Automatización y Gestión de las Pruebas Funcionales usando Herramientas Open Source

Ignacio Esmite, **Mauricio Farías**, Nicolás Farías, Beatriz Pérez

Centro de Ensayos de Software

Agenda

- Contexto
- Herramientas
- Demostración Selenium Core
- Metodología propuesta
- Experiencia
- Futuro

Centro de Ensayos de Software

- Consorcio entre:
  - Cámara Uruguaya de Tecnologías de la Información (CUTI)
  - Universidad de la República de Uruguay
- Servicios
  - Prueba independiente
    - ✓ Testing funcional
    - ✓ Ensayos de plataforma
  - Capacitación en testing
  - Consultoría en testing

CES
CENTRO DE ENSAYOS DE SOFTWARE

Contexto

- Empresas
  - Productos de software en continuo mantenimiento y mejora
  - Exigencias de calidad crecientes
- Necesidades
  - Reducir costos y tiempos en las pruebas de regresión
  - Conjunto de pruebas de humo automatizadas
  - Probar sobre diferentes plataformas

Contexto

- Interés
  - Automatizar las pruebas funcionales
- Herramientas de automatización Open Source
  - Sin costo de licencia
  - Posibilidad de extensión

Terminología

- *Script* de prueba
  - Programa que automatiza la ejecución de una prueba
- *Suite* de prueba
  - Conjunto de *scripts* de prueba

Selenium

- Herramienta para la automatización de pruebas funcionales para aplicaciones web
- Permite
  - Crear pruebas de regresión
  - Probar la aplicación con diferentes navegadores y sobre diferentes plataformas
- Proyecto *open source*. Comunidad OpenQA

Selenium

- Compuesta por
  - Selenium Core
    - ✓Ejecución de pruebas automatizadas
  - Selenium IDE
    - ✓Creación y mantenimiento de pruebas automatizadas
  - Selenium Remote Control
    - ✓Creación de pruebas escritas en lenguajes de programación como Java o C#

Otras herramientas

- Entorno de desarrollo Eclipse
  - Manejar cómodamente suites, scripts, documentación técnica y de gestión
  - Editor de HTML y XML
  - Cliente CVS sencillo
- Mozilla Firefox – Extensiones
  - Firebug
  - XPath Checker
  - XPather

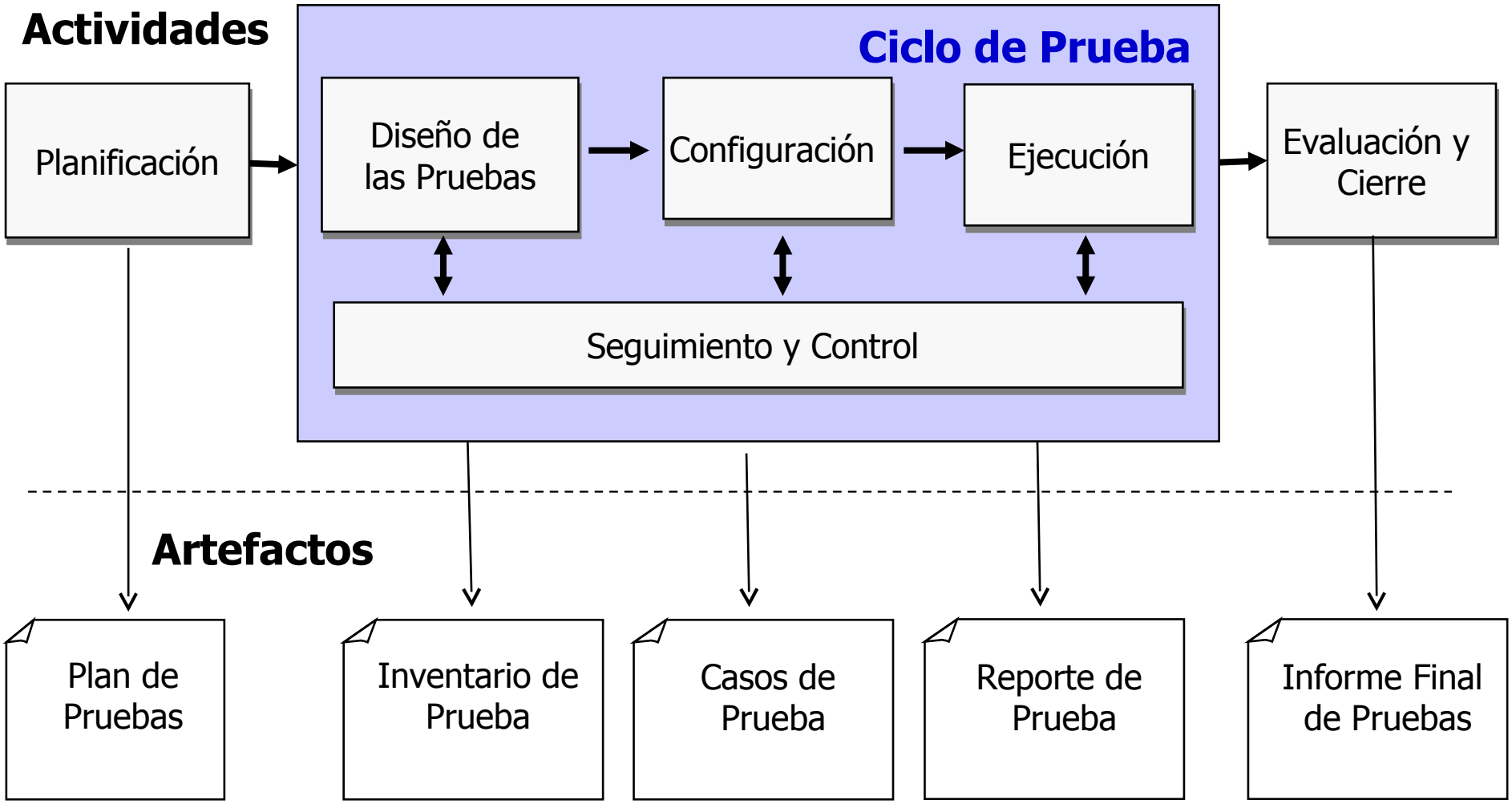
Demo Selenium Core

- DMS 1.0 – Document Management System
- Ejecución de la suite
  - Login Administrator
  - Create User
  - Logout
  - Login User
  - Logout
  - Login Administrator
  - Delete User
  - Logout

¡ Magia !

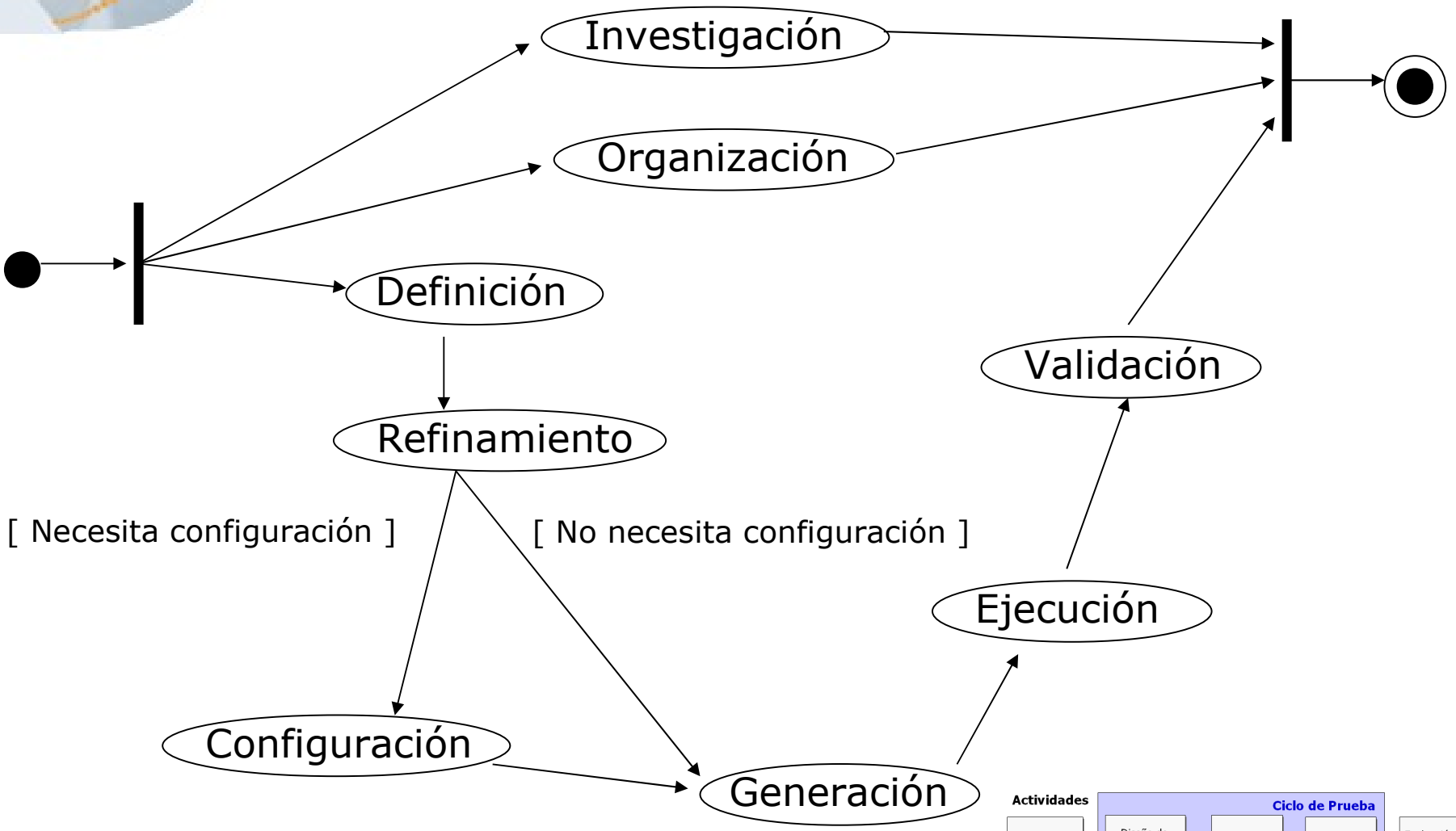
¿ Magia ?

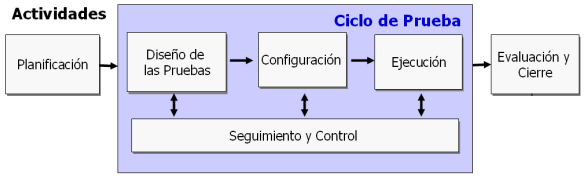
ProTest – Proceso pruebas funcionales



Metodología propuesta

- Necesidad de extender el proceso
- Nuevas actividades de automatización
  - Surgen del trabajo realizado
  - Se fueron ajustando con la experiencia


Investigación
Organización
Definición
Refinamiento
[Necesita configuración]
[No necesita configuración]
Configuración
Generación
Ejecución
Validación


Actividades
Ciclo de Prueba
Planificación
Diseño de las Pruebas
Configuración
Ejecución
Evaluación y Cierre
Seguimiento y Control

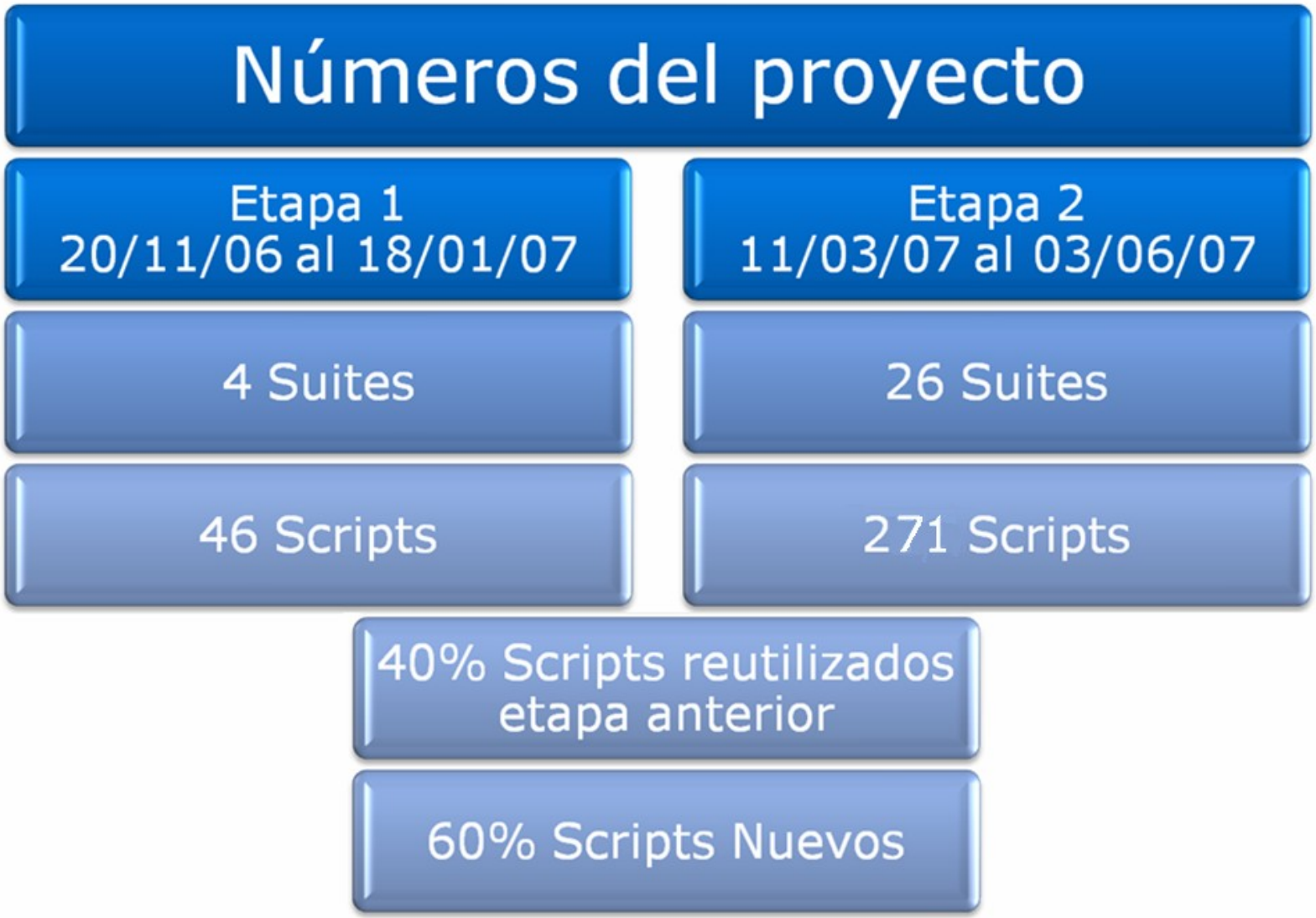
Experiencia

- Varios proyectos
  - Se aplica la metodología
- Objetivo de proyecto particular
  - Automatizar pruebas funcionales
  - Para ejercitar caminos y ciclos funcionales típicos
  - Para probar cada build creado
  - Para probar liberaciones con
    - ✓ Diferentes plataformas de desarrollo
    - ✓ Diferentes DBMS
    - ✓ Diferentes navegadores

Datos del proyecto

- Recursos humanos
  - Líder del proyecto
  - 3 automatizadores

- Etapas
  - Primera etapa
    - ✓2 meses
  - Segunda etapa
    - ✓3 meses


Números del proyecto
Etapa 1
20/11/06 al 18/01/07
Etapa 2
11/03/07 al 03/06/07
4 Suites
26 Suites
46 Scripts
271 Scripts
40% Scripts reutilizados
etapa anterior
60% Scripts Nuevos

Resumen de la experiencia

- Factibilidad de proyectos de automatización de pruebas funcionales utilizando herramientas *open source*
- Selenium
  - simple
  - potente
  - flexible
  - lenguaje
    - ✓ fácil de usar
    - ✓ fácil de aprender
  - no brinda un entorno para gestionar las pruebas

Futuro

FitNesse

- Herramienta para la mejora de
  - Comunicación
  - Colaboración
- Documentar
  - el sistema debe hacer
  - compararlo con lo que el sistema hace
- Pueden participar
  - Usuarios, programadores y testers
  - desde el inicio del desarrollo del sistema
- Implementación como wiki

FitNesse

FitNesse.

TwoMinuteExample

EDIT PAGE

```
[[A One-Minute Description][OneMinuteDescription]]
!1 An Example !-FitNesse-! Test
If you were testing the division function of a calculator application, you might like to see
some examples working. You might want to see what you get back if you ask it to divide 10 by
2. (You might be hoping for a 5!)

In FitNesse, tests are expressed as tables of '''input''' data and '''expected output'''
data. Here is one way to specify a few division tests in FitNesse:

|eg.Division|
|numerator|denominator|quotient?|
|10       |2          |5        |
|12.6     |3          |4.2      |
|100      |4          |24       |

In this style of FitNesse test table (ColumnFixture), each row represents a complete
scenario of example inputs and outputs. Here, the "numerator" and "denominator" columns are
for inputs, and the question mark in the "quotient?" column tells FitNesse that this is our
column of expected outputs. Notice our "10/2 = 5" scenario. Try reading it as a question:
'''"If I give you a numerator of 10 and denominator of 2, do I get back a 5?"'''
!3 Running our test table: Click the Test button
Before we do another thing, let's run this test table. See the little blue and white
'''Test''' button in the upper-left, just below the !-FitNesse-! logo? Click it and see what
happens.

Ah, color! In the green cells, we got back the expected values from our code. When we
```

Save | Spreadsheet to FitNesse | FitNesse to Spreadsheet | - Insert Fixture Table -

Test
Versions
Properties
Refactor
Where Used
RecentChanges
Files
Search
Edit Locally
Edit Remotely

FitNesse.

TwoMinuteExample

▼ Set Up: .FitNesse.SetUp

A One-Minute Description

An Example FitNesse Test

If you were testing the division function of a calculator application, you might like to see some examples working. You might want to see what you get back if you ask it to c

In FitNesse, tests are expressed as tables of **input** data and **expected output** data. Here is one way to specify a few division tests in FitNesse:

eg.Division		
numerator	denominator	quotient?
10	2	5
12.6	3	4.2
100	4	24

In this style of FitNesse test table (ColumnFixture), each row represents a complete scenario of example inputs and outputs. Here, the "numerator" and "denominator" colum the "quotient?" column tells FitNesse that this is our column of expected outputs. Notice our "10/2 = 5" scenario. Try reading it as a question: *"If I give you a numerator of 1*

Running our test table: Click the Test button

Before we do another thing, let's run this test table. See the little blue and white **Test** button in the upper-left, just below the FitNesse logo? Click it and see what happens.

Ah, color! In the green cells, we got back the expected values from our code. When we divided 10 by 2, we expected and got back 5. When we divided 12.6 by 3, we expecte

What about red? A cell turns red when we get back a different value than what we expected. We also see two values: the **expected** value and the **actual** value. Above we e but we got back 25. Ah, a flaw in our test table. That happens!

FitNesse.

TwoMinuteExample

TEST RESULTS

Tests Executed OK

Test
Edit
Versions
Properties
Refactor
Where Used
RecentChanges
Files
Search

Assertions: 2 right, 1 wrong, 0 ignored, 0 exceptions

A One-Minute Description

AN EXAMPLE FITNESSE TEST

If you were testing the division function of a calculator application, you might like to see some examples working. You might want to see what you get back if you ask it to divide 10 by 2. (You might be hoping for a 5!)

In FitNesse, tests are expressed as tables of **input** data and **expected output** data. Here is one way to specify a few division tests in FitNesse:

eg.Division		
numerator	denominator	quotient?
10	2	5
12.6	3	4.2
100	4	33 *expected* 25.0 *actual*

In this style of FitNesse test table (ColumnFixture), each row represents a complete scenario of example inputs and outputs. Here, the "numerator" and "denominator" columns are for inputs, and the question mark in the "quotient?" column tells FitNesse that this is our column of expected outputs. Notice our "10/2 = 5" scenario. Try reading it as a question: *"If I give you a numerator of 10 and denominator of 2, do I get back a 5?"*

RUNNING OUR TEST TABLE: CLICK THE TEST BUTTON

Before we do another thing, let's run this test table. See the little blue and white **Test** button in the upper-left, just below the FitNesse logo? Click it and see what happens.

Ah, color! In the green cells, we got back the expected values from our code. When we divided 10 by 2, we expected and got back 5. When we divided 12.6 by 3, we expected and got back 4.2.

What about red? A cell turns red when we get back a different value than what we expected. We also see two values: the **expected** value and the **actual** value. Above we expected 33 back when we divided 100 by 4, but we got back 25. Ah, a flaw in our test table. That happens!

Futuro

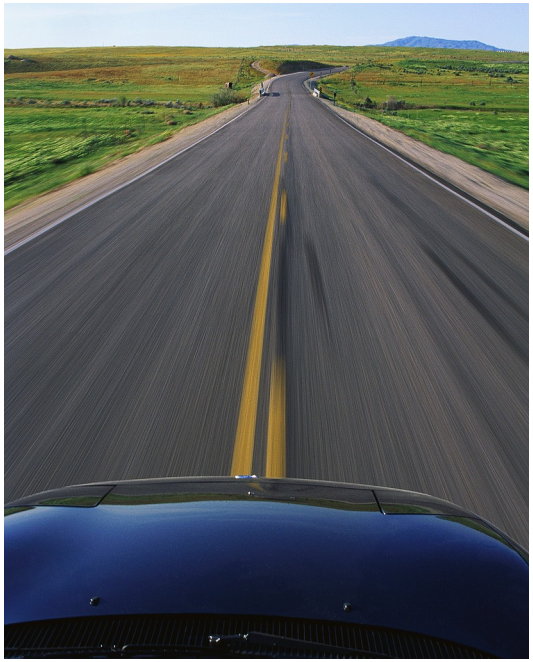
- Selenium Remote Control
  - Implementar suites y scripts más complejos
  - Conocer más a fondo
    - ✓ fortalezas
    - ✓ debilidades
- FitNesse
  - Ayudar a la gestión de scripts
  - Mejorar la comunicación y colaboración en el equipo de proyecto
    - ✓ Proyecto de automatización
    - ✓ Proyecto de desarrollo de la aplicación

Futuro

- FitNesse como interfaz de Selenium Remote Control
  - Crear un lenguaje sencillo para escribir pruebas automatizadas
  - Comandos de Selenium como base
  - Agregar comandos que permitan
    - ✓ varias acciones a la vez
    - ✓ nuevas funcionalidades

Futuro

- Metodología de trabajo
  - Diseñador escribe pruebas en lenguaje simple y abstracto usando estrategias de diseño
  - Automatizador implementa el lenguaje encapsulando problemas técnicos



Gracias,
¿Preguntas?

Mauricio Farías
mfarias@fing.edu.uy
Centro de Ensayos de Software
www.ces.com.uy
ces@fing.edu.uy