Seminar 7 – FitNesse

PV260 Software Quality

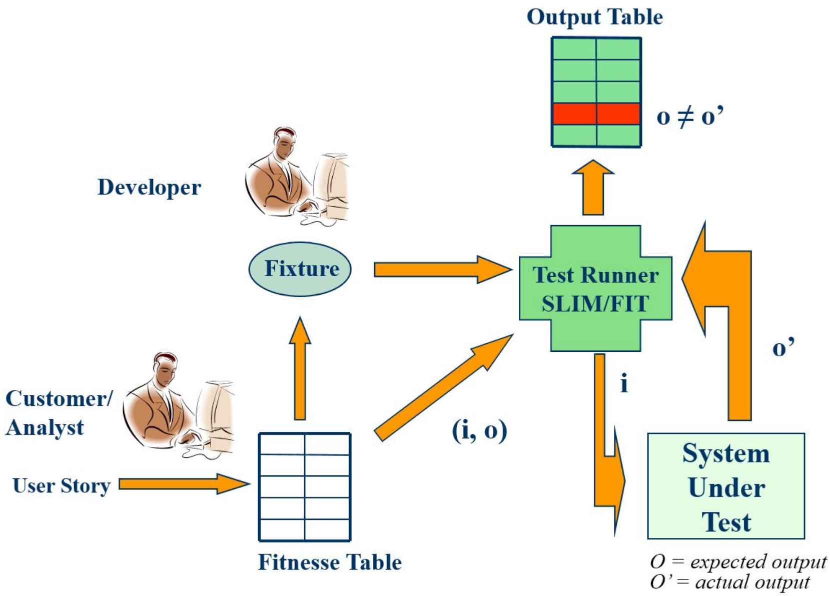
Stanislav Chren, Václav Hála

7. 4. 2015

Introduction

- **F**ramework for **I**ntgrated **T**esting and a wiki system
- Used for acceptance testing
- Easy way to capture user requirements (e.g. in terms of user stories)
- Test cases are written in natural language - both users and developers can write the tests
- Available for various programming languages (Java, C++, C#, Ruby, ...)
- Offers additional extensions for testing of GUI, REST API, ...

How It Works



Output Table
o ≠ o'
Developer
Fixture
Test Runner SLIM/FIT
o'
i
Customer/ Analyst
(i, o)
User Story
System Under Test
Fitnesse Table
O = expected output
O' = actual output

SLIM

- Slim (Simple List Invocation Method) is a lightweight test engine for Fitnesse.
- In the test wiki page, use `!define TEST_SYSTEM{slim}`
- Allows to use various kinds of test tables:
  - Decision table
  - Query table
  - Script table
  - Scenario table
  - ...

Decision Table

- Basic table for comparing inputs with expected outputs

Concatenate strings		
first	second	concatenate?
hello	world	hello world
PV260 Software	Quality	Software Quality

Concatenate strings		
first	second	concatenate?
hello	world	hello world
PV260 Software	Quality	Software Quality

Decision Table - Fixture

- The text in the first cell is mapped to the fixture class name
- Input column names are mapped to the setter methods
- Output column names are mapped to the output methods
- Additional optional methods can be implemented (`beginTable()`, `endTable()`, `reset()`, `execute()`, `table()`)
- The execution flow for each table is as follows:
  1. First, the Fixture object is contructed
  2. `table() method is called`
  3. `beginTable()` is called (e.g. for initialization)
  4. For each row:
     - 4.1 `reset() is called`
     - 4.2 All set methods are called
     - 4.3 `execute()` is called
     - 4.4 All output methods are called and the return values are compared to the table values
  5. `endTable()` is called (e.g. for clean up)

Query Table

- Begins with keyword `Query:`
- The rows in a query table represent the expected results of a query.
- The fixture must contain a parametrized constructor and a `query() method`
- The `query()` method returns list of rows. Each row is a list of result properties. Each result property is list of property name (based on table headers) and property values.
- The result is compared to the table. Missing and surplus results are marked.
- If the table does not specify all values for the properties, they are filled form the query result

Query: employees hired before	1-Dec-2014		
comp number	emp number	name	hire date
488868	123	Bill Mitchell	1-Nov-2000
488868	456	Jan Novak	1-Aug-2014

Query: employees hired before	1-Dec-2014		
comp number	emp number	name	hire date
488868	123	Bill Mitchell	1-Nov-2000
488868 [missing]	125	Janko Hrasko	30-Nov-2014

Download and Running

1. Download `fitnesse-standalone.jar` from `http://fitnesse.org/FitNesseDownload`
2. Run the fitnesse on default port (8080) from cosole by
   `java -jar fitnesse-standalone.jar`
   You can use a different port with `-p portNumber` parameter
3. Access the wiki from the web browser (`http://localhost`)

Task 1

1. Download the `LibraryFitnesse` project from IS
2. Create new test page on the wiki.
3. Copy the contents of the `testPage.txt` (from study materials) to the page.
4. Adjust the `!path` variables to point to your project path. Try to avoid paths with whitespaces. Make sure the provided tests are passing.
5. Create a new decision table and associated fixture for testing the fees for overdue loans. The example headers are:

reader name	book name	loan date	return date	loan type	total overdue fee?

Task 1 Cont.

- There are two types of loans:
  - Short - loan is for 8 hours. Fine for each overdue hour is 10 CZK.
  - Long - loan is for 30 days. Fine for each overdue day is 5 CZK. Additionally, each overdue week (7 days) is further fined with 20 CZK.
- The test should contain at least 5-6 test cases (rows)
- Use the `StaticLibrary` singleton in order to access the books and readers from previous tables.
- In the fixture, use only the public `Library` methods
- In test cases, assume that given book can be loaned only to one person at given time.
- The timestamp string is in the format `dd/mm/yyyy hh:mm`

Task 2

1. Use the following table to initialize the loans in the library system. Implement the related fixture class. Additionally, use it to delete the loans from the previous tests.

Book Loan Setup				
reader name	book name	loan date	return date	loan type
Jan Novak	Book 1	01/01/2015 12:00		long
Janko Hrasko	Book 2	01/01/2015 08:00	01/01/2015 16:00	short
Janko Hrasko	Book 2	05/01/2015 08:00		short

2. Write a Query table and fixture that should verify the loans that are still open at 5/1/2015 00:00.
3. The table should explicitly specify only the loan start time, the reader and book names should be filled by the fixture.
4. You can use the `Arrays.asList()` method to build the resulting lists.

Further Reading

- The FitNesse documentation is also available offline through the downloaded wiki server.
- `http://www.fitnesse.org/FitNesse.UserGuide.WritingAcceptanceTests.SliM`
- `http://schuchert.wikispaces.com/FitNesse.Tutorials`