Domain-Specific Languages in SPLE
Why and How?

Markus Völter
voelter@acm.org
www.voelter.de
@markusvoelter
The Big Picture
Digitalization – What & Why?

So we gave everybody an iPad...

Bringing Products Online
“Computering” Product Development

Accelerate development & maintenance
Simplify expected diversification & customization
Digitalization – How?

People & Culture

IT Infrastructure & Devops

Process, Methodology & Tools
Digitalization – Process, Methodology & Tools

- Business Goals
- Requirements
- Design
- Implementation

Documents → Prototypes

Digitalization – Process, Methodology & Tools

- Business Goals
- Requirements
- Design
- Implementation

Documents → Prototypes
Digitalization – Process, Methodology & Tools

Business Goals

Requirements / Design / Impl

Impl + Non Functional

Automation
Digitalization – Process, Methodology & Tools

Business Goals

Requirements / Design / Impl

Impl + Non Functional

Automation
Digitalization – Process, Methodology & Tools

- Business Goals
- Requirements / Design / Impl
- Impl + Non Functional

Automation

- SPEED
- CUSTOM
Digitalization – Process, Methodology & Tools

Business Goals

Requirements / Design / Impl

Impl + Non Functional

Automation
Digitalization – Process, Methodology & Tools

- Business Goals
- Requirements / Design / Impl
- Impl + Non Functional

Automation
Digitalization – Process, Methodology & Tools

Business Goals

Requirements / Design / Impl

Impl + Non Functional

Domain-Specific Languages
Digitalization – DSLs

Domain-Specific Languages

- Good Abstractions and familiar Notations
- Good Abstractions, Clear Semantics

Test Support & Good Abstractions & Notations

- Verification
- Validation

CI-Server Integration

- Good IDE, with Interpreter

Express the right kind of variability
Configuration vs. Construction
CONFIGURATION

PLATFORM
CONSTRUCTION
COMPARISON

CONFIGURATIVE

Platform = Artifacts
“Selection”
Limited Set of Options
Easy to “assemble” product.

CONSTRUCTIVE

Platform = Tools
“Building”
Unlimited Set of Options.
More effort per Product.
Domain-Specific Languages
A domain-specific language (DSL) is a computer language specialized to a particular application domain.

Abstractions/Metamodel
Notation/Syntax
Analyses + Error Reporting
IDE + other Tools
Model

<abstracts>

Reality

{ ignores details, approximates, emphasizes }
SEPARATION OF CONCERNS

M1  M2  M3

Reality
M1
M2
M3
L1
L2
L3

\[ \text{LM} \exists, \vdash, w, \varphi, \alpha \]
Reality Model

MODEL KINDS

Climate Science
Particle Physics
Social Science
FEM Models

decribe
understand
explain
analyze
predict

Modeling

Manufacturing
Construction
Software Eng

prescribe
derive
manufacture

Model-Driven
Programming is just model-driving using a programming language.

And caring about particular concerns.
Aligning the Semantics of the two Model and Reality is key.
Diverse reasons why you would want to model beyond reducing the coding effort by generating the implementation from more concise models.
Automated Execution is essential to "connect" the models to the actual system.
Modeling vs. Programming
Programming is just model-driving using a programming language.

And caring about particular concerns.
<table>
<thead>
<tr>
<th>High-level, Domain-Specific Concepts</th>
<th>Model-Driven</th>
<th>Programming</th>
<th>21st Century MD</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-definable Abstractions</td>
<td></td>
<td></td>
<td>DSMs</td>
</tr>
<tr>
<td>Focus on Behavior and Algorithms</td>
<td></td>
<td></td>
<td>Language Extension</td>
</tr>
<tr>
<td>Multiple Integrated Languages</td>
<td></td>
<td></td>
<td>Expr., Math, State M.,</td>
</tr>
<tr>
<td>Notational Freedom</td>
<td></td>
<td></td>
<td>Txt, Tables, Math, Diag</td>
</tr>
<tr>
<td>Type Systems (vs. Constraint Checks)</td>
<td></td>
<td></td>
<td>Separated Concerns</td>
</tr>
<tr>
<td>Focus on Technical Concerns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focus on Execution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focus on Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Powerful, productivity-focused IDEs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration of Non-Programmers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language Modularity and Comp'n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example DSLs
Funktionsmodell VKzahlbtgTF

Formale Beschreibung

Funktion: VKzahlbtgTF
Enthaltende Quelldatei: vmsctfa
Produkt-Typ: Produkt-Typen auswählen
PK-Typ: PK-Typ auswählen
Status: Status auswählen

Parameter-Attribute:
- tvk_el_ptr: tvk_el<> E Beschreibung hinzufügen
- buzzbrf: Ganzzahl A Beschreibung hinzufügen
- tech_ptr techptr : Beschreibung hinzufügen

Rückgabe-Typ: Kommazahl
Verwendete VADM-Attribute: ...

Aufgerufene Funktionen:
- VKversartTF (tvk_ptr: tvk_el<> E; tech_ptr techptr ): VERSART

Beschreibung

Berechnet den Zahlbeitrag auf Vertragskomponenten-Ebene zurück

Hilfsvariablen
- vkzb: Kommazahl Beschreibung hinzufügen

Verarbeitungen

<table>
<thead>
<tr>
<th>spk_typ_id</th>
<th>Beschreibung hinzufügen</th>
<th>Bemerkung</th>
</tr>
</thead>
</table>
| PK_TYP_ID.KAPITAL_KONTO | If (:vtrk_zustand = ZUSTAND.BPFL) 
  vkzb = :vtrk_zb 
  End If :vtrk_zustand = ZUSTAND.BPFL | Beschreibung hinzufügen |
| PK_TYP_ID.LV_TARIF | If (:stamm_ptr <> NULL) 
  If (:zustand = ZUSTAND.BPFL) 
  vkzb = :vtzb 
  If (VKversartTF (tvk_el_ptr; tech_ptr) = VERSART.BUZB) 
  buzzbrf = 0 
  End If VKversartTF(tvk_el_ptr; tech_ptr) = VERSART.BUZB 
  End If :zustand = ZUSTAND.BPFL 
  End If :stamm_ptr <> NULL |
| Andernfalls        | Fehler (PK_TYP_NICHTIMPLEMENTIERT)                            | Beschreibung hinzufügen |

return vkzb
**Decision Table**

\[
\text{BpScoreDecisionTable}(\text{sys: bpRange}, \text{dia: bpRange}) =
\]

<table>
<thead>
<tr>
<th>dia</th>
<th>&lt;= 50</th>
<th>[51..90]</th>
<th>[91..95]</th>
<th>[96..100]</th>
<th>[101..109]</th>
<th>&gt;= 110</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys &lt;= 90</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>[91..140]</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>[141..150]</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>[151..160]</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>[161..179]</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>&gt;= 180</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

**Decision Tree**

\[
\text{DiarrheaStoolsDecisionTree}(\text{score: DiarrheaStoolsOverBaseline}, \text{patientHasAnySymptom: boolean}, \text{goToStartBrat: boolean})
\]

```
DiarrheaReco1

patientHasAnySymptom

score >= 7

DiarrheaReco3

score in [4..6]

DiarrheaReco2

goToStartBrat

DiarrheaRecoSBrat

DiarrheaRecoCBrat
```

**Type**

- temperature: number[36,42]\{1\}
- measuredTemp: number[35,43]\{2\}

```
val T_measured: measuredTemp = 42.22
val T_calibrated: temperature = T_measured * 0.93
```

**Error**

```
Error: type number[32.55,39.99]\{4\} is not a subtype of number[36,42]\{1\}
```
D : Kommutationswerte

Ergebnistyp: number[3]
Laufvariable: x
Parameter: i, geschlecht, q

\[ D_x := l_x \times \frac{1}{(1 + i)^x} \]

l : Lebende im Jahr x

Ergebnistyp: number[0]
Laufvariable: x
Parameter: geschlecht, q

\[ l_0 := \text{startwertLebende} \]
\[ l_x := l_{x-1} \times (1 - q.\text{lookup}(x, \text{geschlecht})) \]
state machine Game (initialPlayers: list<party>) {
  var players = initialPlayers
  var pendings = nopendings
  var bids = nobids
  var lastWinner: opt<party> = none

  event requestAccess
  event startVoting
  event voteCandidate(candidate: party, vote: boolean)

  event terminatePendings
  event play
  event offerBid(money: bid)
  event endGame

  fun clearGame(RM) () {
    lastWinner := none
    bids := nobids
  }
}

state playing [senderIs(players)] {
  entry: clearGame(RM(EMPTY))
  state bidding [takeTurns(players|ordered|after 500 boot out)] (initial) {
    on offerBid(money): bids := bids.put(sender->money)
    if [timeInState > 2000] -> finished
  }
  state finished {
    entry: {
      val maxOfferedValue = bids.values.max
      lastWinner := bids.keys.findFirst(bids[it] == maxOfferedValue)
    }
    on endGame -> setup
  }
  exit: clearGame(RM(EMPTY))
}
Unterhaltsvorschuss

Zeitangabe: laufend
Häufigkeit: monatlich einmal
Leistungskontext:
Leistungsart: Leer
Zählart: uvg

Anspruch Beginn: Anfang – Unbegrenzt: junger Mensch.geburtsdatum
Anspruch Ende: 01.01.1800 - 31.12.9999 : \text{min}\{\text{jung}\text{er Mensch}.geburtsdatum + 12 Jahre, \\
\text{datum} + 72 \text{ Monate} - \text{Anzahl Monate mit uvg}\}

Zeitraum für Berechnung: Anfang – Unbegrenzt: \{standardzeitraum, standardzeitraum\}

zweckgebundene Leistung: ☐
dem Grunde nach: ☐

Zeitraumbezogene Daten
nullwerte Anzeigen: boolean = 01.01.1800 - 31.05.2016 : true
01.06.2016 - Unbegrenzt : false
berechnungsart : berechnungsarttyp = 01.01.1800 - 31.12.9999 : dreißigstel

Bezugsojekte:
Attribute: bemerkung : string wird validiert
antragsdatum : Datum
rule TimingPattern_07_ObjectsAcrossRows_Modified_TandA_897_1Iteration

description: 1381R3.ARCHD.0609.TimingPatternLanguage_TestScenarioMap.xlsx

parameters: TimePeriodObjectTypA4

patterns:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>TimePeriodObjectTypA4</th>
<th>TimePeriodObjectTypA6</th>
</tr>
</thead>
<tbody>
<tr>
<td>scenario</td>
<td>&lt; TimePeriodSpecifer2::Duration = 24 Hours &gt;</td>
<td></td>
</tr>
<tr>
<td>scenario</td>
<td>&lt; TimePeriodSpecifer3::Duration = 15 Minutes &gt;</td>
<td></td>
</tr>
</tbody>
</table>

database databaseOneElementAcrossRows

<table>
<thead>
<tr>
<th>Type</th>
<th>Begin</th>
<th>End</th>
<th>Duration</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTimePeriodObjectTypA</td>
<td>500</td>
<td>550</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

database databaseOneAndMoreIterationsHappy

<table>
<thead>
<tr>
<th>Type</th>
<th>Begin</th>
<th>End</th>
<th>Duration</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTimePeriodObjectTypA</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>eTimeSpikeObjectTypA</td>
<td></td>
<td></td>
<td>86000</td>
<td></td>
</tr>
<tr>
<td>eTimePeriodObjectTypA</td>
<td>86020</td>
<td>86030</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Damage potentials</td>
<td>Very low</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>-----</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>Beyond high</td>
<td>Low risk</td>
<td>Low risk</td>
<td>Low risk</td>
<td>Low risk</td>
</tr>
<tr>
<td>High</td>
<td>Low risk</td>
<td>Low risk</td>
<td>Moderate risk</td>
<td>High risk</td>
</tr>
<tr>
<td>Moderate</td>
<td>Low risk</td>
<td>Moderate risk</td>
<td>High risk</td>
<td>High risk</td>
</tr>
<tr>
<td>Very high</td>
<td>Moderate risk</td>
<td>High risk</td>
<td>High risk</td>
<td>High risk</td>
</tr>
</tbody>
</table>

### Required Attack Potentials

<table>
<thead>
<tr>
<th>Beyond high</th>
<th>High</th>
<th>Moderate</th>
<th>Low</th>
<th>Very low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low risk</td>
<td>Low risk</td>
<td>Moderate risk</td>
<td>Low risk</td>
<td>Moderate risk</td>
</tr>
<tr>
<td>Moderate risk</td>
<td>High risk</td>
<td>High risk</td>
<td>Very high risk</td>
<td>Very high risk</td>
</tr>
<tr>
<td>High risk</td>
<td>Very high risk</td>
<td>Very high risk</td>
<td>Very high risk</td>
<td>Very high risk</td>
</tr>
</tbody>
</table>

### S2.16 Confidentiality UpdateRequest

- **Name**: Confidentiality UpdateRequest
- **Title**: Confidentiality UpdateRequest
- **Security Goal Classes**: Confidentiality
- **SUD Elements**: UpdateRequest
- **CRIERI**: DPE
- **Criteria**: SAF, ITA, LAM
- **Depends on**: S2.37, S2.40

### S2.17 Availability UpdateRequest

- **Name**: Availability UpdateRequest
- **Title**: Availability UpdateRequest
- **Security Goal Classes**: Availability
- **SUD Elements**: UpdateRequest
- **CRIERI**: DPE
- **Criteria**: NSG, SAF, S2.38, S2.39

### S2.18 Authenticity UpdateRequest

- **Name**: Authenticity UpdateRequest
- **Title**: Authenticity UpdateRequest
- **Security Goal Classes**: Authenticity
- **SUD Elements**: UpdateRequest
- **CRIERI**: DPE
- **Criteria**: IIS

### S2.19 Confidentiality UpdateContainer

- **Name**: Confidentiality UpdateContainer
- **Title**: Confidentiality UpdateContainer
- **Security Goal Classes**: Confidentiality
- **SUD Elements**: UpdateContainer
- **CRIERI**: DPE
- **Criteria**: SAF, ITA, LAM

### S2.20 Availability UpdateContainer

- **Name**: Availability UpdateContainer
- **Title**: Availability UpdateContainer
- **Security Goal Classes**: Availability
- **SUD Elements**: UpdateContainer
- **CRIERI**: DPE
- **Criteria**: NSG

### S2.21 Authenticity UpdateContainer

- **Name**: Authenticity UpdateContainer
- **Title**: Authenticity UpdateContainer
- **Security Goal Classes**: Authenticity
- **SUD Elements**: UpdateContainer
- **CRIERI**: DPE
- **Criteria**: IIS, SAF, S2.38, S2.39

### S2.22 Confidentiality ConfigurationProfiles

- **Name**: Confidentiality ConfigurationProfiles
- **Title**: Confidentiality ConfigurationProfiles
- **Security Goal Classes**: Confidentiality
- **SUD Elements**: ConfigurationProfiles
- **CRIERI**: DPE
- **Criteria**: SAF, ITA, LAM

### S2.23 Availability ConfigurationProfiles

- **Name**: Availability ConfigurationProfiles
- **Title**: Availability ConfigurationProfiles
- **Security Goal Classes**: Availability
- **SUD Elements**: ConfigurationProfiles
- **CRIERI**: DPE
- **Criteria**: NSG

### S2.24 Authenticity ConfigurationProfiles

- **Name**: Authenticity ConfigurationProfiles
- **Title**: Authenticity ConfigurationProfiles
- **Security Goal Classes**: Authenticity
- **SUD Elements**: ConfigurationProfiles
- **CRIERI**: DPE
- **Criteria**: SAF, S2.38, S2.39

### S2.25 Availability D4g

- **Name**: Availability D4g
- **Title**: Availability D4g
- **Security Goal Classes**: Availability
- **SUD Elements**: D4g
- **CRIERI**: DPE
- **Criteria**: NSG

### S2.26 Authenticity ProfileSelection

- **Name**: Authenticity ProfileSelection
- **Title**: Authenticity ProfileSelection
- **Security Goal Classes**: Authenticity
- **SUD Elements**: ProfileSelection
- **CRIERI**: DPE
- **Criteria**: SAF, S2.38, S2.39

### S2.27 Confidentiality WebsiteData

- **Name**: Confidentiality WebsiteData
- **Title**: Confidentiality WebsiteData
- **Security Goal Classes**: Confidentiality
- **SUD Elements**: WebsiteData
- **CRIERI**: DPE
- **Criteria**: SAF, ITA, LAM

### S2.28 Availability WebsiteData

- **Name**: Availability WebsiteData
- **Title**: Availability WebsiteData
- **Security Goal Classes**: Availability
- **SUD Elements**: WebsiteData
- **CRIERI**: DPE
- **Criteria**: NSG

### S2.29 Authenticity WebsiteData

- **Name**: Authenticity WebsiteData
- **Title**: Authenticity WebsiteData
- **Security Goal Classes**: Authenticity
- **SUD Elements**: WebsiteData
- **CRIERI**: DPE
- **Criteria**: SAF, S2.38, S2.39

### S2.30 Confidentiality GCU

- **Name**: Confidentiality GCU
- **Title**: Confidentiality GCU
- **Security Goal Classes**: Confidentiality
- **SUD Elements**: GCU
- **CRIERI**: LTO

### S2.31 Availability GCU

- **Name**: Availability GCU
- **Title**: Availability GCU
- **Security Goal Classes**: Availability
- **SUD Elements**: GCU
- **CRIERI**: NSG

### S2.32 Integrity GCU

- **Name**: Integrity GCU
- **Title**: Integrity GCU
- **Security Goal Classes**: Integrity
- **SUD Elements**: IIS

### S2.33 Compliance GCU

- **Name**: Compliance GCU
- **Title**: Compliance GCU
- **Security Goal Classes**: Compliance
- **SUD Elements**: Compliance
- **CRIERI**: DPE
- **Criteria**: SAF, ITA, LAM

### S2.34 Availability GCU

- **Name**: Availability GCU
- **Title**: Availability GCU
- **Security Goal Classes**: Availability
- **SUD Elements**: GCU
- **CRIERI**: NSG

### S2.35 Authenticity GCU

- **Name**: Authenticity GCU
- **Title**: Authenticity GCU
- **Security Goal Classes**: Authenticity
- **SUD Elements**: GCU
- **CRIERI**: SAF, S2.38, S2.39

### S2.36 Integrity GCU

- **Name**: Integrity GCU
- **Title**: Integrity GCU
- **Security Goal Classes**: Integrity
- **SUD Elements**: IIS

### S2.37 Confidentiality GCU

- **Name**: Confidentiality GCU
- **Title**: Confidentiality GCU
- **Security Goal Classes**: Confidentiality
- **SUD Elements**: GCU
- **CRIERI**: LTO

### S2.38 Availability GCU

- **Name**: Availability GCU
- **Title**: Availability GCU
- **Security Goal Classes**: Availability
- **SUD Elements**: GCU
- **CRIERI**: NSG

### S2.39 Authenticity GCU

- **Name**: Authenticity GCU
- **Title**: Authenticity GCU
- **Security Goal Classes**: Authenticity
- **SUD Elements**: GCU
- **CRIERI**: SAF, S2.38, S2.39

### S2.40 Integrity GCU

- **Name**: Integrity GCU
- **Title**: Integrity GCU
- **Security Goal Classes**: Integrity
- **SUD Elements**: IIS
public functional component DriveTrain {
    produces SpeedFromEngine
    produces EngineStatus
    produces Gear where it < gearsCount
    consumes RoadConditions
    param int gearsCount
    consumes DrivingCommands
}

```
vector<int16, 3> aVector = \[
\begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix} \times 512;
\]

vector<int16, 3> resultOfCrossProduct = aVector \times aVector;

matrix<int16, 2x3> aMatrix = \[
\begin{bmatrix}
1 + 2 & 2 \times 7 & 42 \\
3 & 51 & 24
\end{bmatrix}
\]

matrix<int16, 3x2> transposedMatrix = aMatrix^T

int32 averageIntArray(int32[] arr, int32 size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum / size;
}

averageIntArray (function)
val salary : TT[currency] = TT | /2017 01 01/  =>  5.000 EUR |
| /2017 05 01/  =>  6.000 EUR |

\[
\begin{align*}
\begin{array}{c}
\begin{array}{c}
 a \\
 a_1 \\
 a_2 \\
 a_3 \\
\end{array} \\
+ s = \\
\begin{array}{c}
 a_1 + s \\
 a_2 + s \\
 a_3 + s \\
\end{array} \\
\end{array} \\
\begin{array}{c}
\begin{array}{c}
 a \\
 a_1 \\
 a_2 \\
 a_3 \\
\end{array} \\
+ b \\
\begin{array}{c}
 b_1 \\
 b_2 \\
\end{array} = \\
\begin{array}{c}
 a_1 + b_1 \\
 a_2 + b_1 \\
 a_2 + b_2 \\
 a_3 + b_2 \\
\end{array}
\end{align*}
\]
calculation for Tax
depends Salary as s
  valid from /2017 01 01/
calculate [monthly] {
  ...
}

calculation for Tax
depends Salary as s
  valid from /2017 01 01/
  SomeOtherThing as t
  if ctx.employment.person.homeAddress.state == BW
calculate [monthly] {
  ..
}

result data [monthly] Salary {
  employment -> Employment
  amount : currency
}

result data [monthly] Salary from /2017 10 01/ {
  employment -> Employment
  amount : currency
  taxFree : boolean
}
More than Language

GOOD

Language
- Abstractions
- Notations

Great IDE
- Syntax Coloring
- Code Completion
- Goto Definition
- Collaboration
- Migration

Analyses
- Relevant
- Good Errors
- Type System!

GREAT

Refactorings
- Aligned with Processes

Testing
- Write Tests
- Run them in IDE
- Report Back

Debuggers
- Animate Execution
- Simulators

GOOD

GREAT
Language
Workbenches
DSL Process

1. Understand Domain
2. Design & Implement Language + IDE
3. Validate with Users
4. Generator/Interpreter
5. Deploy / Use
DSL Process – Meta?

Design & Implement Language + IDE

Generator/Interpreter
DSLs are the Tools

Design & Implement Language + IDE

Generator/Interpreter

Language Workbench
DSLs are the Tools

Tools for efficiently building ecosystems of languages and their IDEs

Language Workbench
DSLs are the Tools

Language Workbench
MPS Language Workbench

Language Workbench
Open Source, by JetBrains
Very Powerful
Used for years by itemis and others
Vast Experience
MPS Language Workbench

- Structure: Concepts, Properties, Inheritance, Relationships
- Editor: Projection Rules, Side Transformations, Intentions
- Type System: Typing Rules, Type Checks, Other Validations
- Transformations: Reduction Rules, Weaving Rules, Transformation Priorities
- Constraints: Scopes, Usage Restrictions, Property Value Limitations

+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...
MPS: Notational Freedom

100% crucial for acceptance in domain!

Rabbit Hole

Modeling is more than diagrams!
MPS: Language Composition

SPLE on Language Level!
Other Language Workbenches

{S} spoofax
xttext
Rascal
The Whole Platform
MontiCore

TU Delft
itemis/Typefox
CWI Amsterdam
Solmi/Persiani
Metacase
RWTH Aachen
The Big Picture (Revisited)
Modeling vs. Programming

Modeling ≠ Programming

Modeling:
- Performance
- Scalability
- Robustness
- Deployment

Programming:
- Formulas, Rules
- Data Structures
- Tables
- Values
The concepts in the DSL are well aligned with the domain.

This makes learning the language much simpler.

And there is tool support that helps create valid models.
Meaningful Validation

More Efficient Development

Integrate Domain Experts

Decouple from Technology
MD* and Agile is in Conflict. 

Project 1
Language Development

Depend on, use
System Development

Later:

Project 1
Language Development

Dep’d on, use
System Development

Project 2

...
MD* and Agile is in Conflict.

Manage like any other intra-project dependency.

Evolution of client code is easier than for F/L/P because of migration support!
MD* and Agile is in Conflict.

Manage like any other 3rd party dependency:
- Development Roadmap
- Issue Tracker
- Release Notes

Later:

Project 1
Language Development
- Framework
- Library
- Platform

Dep’d on, use

Project 2
System Development
...
MD* and Agile is in Conflict.

Models and DSLs are an Enabler for Agility: Integration of Domain Experts „Living“ Requirements Decoupled Fachlichkeit & Technik
MD* and Agile is in Conflict.

Leading LWBs are so productive, you can literally sit with the domain experts and interactively prototype languages (and then clean up later)

I’ve looked at the implementation of the language in MPS, but I didn’t find much. Is this all there is? Where’s the magic?

[Customer]
Expressing Configurative Variability with DSLs
Write each product from scratch

Languages are optimized for the domain
Syntax is concise, very little boilerplate
Just write every product from scratch.

[Works only for simple cases]
[Is at odds with SE best practices]
Design Variability into the Language Similar to object-oriented programming

[Scales very well, esp. with verification]
[Might be harder to understand for DEs]
Use Variability Patterns in the DSL

calculation for Tax
depends Salary as s
valid from /2017 01 01/
calculate [monthly] { ...
}
calculation for Tax
depends Salary as s
valid from /2017 07 01/
SomeOtherThing as t
if ctx.employment.person.homeAddress.state == BW
calculate [monthly] { .. }

23/01/2018
Feature model for high-level variability
Interactive, solver-backed configuration
Variation points in DSL code.
here: system modeling and performance simulation
SPLE on Meta Level
Projects are evil!

Ideally each product is just a side effect of the constant evolution of the product line.

Don’t just win the current game, make sure you also prepare for the next.

We need industry-strength language and IDE modularity!
DSL Development

**GPL Extension**
Reuse GPL incl. Expressions and TS
Add/Embed DS-extensions
Compatible notational style
Reduce to GPL

**New Language**
Analyze Domain to find Abstractions
Define suitable, new notations
Rely on existing behavioral paradigm
Reuse standard expression language
Interpret/Generate to one or more GPLs

**Formalization**
Use existing notation from domain
Clean up and formalize
Generate/Interpret
Often import existing „models“

---

**Domain-Specific Data Structures**

**Domain-Specific Behaviors**
Based on existing paradigms such as imperative, functional, declarative, data flow, state-based

**Functional Expressions**

**Existing Domain Notation (Informal)**

**Formalized Language**
DSL Development

**GPL Extension**
- Reuse GPL incl. Expressions and TS
- Add/Embed DS-extensions
- Compatible notational style
- Reduce to GPL

**New Language**
- Analyze Domain to find Abstractions
- Define suitable, new notations
- Rely on existing behavioral paradigm
- Reuse standard expression language
- Interpret/Generate to one or more GPLs

**Formalization**
- Use existing notation from domain
- Clean up and formalize
- Generate/Interpret
- Often import existing „models“
An extensible set of integrated languages for embedded software engineering.

<table>
<thead>
<tr>
<th>User Extensions</th>
<th>User-defined Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages shipped with mbeddr</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>C99</td>
</tr>
<tr>
<td>Logging</td>
<td></td>
</tr>
<tr>
<td>Utilities</td>
<td></td>
</tr>
<tr>
<td>Messaging</td>
<td></td>
</tr>
<tr>
<td>Components</td>
<td></td>
</tr>
<tr>
<td>Physical Units</td>
<td></td>
</tr>
<tr>
<td>State Machines</td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td></td>
</tr>
<tr>
<td>Importer</td>
<td></td>
</tr>
</tbody>
</table>

**Platform**
- Libraries for web server, node navigation, additional notations, pattern matching, palettes, XML processing, debugging...

**MPS**
- Syntax Highlighting, Code Completion, Goto Definition, Find Usages, Type Checking, Data Flow Analysis, Refactoring, Versioning, Debugging

**Foundation**
- C Compiler & Debugger
- PlantUML
- Latex
- HTML
- CBMC
- Z3
- Sat4J

**Implementation**
- Process
- Analysis
### User Extensions

<table>
<thead>
<tr>
<th>Platform</th>
<th>Libraries for web server, node navigation, additional</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPS</td>
<td>Syntax Highlighting, Code Completion, Goto Definition,</td>
</tr>
<tr>
<td>Foundation</td>
<td>C Compiler &amp; Debugger</td>
</tr>
</tbody>
</table>

#### Bottom-Up Development

Growing the Language
C99 itself is broken down into modules that can be reused separately.
Extension Hooks defined into the C99 language
IAssignmentLike: =, +=, *=, etc.

IFunctionLike: function, method

IBinaryLike: +-*/ but also . ?: !

IVoidTypeContainer: function, ftype
DSL Development

**GPL Extension**
- Reuse GPL incl. Expressions and TS
- Add/Embed DS-extensions
- Compatible notational style
- Reduce to GPL

**New Language**
- Analyze Domain to find Abstractions
- Define suitable, new notations
- Rely on existing behavioral paradigm
- Reuse standard expression language
- Interpret/Generate to one or more GPLs

**Formalization**
- Use existing notation from domain
- Clean up and formalize
- Generate/Interpret
- Often import existing „models“

**KernelF**

**Domain-Specific Data Structures**
- Domain-Specific Behaviors
  based on existing paradigms such as imperative, functional, declarative, data flow, state-based
- **Functional Expressions**
Abstract Concepts

Expression
Type
IToplevelContent
IDotTarget
Concept Removals

MPS Constraints can be used to effectively reduce the language.
Exchangeable Primitives

PTF.create<Type>()

Plus an extension infrastructure to contribute types.
Structure vs. Types

Types, such as `ListType` or `IRecordType` can be used for custom language concepts.
New syntax can be defined for existing language concepts.
## Extensions Overview

<table>
<thead>
<tr>
<th>IDE</th>
<th>Salary/Tax</th>
<th>Smart Contracts</th>
<th>Healthcare</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alternative Rules</td>
<td>Live Values</td>
<td>Simulator with Phone UI</td>
</tr>
<tr>
<td></td>
<td>Rules for Data Items</td>
<td>Diff of Live Values</td>
<td>Debugger for State Machine</td>
</tr>
<tr>
<td></td>
<td>Projection for a given Date</td>
<td></td>
<td>Language-Specific Extensions to Functional Debugging</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tracing and Testing</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Structure</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Data</td>
<td>Result Data.</td>
<td>Time Series</td>
<td>Components</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Test Scenarios</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Behaviors</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculation Rules &amp; Dependencies</td>
<td>Interactors &amp; Processes</td>
<td>(Special) State Machines</td>
<td></td>
</tr>
<tr>
<td>Variants</td>
<td>State Machines &amp; Interceptors</td>
<td>Test Generation &amp; Mutation</td>
<td></td>
</tr>
<tr>
<td>Validity</td>
<td>Context Arguments</td>
<td>Coverage Measurement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Live Values</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functional &amp; Types</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Date Types</td>
<td></td>
<td>Time &amp; Duration Types</td>
<td></td>
</tr>
<tr>
<td>Currency Types</td>
<td></td>
<td>Decision Trees and Tables</td>
<td></td>
</tr>
<tr>
<td>Temporal Types</td>
<td></td>
<td>Sentence-like Function Calls</td>
<td></td>
</tr>
</tbody>
</table>
Wrap Up
Efforts?

\[ \approx \]
Size of language
Complexity of type system
Complexity of generator
# of additional tools/services

\[ \approx^{-1} \]
Capabilities of Language Workbench
Reuse of Base Languages & FWs
Skills of dev team and domain experts
(Abstraction Potential)
Using language workbenches and domain-specific languages for safety-critical software development

Abstract

Language workbenches support the efficient creation, integration, and use of domain-specific languages. Typically, they execute models by code generation to programming language code. This can lead to increased productivity and higher quality. However, in safety-/mission-critical environments, generated code may not be considered trustworthy, because of the lack of trust in the generation mechanisms. This makes it harder to justify the use of language workbenches in such an environment. In this paper, we demonstrate an approach to use such tools in critical environments. We argue that models created with domain-specific languages are easier to validate and that the additional risk resulting from the transformation to code can be mitigated by a suitably designed transformation and verification architecture. We validate the approach with an industrial case study from the healthcare domain. We also discuss the degree to which the approach is appropriate for critical software in space, automotive, and robotics systems.
Lessons learned from developing mbeddr: a case study in language engineering with MPS

Authors
Markus Voelter, Bernd Kolb, Tamás Szabó, Daniel Ratiu, Arie van Deursen

Abstract

Language workbenches are touted as a promising technology to engineer languages for use in a wide range of domains, from programming to science to business. However, not many real-world case studies exist that evaluate the suitability of language workbench technology for this task. This paper contains such a case study. In particular, we evaluate the development of mbeddr, a collection of integrated languages and language extensions built with the JetBrains MPS language workbench. mbeddr consists of 81 languages, with their IDE support, 34 of them C extensions. The mbeddr languages use a wide variety of notations—textual, tabular, symbolic and graphical—and the C extensions are modular; new extensions can be added without changing the existing implementation of C. mbeddr’s development has spanned 10 person-years so far, and the tool is used in practice and continues to be developed. This makes mbeddr a meaningful case study of non-trivial size and complexity. The evaluation is centered around five research questions: language modularity, notational freedom and projectional editing, mechanisms for managing complexity, performance and scalability issues and the consequences for the development process. We draw generally positive conclusions; language engineering with MPS is ready for real-world use. However, we also identify a number of areas for improvement in the state of the art in language engineering in general, and in MPS in particular.
SPLE is way more than configurative variability. Not everything can be “predefined” and “selected”.

DSLs are „platforms“ for non-config variability. They make the construction process reusable.

Language Workbenches enable DSLs by reducing effort to build, compose and maintain them.

DSLs enable meaningful integration of domain experts ... making an important contribution to real digitalization.

This is a major contributor to business agility. Because domain experts are not just „customers“.