Maintaining Core Business Knowledge with Language Workbenches

Markus Völter
1 Business Knowledge and Software
2 Language Workbenches
3 JetBrains MPS
4 Examples and Demo
4 Excel
6 The Big Picture
7 Summary
Business Knowledge and Software
What makes a business tick
Distinguishes the business

Business Rules
(Financial) Calculations
Data Structures
Mappings or Queries
Validations
Scientific Processes
Contracts
Processes
UIs
What makes a business tick
distinguishes the business

Contributed *not* by developers

... but is typically implemented in **software**
SO HOW DOES IT GET INTO THE SOFTWARE?

Contributed not by developers

... but is a typically implemented in software
Reality
Reality
Goal!?
Let Business/Domain people contribute directly!

Give them expressive, productive tools to do so!
Expressivity for Core Domain Knowledge

User-Friendly Notation
Great Tool/IDE

Testing

Meaningful Analyses

Synthesis of Software
Language Workbenches
An old idea from the 1970s.

BUT...
Language Workbench
(Martin Fowler, 2004)

Freely define languages and integrate them
Language Workbench
(Martin Fowler, 2004)

language definition
implies
IDE definition

powerful editing testing refactoring debugging groupware

Language Workbench (MarUn Fowler, 2004)
Language Workbench
(Martin Fowler, 2004)

support for “classical” programming and modeling
There’s no difference!
LWBs make Languages Easier

Blur the distinction between programming and modeling.

Several different LWBs exist.

http://languageworkbenches.net
JetBrains MPS
A Language Workbench – a tool for defining, composing and using ecosystems of languages.
Open Source
Apache 2.0
http://jetbrains.com/mps
V 3.2.1 has just been released.
V 3.2.x to be released soon :-}
[Language Workbench]

Comprehensive Support for many aspects of Language Definition.

+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...
[Projectional Editing]

Parsing

- Concrete Syntax
- Abstract Syntax Tree

Projectional Editing

- Concrete Syntax
- Abstract Syntax Tree
[Projectional Editing] Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical
[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

```c
// A documentation comment with references
to @arg(data) and @arg(dataLen)
void aSummingFunction(int8[] data, int8 dataLen) {
    int16 sum;
    for (int8 i = 0; i < dataLen; i++) {
        sum += data[i];
    }
} aSummingFunction (function)
```

Mathematical

```c
double midnight2(int32 a, int32 b, int32 c) {
    return \(-b + \sqrt{b^2 - \sum_{i=1}^{4} a * c}\) / 2 * a;
}
```

Tables

```c
int16 decide(int8 spd, int8 alt) {
    return spd > 0 spd > 100 otherwise 0;
    alt < 0 | 1 | 1 |
    alt == 0 | 10 | 20 |
    alt > 0 | 30 | 40 |
    alt > 100 | 50 | 60 |
} decide (function)
```
[Projectional Editing]
Language Composition

Separate Files

Type System
Transformation
Constraints

In One File

Type System
Transformation
Constraints
Syntax
IDE

50+ extensions to C
10+ extensions to requirements lang.
Projectional Editing provides syntactic flexibility and lang. extensibility.

Usability Issues are mostly solved.

MPS is great, but alternatives exist.
Examples
Insurance Contract Definition: Insurance Math, data types, business/payment rules

End Users: Thousands

Size of System/Data: ??

Language Users: Dozens

Language Developers: 3
Life Product

Example product

1 Overview

1.1 Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam risus ante, bibendum ut mattis egestas velit. Quisque venenatis faucibus tellus consequat quam eu dui dictum sollicitudin. Duis tempus justo magna. Nunc lobortis libero sed non sagittis sed, vulputate quis nunc. Integer scelerisque congue scelerisque, sapien sapien pellentesque pretium du sit amet consectetur a 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
Life Product

Insurance Configuration Tool

1 Overview

1.1 Description


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
Example product
- Financial cover
  - Premium waiver
  - Total permanent disability

Example product
- Financial cover
  - Premium waiver
  - Total permanent disability
  - Death in Accident
  - Natural Death
  - Death by Terrorist Attack
Business Application

Insurance Configuration Tool

Formula Library  FormulaLibrary01 for Rule Set Type IEEE_RST

```
numeric formulaOne = 3 < 4  # 12 + 7 + CATV - \sum_{in = 1}^{6} (in + 9 + \sum_{out = 1}^{in} (in + out))

Testsuite
Show Test Suite

numeric dummy = 3 > 2  # \sum_{index = 1}^{6} (index + 1)

Testsuite
Show Test Suite
```
Business Application
Insurance Configuration Tool

Formula Library  FormulaLibrary01 for Rule Set Type IEEE_RST

numeric formulaOne = 3 < 4  |  12 + 7 + CATV - \sum_{in = 1}^{6} (in + 9) + \sum_{out = 1}^{in} (in + out)

Testsuite
Show Test Suite  Run All  Clear Evaluation

<no name>  CATV
0

Expected  9
Result  0

<<Test Case>>

numeric dummy = 3 > 2  |  \sum_{index = 1}^{6} (index + 1)

Testsuite
Show Test Suite  Run All  Clear Evaluation

test

Input Variables  Output Variables  result
Expected  0
Result  0

<<Test Case>>
Rule Set Type: DemoRuleSetType

Business objects

person : Person

Variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRMI</td>
<td>int</td>
</tr>
<tr>
<td>FR</td>
<td>int</td>
</tr>
<tr>
<td>NN</td>
<td>int</td>
</tr>
<tr>
<td>TT</td>
<td>int</td>
</tr>
<tr>
<td>J</td>
<td>int</td>
</tr>
<tr>
<td>A3</td>
<td>int</td>
</tr>
<tr>
<td>G3</td>
<td>int</td>
</tr>
<tr>
<td>ANUI</td>
<td>int</td>
</tr>
<tr>
<td>X</td>
<td>int</td>
</tr>
</tbody>
</table>

Parent:

<no parent>

Libraries:

Standard
Extra
[Business Application]
Insurance Configuration Tool

Rule Set Type: DemoRuleSetType

Business objects

person : Person

Variables:

PRMI : int
FR : int
NN : int
TT : int
J : int
A3 : int
G3 : int
ANUI : int
X : int

Parent: <no parent>

Libraries:

Standard
Extra

<no business objects>

Variables: <no variables>

Parent: <no parent>

Libraries: <no libraries>
**Rule Set**

*rule set* DemoRuleSet2 *is of type* DemoRuleSetType

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Save</th>
<th>Print</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU0</td>
<td>int</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>CATEG</td>
<td>string</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>CATEG1</td>
<td>double</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

**Premio Calculation**

\[
\text{PREMIO} = \begin{cases} 
A1 > 10 & \rightarrow \text{EU0} \\
\text{<always>} & \rightarrow \text{FLAG} 
\end{cases}
\]

\[
\text{FLAG} = \begin{cases} 
\text{CATEG1 equals 60 or CATEG1 equals 63 or CATEG1 equals 64} & \rightarrow 160 \\
\text{PREMIO equals 0} & \rightarrow 162 \\
\text{CATEG1 > 0 or substr(inga[4], 1, 1) equals "V"} & \rightarrow 163 \\
\text{<always>} & \rightarrow \text{PREMIO + FLAG} 
\end{cases}
\]

\[
\text{PREMIO} = \text{<always>} \rightarrow \text{round(PREMIO * (1 + factacer), 0)}
\]
[Tax/Benefits Application]

Dutch Tax Agency (Evaluation)

Benefits Calculation:

End Users: Hundreds

Size of System/Data: Process data for 6 million citizens, 250 data types, 160 business concepts, 400 message types, 100 business functions, 10 services

Language Users: 15

Language Developers: 2
bloedverwanten: lijst van Burgers zijn gedefinieerd als {
    Een bloedverwant is een Burger die
    bloedverwant in rechte lijn is of die
    bloedverwant in tweede graad zijlijn is
    Einde declaratie
}

bloedverwanten in rechte lijn: lijst van Burgers zijn gedefinieerd als {
    Een bloedverwant in rechte lijn is een Burger die
    nakomeling is of die
    voorouder is
    Einde declaratie
}

bloedverwanten in tweede graad zijlijn: lijst van Burgers zijn gedefinieerd als {
    Een bloedverwant in tweede graad zijlijn is een ouder.kind met
    ouder.kind ongelijk het actuele voorkomen
    Einde declaratie
    ' dus: broer of zus (incl. erkend kind van ouder)
}

bloed- of aanverwanten in rechte lijn: lijst van Burgers zijn gedefinieerd als {
    Een bloed- of aanverwant in rechte lijn is een Burger die
    bloedverwant in rechte lijn is of die
    aanverwant in rechte lijn is
    Einde declaratie
}
Telco Price Calculations: Data structures, math rules, business rules, contract definitions

End Users: --

Size of System/Data: --

Language Users: --

Language Developers: 1
Data Contract

proxy for Customer.Customer

core data entity BillingRegion

| code [key]: | string | references: |
| name:      | string |
| baseMinPrice: | float |
| maxRebateFactor: | float |

entity Contract

| starts: date | customer: Customer 1 | contracts 0..* |
| ends: date   | applicableTariff: Tariff 1 |

entity Tariff

| attributes: | references: |
[Business Application]

Telco Demo App

Contract
- starts: date
- ends: date

Tariff
- applicableTariff: 1

Customer
- customer: 1

BillingRegion
- code [key]: string
- name: string
- baseMinPrice: float
- max RebateFactor: float
### Core Data Default Regions for entity Billing Region

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Base</th>
<th>Min Price</th>
<th>Max</th>
<th>Rebate</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>BW</td>
<td>Baden Württemberg</td>
<td>0.20</td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BY</td>
<td>Bayern</td>
<td>0.20</td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BE</td>
<td>Berlin</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BB</td>
<td>Brandenburg</td>
<td>0.10</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HB</td>
<td>Bremen</td>
<td>0.20</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HH</td>
<td>Hamburg</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HE</td>
<td>Hessen</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MV</td>
<td>Mecklenburg-Vorpommern</td>
<td>0.10</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NI</td>
<td>Niedersachsen</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NW</td>
<td>Nordrhein-Westfalen</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP</td>
<td>Rheinland-Pfalz</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>Saarland</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SN</td>
<td>Sachsen</td>
<td>0.10</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>Sachsen-Anhalt</td>
<td>0.10</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SH</td>
<td>Schleswig-Holstein</td>
<td>0.15</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TH</td>
<td>Thüringen</td>
<td>0.10</td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Calculations CustomerBasic for Customer

import:
- TimeUnits
- BusinessRequirements

Node: isRebated [FlagVar]
Kind: implements
1st Target: Users should be rebated

flag isRecentlyActive := entity.calls.last.startTime.isOlderThan(30 day)
flag isRebated := magic of type boolean[T]

A couple of statistics about the last month's activity

value callsLastMonth := entity.calls.where(!it.startTime.isOlderThan(30 day))
flag activeThisMonth := !callsLastMonth.isEmpty
value devicesUsedLastMonth := callsLastMonth.select(it.sourceDevice).distinct

callsLastMonth.size
value totalPriceLastMonth := \sum_{i=0}^{\text{callsLastMonth.size}} \text{callsLastMonth.at}(i).price.value

value averageCallPriceLastMonth := \frac{\text{totalPriceLastMonth}}{\text{callsLastMonth.size}}

Some random examples.
value example := all[Call].first.customer.calls.first.startTime
Calculations Call Calculations for Call

**Imports:** \( \Sigma \) CustomerBasic

- flag isLocal := magic of type boolean
- flag isLongDistance := magic of type boolean
- flag isRoaming := magic of type boolean

value cust := entity.customer
value pricingFactor := | isLocal | isLongDistance | isRoaming | otherwise 1
| cust isRebated | 0.5 | 0.6 | 0.8 |
| !cust isRebated | 0.8 | 0.9 | 1.0 |

Here is a comment added in the gutter, just as in MS Word.

22/09/14 08:19 (13 s ago) by markusvoelter
**Business Application**

**Telco Demo App**

```plaintext
contract BaseContract specializes <no baseContract> imports: << ... >>
Context Objects:
c: Customer

[final] assign callsThisMonth
callsThisMonth := c.callsLastMonth

assign amountThisMonth
amountThisMonth := 0

[final] store storeBill
c.bills := new MonthlyBill {
    amount := amountThisMonth
}
```
contract FlatrateContract specializes BaseContract

Context Objects:
  c: Customer

conditional assign overrides BaseContract.amountThisMonth as of 16/8/2014
amountThisMonth := c.isRebated | 40 |
| otherwise | 50 |

conditional assign overrides BaseContract.amountThisMonth as of 20/8/2014
amountThisMonth := c.isRebated | 40 |
| otherwise | 60 |
contract FlatrateContract specializes BaseContract

Context Objects:
  c: Customer

[final] assign BaseContract.callsThisMonth
callsThisMonth := c.callsLastMonth

[final] store BaseContract.storeBill
  
c.bills := new MonthlyBill {
    amount := amountThisMonth
  }

conditional assign overrides BaseContract.amountThisMonth as of 16/8/2014[T]
amountThisMonth :=
  | c.is Rebated | 40 |
  | otherwise     | 50 |

conditional assign overrides BaseContract.amountThisMonth as of 20/8/2014[T]
amountThisMonth :=
  | c.is Rebated | 40 |
  | otherwise     | 60 |
**Business Application**

Telco Demo App

```plaintext
rule checkStuff
  given  anything
  when   \[
    \text{the } customer.calls.size \text{ is equal to 10} \]
    and
    \[
    \text{the } call.endTime \text{ is smaller than 20} \]
  then  \[
    \text{set } call.price \text{ to 20} \]
    \[
    \text{execute cancelContract with } customer \]
```
Assessment: UnusedCode
query: unused code
sorted: must be ok: hide ok ones:
last updated: Sep 18, 2014 (3 days ago) by markusvoelter

BaseContract
| storeBill

CustomerBasic
| example
| isMale
| activeThisMonth

FlatrateContract
| FlatrateContract.amountThisMonth
| FlatrateContract.amountThisMonth

total 11, new 0, ok 1
1 | Initially you have no points.
InitialNoPoints /functional: tags

Add Comment  Add Other Data  Add Child Requirement  Add Next Requirement

[ When the game starts, you have no points. ]

workpackage initial scope: 1 responsible: peter prio: 1 effort: 1 days

2 | Once a flight lifts off, you get 100 points
PointsForTakeoff /functional: tags

Add Comment  Add Other Data  Add Child Requirement  Add Next Requirement

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat
enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum
ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum
velit. Quisque venenatis faucibus tellus consequat rhoncus.

3 | The factor of points
PointsFactor /functional: tags

Add Comment  Add Other Data  Add Child Requirement  Add Next Requirement

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat
enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum
ut mattis eget, convallis sit amet nunc.
5 Excel
GOOD

Extremely Widely Used for End-User Programming.

Easy to Understand, spreadsheet metaphor is intuitive.

Direct Manipulation, „Program as State“

Formatting: Nice-Looking „Documents“
BAD

Limited Typing, Limited Modularity

No support for unit tests

Limited support for separating programs from data („templates“)

A lot of duplication (Formulas!)

Fragile (Cross-References, Movement)

Forces users to do everything with tables
BAD
Limited Typing, Limited Modularity

No support for unit tests

Limited support for separating programs from data („templates“)

A lot of duplication (Formulas!)

Fragile (Cross-References, Movement)

Forces users to do everything with tables
What I did last weekend?
Many Notations: Spreadsheet, Functions, DSLs, REPL, ...

Modularity and Visibility

Type Checks, Range Constraints

Cross-Table References.

Spreadsheets are just another language.
ALTERNATIVE

And: you can easily generate efficient code from a spreadsheet „program“.
The Big Picture
System: LWB

Lang 1 → Lang 2

↓

Lang 4

Lang 2 ← Lang 3

⋯

Lang 5

⋯

Lang N
[LOBA]

Business Rules
(Financial) Calculations
Data Structures
Mappings or Queries
Validations
Scientific Processes
Contracts
Processes
UI

Core Business Knowledge
Most business people are able to and want to express themselves precisely!

Let’s give them the tools to do it!
A new Paradigm for Applications

„Language“ interpreted liberally – new styles for languages.
A new Paradigm for Applications

„Language“ interpreted liberally – new styles for languages.
Expressions
  „Code“
  Code Completion
  Error Highlighting
  Version Control
  Refactoring
  Debugging

Tables
  Graphical
  Prose + Code
  Math Notation
  Live Interpretation
  Tree Views
  Visualizations
  Rigid Structures
  Helper Buttons

Language

Form
Business oriented languages are very different from what we have learned about languages for developers. LWBs let you build such languages.

Let’s combines the best of Languages and Form-based Tools.
Language Workbenches enable developers to build really expressive tools for business people to work with data effectively.
Summary
Expressivity for Core Domain Knowledge
Build Language for Domain!
User-Friendly Notation
Great Tool/IDE
You‘ve seen the demos.
Testing
An integrated DSL for testing.
Meaningful Analyses
Types, Consistency, Checking
Synthesis of Software Code Generation.
Fundamentally still manual, no AI. But much better tooling.
Become Language Engineering Experts.
Focus on architecture & technology, engineering
If you have to build a business app, consider using an LWB as the foundation, and recasting the “application” as a set of languages.
open source
[Read & Learn]

DSL Engineering
Designing, Implementing and Using Domain-Specific Languages
Markus Voelter

Generic Tools, Specific Languages
Markus Voelter

The MPS Language Workbench
Fabien Campagne
Volume I

mbeddrr

MPS Meta Programming System
One more thing
LWBs on the Web

Xtext and MPS teams are working on it

It will be months to years until these are finished/robust.

Other web-LWBs (prototypes) exist: http://languageworkbenches.net
Two more more things

Transactional data stays in databases.
Two more more things

We don‘t want to replace the math wiz‘ tools.
Thank you!