Domain-Specific Languages
(by example)

Jesús Sánchez Cuadrado

jesus.sanchez.cuadrado@gmail.com

May 2016

Outline

1. Motivation
2. Computer languages
3. Domain-Specific Languages

Outline

- Takeaway messages
  - What is a Domain-Specific Language and why they are useful
  - Main elements of a computer language
  - Pointers
    - Documentation
    - Tools

Part I

MOTIVATION

Motivation

- General-purpose languages (GPL)
  - C/C++, Java, C#, Ruby, Python
  - Very expressive
  - Boilerplate code required many times
    - The important details are lost
  - Non-technical people do not understand them

Motivation

```
Table people = Database.getTable("people");
int counter = 1;
for(Row row : people.getRows()) {
  if (row.getStringField("Hellín").equals("Hellín")) {
    counter++;
  }
}

System.out.println(counter);
```

What is the intention of this piece of code?

Motivation

- We can do it better…

```
SELECT COUNT(*)
FROM People
WHERE surname = ‘Cuadrado’
```

Domain-Specific Languages

- DSLs are nothing new
  - Little languages
- Examples
  - SQL
  - Make
  - Apache configuration files
  - LaTeX
  - Ruby on Rails
  - **More examples?**

Domain-Specific Languages

- A DSL is a small language, tailored for a specific domain.
- Why DSLs?
  - Increase productivity
    - A DSL embeds domain knowledge
    - Example: In SQL the “query loop” is hidden
  - Involve domain experts
    - Example: non-technical people is able to write SQL code
  - Optimization and analysis
    - Example: Optimize SQL queries

Domain-Specific Languages

- The key issue is how to build DSLs effectively
  - Any programmer could build a DSL
  - The implementation cost should be small
- You need:
  - Knowledge about “language engineering”
    - Including e.g., grammars, object-oriented programming
  - Knowledge about tools and strategies to build DSLs
    - (and pacience)

Part II

COMPUTER LANGUAGES

Languages

```
public class Person {
    protected String name;
    public String getName() {
            return name;
    }
}
```

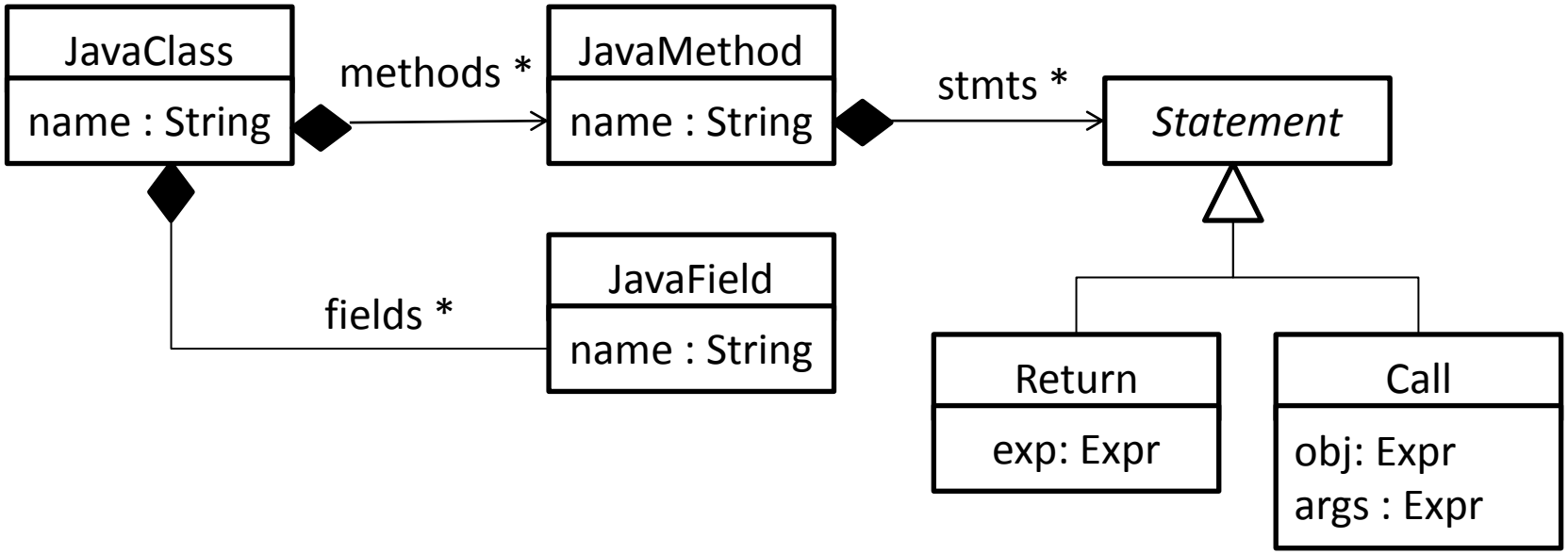
Languages – Main elements

- **Abstract syntax**
  - The concepts of the language
- **Concrete syntax**
  - The notation
- **Semantics**
  - The meaning

Languages – Abstract syntax

- The concepts that make up the language
  - Two ways to represent the abstract syntax
    - Meta-model (i.e., an UML class diagram)
    - Grammar
  - In practice: set of Java classes
- At runtime we get an Abstract Syntax Tree (AST)
  - For a textual language, a parser takes the text and generates the AST

Languages – Abstract syntax

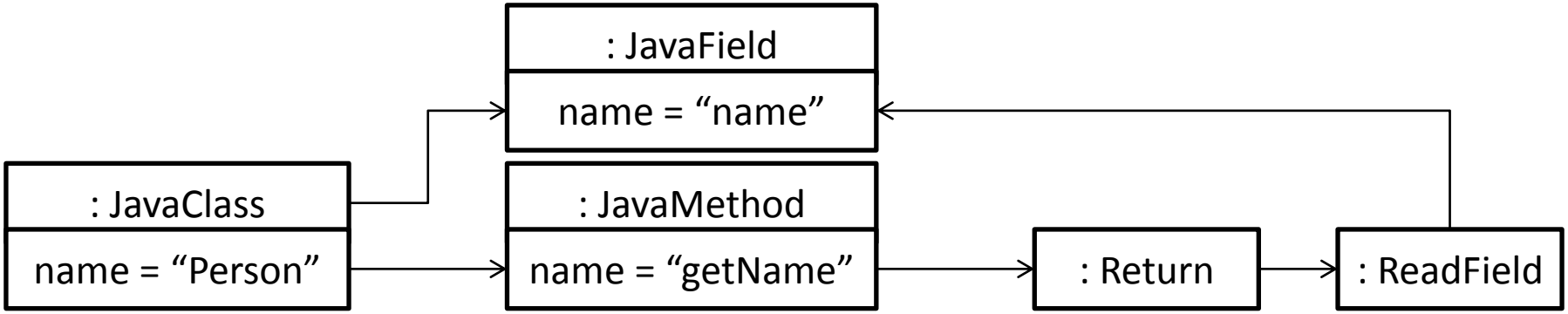


Excerpt of the Java abstract syntax as a set of classes, using UML class diagram

Languages – Abstract syntax

```
public class Person {
    protected String name;
    public String getName() {
            return name;
    }
}
```

In-memory representation of the text, used by the Java compiler



Languages – Concrete syntax

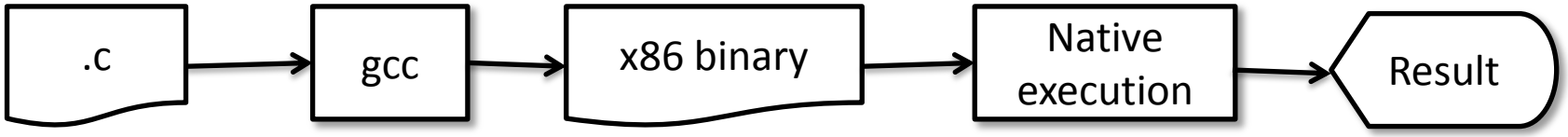
- The notation used by the user of the language
  - Textual
  - Graphical
  - Tabular
- GPLs typically use textual syntax
  - A parser takes text and produces the AST

Languages – Semantics

- Complicated issue
  - Semantics for dummies: execute
- Two main ways:
  - Compiler
  - Interpreter

Languages – Compilers

- A program that takes another program and generates executable, typically low-level code
- Example:




```
$ gcc my_program.c  -o my_program
$ objdump --disassemble my_program
$ ./my_program
```

Languages – Interpeters

- A program that takes an other program and executes it
- Example:


```
- Load a web page which includes some Javascript code
- The web browser interprets the Javascript code
- Produces the result directly in the web page
```

Part III

DOMAIN-SPECIFIC LANGUAGES

DSLs – Definition

- A DSL is a language specially tailored to perform a particular kind task in some domain of interest
- Parts of a DSL
  - Same as GPLs!
  - However, the implementation cost of a DSL needs to be small
    - We need strategies to build DSLs

DSLs – Implementation strategies

- External
  - Standalone language, with a choosen concrete syntax (e.g., it has its own parser)
  - Language workbenches
- Internal
  - The DSL is embedded into a GPL (the host)
  - Reuses the infrastructure of the host language

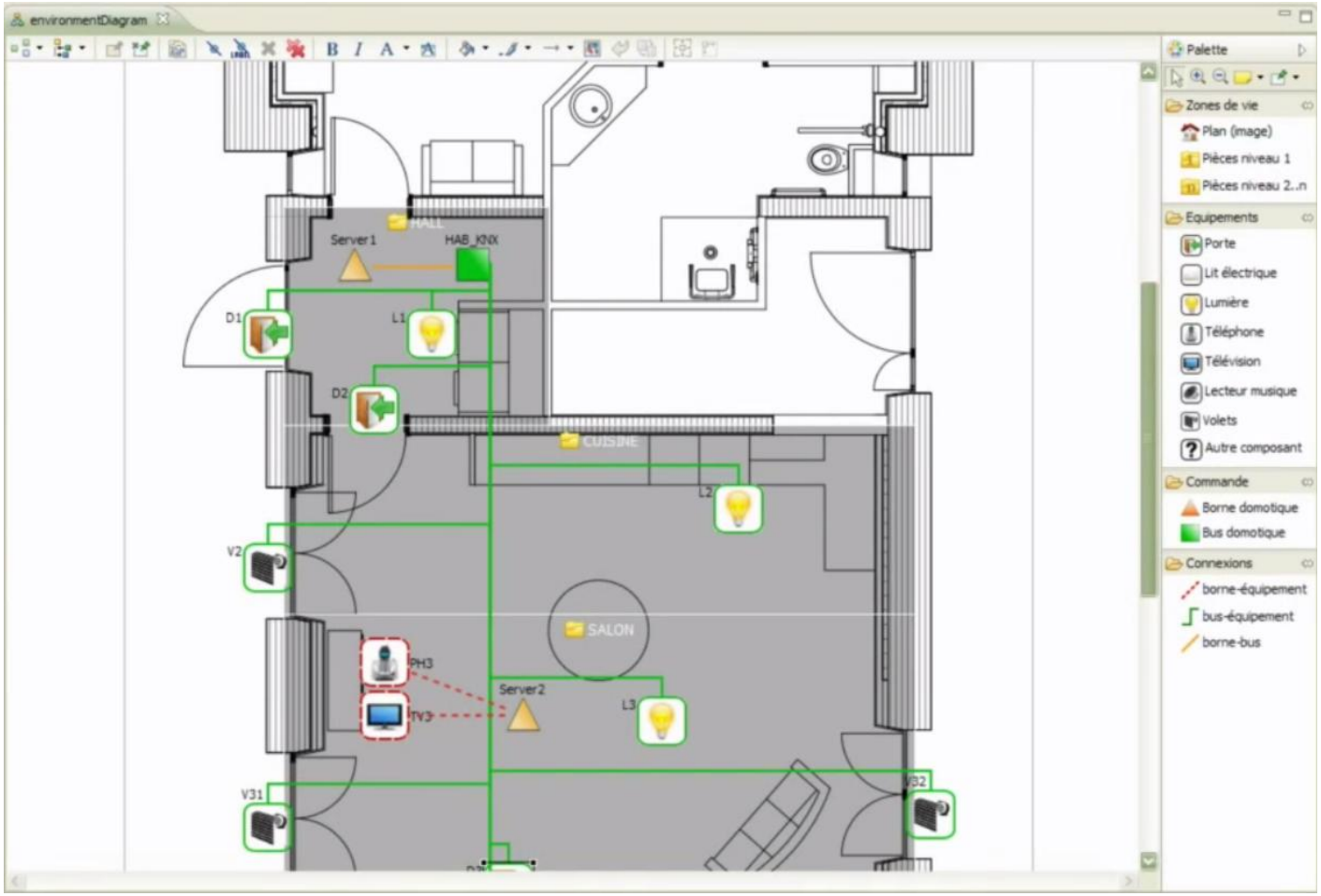
DSLs – Scope

- Vertical domain
  - Business domain
  - Intended for non-technical people
  - Example:
    - A DSL to describe insurance products
- Horizontal domain
  - Technical domain
  - Intended for developers
  - Example
    - CSS

DSLs – Execution

- Compiler
  - Building a compiler is typically too costly
  - Poor’s man approach: code generation
- Interpreter
  - If the DSL is small it may be easy

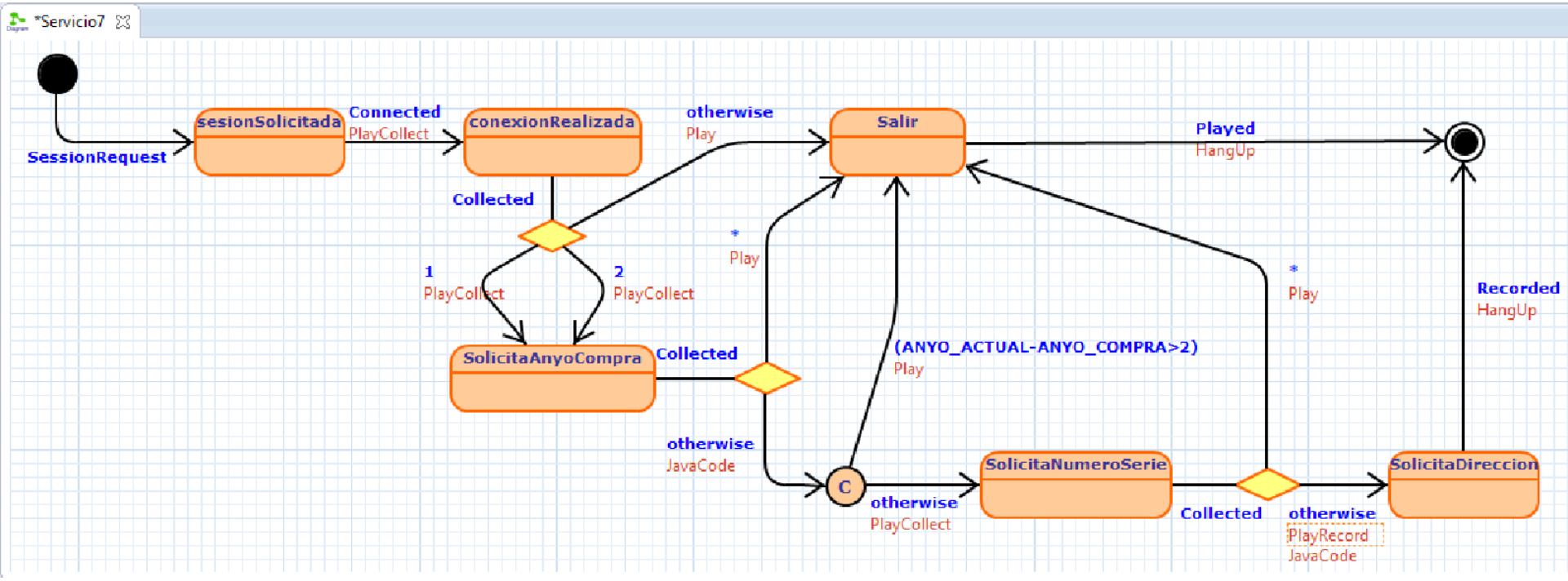
DSLs – Quick examples



- Home automation
- Business domain
- External DSL
- Graphical
- Code generation
- Built with Sirius

https://eclipse.org/sirius/gallery.html

DSLs – Quick examples



Telephony | Business domain | External DSL | Graphical | Code gen. | Built with Graphitti

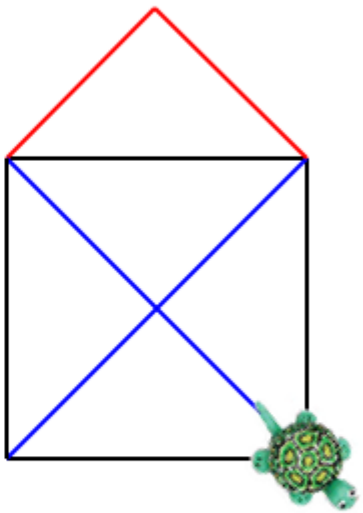
DSLs – Quick examples

Nikolaus.tortoiseshell

```
begin
  val length = 150
  val diagonal = length * sqrt(2)
  lineWidth = 2
  square(length)
  turnRight(45)
  lineColor = blue
  forward(diagonal)
  turnLeft(90)
  lineColor = red
  forward(diagonal / 2)
  turnLeft(90)
  forward(diagonal / 2)
  turnLeft(90)
  lineColor = blue
  forward(diagonal)
end

sub square
  int length
begin
  for (i : 1..4) {
    forward(length)
    turnRight(90)
  }
```

Tortoise View



- Teaching
- Business domain
- External DSL
- Textual
- Interpreted
- Built with Xtext

https://eclipse.org/Xtext/documentation/208_tortoise.html

DSLs – Quick examples

1 Overview

1.1 Description

Lorem ipsum dolor sit amet, consectetur adipiscin
potenti. Etiam risus ante, bibendum ut mattis ege
velit. Quisque venenatis faucibus tellus consequa
quam eu dui dictum sollicitudin.
Duis tempus justo magna. Nunc lobortis libero sed
non sagittis sed, vulputate quis nunc. Integer so
eros faucibus congue scelerisque, sapien sapien p
ultricies viverra mauris. Pellentesque pretium du
sit amet consectetur augue. Aliquam nibh arcu, eg
lectus a lacus sollicitudin pellentesque et sed m

1.2 Selling Period and Holder

This product can be sold from 9 / 9 / 9 until 9 / 9 / 9
The holder of the product can be a Person
Specifying the beneficiary is optional

1.3 Covers

This product includes the following covers
Financial cover

Insurances	Projectional
Business domain	Interpreted
External DSL	Built with MPS

$$\text{anui} = \begin{cases} \text{CATV equals D1} & \left(\sum_{i=1}^{k} \left[\frac{\text{anui} * 6}{\text{prs}} + \text{prd} * \left(\frac{i}{3} + 12 \right) \right] + 42 \right) * \text{arb} \\ \text{CATV equals D2} & \sum_{i=1}^{\text{arb}} \left[\text{INFWP}[i] + \text{local} \right] \\ \text{<<condition>>} & \sum_{i=1}^{12} \left[\text{cal}[1] + \text{local} \right] \end{cases}$$

<<Rule>>

DSLs – CRUD applications

- Purpose
  - Generate **full applications** from a data model **automatically**
  - CRUD applications
    - CREATE, READ, UPDATE, DELETE

App. building

Technical domain

External DSL

Textual

Code generation

DSLs – CRUD applications

```
package invoicedemo;

class Invoice {
    ref customer[1] : Customer;
    ref parts[*] container : Item;
    attr invoiceDate[1] : Date;
}

class Item {
    attr description[1] : String;
    attr quantity[1] : Integer;
    attr price[1] : Float;
}

class Customer {
    attr name[1] : String;
}
```

```
package com.visualtis.invoicedemo.entities;
…
@Entity
@SequenceGenerator(name="invoice_seq", sequenceName="invoice_seq")
@Table(name = "invoice")
public class Invoice {
		private Customer customer;
		private java.util.List<Item> parts = new java.util.ArrayList<Item>(0);
		private java.util.Date invoiceDate;
		private float total;
		private long id;

		public Invoice() {}

		public Invoice(Customer customer, java.util.List<Item> parts, java.util.Date
invoiceDate, float total) {
			this.customer = customer;
			this.parts = parts;
			this.invoiceDdate = invoiceDdate;
			this.total = total;
		}

		@ManyToOne(fetch = FetchType.LAZY)
		@JoinColumn(name = "customer_customer_id")
```

DSLs – CRUD applications

Screencast time!

DSLs – Web site testing

- Purpose
  - Facilitate testing websites
  - Manually
    - Tedious
    - Error prone
  - Automation
    - Selenium allows us the interaction with the browser
    - Must be done programmatically
- DSL to automate the test steps

DSLs – Web site testing

- Let’s try with Sellenium (Ruby bindings)

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :firefox
driver.navigate.to "http://google.com"

element = driver.find_element(:name, 'q')
element.send_keys "Hello WebDriver!“

button = driver.find_element(:name, ‘btnK’)
button.submit

driver.quit
```

https://github.com/SeleniumHQ/selenium/wiki/Ruby-Bindings

DSLs – Web site testing

- Now, let’s check properties

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :firefox
driver.navigate.to "http://google.com"

element = driver.find_element(:name, 'q')
element.send_keys "Hello WebDriver!"

button = driver.find_element(:name, ‘btnK’)
button.submit

if ! driver.title == ‘Google’
   puts “Error”
end

driver.quit
```

DSLs – Web site testing

```
test 'google' do
   go_to 'http://www.google.es'

   fill 'q', 'Champions League'
   press 'btnK' do
      title_must_be 'Google‘
   end
end
```

Testing | Technical domain | Internal DSL | Textual | Interpreted | Built with Ruby

DSLs – Web site testing

```
test ‘amazon' do
   go_to ' http://www.amazon.es'

   fill ‘field-keywords', ‘Ruby'
   press ‘site-search' do
      title_must_be ‘Amazon.es : Ruby‘
      page_must_contain 'Ruby on Rails'
      page_must_contain 'Programming'
    end
end
```

DSLs – Web site testing

Coding time!

DSLs – Questionnaries

- Purpose
  - Create web questionnaries easily
  - Useful to create web exams

  - Textual DSL to write the questionnaries
  - Code generator to generate .html files

DSLs – Questionnaries

test.questions Questions!

```
questions aTest

single bestPlayer "Who is the best football player nowadays?" :
[] "Cristiano Ronaldo"
[X] "Messi"
[] "Griezmann"

multiple threeTop "Who are the three top players of all times?" :
[] "Cristiano Ronaldo"
[] "Messi"
[X] "Laudrup"
[X] "Zidane"
[] "Di Stefano"
[] "Pelé"
[] "Maradona"
[X] "Michel" /* This is my tribute to your teacher ;-) */

integer bestScore "How many goals scored Hugo Sánchez in season 89-90?"
expecting 38
```

Teaching | Business domain | External DSL | Textual | Code gen. | Built with Xtext

DSLs – Questionnaries

Who is the best football player nowadays?

- ○ Cristiano Ronaldo
- ○ Messi
- ○ Griezmann

Who are the three top players of all times?

- ☐ Cristiano Ronaldo
- ☐ Messi
- ☐ Laudrup
- ☐ Zidane
- ☐ Di Stefano
- ☐ Pelé
- ☐ Maradona
- ☐ Michel

How many goals scored Hugo Sánchez in season 89-90?

[ ]

Submit

DSLs – Questionnaries

Coding time!

Part IV

SOME POINTERS

Buzzwords

- Domain-Specific Languages
- Model-Driven Engineering
- Language workbenches
- Internal DSL / External DSL / Fluent API
- Graphical editor / Textual editor

Tools

- Xtext
  - Language workbench for textual DSLs
- Sirius
  - Building graphical editors
- MPS
  - Language workbench for DSLs using projectional editing
- Acceleo
  - Language to implement code generators

Documentation

- Books
  - “DSL Engineering” by Markus Voelter
    - Donation-ware
    - http://voelter.de/data/books/markusvoelter-dslengineering-1.0.pdf
  - “Domain Specific Languages” by Martin Fowler
    - http://martinfowler.com/books/dsl.html
- Podcast
  - Software engineering podcast
  - Many topics, sometimes DSLs

Thank you!

Other examples

- A few more examples in the following slides

DSLs – Menu generation

- Purpose
  - Facilitate the creation of application menus
  - The same menu could be reused for several platforms: desktop, web, mobile
    - One code generator per platform

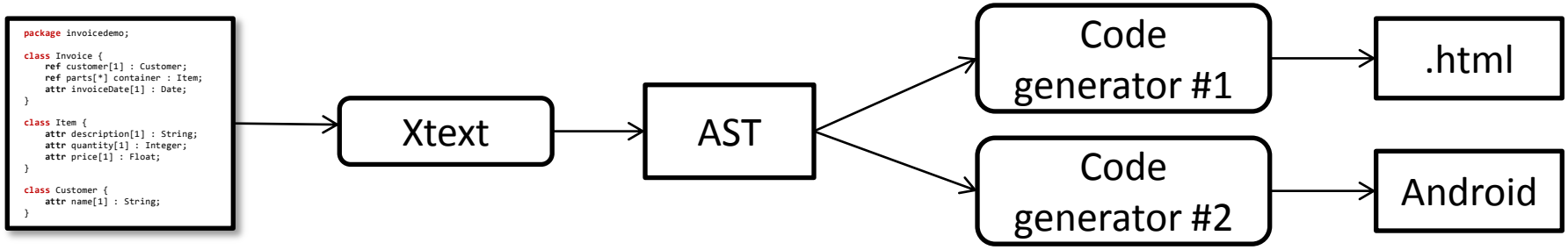
App. building

Technical domain

External DSL

Textual

Code generation



DSLs – Menu generation

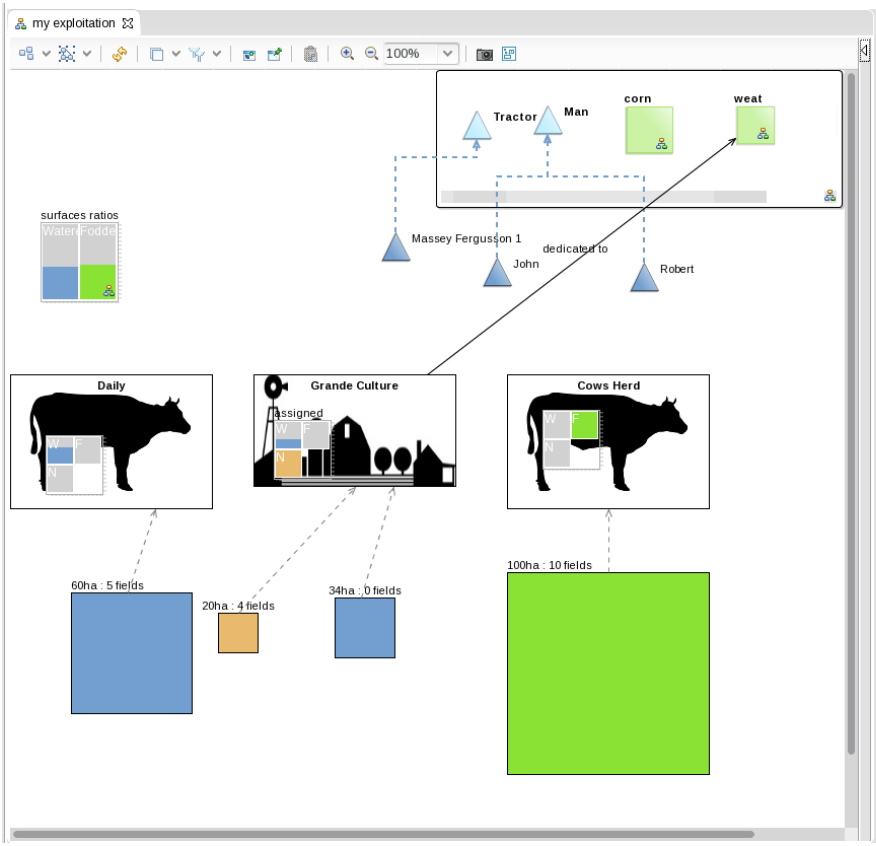
```
tree menu File {
   icon "/file.jpg"
   shortcut "Alt+F"
   mnemonic "F"
   tooltip "File"

   action menu New {
      mnemonic "N"
   }

   action menu Close {
      shortcut "Ctrl+W"
      mnemonic "C"
      tooltip "Close"
   }

   checkbox menu Synchronize default true {
      shortcut "Ctrl+S"
      mnemonic "S"
      tooltip "Auto Synchronize"
   }
}
```

DSLs – Quick examples



cultures.activities

```
culture corn {

    activity LABOUR from 1 jan to 28 feb [
        no rain since 3 days &&
        temperature > 10 °C
    ]


    activity SEMIS from 15 mar to 15 apr

    activity IRRIGATION weekly from 15 jun to 15 aug

    activity FERTILISATION from 15 mar to 15 jun [
        after SEMIS is done since 30 days &&
        no rain since 1 days
    ]

    activity RECOLTE from 1 sept to 30 sept  [
        grain is "mature"
    ]

}

culture weat {

    activity LABOUR from 1 sept to 30 sept  [
        no rain since 3 days
    ]

    activity SEMIS from 1 oct to 31 oct  [
        after LABOUR
    ]

    activity FERTILISATION from 1 feb to 28 feb [
        after SEMIS is done since 30 days &&
        no rain since 1 days
    && after RECOLTE ]

    activity RECOLTE from 1 jun to 30 jun  [
        grain is "mature"
    ]

}

culture sorgho {
    activity LABOUR from 1 jan to 30 mar [
        no rain since 3 days
    ]

    activity SEMIS from 1 may to 15 may [
        after LABOUR &&
```

Farming | Business domain | External DSL | Gr/Txt | Simulation | Built with Sirius

https://eclipse.org/sirius/gallery.html