cucumber features/

- Particular Feature

  > cucumber features/<file>.feature

- Particular Test case scenario

  > cucumber features/ --tags @signin

Cucumber.yml

```
default: --format progress features
html_report: --format html --out=features_report.html  features
signup_report :  --tags @signup features
```

Execution:

```
> cucumber --profile html_report
> cucumber --profile signup_report
```

Feature:

```
@signup
Scenario:ABC        Scenario:XYZ
 Given...            Given...
  When...             When...
  Then...             Then...
```

Ambiguous Steps !!

Scenario1: Different users view reports ***When the user "xyz" access the reports page*** *Then user "xyz" should see all his reports*	Feature
When /^the user (.) access the reports page$/ do | user_name |* *# Ruby code here* *End*	Step-Defn
Scenario2: User ”xyz” view different pages ***When the user xyz access the ”reports” page*** *Then user xyz should see all his ”reports”*	Feature
When /^the user xyz access the (.) page$/ do | page_name |* *# Ruby code here* *End*	Step-Defn

Redundant Steps !!

Scenario: User ”xyz” view his reports

When the user ”xyz” access the ”reports” page

Then user ”xyz” should see all his reports

Reports.feature

Scenario: User ”xyz” view his transactions

When the user ”xyz” access the ”transactions” page

Then user ”xyz” should see all his transactions

Trans.feature

```
When /^the user (.*) access the (.*) page$/ do | user_name, page  |
        # Ruby code here
End
```

”Spork” (What & Why)

- To speed up the running of an application’s tests
- Preloads the Rails Environment once at the start
- The whole Rails initialization process is skipped, saving valuable seconds off of your spec runs
- A little analysis reveals that execution time gets reduced by about 3.5 min or so in case of 900 examples

When to restart Spork ??

- No need to restart the Spork server if there are any changes in your test cases written under ”***model-spec***” or say ”***controller-spec***”

- Need to restart the Spork server for any changes in the ***configuration*** in your application.

  examples:

  - change in Factory's

  - spec_helpers

  - spec/support/...

  - .rspec

Get Spork working on Rails 3 & Rspec 2

- Add ”**Spork**” gem into your Gemfile
- Configure the option **--drb** on a new line in your **.rspec** file
- Modify your **spec_helper.rb** as follows:

```
require 'spork'
  Spork.prefork do
     ENV["RAILS_ENV"] ||= 'test'
     require File.expand_path("../../config/environment", __FILE__)
     require 'rspec/rails'
     ...
  end
```

- Finally start the ”Spork” server as *:$ bundle exec spork*

.rspec

```
--color
--drb
```

Get Spork working on Rails 3 & Cuke

- Add "**Spork**" gem into your Gemfile
- Configure the option **--drb** on a new line in your **cucumber.yml** file
- Modify your **features/support/env.rb** as follows:

```ruby
require 'spork'
  Spork.prefork do
     ENV["RAILS_ENV"] ||= 'test'
     require File.expand_path(File.dirname(__FILE__) + '/../../config/environment')
    require 'cucumber/rails/rspec'
  end
```

Cucumber.yml

```
pretty_format: --drb --format pretty features
```

- Finally start the "Spork" server as *: $ bundle exec spork cucumber*

Thank You !!