

BDD!

Cual la historia?

2003

- Dan North crea JBehave. Comenzó como un experimento para ver como podría haber sido JUnit si se hubiera concebido desde un principio como una herramienta para hacer TDD

2005

- Algunas ideas de JBehave fueron incorporadas por Dave Astels en RSpec

2007

- Dan North inspirado por el la tracción generada por RSpec crea RBehave, el cual trae como novedad la posibilidad de especificar comportamiento en texto plano, idea que luego daría origen a lo que en la actualidad se conoce como sintaxis Gherkin

2008

- Reescriben completamente desde cero JBehave incorporan soporte para especificaciones en texto plano

2008

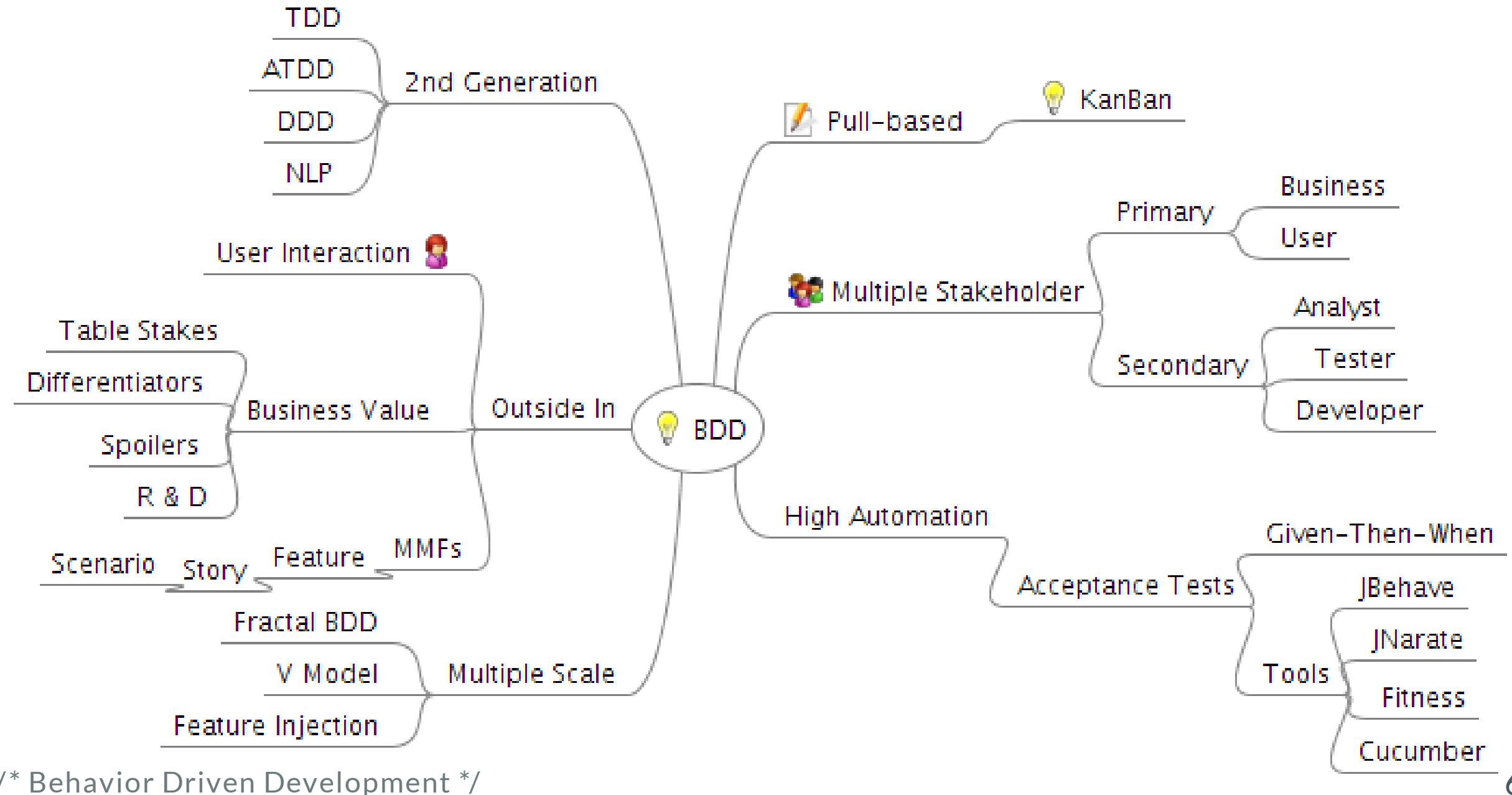
- Nace Cucumber como remplazo del motor de especificaciones de RSpec. Se estandariza la sintaxis de especificacion de texto plano, y se la bautizó como Gherkin

Qué es BDD?

Dan North:

BDD is a second-generation, outside-in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology. It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters.

Qué es BDD?



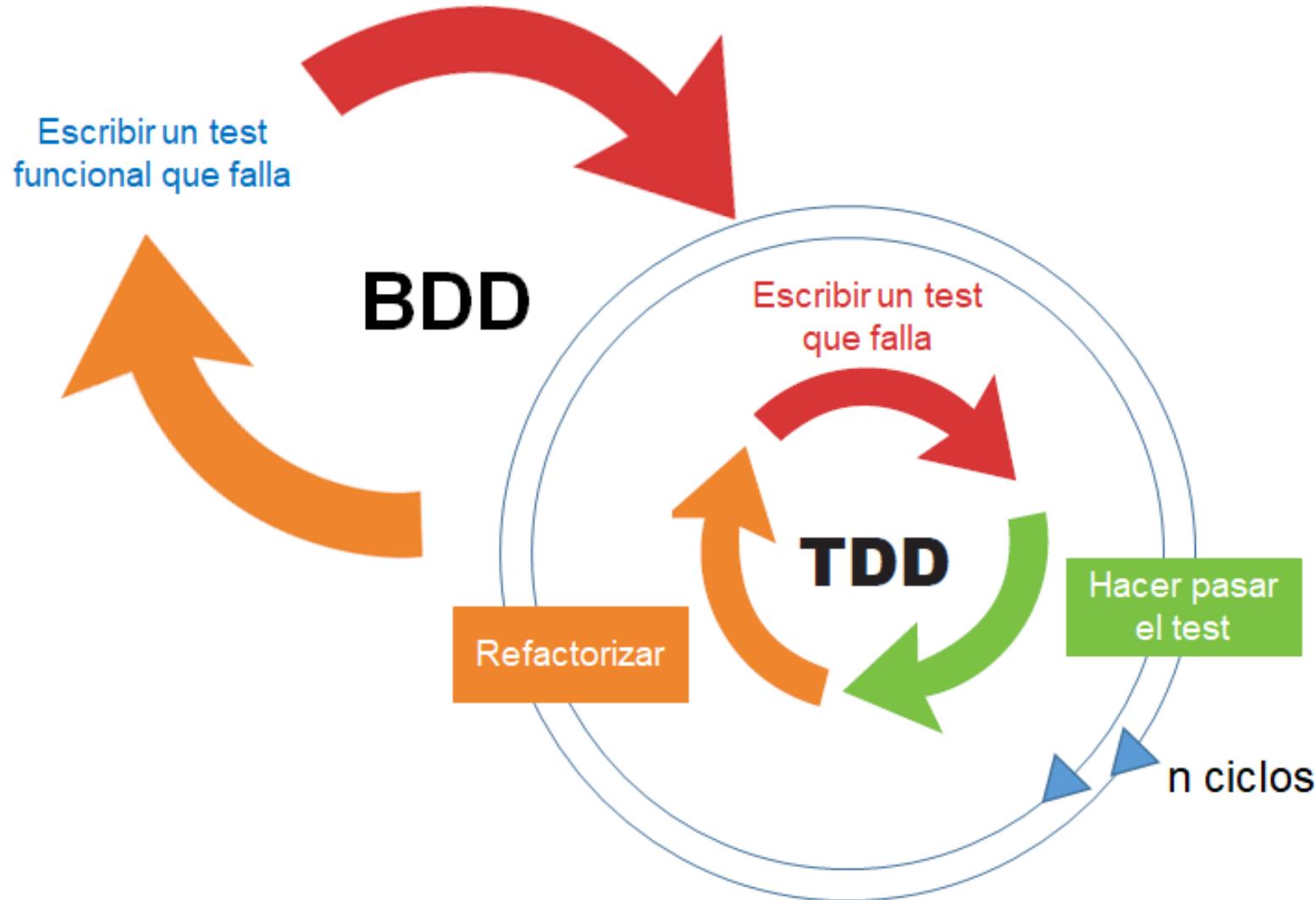
Qué no es BDD?

BDD no es E2E testing

Cómo se vincula con TDD?

TDD	BDD
Visión del programador	Visión del cliente
Una aproximación de bajo nivel	Una aproximación desde el punto de vista del usuario
Verifica si la implementación de la funcionalidad es correcta	Verifica si la aplicación se comporta como el usuario quiere que se comporte

Cómo se vincula con TDD?



Como aplicar BDD?

Cómo aplicar BDD?

En papel...

Luego ir a codear los tests y ya

BDD es un proceso,
no una herramienta



Cómo aplicar BDD?

Pasar requerimientos del cliente a una especificación formal y luego automatizar los tests



Cómo aplicar BDD?

Armar y mantener la especificación junto con el cliente



Cómo escribir una especificación en BDD?

Cómo escribir una especificación en BDD?

Usando Gherkin:

Necesidad del negocio: Dar una charla de BDD

Escenario: Se explica el formato de especificación de manera clara
Dado que se está presentando la charla
Cuando se muestre el ejemplo de especificación
Entonces debe entenderse claramente
Y debe haber un chiste sobre recursión

Feature: Account operations

Scenario: Extract money from account

Given I have an account with \$80

When I extract \$10 from my account

Then I should have \$70 in my account

Scenario: Deposit money into account

Given I have an account with \$300

When I deposit \$150 into my account

Then I should have \$450 in my account

Gauge

Find movies playing near me

=====

The System Under Test in this example is a web application to find and book movie tickets

Search for movies

- * Specify location as "Bangalore"
- * Search for movie "Star Wars"
- * Verify that "INOX" is playing "Star Wars" at "7:30pm"

Fit: Framework for Integrated Test

The screenshot shows a Microsoft Word document window titled "result.htm - Microsoft Word". The menu bar includes File, Edit, View, Insert, Format, Tools, Table (which is highlighted), Window, Documents To Go, Help, and a close button. Below the menu is a toolbar with various icons. The main content area contains the following text:

Basic Employee Compensation

For each week, hourly employees are paid a standard wage per hour for the first 40 hours worked, 1.5 times their wage for each hour after the first 40 hours, and 2 times their wage for each hour worked on Sundays and holidays.

Here are some typical examples of this:

Payroll Fixtures	Weekly Compensation		
StandardHours	HolidayHours	Wage	Pay()
40	0	20	\$800
45	0	20	\$950
48	8	20	\$1360 <i>expected</i> \$1040 <i>actual</i>

Concordion

To help personalise our mailshots we want to have the first name and last name of the customer. Unfortunately the customer data that we are supplied only contains full names.

The system therefore attempts to break a supplied full name into its constituents by splitting around whitespace.

Example

The full name Jane Smith is broken into first name `Jane` and last name `Smith`.

```
The full name [Jane Smith](- "#name") is [broken](- "#result = split(#name)")  
into first name [Jane](- "?=#result.firstName") and last name [Smith](- "?=#result.lastName").
```

Cómo se organiza una especificación en Gherkin?

Feature:

Colección de escenarios/ejemplos. Se puede pensar tanto como un caso de uso como una historia de usuario.

Scenario / Example:

Una serie de pasos que describen el comportamiento deseado del sistema... en otras palabras, un test.

Step:

Una oración que describe qué debería ocurrir.

Qué debe tener cada escenario?

Given:

Una precondición necesaria para que se cumpla el escenario.

Then:

La acción a la cual le estamos definiendo el comportamiento.

When:

Una postcondición que debe cumplirse luego de realizar la acción.

Puntos a tener en cuenta:

- Usar lenguaje ~~ubicuo~~ fácil de entender
- Descripciones lo más cortas posibles. Que sean realmente consisas y que vayan al punto del asunto, sin dar vueltas. Tienen que entrar en un renglón.
- Mantenga persona y tiempo verbal
- Reutilizar steps
- Reutilizar steps

Cómo testear una especificación?

Cómo **testear** automatizar una especificación?

Cómo automatizar una especificación?

Mantra:

- El test es la especificación
- La especificación es el test
- El resto es código plasticola



Cómo automatizar una especificación?

Depende del tipo de especificación:

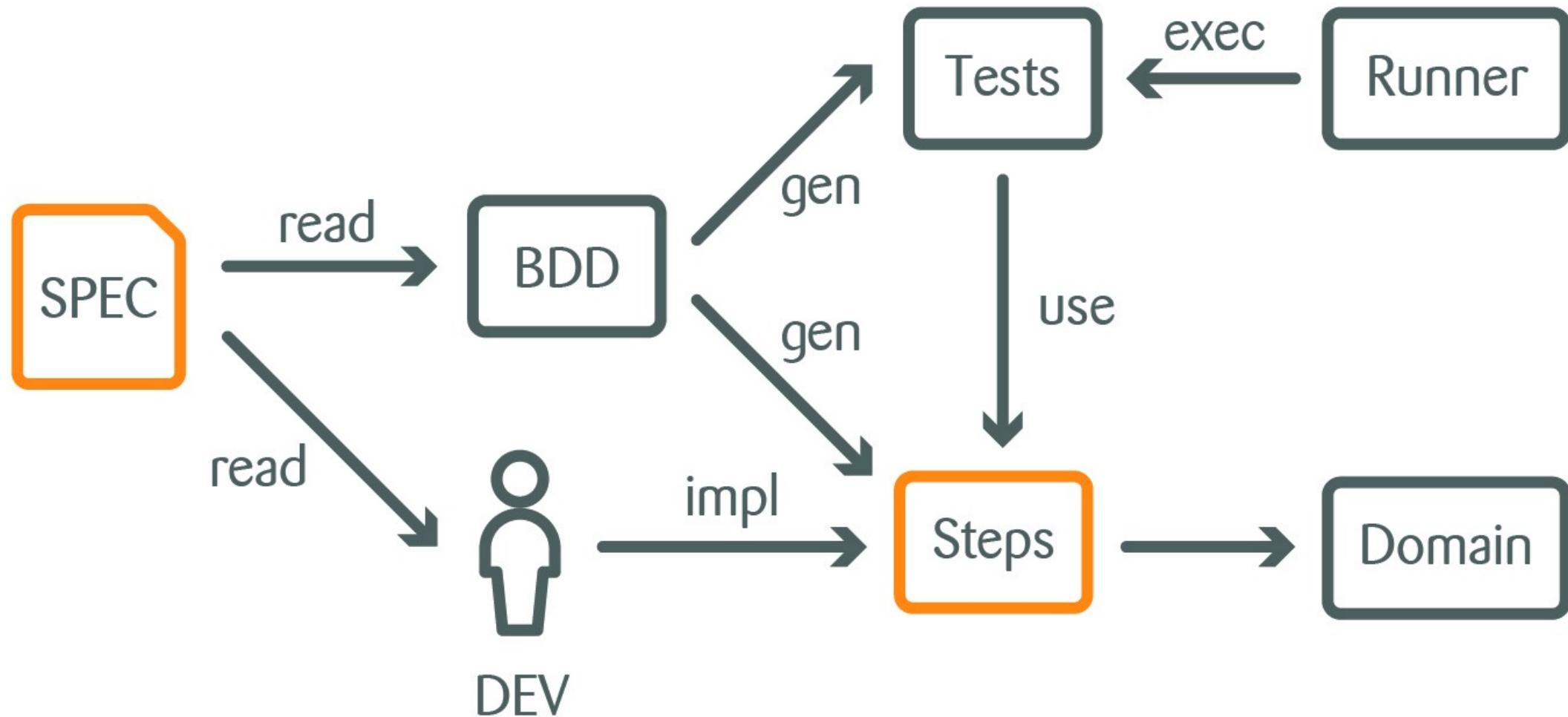
- Gherkin: con Cucumber, Behat, SpecFlow, etc...
- Gauge
- Fit
- Concordión

Cómo automatizar una especificación?

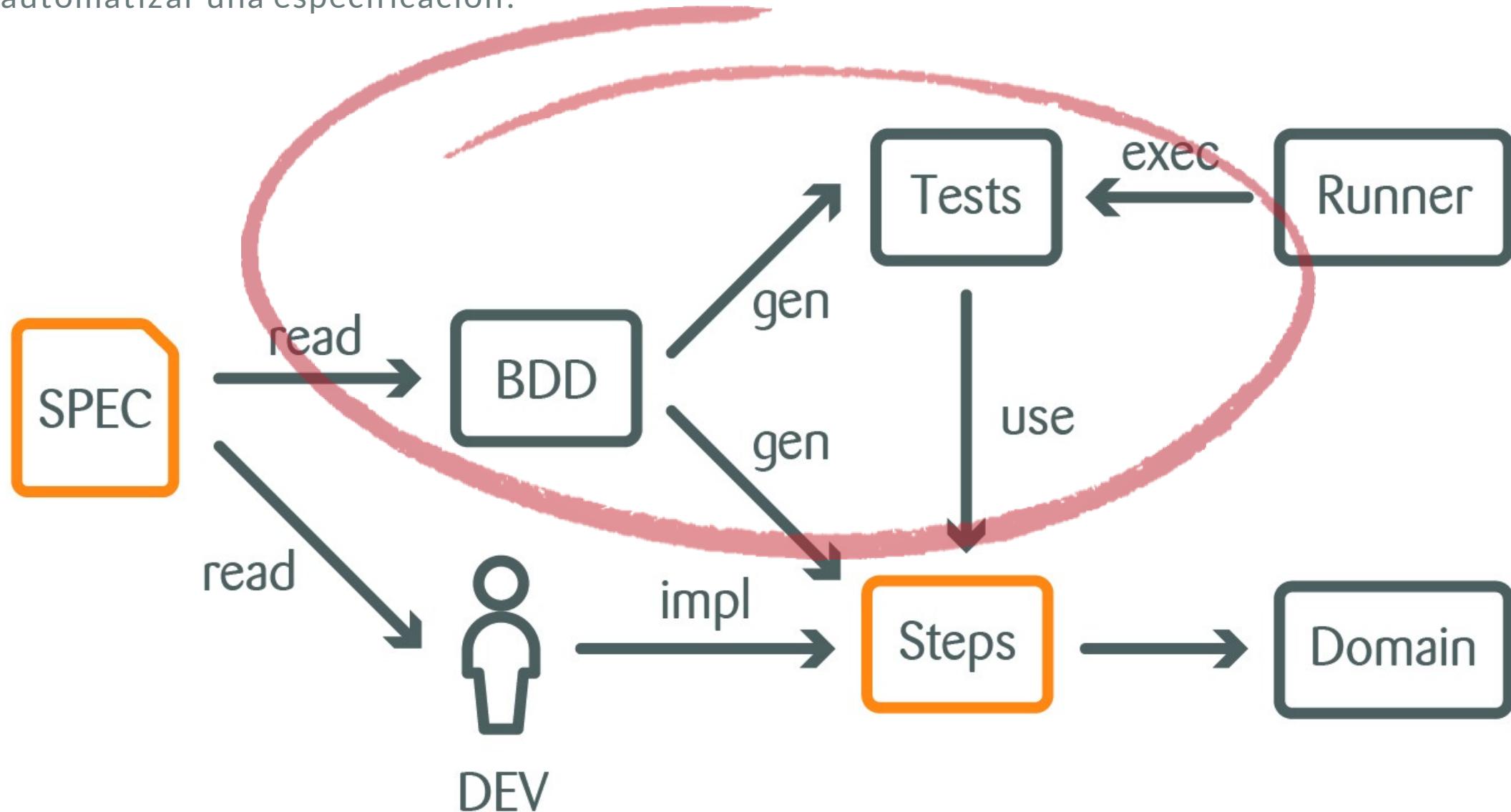
Para Gherkin:

- El step es la unidad ejecutable
- Alcanza con implementar cada step
- Escribir una función que matchee con el step

Cómo automatizar una especificación?



Cómo automatizar una especificación?



Cómo automatizar una especificación?

Feature: Greeting

Scenario: Say hello

When the greeter says hello

Then I should have heard "hello"

```
When('the greeter says hello', function () {
  this.whatIHeard = new Greeter().sayHello();
});
```

```
Then('I should have heard {string}', function (expectedResponse) {
  assert.equal(this.whatIHeard, expectedResponse);
});
```

Cómo automatizar una especificación?

Feature: Greeting

Scenario: Say hello

When the greeter says hello

Then I should have heard "hello"

```
/**  
 * When the greeter says hello  
 */  
function theGreeterSaysHello() {  
    this->whatIHeard = (new Greeter())->sayHello();  
}
```

Cómo automatizar una especificación?

- Reutilizar steps es clave
- Parametrizar con criterio

Aunque no parezca, estamos escribiendo nuestro propio DSL para especificar el comportamiento del sistema



Un ejemplo

Un ejemplo

- Setup del proyecto
- Escribir el primer escenario
- Definir los steps
- Hacer pasar el test
- Escribir otro escenario
- Definir los steps
- Hacer pasar el test...



Manos a la obra

Cosas importantes a mencionar

- “ En las reuniones de BDD juntar diversos roles para que... ”
- “ Si no pones distintos perfiles en la reunion no sería BDD ”
- “ Spec sirve de doc siempre y cuando esté acualizado ”

Más material

- Cucumber framework: cucumber.io/docs/guides/overview/
 - cucumber-js: github.com/cucumber/cucumber-js
- Behat framework: docs.behat.org/en/latest/
- Specification by example: bib.convdocs.org/v1856/?download=file

Preguntas?