SLIM and the future of FitNesse

Gojko Adzic
http://gojko.net
gojko@gojko.com
http://twitter.com/gojkoadzic

Is FIT dead?

- FIT/FitNesse were “The acceptance testing toolkit”
- Java FIT has not been developed for a while
- Lots of differences between implementations in .NET, Python, Java
- FitNesse went through a few years of stagnation
- Other tools (xSpec) and ideas emerging

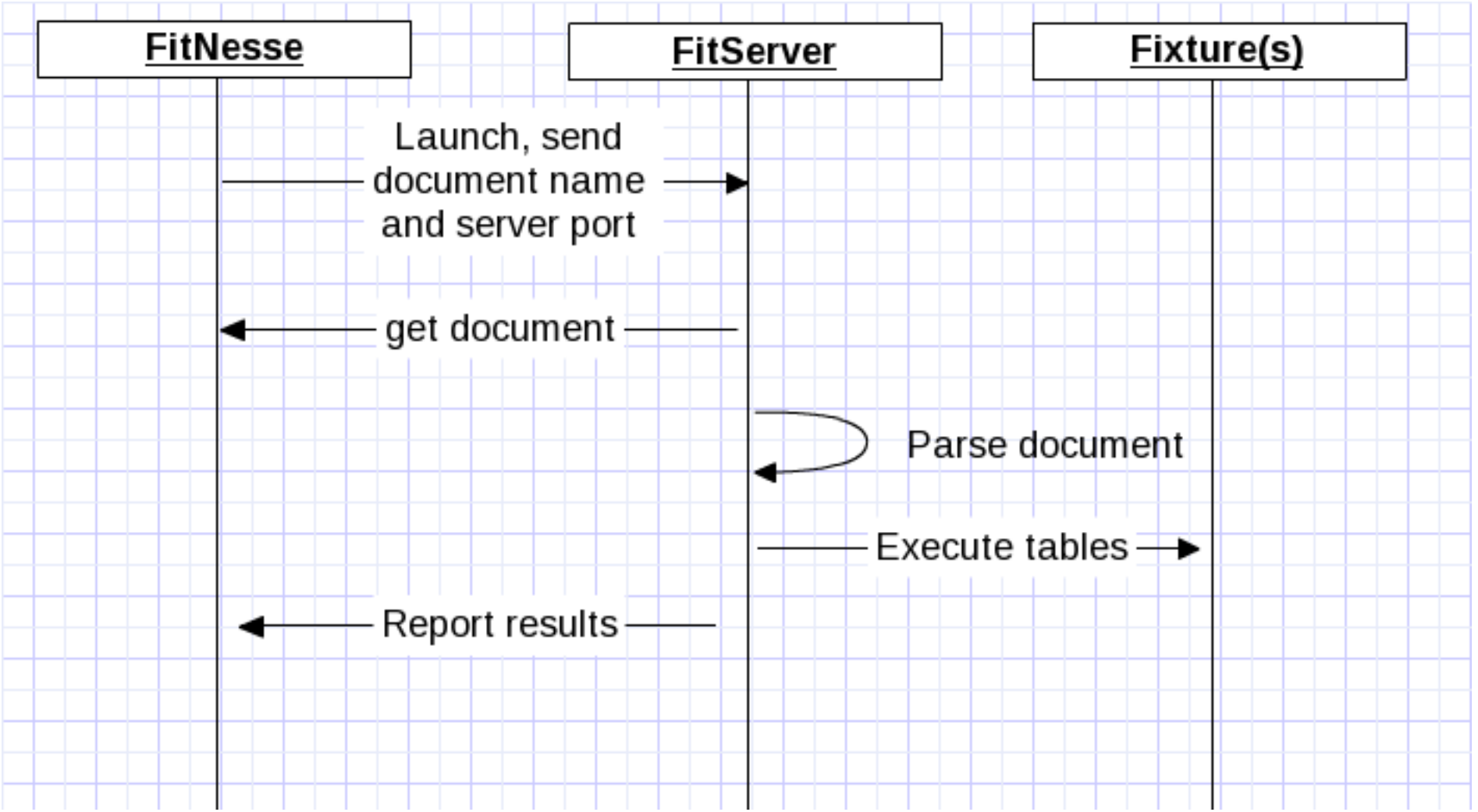
FitNesse fights back

- OM finally showing interest in updating it
- Several releases over the last few months, major updates
- Move to GitHub, project restructured
- Version control, new widgets
- SLIM

So what is SLIM?

- A new test runner
- No dependencies on FIT
  - So no GPL!
- Promises to bring more compatibility and easier platform porting

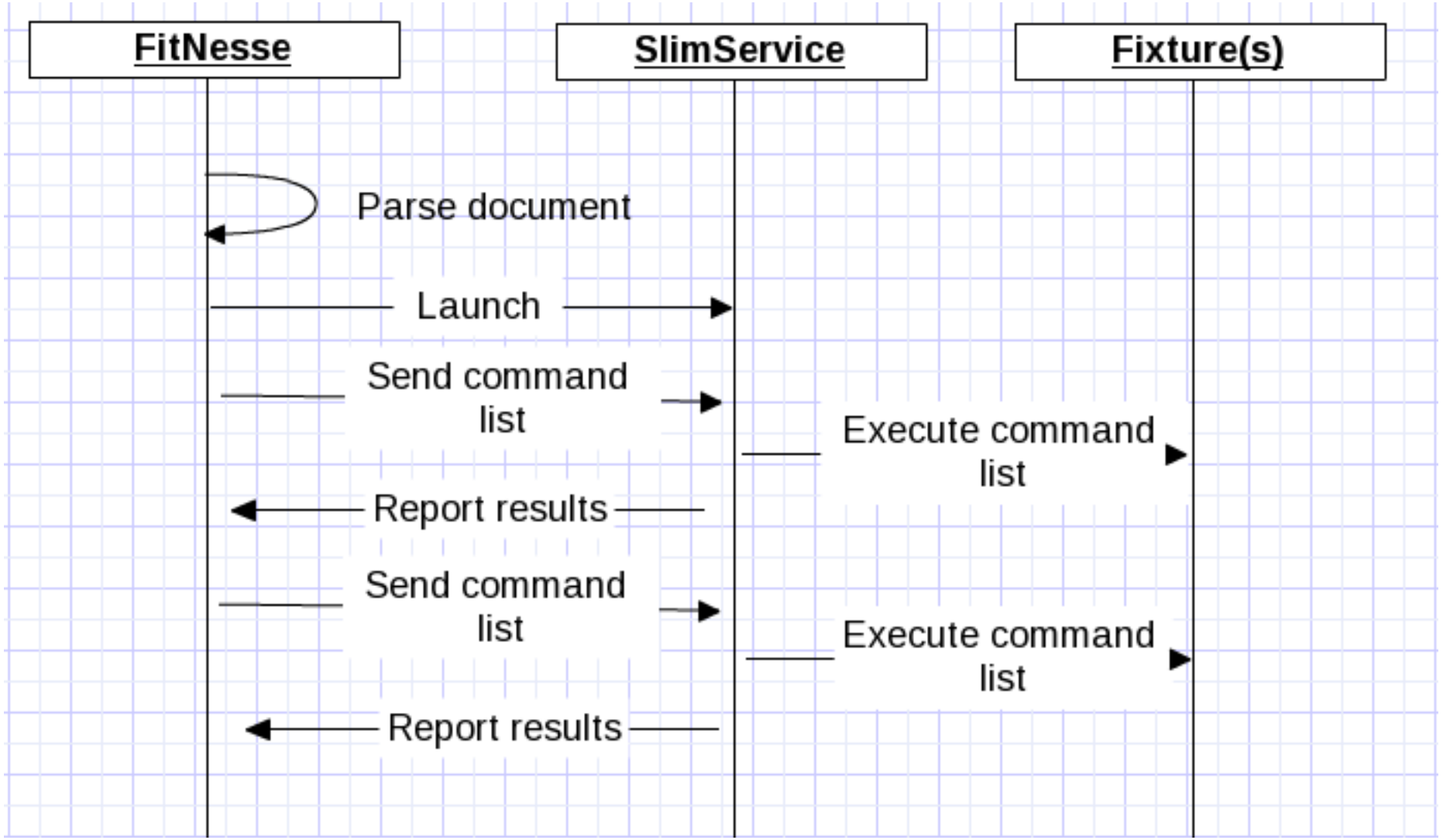
FIT inside FitNesse



FIT inside FitNesse

- FitNesse is responsible to render the page
  - Includes
  - Markup variables
- FitServer is responsible for parsing
  - Parsing systems, syntax
  - Cell handlers
  - Symbols
- Fixtures responsible for executing Parse trees
  - List? Calculation? Free form?
  - Interpreting results

Slim inside FitNesse



Slim inside FitNesse

- FitNesse responsible for parsing and execution
  - Common syntax
  - Test control (calculation? list? query? free form?)
- SlimService responsible for interpreting commands
  - Find fixture
  - command execution
- Fixtures responsible for implementation
- FitNesse responsible for interpreting results

The promise of Slim

- A lot of the work delegated to FitNesse, so easier to build Slim ports
- Common syntax, test control etc across ports
- Common fixture types
- Works on commands, not on tables, so possibly non tabular formats

Turning it on

- !define TEST_SYSTEM {slim}
  - suite/root page
  - You can still use imports
  - No “Fixture” class, use pojos

Decision table (aka Slim Column fixture)

Set Up: *.SlimTest.SetUp* *(edit)* *Expand*

Import
slimtest

Concatenate Strings		
First	Second	concatenate?
hello	world	hello world
so long and thanks	for all the fish	so long and thanks for all the fish

DT code

```
1  package slimtest;
2
3  public class ConcatenateStrings {
4      public String first;
5      public String second;
6      public String concatenate(){
7          return first+" "+second;
8      }
9      public String getFirst() {
10         return first;
11     }
12     public void setFirst(String first) {
13         this.first = first;
14     }
15     public String getSecond() {
16         return second;
17     }
18     public void setSecond(String second) {
19         this.second = second;
20     }
21 }
22
```

- Looks as a column fixture
- Works as a column fixture
- Huge code differences!
  - No inheritance
  - JavaBean getters/setters

Script table (aka Slim DoFixture)

script	create players				
player registers with name	Mike	postcode	12345	and balance	120.01
check me	5				
check	get double	3	is	6	
player registers with name	Tom	postcode	181818	and balance	20.01

Script table

```
1  package slimtest;
2
3  import java.math.BigDecimal;
4
5  public class CreatePlayers {
6     static{
7        fitnesse.slim.Slim.addConverter(
8              BigDecimal.class, new BigDecimalConverter());
9     }
10    public void playerRegistersWithNamePostcodeAndBalance(
11          String name, String postCode, BigDecimal balance){
12       Player.addPlayer(name, postCode, balance.doubleValue());
13    }
14    public boolean checkMe(int x){
15       return x==5;
16    }
17    public int getDoubleIs(int x){
18       return x*2;
19    }
20
21 }
22
```

- Similar to DoFixture method naming
- boolean methods are tests
- Keywords similar to DoFixture

Script table keywords

- Check/Check not – value test pass/fail
- Ensure/Reject - bool test pass/fail
- Note - comment
- Show – display result
- Start – sets up a different system under test

- Optional constructor arguments after class name

Query table (aka Slim row/array fixture)

query:list players		
name	post code	balance
Mike	12345	120.01
Tom	181818	20.01

Query table

```
1  package slimtest;
2
3  import java.util.ArrayList;
6
7  public class ListPlayers {
8      public List<Object> query(){
9  //      can't do this!
10 //      return new ArrayList<Object>(Player.players);
11         ArrayList<Object> objects=
12                 new ArrayList<Object>();
13         for(Player p:Player.players){
14             objects.add(list(list("name",p.name),
15                     list("balance",p.balance),
16                     list("post code",p.postCode)));
17         }
18         return objects;
19     }
20 }
21
```

- Mandatory query method
- List of list of lists of properties!!!
- Yikes!
- Optional constructor argument in table

Symbols

- $*name*= sets the symbol
- $*name uses the symbol*
- $*name*= also in script tables as first cell

Concatenate Strings		
first	second	concatenate?
hello	world	$sentence=
so long	and thanks	$SOLONGANDTHANKS=
$SOLONGANDTHANKS for all the	fish	so long and thanks for all the fish
hey and	hello world	hey and $sentence

Scenario tables (macros/scripts)

scenario	Create Three Players With postcode	postcode			
player registers with name	Mike	postcode	@postcode	and balance	100
player registers with name	Tom	postcode	@postcode	and balance	200
player registers with name	John	postcode	@postcode	and balance	300

script	Create Players
Create Three Players With Postcode	XX1YY2

Scenario tables

- Allow you to reuse fixtures and script with FitNesse
  - Not sure that I'd want to do this, but people often ask for it
- Arguments start with @
- Scenarios can be used within script or decision tables

Conclusions

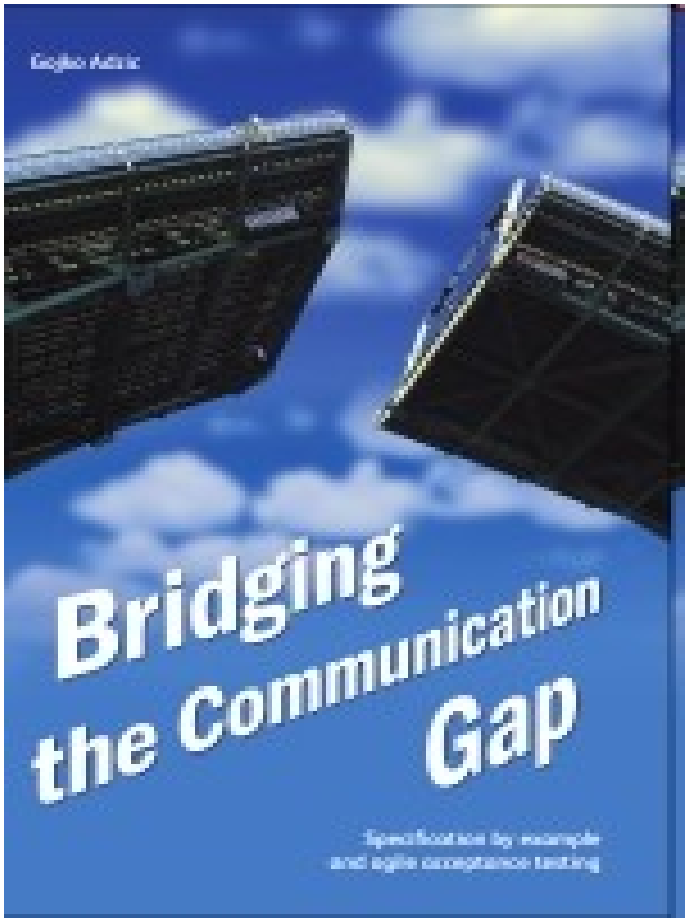
- I don't use it yet
  - FIT much easier to program
    - Lists are a bit too raw
    - Implicit interface dependency – works on pojo but imposes constraints!
  - Less features than .NET/Python FIT
  - Don't really like all the technical stuff in tests
- Interesting for the future, especially if you plan to use FitNesse
  - Will become more feature rich eventually
  - Waiting for “slimlibrary”

Trinidad

- In-process test runner for FIT and Slim
- Works from FitNesse wiki files, but without the server
- Junit/Maven integration
- Debug fixtures from your IDE
- Java only at the moment
- .NET version planned
- http://fitnesse.info/trinidad

Bridging the Communication Gap

- learn how to improve communication between business people and software implementation teams
- find out how to build a shared and consistent understanding of the domain in your team
- learn how to apply agile acceptance testing to produce software genuinely fit for purpose
- discover how agile acceptance testing affects your work whether you are a programmer, business analyst or a tester
- learn how to build in quality into software projects from the start, rather than control it later



http://www.acceptancetesting.info