Language? Form? IDE? Application?
Towards Language-Oriented Business Apps

Markus Voelter
independent/itemis
voelter@acm.org
www.voelter.de
@markusvoelter

Bernd Kolb
itemis
kolb@itemis.de
www.itemis.de
@berndkolb
1 About mbeddr
2 mbeddr Demo 1
3 JetBrains MPS
4 mbeddr Demo 2
5 Generalization
6 LOBA
7 Conclusions
About mbeddr
About mbeddr

Language Engineering Embedded Software

An extensible collection of integrated languages for embedded software engineering.
About mbeddr

Language Engineering Embedded Software

An IDE + Debugger for all of them
About mbeddr

Language Engineering Embedded Software

Open Source
Eclipse Public License

http://mbeddr.com
http://www.eclipse.org/proposals/technology.mbeddr/

itemis  fortiss  BMW Car IT
mbeddr Demo I
About mbeddr

Built on JetBrains MPS

A Language Workbench
About mbeddr

Built on JetBrains MPS

Open Source
Apache 2.0
http://jetbrains.com/mps
About MPS

Rich Set of Language Aspects

+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...
About MPS

Projectional Editing

**Parsing**

- Concrete Syntax
- Abstract Syntax Tree

**Projection**

- Concrete Syntax
- Abstract Syntax Tree
About MPS

Notational Flexibility

Regular Code/Text

Mathematical

Tables

Graphical
About MPS

Language Composition

Separate Files

- L2
- L1

Type System
Transformation
Constraints

In One File

Type System
Transformation
Constraints
Syntax
Editor/IDE
4
mbeddr Demo II
Generalization
Thought Process

From Data Formats To Languages

Structure, Constraints, Semantics

Data Format + Syntax + IDE

Language
Thought Process

Language Engineering

Language Reuse
Language Modularization
Language Composition

Language Engineering
Thought Process

Language Engineering

Languages

Language Engineering

Text Math Graphics
Tables Symbols Forms

Syntactic Diversity
But does this really work?
Generic Tools, Specific Languages

Ingredients

Specific Languages

Language Engineering

Syntactic Diversity

Languages

Language Workbenches

(we don’t have to reimplement editors and synchronizers)
Generic Tools, Specific Languages

Ingredients

Specific Languages

Support

Languages

Language Engineering

Syntactic Diversity

Generic Tools

Language Workbenches
Language Workbenches

Typical Features

Language Definition, Reuse, Extension, Composition

Mixing Notations

Type Systems, Constraints, Transformation, Interpretation
Language Workbenches

Typical Features

- Goto Definition/Find Usages
- Error Markup/Quick Fixes
- Syntax Highlighting
- Code Completion
- Search/Replace
- Refactoring
- Debugging
- Reporting
- Visualization
- Version Control
Language Workbenches

Typical Features

for any Language!
Language Workbenches act as the foundation for IDEs for any language.
LOBA

why
where we are
what is missing
LOBA  why  where we are  what is missing
Language Workbenches act as the foundation for IDEs for any language.
LOBA: Why

For which kinds of Systems?

Language Workbenches act as the foundation for IDEs for (m)any applications.
many applications?

- Structured or Formalizable
- Mathematical
- Data-Oriented
- Language-y
LOBA: Why

For which kinds of Systems?

many applications?

- Data Models
- Pricing Calculations
- Financial Calculations
- Business Rules
- Contracts
- Highly Structured Requirements
A language may be hiding behind many of these!
LOBA: Why

[Motivation] Languages!

But: users don’t want to be programmers!
LOBA: Why

[Motivation] Languages!

Combine the best of Applications/Forms/UIs and Languages and IDEs.
Applications/Forms/Uis vs. Languages + IDEs

Structure
User Guidance
Tables
Views

Expressions
Complex Structures
Code Completion
Type Checking
Debugging
Refactoring
LOBA: Where we are.

[Notation] Math

\[
\text{int other(a : int, b : int) ==> } a + b + \sum_{i = 1}^{5} i + \prod_{p = 1}^{3} p
\]

\[
\text{local = } A1 \Rightarrow \sum_{i = 1}^{\text{NN}} \left( (D(X + \text{ANUI} + i - 1) - D(X + \text{ANUI} + i)) \cdot \left(1 - \frac{\text{TM18}[i]}{\text{TM17}}\right) \right) \frac{D(X + \text{ANUI})}{\text{D}(X + \text{ANUI})}
\]

\[
\text{int rate(age : int) ==> } 1 + \frac{\text{ANUI} + \frac{\text{age}}{\text{AOPS} - 9}}{4 * 5 + \sum_{i = 8}^{12} i * 8} + \text{in01}
\]
LOBA: Where we are.

[Notation] Math II
LOBA: Where we are.

[Notation] Tables

<table>
<thead>
<tr>
<th>sensorOmega</th>
<th>designOmega</th>
<th>curTime</th>
<th>torque</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0 s</td>
<td>-23 Nm</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.1 s</td>
<td>-38.5 Nm</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.2 s</td>
<td>-47.5 Nm</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.3 s</td>
<td>-47.5 Nm</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.4 s</td>
<td>-36 ±0.001</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.5 s</td>
<td>9 ±0.001</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.6 s</td>
<td>236.25 ±0.001</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.7 s</td>
<td>2023 ±0.001</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.8 s</td>
<td>22093 ±0.001</td>
</tr>
<tr>
<td>5 radps</td>
<td>10 radps</td>
<td>0.9 s</td>
<td>379457.5 ±0.001</td>
</tr>
</tbody>
</table>
### LOBA: Where we are.

**[Notation]** Tables II

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Default</th>
<th>Description</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLB_Time</td>
<td>double</td>
<td>s</td>
<td>0.1</td>
<td>Time in seconds</td>
<td>range 0.00 .. 1.0E16</td>
</tr>
<tr>
<td>Temperature_K</td>
<td>double</td>
<td>K</td>
<td>300.0</td>
<td>Temperature in Kelvin</td>
<td>range 223.0 .. 1773.0</td>
</tr>
<tr>
<td>Temperature_C</td>
<td>double</td>
<td>°C</td>
<td>25.0</td>
<td>Temperature in Celsius</td>
<td>range -50.0 .. 1250.0</td>
</tr>
<tr>
<td>Torque</td>
<td>double</td>
<td>Nm</td>
<td>0.0</td>
<td>Torque in Nm</td>
<td>&lt;no elements&gt;</td>
</tr>
<tr>
<td>Inertia</td>
<td>double</td>
<td>kgm²</td>
<td>0.0</td>
<td>Inertia in kg m square</td>
<td>min 0.00</td>
</tr>
<tr>
<td>motor_speed</td>
<td>double</td>
<td>rad/s</td>
<td>&lt;none&gt;</td>
<td>Motor speed in rad per sec</td>
<td>range 0.00 .. 100000.0</td>
</tr>
<tr>
<td>shaft_speed</td>
<td>double</td>
<td>rad/s</td>
<td>&lt;none&gt;</td>
<td>Output Shaft Speed</td>
<td>range -20000.0 .. 20000.0</td>
</tr>
<tr>
<td>motor_power</td>
<td>double</td>
<td>W</td>
<td>&lt;none&gt;</td>
<td>Motor power in Watts</td>
<td>range -100000.0 .. 100000.0</td>
</tr>
<tr>
<td>coolant_flowrate</td>
<td>double</td>
<td>m³/s</td>
<td>&lt;none&gt;</td>
<td>Coolant volume flow rate</td>
<td>range 0.0 .. 3.0</td>
</tr>
</tbody>
</table>
LOBA: Where we are.

[Notation] Tables III
LOBA: Where we are.

[Notation] Graphical

```
compositeblock Experiment [ double input2 ] => [ double result
                                 double result2
                                 double max
]
parameters { double value1; }
```

Diagram:

- `input2` connected to `adder2` with `add`
- `adder2` connected to `c1` with `c`
- `c1` connected to `adder` with `add`
- `adder` connected to `result`
- `result` connected to `max`
- `result2` connected to `max`
4.1 Price Depends on Country and Price Group

The price of the phone call depends on a number of factors. Among them are the #country and the #pricegroup.

The actual #actMinPrice is computed from the #baseMinPrice with the following equation, the #pricefactor is determined by the table below:

#(actMinPrice = baseMinPrice * priceFactor / 100).

<table>
<thead>
<tr>
<th></th>
<th>Germany</th>
<th>Italy</th>
<th>Spain</th>
<th>GreatBritain</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLATINUM</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>GOLD</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>SILVER</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
**Rule Set Type** DemoRuleSetType

<table>
<thead>
<tr>
<th>Business objects</th>
<th>Variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>person: Person</td>
<td>PRMI: int</td>
</tr>
<tr>
<td>policy: Policy</td>
<td>FR: int</td>
</tr>
<tr>
<td></td>
<td>NN: int</td>
</tr>
<tr>
<td></td>
<td>TT: int</td>
</tr>
<tr>
<td></td>
<td>J: int</td>
</tr>
<tr>
<td></td>
<td>A3: int</td>
</tr>
<tr>
<td></td>
<td>G3: int</td>
</tr>
<tr>
<td></td>
<td>ANUI: int</td>
</tr>
<tr>
<td></td>
<td>X: int</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parent</th>
<th>Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;no parent&gt;</td>
<td>Standard</td>
</tr>
<tr>
<td></td>
<td>Extra</td>
</tr>
</tbody>
</table>
rule set DemoRuleSet2 is of type DemoRuleSetType

EU0 : int [ save false print false ]
CATEG : string [ save false print false ]
CATEG1 : double [ save true print true ]

PREMIO = [ A1 > 10 => EU0
<always> => FLAG ]

FLAG = [ CATEG1 equals 60 or CATEG1 equals 63 or CATEG1 equals 64 => 160
PREMIO equals 0 => 162
CATEG1 > 0 or substr(inga[4], 1, 1) equals "V" => 163
<always> => PREMIO + FLAG ]

PREMIO = [ <always> => round(PREMIO * (1 + factacer), 0) ]
1. Initially you have no points.
   `InitialNoPoints /functional: tags`
   
   [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`
   
   [Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat
einnm 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`
   
   [Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat
einnm arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum
ut mattis eget, convallis sit amet nunc.]
LOBA: Where we are.

[Guided Editing] Code Completion

calculation MTBF: At some time this had been calculating some kind of mean time between failure. No longer does.

parameters: 
- `int32 t_ds: start of downtime` => `int32`
- `int32 t_us: start of uptime`
- `int32 n_fail: # of failures`

\[
t_{ds} - t_{us}
\]

\[
\text{result} = \text{t_ds} \^ \text{params (m.t.main.req.FlightJudgementRules)}
\]

\[
\text{n_fail} \text{t_us} \^ \text{params (m.t.main.req.FlightJudgementRules)}
\]

tests: MTBF(100, 100, 2) == 35

\[
\sum_{i=1}^{k} \frac{anui \times 6 + prd \times (iii + \frac{iii}{3} + 12)}{arb \times (anui - k)}
\]

\[
\sum_{i=1}^{12} \text{cal} \left[ \begin{array}{c}
\text{product} \\
\text{product}
\end{array} \right]
\]

\[
\text{variables (i2sdemo.com.IEEE_RST)}
\]

\[
\text{variables (i2sdemo.com.IEEE_RST)}
\]

\[
\text{variables (i2sdemo.com.IEEE_RST)}
\]

\[
\text{Product}
\]
Business Apps

[Context Aware] Different Projections

1. Initially you have no points.
   InitialNoPoints /functional: tags

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

3. The factor of points
   PointsFactor /functional: tags

4. Points you get for each trackpoint
   InFlightPoints /functional: tags

   4.1 Price Depends on Country and Price Group
       priceDep /functional: status=accepted, @pricing

   4.2 For each trackpoint where you go more than 100 mps, you get 10 points
       FasterThan100 /functional: tags

   4.3 For each trackpoint where you go more than 200 mps, you get 20 points
       FasterThan200 /functional: tags
exported statemachine FlightAnalyzer initial = beforeFlight {
    in event next(Trackpoint* tp) < no binding >
    in event reset() < no binding >
    out event crashNotification() => raiseAlarm
    readable var int16 points = 0
    state beforeFlight {
        on next [tp->alt == 0 m] => airborne
        exit { points += TAKEOFF; }
    } state beforeFlight
    state airborne { ... }
    state landing {
        on next [tp->speed == 0 mps] => landed
        on next [tp->speed > 0 mps] => landing
        on reset [ ] => beforeFlight
    } state landing
    state landed {
        entry { points += LANDING; }
        on reset [ ] => beforeFlight
    } state landed
    state crashed {
        entry { send crashNotification(); }
    } state crashed
    }

<table>
<thead>
<tr>
<th>State</th>
<th>Transition Conditions</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>beforeFlight</td>
<td>[tp-&gt;alt == 0 m] =&gt; airborne</td>
<td></td>
</tr>
<tr>
<td>airborne</td>
<td>[tp-&gt;speed == 0 mps] =&gt; crashed</td>
<td>=&gt; beforeFlight</td>
</tr>
<tr>
<td></td>
<td>[tp-&gt;speed &gt; 0 mps] =&gt; landing</td>
<td>=&gt; beforeFlight</td>
</tr>
<tr>
<td></td>
<td>[tp-&gt;speed &gt; 200 mps &amp;&amp; tp-&gt;alt == 0 m] =&gt; airborne</td>
<td>=&gt; beforeFlight</td>
</tr>
<tr>
<td></td>
<td>[tp-&gt;speed &gt; 100 mps &amp;&amp; tp-&gt;speed &lt;= 200 mps &amp;&amp; tp-&gt;alt == 0 m] =&gt; airborne</td>
<td>=&gt; beforeFlight</td>
</tr>
<tr>
<td>landing</td>
<td>[tp-&gt;speed == 0 mps] =&gt; landed</td>
<td>=&gt; beforeFlight</td>
</tr>
<tr>
<td>landed</td>
<td>[tp-&gt;speed == 0 mps] =&gt; landing</td>
<td>=&gt; beforeFlight</td>
</tr>
<tr>
<td>crashed</td>
<td>[tp-&gt;speed == 0 mps] =&gt; crashed</td>
<td>=&gt; beforeFlight</td>
</tr>
</tbody>
</table>
LOBA: Where we are.

[Context Aware] Visualization
LOBA: Where we are.

[Context Aware] Visualization II
4.1 Price Depends on Country and Price Group

```java
priceDep /functional: status=accepted, @pricing

The price of the phone call depends on a number of factors. Among them are the #country and the #pricegroup.

The actual #actMinPrice is computed from the #baseMinPrice with the following equation, the #priceFactor is determined by the table below:

#(actMinPrice = baseMinPrice * priceFactor / 100).
```

<table>
<thead>
<tr>
<th></th>
<th>Germany</th>
<th>Italy</th>
<th>Spain</th>
<th>Great Britain</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLATINUM</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>GOLD</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>SILVER</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Error: type double is not a subtype of uint32
LOBA: Where we are.

[Live Code] Interpreted Tests

calculation PointForATrackpoint: This rule computes the points awarded for a Trackpoint. It does so by taking into account the @alt and the @speed passed as arguments.

parameters: [ int16 alt: current altitude of the trackpoint ] => int8
[ int16 speed: current speed of the trackpoint ]

result = BASEPOINTS *

<table>
<thead>
<tr>
<th>alt &gt; 2000</th>
<th>alt &gt; 1000</th>
<th>otherwise 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed &gt; 180</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>speed &gt; 130</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

tests: PointForATrackpoint(500, 100) == 0
PointForATrackpoint(500, 1200) == 0
PointForATrackpoint(1100, 200) == 10
PointForATrackpoint(2100, 140) == 120
PointForATrackpoint(2100, 200) == 300

Error: failed; expected 120, but was 100
calculation PointForATrackpoint: 

This rule computes the points awarded for a Trackpoint. It does so by taking into account the @alt and the @speed passed as arguments.

parameters: 

- int16 alt: current altitude of the trackpoint
- int16 speed: current speed of the trackpoint

=> int8

result = 

\[
\begin{array}{c|c|c}
\text{BASEPOINTS} \cdot & 100 & 10 \\
\hline
10 & \text{true} & \text{false} \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{true} & \text{false} & \text{otherwise 0} \\
\hline
\text{alt > 2000} & 2100 & 2100 \\
\text{alt > 1000} & 140 & 140 \\
\hline
\text{speed > 180} & 30 & 15 \\
\text{speed > 130} & 10 & 20 \\
\hline
\end{array}
\]

tests: 

- PointForATrackpoint(500, 100) == 0
- PointForATrackpoint(500, 1200) == 0
- PointForATrackpoint(1100, 165) == 200
- PointForATrackpoint(2100, 140) == 120
- PointForATrackpoint(2100, 200) == 300
<table>
<thead>
<tr>
<th>Language Workbenches</th>
</tr>
</thead>
<tbody>
<tr>
<td>All the IDE Support We Expect</td>
</tr>
</tbody>
</table>

- Goto Definition/Find Usages
- Error Markup/Quick Fixes
- Syntax Highlighting
- Code Completion
- Search/Replace
- Refactoring
- Debugging
- Reporting
- Visualization
- Version Control
LOBA

where we are

what is missing

why
LOBA: What is missing/Challenges

Apparent Tool Complexity

Too many (too big) menus and buttons
LOBA: What is missing/Challenges

Need for Simplified Version Control

Too many options. Locking? Realtime?
LOBA: What is missing/Challenges

Some Shortcomings in MPS

- Cross-model generation
- Projection of computed collections
- Better Graphical Editing
- Type System Performance
- Some Editor Usability

Adressed by JetBrains in 2014.
Users may not be used to this approach.

Training is important.

Productivity more useful than learnability.
Modularity, Reuse, Inheritance, ...

Users may not know about these things, but they may still be necessary for efficiency reasons.
Applications hide Languages

Limited Tool Support for them

LWBs are useful alternative

Connect Business & IT
Language? Form? IDE? Application?
Towards Language-Oriented Business Apps

THE END.

Markus Voelter
independent/itemis
voelter@acm.org
www.voelter.de
@markusvoelter

Bernd Kolb
itemis
kolb@itemis.de
www.itemis.de
@berndkolb