Language-Oriented Business Applications

Fachexperten als Programmierer

Erfahrungen aus zehn Jahren Projektarbeit

Markus Völter
voelter@acm.org
www.voelter.de
@markusvoelter

Bernd Kolb
kolb@itemis.de
www.itemis.de
@berndkolb
Nothing teaches us better than our own experiences!
3. Product Definition Languages in the Insurance Domain

2. Benefits calculation languages for governments

7+. Languages for (non-programmer) in technical domains

* Languages for use by programmers
Stakeholder integration, Scalable Business, Document Generation + Certification

Code Complexity, Frameworks (Autosar), Product Lines

Reduction of Accidental Complexity in Code, Process Conformance (Docs)

A powerful language and IDE for existing frameworks (Industry Robots, ROS)

Precise Specification and Implementation of Insurance Products („Rules“)

Multi-Paradigm Programming, not just Simulink and C

Consistent Derived Documents

Changing Regulations, Fast Implementation, End User Empowerment
Business Knowledge and Software
It’s what makes a business tick. Distinguishes the business.

- Business Rules
- (Financial) Calculations
- Data Structures
- Mappings or Queries
- Validations
- Scientific Processes
- Contracts
- Processes
- UIs
It's what makes a business tick. Distinguishes the business.

Contributed not by developers... but typically implemented in software.
SO HOW DOES IT GET INTO THE SOFTWARE?

Contributed not by developers

... but typically implemented in software
Trends/Challenges
Complexity
Mass Customization
Time To Market
Reality
Reality
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
Not a software engineer. Does not care about „software stuff“ But understands the domain very well.
Not a software engineer. Does not care about „software stuff“.
But understands the domain very well.

He is a professional, not a „casual hacker“.
Structure/Guid.  +  Mixed
Notation            *
Views               -  Text
IDE/Tool            -  1
Learn/Effective     L  Powerful
Business oriented languages are very different from what we have learned about languages for developers. LWBs let you build such languages.
An old idea from the 1970s.

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

Freely define languages and integrate them
Language Workbench

(Martin Fowler, 2004)

powerful editing testing refactoring debugging groupware

language definition implies IDE definition
Language Workbench
(Martin Fowler, 2004)

There’s no difference!

support for "classical" programming "classical" and modeling

There’s no difference!
LWBs make Languages Easier

Blur the distinction between programming and modeling.

Several different LWBs exist.

http://languageworkbench.es.net
A Language Workbench – a tool for defining, composing and using ecosystems of languages.
Open Source
Apache 2.0
http://jetbrains.com/mps
V 3.3 is current
V 3.4 to be released Summer 2016
[Language Workbench]
Comprehensive Support for many aspects of Language Definition.

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

Parsings

Projectional Editing

Concrete Syntax

Abstract Syntax Tree

Concrete Syntax

Abstract Syntax Tree
[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical
Regular Code/Text

```c
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;
} midnight2 (function)
```

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)
```

Graphical

```
Cst.Customer

Contract
starts: date
ends: date

Tariff
attributes
```

```c
cust 1

trf 1
```
[Projectional Editing]
Language Composition

Separate Files
Type System
Transformation
Constraints

In One File
Type System
Transformation
Constraints
Syntax
IDE
[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.
Most business people are able to and want to express themselves precisely!

Let’s give them the tools to do it!
Examples: Code
exported  compositeblock CbValidatedCommand extends <no extends> realizes IVValidatedCommand

- boolean down
- boolean neutral
- boolean up

⇒ boolean validatedDown
  boolean validatedNeutral
  boolean validatedUp

contract

post(0) OnlyOneCommand: !(validatedNeutral && validatedUp) &&
   !(validatedNeutral && validatedDown) && !(validatedUp && validatedDown));
post(1) OnlyOneCommandImplies: (validatedNeutral -> !validatedUp && !validatedDown)
   || (validatedUp -> !validatedNeutral && !validatedDown)
   || (validatedDown -> !validatedNeutral && !validatedUp);
post(2) AtleastOneCommand: validatedNeutral || validatedUp || validatedDown;
exported atomicblock AbCheckUpDown realizes ICheckUpDown

contract

\[
\begin{align*}
\text{post}(0) & \quad \text{IfResetFalse: } !(\text{reset} \&\& \text{checkedUpDown}); \\
\text{post}(1) & \quad \text{DesignedToFail: } \text{reset} \mid\mid \text{checkedUpDown}; \\
\text{post}(2) & \quad \text{IfResetFalseImplies: } \text{reset} \rightarrow \text{checkedUpDown} == \text{false};
\end{align*}
\]

ccode { checkedUpDown = upDown \&\& !reset; }

exported exprblock EbCheckUpDown

\[
\begin{align*}
\begin{bmatrix}
\text{boolean} & \text{reset} \\
\text{boolean} & \text{upDown}
\end{bmatrix} \Rightarrow 
\begin{bmatrix}
\text{boolean} & \text{checkedUpDown}
\end{bmatrix}
\]

\Rightarrow \text{checkedUpDown} = \text{upDown} \&\& \!\text{reset};

exported compositeblock CbCheckUpDown extends <no extends> realizes ICheckUpDown

contract

\[
\begin{align*}
\text{post}(0) & \quad \text{IfResetFalse: } !(\text{reset} \&\& \text{checkedUpDown}); \\
\text{post}(1) & \quad \text{DesignedToFail: } \text{reset} \mid\mid \text{checkedUpDown}; \\
\text{post}(2) & \quad \text{IfResetFalseImplies: } \text{reset} \rightarrow \text{checkedUpDown} == \text{false};
\end{align*}
\]

exported compositeblock CbCheckUpDownFco extends <no extends> realizes ICheckUpDown

contract

\[
\begin{align*}
\text{post}(0) & \quad \text{IfResetFalse: } !(\text{reset} \&\& \text{checkedUpDown}); \\
\text{post}(1) & \quad \text{DesignedToFail: } \text{reset} \mid\mid \text{checkedUpDown}; \\
\text{post}(2) & \quad \text{IfResetFalseImplies: } \text{reset} \rightarrow \text{checkedUpDown} == \text{false};
\end{align*}
\]
```
atomicblock PassengerModel[] =>
  [boolean up
   boolean down
   boolean neutral]

code {
  up = false;
  down = false;
  neutral = true;
};

atomicblock PassengerWindow[boolean moveUp
                           boolean moveDown] =>
  [uint8/cm/ ->position
   uint8/cm/ ->obstaclePosition]

state [uint8/cm/ ->position:windowPosn = 0 cm;]
code {
  obstaclePosition = 0 cm;
  position = gswitch uint8/cm/ {
    case moveUp: windowPosn + 1 cm
    case moveDown: windowPosn - 1 cm
    default: windowPosn
  };
  windowPosn = position;
};

sim-only(dense) compositeblock ClosedLoopModel extends <no extends>[ ] => [uint8/cm/ ->position]
```
Explorability of the Language

```c
int32 sumUpIntArray(int32 [] arr, int32 size) {
    return \sum_{i=0}^{size} arr[i];
} sumUpIntArray (function)

int32 averageIntArray(int32 [] arr, int32 size) {
    return \frac{\sum_{i=0}^{size} arr[i]}{size};
} averageIntArray (function)
```
Math Notations

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

vector<int16, 3> resultOfCrossProduct = aVector x aVector;

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

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

int32 averageIntArray(int32[] arr, int32 size) {
    size
    \sum_{i = 0}^{\text{size}} arr[i]

    return \frac{\text{size}}{\sum_{i = 0}^{\text{size}} arr[i]};
}

averageIntArray (function)
#constant TEMP_BUFFER_SIZE = 10;

TACQA = Instance of TemperatureAcquisition with mnemonic tail A and the Numeric Id 350
[SENSOR = SensorA]
TACQB = Instance of TemperatureAcquisition with mnemonic tail B and the Numeric Id 351
[SENSOR = SensorB]

Component TemperatureAcquisition with Base Mnemonic: TACQ
  Short Description: acquisition of temperatures
  Description: The components acquires the measurements of an assigned set of thermistors
{
  Attribute (hidden) int32/rawTemp/[TEMP_BUFFER_SIZE] MEASURED = <no init> ; // measured raw values
  Attribute (hidden) uint32 ACQCNT = 0 ; // index for filling data acquisition buffer
  Attribute (readwrite) tempSensor SENSOR (Id= 2 ) = <no init> ; // selected sensor for this component

  ModeChart TCSACQ (Id= 3 ) initial = OFF {
    Trigger tcsAcquisition
    Mode OFF {
      << ... >>
    }
    Mode ON {
      entry { ACQCNT = 0; }
      on trigger tcsAcquisition {
        // measure a value
        MEASURED[ACQCNT] = readTemperature(SENSOR);
        ACQCNT = (ACQCNT + 1) % TEMP_BUFFER_SIZE;

        // calculate average of the \$top(TEMP_BUFFER_SIZE) latest measurements and convert to °C
        PLUS150.AVTEMP = convert[ TEMP_BUFFER_SIZE - 1 \( \sum_{idx = 0}^{\text{TEMP_BUFFER_SIZE}} \text{MEASURED}[idx] \) / \text{TEMP_BUFFER_SIZE} -> °C];
      }
    }
  }

  Activity startAcquisition with Numeric Id 1
  ...
  { TCSACQ.setMode(ON); }

  Activity stopAcquisition with Numeric Id 2
  ...
  { TCSACQ.setMode(OFF); }
} Component TemperatureAcquisition
**Activity** enableTcs with Numeric Id 1 is commandable by TC(150,1)

Short Description: enable thermal control

Description: The thermal control heats the system if it is too cold. The switching hysteresis can be configured.

Constraints:

- 0: TCSCONTR.inMode(OFF) // switching on is possible only if the TCS is off

In-Parameter:

```c
int16/°C/ upperThreshold: constrained : <no constraint> // upper switching threshold
int16/°C/ lowerThreshold: constrained : lowerThreshold < upperThreshold // lower switching threshold
component<TemperatureAcquisition> acq: constrained : <no constraint> // acquisition component instance to use
```

```c
REQUEST acq.startAcquisition ( << ... >> ) --> ( << ... >> )
  on error do nothing special
  on error abort
  UPTH = upperThreshold;
  LOPTH = lowerThreshold;
  DELAY for 10 s
  TCSCONTR.setMode(ON);
  TELEMETRY (150,11)
```

Description: Report switching on in a dedicated packet that reports the initial temperature.

```c
[initialTemp : int32/°C/ = FUS150.AVTEMP // initial temperature when starting thermal control]
```

**Activity** disableTcs with Numeric Id 2 is commandable by TC(150,2)

Short Description: disable thermal control

Description:

Constraints:

- 0: TCSCONTR.inMode(ON) // switching off is possible only if the TCS is on

In-Parameter:

```c
<< ... >>
```

```c
TCCONTR.setMode(OFF);
REQUEST TACCA.stopAcquisition ( << ... >> ) --> ( << ... >> )
  on error do nothing special
REQUEST TACCQ.stopAcquisition ( << ... >> ) --> ( << ... >> )
  on error do nothing special
```

Component ThermalControlSystem
TCSCONTR
thermal control

OFF
thermal control is inactive

TC(150,1) enableTcs
TC(150,2) disableTcs

ON
thermal control is active

trigger tcsControl
periodically triggered for altering the heater power state according to the measured values
exit
disable all heaters

TCSCONTR

OFF

TC(150,1) enableTcs

TC(150,2) disableTcs

ON

PUS128 TelemetryService

enableTcs

TACQ_startAcquisition

DELAY 10 s

TCSCONTR->ON

TM(150,11); queue=10

ProcessTelemetry

TCSACQ->ON
Examples: Engineering
Requirements

This document lists the requirements for the passenger-side power window controller. Power window controller controls lifting and lowering of the passenger-side window. Both the driver and passenger have up and down buttons that they can use to lift or lower the window.

Driver Operation

DriverOperation /functional: status=accepted
created by sundu at Feb 23, 2016 (4 months ago)

Driver shall be able to operate the passenger side window: lift the window by pressing the relevant button upwards, and lower it by pressing the relevant button downwards.
variability Scooter

Feature model

[Wheels]
- [front Wheel]
- [rear Wheel]

[Engine]
- [power]

[Electrical]

[Battery]
- [capacity]

[Combustion]
- [Diesel]
- [Gasoline]

Feature model

[Wheel]

[Brake]
- [Simple]
- [Carbon]

[PreassureSensor]
- [maxPressure]
Functional Expressions

variables:

```plaintext
type, value
```

```plaintext
int x1 = 10 * (1 + 2) => 30
int x2 = 20 => 20
boolean b1 = true || !false => true
int b2 = if [ b1 then 12 else 13 ] => 12
list<int> intList = list(1, 2, 3) => [1, 2, 3]
int three = intList.last => 3
list<int> t2 = intList.where[it > 2] => [3]
boolean allEl = intList.all[it > 0] => true
int surprise2 = doWithTwoInts(:add, 1, 3) => 4
[int, int] tuple = [1, 2] => [1, 2]
int one = tuple[0] => 1
int c1 = alt[x1 < 0 && x2 > 1 => 2]
[ x1 > 0 && x2 == 1 => 1] => 0
int c2 =
```

```plaintext
<table>
<thead>
<tr>
<th>x2</th>
<th>x1 &lt; 0</th>
<th>x1 == 0</th>
<th>x1 &gt; 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
```

int complicated = {
  val t1 = 10 + 20
  val t2 = t1 + 30
  t2
}

functions:

```plaintext
fun add(int a, int b) : int = a + b
fun doWithTwoInts((int, int => int) fun, int a, int b) : int = fun.exec(a, b)
fun anotherFun(option<int> i) : int = with some i as x => x + 1 none 20
fun giveMeAnInt() : int = anotherFun(some(10))
fun getStreets(Person p) : collection<string> = p.workedAt.offices.street
```
public data GenericSpeed : int where it.range[-50..300]
data SpeedFromRoad extends EngineRPM
data SpeedFromEngine extends GenericSpeed where it.range[0..300]
data SpeedFromRoad_FWD extends SpeedFromEngine where it.range[0..250]

public data Pos_Lat : int where it.range[-90..90]
public data Pos_Long : int where it.range[-180..180]
public data Position {
    Pos_Lat
    Pos_Long
}

public data EngineRPM : int where it.range[0..10000]
public data EngineOnOff : enum[on, off]
public data EngineStatus {
    EngineOnOff
    EngineRPM
}

// 0 means backwards :-)
public data Gear : int where it.in[1, 2, 3, 4, 5, 6, 7, 8]

data RoadConditions : enum[dry, wet, snowy, icy]

public data Req_Accelerator : int where it.range[0..100]
public data Req_Brake : int where it.range[0..100]
public data Req_Gear : int where it.in[1, 2, 3, 4, 5, 6, 7, 8]

public data DrivingCommands {
    Req_Accelerator
    Req_Brake
    Req_Gear
}
Hierarchical Components

```java
public functional component DriveTrain {
    produces SpeedFromEngine
    produces EngineStatus
    produces Gear where it < gearsCount
    consumes RoadConditions
    param int gearsCount
    consumes DrivingCommands
}

functional component Car {
    ActuatorBox
    LocationServices
    DriveTrain gearsCount = 7
    RoadSensors
    HeadUnit
    Gearbox
}
```
Component Modeling II
Activity Modeling
Safety Analysis I

@hazard(Ommision of Braking Function) - 8.65E-9

-or-

@subsystem(BrakePedal)
malfunctions - 2.5 FIT

@subsystem(DistanceSensor2)
malfunctions - 10.0 FIT

@subsystem(DistanceSensor1)
malfunctions - 10.0 FIT
Safety Analysis II

G1
Braking system is acceptably safe to operate
@req_module(BrakingFunctionRequirements)
@req_module(BrakingFunctionSafetyRequirements)

G2
All identified hazards are mitigated

S1
Argument over each hazard from
@func_hazards(BrakingFunctionRequirements)

G3
Hazard @hazard(Unintended Braking) is sufficiently mitigated

G4
Probability of
@hazard(Commission of Braking Function) occurrence < 10^-8 per h

G3
Probability of
@hazard(Unintended Braking) occurrence < 10^-7 per h

S2
Fault Tree Analysis
@fault_tree(CommissionOfBrakingFunctionFT)

S1
Two channel architecture
@arch(BrakingFunctionArchitecture)
Performance Modeling I

hardware MySystem

imports << ... >>

processor ARM_Multicore

resource Harddisk {
  unit MB
  interface standard {
    bandwidth 32.0
    induced load 20%
    cst 5%
  }
  interface dire {
    bandwidth 32
    induced load 1%
    cst 1%
  }
}

pool MainMemory {
  unit MB
  maximum 4096
}

components SimbenchTest

functional component Harddisk {...}

functional component GraphicsSubsystem {...}

data MapData

functional component NavigationSubsystem {
  sim {
    on init {...}
    on start_routing {
      2300 ms (ready: 2300ms)
      prepare {
        use 1500
      }
      5800 ms
      loadData {
        read MapData : 50 MB via Harddisk.standard
        allocate MainMemory : 250 MB
        use 1000
      }
      5362 ms
      map_visible {
        precondition GraphicsSubsystem::available
        allocate Display : 1 items
        use 500
      }
      5862 ms
    }
    << ... >>
  }
}
Performance Modeling II

Resources capacity also visible

CPU capacity utilisation gaps!
Examples: Business
4 Points you get for each trackpoint

InFlightPoints /functional: tags


Duis tempus justo magna. Nunc lobortis libero sed eros interdum aliquet ele. It uses @req(PointsFactor) sdf @cfmod(ArchitecturalComponents) to calculate the total points.

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] => (uint8  || int8 )
[int16 speed: current speed of the trackpoint]

result = 200

\[
(10 \cdot \text{BASEPOINTS}) \cdot 10
\]

false true otherwise 0

\[
\begin{array}{c|cc}
\text{false} & 1100 \text{alt > 2000} & 1100 \text{alt > 1000} \\
\hline
165 \text{speed > 180} & 30 & 15 \\
165 \text{speed > 130} & 10 & 20 \\
\end{array}
\]

tests: PointForATrackpoint(500, 46) == 0
PointForATrackpoint(500, 1200) == 0
PointForATrackpoint(1100, 165) == 200
PointForATrackpoint(2100, 140) == 100
PointForATrackpoint(2100, 200) == 300
# Rigid Structures

<table>
<thead>
<tr>
<th>Rule Set Type</th>
<th>DemoRuleSetType</th>
</tr>
</thead>
</table>

## 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>

## Libraries

- **Standard**
- **Extra**

## Parent

- **<no parent>**
Calculation Rules

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 = [ <always> => round(PREMIO * (1 + factacer), 0) ]
Prose-Like Language for Calc Rules

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 ongeliijk 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
}
# Textual Notation for Data Modeling

**Data Contract**

**proxy for** Customer.Customer

<table>
<thead>
<tr>
<th>core data entity BillingRegion</th>
</tr>
</thead>
<tbody>
<tr>
<td>code [key]: string</td>
</tr>
<tr>
<td>name: string</td>
</tr>
<tr>
<td>baseMinPrice: float</td>
</tr>
<tr>
<td>maxRebateFactor: float</td>
</tr>
</tbody>
</table>

**entity Contract**

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

**entity Tariff**

| attributes: | references: |
Diagrams for Data Modeling

Contract

starts: date
ends: date
contracts 0..*

Tariff

applicableTariff 1
attributes

customer 1

Customer.Customer

BillingRegion

code [key]: string
name: string
baseMinPrice: float
maxRebateFactor: float
# Tables for Reference Data

## Core Data: Default Regions for entity BillingRegion

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

\[ \text{import: } \Sigma \text{ CustomerBasic} \]

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

| Value cust := entity.customer |
| Value pricingFactor := |

<table>
<thead>
<tr>
<th>cust.isRebated</th>
<th>isLocal</th>
<th>isLongDistance</th>
<th>isRoaming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>!cust.isRebated</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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

22/09/14 08:19 (13 s ago) by markusvoelter
Calculations CustomerBasic for Customer

---

**Imports:**
- TimeUnits
- BusinessRequirements

---

**Node:** isRebated [FlagVar]

**Kind:** implements

**1st Target:** Users should be rebated

- Some users should get cheaper phone calls. The reasons for the rebates are outlined below.

```plaintext
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`

- `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`
Tests executed in the Editor

```
structure group Calculate and Test calls
  
  flag hasEverMadeACall :: !entity.callsOfCustomer.isEmpty
  value amount of calls :: ((hasEverMadeACall)) ? entity.callsOfCustomer.size : 0

  tests:
    | (entity := Peter M) == 0 actual: 2
    | (entity := Peter M) == 2
    | (entity := Hanna B) == 2 actual: 3
    | (entity := Hanna B) == 3

  endtests

  value all calls :: entity.callsOfCustomer

  value discountFactor :: magic of type double

  value current price :: \[ \sum_{i=0}^{\text{amount of calls} - 1} (\text{all calls.at(i).price.value}) \times \text{discountFactor} \]

  tests:
    | (entity := Hanna B, discountFactor := 0.9) == 10.8
    | (entity := Hanna B, discountFactor := 1.0) == 8.55 actual: 12.0
    | (entity := Peter M, discountFactor := 1.0) == 0.5 actual: 4.9
    | (entity := Peter M, discountFactor := 1.0) == 4.9

  endtests

  value averageCallPrice :: \[ \frac{\text{current price}}{\text{amount of calls}} \]

  tests:
    | (entity := Hanna B, discountFactor := 1.0) == 4.0
    | (entity := Hanna B, discountFactor := 1.0) == 2 actual: 4.0
    | (entity := Peter M, discountFactor := 1.0) == 2.45

  endtests
```
BDD-style Tests for Business Rules

```plaintext
rule checkStuff
  given anything
  when [the customer.calls.size is equal to 10] and
       [the call.endTime is smaller than 20]
  then [set call.price to 20]
       [execute cancelContract with customer]
```
Assessments

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
Embedded Buttons in Editors

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
   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
   [ ]
   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.
Formale Beschreibung

Programmquelle: vmcfo2.c
Produkt-Typ: Fonds, HSR  PK-Typ: Kapital-Konto
Status: 18.1

Parameter-Attribute

| znt_param |

Verwendete VADM-Attribute

<table>
<thead>
<tr>
<th>rg_kk_fo_beta_satz</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>rg_kk.beta_satz</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.kz_zus_gar</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.zm</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.vtrk_zb</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.zw</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.ko_reid</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.kz_mandant</td>
<td>E</td>
</tr>
<tr>
<td>rg_kk.beta_satz_fakt</td>
<td>E</td>
</tr>
</tbody>
</table>

Rückgabe-Attribut

| satz_beta |

aufgerufene Funktionen

Kommzahl MDI (Kommzahl a; Kommzahl b)
Kommzahl MAX (Kommzahl a; Kommzahl b)
rg_kk_beta_dp_satzTF |
rg_kk_beta_up_satzTF |

Beschreibung

In dieser Funktion wird der Kostensatz β ermittelt.

Hilfsvariablen

| grenze     |
| fak_beta   |
| beta_dp_satz_hilf |
| beta_up_satz_hilf |

Verarbeitungen

<table>
<thead>
<tr>
<th>rg_kk_fo_beta_satz</th>
<th>Berechnung</th>
<th>Bemerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>s = satz_beta</td>
<td>Standard</td>
</tr>
<tr>
<td></td>
<td>Falls znt_param &lt;= 120 und rg_kk.kz_zus_gar = JA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>satz_beta = satz_beta * MIN (0,01 + MAX (znt_param - 12; 0); 1) sonst</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vorarbeitung hinzufügen</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ende Falls znt_param &lt;= 120 und rg_kk.kz_zus_gar = JA</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>s = satz_beta</td>
<td>PF</td>
</tr>
<tr>
<td></td>
<td>Falls grenze &lt; 10000.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>satz_beta = satz_beta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sonst</td>
<td>GULPP</td>
</tr>
<tr>
<td></td>
<td>satz_beta = 0.074</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ende Falls grenze &lt; 10000.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>s = satz_beta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Falls znt_param &lt;= 156 und rg_kk.kz_zus_gar = JA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fak_beta = 0,03 * MIN (znt_param / 12 - 1; 1)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>satz_beta = satz_beta</td>
<td>FV Standard</td>
</tr>
<tr>
<td></td>
<td>Falls znt_param</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z-D ab TV 8</td>
<td></td>
</tr>
</tbody>
</table>
Funktionenmodell Geometrisches Mittel

Formale Beschreibung

Funktion: Geometrisches Mittel
Programmquelle: Programmquelle auswählen
Produkt-Typ: Produkt-Typen auswählen
PK-Typ: PK-Typ auswählen
Status: Status auswählen

Parameter-Attribute
\[ a \]
\[ b \]

Verwendete VADM-Attribute
Keine verwendeten VADM-Attribute, werden automatisch hinzugefügt

Rückgabe-Attribut
result

aufgerufene Funktionen
Keine aufgerufenen Funktionen, werden automatisch hinzugefügt

Beschreibung
Berechnet das geometrische Mittel der Parameter

Hilfsvariablen
Hilfsvariable hinzufügen

Error: Quadratwurzel ist nur für positive Zahlen erlaubt
\[ \text{result} = \sqrt{a - b} \]
\[ \text{result} = \sqrt{a + b} \]

return result
Observations:
1) Every domain has **different reasons** why language engineering is used.
2) No domain has **only one reason** – it is always a combination
3) Languages, Editors, Type Systems and good IDEs are **always** important!
Morgen um 10:45:

Der Geekige Teil

Not your Daddy’s Language

#Architecture & Design

4
Lessons Learned
How to make People precise?
Precision

!=

Programming
Precision ^= Programming
Precision != { Formulas, Rules, Data Structures, Tables, Values }

Performance
Scalability
Robustness
Deployment

Programming
Precision ≠ Formalization

Formulas, Rules
Data Structures
Tables
Values
Precision \{ Formulas, Rules, Data Structures, Tables, Values \}

\text{Greek Letters, Analyses, Proofs} \implies \text{Formalization}
Benefits of being precise

Make changes to system without waiting for IT
Directly Test and Debug business knowledge
Explore Alternatives and Experiment
How to get business people to be precise

Willingness to take responsibility
Very good fit with domain „Friendly“ Abstractions and Notations
Good Tools (see later)
Education and Training
How to get business people to be precise

Technical People: „It‘s not my job!“. (and it really isn‘t)
A hybrid of many worlds
Expressions
Code Completion
Syntax Highlighting
Error Markup
Version Control
Refactoring
Debugging
Scalability
Code Reviews

Languages/IDEs
Abstraction Levels
Multiple Abstractions
Multiple Notations
Live Execution
„Visible Computation“
Document-Oriented

Applications
Forms

Languages/IDEs

Wikis

Modeling Tools

Spreadsheets
Why Version Control
Why Version Control

Consistency across Team
Why Version Control

Consistency across Team Development History
Why Version Control

Consistency across Team Development History Time Machine
Why Version Control

Consistency across Team Development History Time Machine Branching (Feature, Version)
Why Version Control

Consistency across Team Development History
Time Machine
Branching (Feature, Version)
Support Staging
Use Staging
Production System

Live for Customers
Real Data
Mission Critical
Test System
For Business People
Real-Like Data
May Have Bugs

Production System
Live for Customers
Real Data
Mission Critical
Change

Test System
- For Business People
- Real-Like Data
- May Have Bugs

Production System
- Live for Customers
- Real Data
- Mission Critical

Integration Tests
Simulations
Reviews
How do you achieve consistency?
Strict Language
Cross-References
Modularization and Reuse

Automatic Derivation based on rules (transformation, generation)
Common Repository
Version Control System
Periodic, Global Checks/Reports
Influences on the Language
Domain Structure

Non Functionals
Permissions, IP, Sharing

User Skills

Model Purpose
Analyze, Generate

Tool Capabilities
Notations, Editing, Scale

Software Engineering Practices

Sep. of Concerns
Different Views

Educate, Put results in context

Get a better tool :-)
The Language is not Enough
Requirements on the tool
Be a great LWB obviously

Support all the language goodness we talked about so far.
Productivity

Quickly evolve the language as the (understanding of) domain changes
Performance

Nobody wants to work with a sluggish tool
Scalability

Non-trivial languages and significant model sizes
Evolution Support

Migrate existing models as the languages evolve.
Friendliness

Don’t overwhelm end users with too much „cruff“
Explorability

Ensure the language can be explored
A tool is not enough
Methodology = Process + Tool (+ Metrics)
Precision/Consistency refers to Artifacts and not to a rigid Process.
Discipline: do the right thing.

Define what is „right“
Force People?
Tool should makes the right thing easy.
Tool should make the right thing easy.

Error Messages
Process-Guidance in the tool
Checklists to finish manual processes
Tool must fit the process!
Does this scale?
Does the approach scale?

If structure, formalization, and tool support don’t scale, then what will?

What are the alternatives?
  Excel?
  Wikis?
  Prose Documents?
Do the tools scale?

In terms of overall system size?
Yes, the system has to be broken down into models of manageable size, as usual. This requires some thought.

In terms of team size?
Yes, since we rely on established version control systems (git) to deal with groupware aspects; and yes, diff/merge works as expected.

In terms of language complexity?
Yes, in particular, since you can modularize the language definitions.
Can I find the people to do this?

Yes, but it is a significant change, so:
- it may be a significant education/training effort.
- a few people might not get it
- a few people may not want to do it.
This is a threat!
Precision and Formality
Different Processes
Higher Efficiency

-> New Skills
-> Role Change
-> Job Loss

Automation
Focus on Engineering
Empower Business Ppl

-> Job Loss
-> Role Change
-> Less Importance
Some people are afraid of this.
Take them seriously.
A change of culture that must be managed!
We tried it before, and it failed.
The UML tool was a bad choice
  -> ok, choose a better one :-)  

Hard to represent business logic in UML.
  -> oh, really?? Who would have thunk.

Generate Class-Skeletons, fill in app logic.
  -> how and why does this solve the challenges??

Round-Tripping did not work.
  -> never works, but why use it?

Such an approach is completely pointless!!
Rule Language

No tests and debuggers for end users
-> hard to be sure about things

Language not expressive enough (tables)
Tool too limited to enhance expressivity
-> tedious to express many algorithms

Parts still had to be programmed manually
-> overall process more complex, not simpler

The right direction, but not good enough.
How is this not an EDM?
Requires coordination with the whole enterprise – never works.
Language Modularization, Composition and Extensions

Narrow interfaces between languages
(and between the models build with the languages)

Delayed global consistency checks
(in contrast to local, eager checks)

Can be limited to one or more subdomains
Why now?
What has changed?
Complexity rises, time to market reduces, variability increases.

What is the alternative?

Tools have gotten better in terms of flexibility, usability, scalability.

It seems realistic now.
Contraindications
No structure in domain
  -> language would be too low level

No availability of domain experts
  -> cannot retrieve knowledge for building the language

No resources available
  -> initially it will be additional work...

Immature Organization
  -> never heard of unit test, CI and VCS? Bad sign!
How do you introduce this?
YOU NEVER KNOW HOW STRONG YOU ARE... UNTIL BEING STRONG IS THE ONLY CHOICE YOU HAVE.
1 Agree this is the right way

Self-Learning and considering alternatives
Consulting & Look at relevant similar cases
Analysis of your own situation

2 Prototype it

Possibly with external help to learn tool and guide
Small but meaningful sub problem
Evaluate Approach and tools
Integrate Stake Holders -> Sales Job!

3 Go for the real thing

See next slides.
Create a dedicated team/organization whose goal it is to be successful with the approach.
Create a dedicated team/organization whose goal it is to be successful with the approach.

Decouple from Daily Business.

Staff with people who are driven, open to change and good communicators.
Introducing the Approach

Step by Step 1
Vertical Slice through Domain, then expand

Step by Step 2
Increasing Levels of Formality

Prose
Prose + Glossary
Prose + Glossary + Calculation Rules + Code Generation
...
Introducing the Approach

Step by Step 1
Vertical Slice through Domain, then expand

Step by Step 2
Increasing Levels of Formality

Keep the end goal (formalization, automation) in sight, otherwise it is hard to justify „strange tools“ as opposed to a Wiki, e.g.
A real Example: Legacy

Specify/Program

Insurance Programs

Write formal code in a DSL mixed with tables and text
No tool support whatsoever
No testing (except inspection)
No reuse
No modularity
No variability
A real Example: Legacy

Specify/Program

Insurance Programs

Write formal code in a DSL mixed with tables and text
No tool support whatsoever
No testing (except inspection)
No reuse
No modularity
No variability
A real Example: Legacy

Specify/Program

Insurance Programs

Write formal code in a DSL mixed with tables and text

No tool support whatsoever
No testing (except inspection)
No reuse
No modularity
No variability
A real Example: Legacy

Specify/Program

Insurance Programs

„Pixelcrap“

Write formal code in a DSL mixed with tables and text
Printed, PDF

No tool support whatsoever
No testing (except inspection)
No reuse
No modularity
No variability
A real Example: Legacy

Write formal code in a DSL mixed with tables and text

No tool support whatsoever
No testing (except inspection)

No reuse
No modularity
No variability

Specify/Program

Insurance Programs

„Pixelcrap“

Implement

C Code

Printed, PDF

Developer reads „spec“
Very idiomatic implementation

Dev acts as a human compiler and implements it in C
A real Example: Legacy

Specify/Program

Insurance Programs

„Pixelcrap“

Implement

C Code

Specify/Program

Printed, PDF

Developer reads „spec“
Very idiomatic implementation
Dev acts as a human compiler and implements it in C

Debugging directly in C
Search-for-use by text search
Don’t trust the documents – may be outdated!

Write formal code in a DSL mixed with tables and text
No tool support whatsoever
No testing (except inspection)
No reuse
No modularity
No variability
A real Example: Current

Specify/Program/Test/Debug

Insurance Programs

Write formal code in a DSL mixed with tables and text
Now with IDE support and executable tests

The same notation!
Funktionenmodell berbwvekFF

Formale Beschreibung

Funktion: berbwvekFF
Programmquelle: vmsctfal.c
Produkt-Typ: Fonds  PK-Typ: Kapital-Konto
Status: 18.1

Parameter-Attribute

lkm_akt_param
lkm_faell_param
ber_zweck_param
kz_rzw_param

Verwendete VADM-Attribute

Keine verwendeten VADM-Attribute, werden automatisch hinzugefügt

Rückgabe-Attribut

bwvek

aufgerufene Funktionen

VTRKernabtfgaellFF (a)
bwbweinzelFF (a; b; c)

Beschreibung

Die Funktion liefert den Barwert per @lkm_akt_param des vorschüssigen Zahlungsstroms der Höhe 1 von Monat @lkm_akt_param bis @lkm_faell_param – jeweils einschließlich. Zahlungszeitpunkte sind jeweils die Monatsbeginne, also @lkm_akt_param bis @lkm_faell_param – 1#. Der Parameter @kz_rzw_param steuert die zu berücksichtigende Zahlweise des Zahlungsstroms. Mögliche sind zur Zeit nur die Auspragungen 0 (Zahlungen zu den Beitragsfälligkeiten) und 12 (monatliche Zahlungsweise).

Hilfsvariablen

kz_bf_hilf

Verarbeitungen

Schleife über lkm_faell_hilf = lkm_akt_param bis lkm_faell_param

Falls kz_rzw_param = 12

kz_bf_hilf = 1
sonst

kz_bf_hilf = VTRKernabtfgaellFF (lkm_faell_hilf)

Ende Falls kz_rzw_param = 12

bwvek = bwvek + kz_bf_hilf * bwweinzelFF (lkm_akt_param; lkm_faell_hilf = 1; ber_zweck_param)

Ende Schleife über lkm_akt_param bis lkm_faell_param

return bwvek
A real Example: Current

**Funktionenmodell berbwekFF**

**Formale Beschreibung**

- **Funktion:** berbwekFF
- **Programmquelle:** vmsctfa1.c
- **Produkt-Typ:** Fonds
- **PK-Typ:** Kapital-Konto
- **Status:** 18.1

**Parameter-Attribute**

- lkm_akt_param
- lkm_faelle_param
- ber_zweck_param
- kz_rzw_param

**Verwendete VADM-Attribute**

- Keine verwendeten VADM-Attribute, werden automatisch hinzugefügt

**Rückgabe-Attribut**

- *bwvek*

**aufgerufene Funktionen**

- VTRKernmrbgtfaellFF (a)
- BerbeweinzelFF (a; b; c)

**Beschreibung**

Die Funktion liefert den Barwert per @lkm_akt_param des vorschüssigen Zahlungsstroms mit @lkm_akt_param bis @lkm_faelle_param - jeweils einschließlich. Zahlungszeitpunkt wird mit @lkm_akt_param - 1 bis @lkm_faelle_param - 1. Der Parameter @kz_rzw_param steuert die zu berücksichtigenen Zahlwert des Zahlungsstroms. Möglich sind zur Zeit nur die Ausprägungen 0 (Zahlung und Schleife über lkm_faelle_hilf = lkm_akt_param bis lkm_faelle_param

**Verarbeitungen**

Schleife über lkm_faelle_hilf = lkm_akt_param bis lkm_faelle_param

- Falls @kz_rzw_param = 12
  - kz_bf_hilf = 1
  - sonst
    - kz_bf_hilf = VTRKernmrbgtfaellFF(lkm_faelle_hilf)
- Ende Schleife über lkm_faelle_param bis lkm_faelle_param
  - return bwvek

**Formale Beschreibung**

- **Funktion:** berbwekFF
- **Programmquelle:** vmsctfa1.c
- **Produkt-Typ:** Fonds
- **PK-Typ:** Kapital-Konto

**verwendete Attribute**

- lkm_akt_param
- lkm_faelle_param
- ber_zweck_param
- kz_rzw_param
- bwvek

**aufgerufene Funktionen**

- berbeweinzelFF
- VTRKernmrbgtfaellFF

**Verarbeitungen**

Die Funktion liefert den Barwert per lkm_akt_param des vorschüssigen Zahlungsstroms der Höhe 1 von Monat lkm_akt_param bis lkm_faelle_param - jeweils einschließlich. Zahlungszeitpunkte sind jeweils die Monatsbeginne, also lkm_akt_param - 1 bis lkm_faelle_param - 1.

Der Parameter @kz_rzw_param steuert die zu berücksichtigende Zahlwert des Zahlungsstroms. Möglich sind zur Zeit nur die Ausprägungen 0 (Zahlungen zu den Beitragsfälligkeiten) und 12 (monatliche Zahlungsweise).

Schleife über lkm_faelle_hilf = lkm_akt_param bis lkm_faelle_param

- Falls @kz_rzw_param = 12
  - kz_bf_hilf = 1
  - sonst
    - kz_bf_hilf = VTRKernmrbgtfaellFF(lkm_faelle_hilf)
- Ende Falls @kz_rzw_param = 12
  - bwvek = bwvek
  - + kz_bf_hilf * berbeweinzelFF(lkm_akt_param, lkm_faelle_hilf - 1, ber_zweck_param)
- Ende Schleife
  - return bwvek
A real Example: Current Insurance Programs

Write formal code in a DSL mixed with tables and text

Now with IDE support and executable tests

The same notation!
A real Example: Current Insurance Programs

Write formal code in a DSL mixed with tables and text. Now with IDE support and executable tests. The same notation!
A real Example: Current

Specify/Program/Test/Debug

M3

Insurance Programs

Generate

C Code

Write formal code in a DSL mixed with tables and text

Exactly the same C code.

The same notation!
A real Example: Future

Specify/Program/Test/Debug

M3
Insurance Programs

Generate

C Code

Still exactly the same C code, or improved as needed.

Incremental Refinement/Refactoring of languages:

Partially automated migration of models
Add model natural notations (insurance-specific, math)
Add Support for modularity, reuse, variants
Why is this an initiative by engineers?
Business people don’t feel the pain

-> the developers find inconsistencies and problems

They don’t necessarily know the ways to solve the problem

-> don’t have the ideas of how to do it better

And by the way:
We know many organizations where the business people want to be involved more directly, but the technical people don’t know how to do it.
Is this the next legacy system?
Today’s software is tomorrow’s legacy system.

Or is it?
Today’s software is tomorrow’s legacy system.

Business change is hard
Technology change is hard

Separation of Concerns
Keep BL free of technology
Make it „portable“
Existing models become incompatible with new language

⇒ Language Versions

Migration Scripts
Runtime Tech outdated, uncool or slow

⇒ Keep Lang Technology
Keep Models
Build new Generator
Language Tech outdated, uncool
⇒ Build new Tool
Migrate Data Simple, because it well-defined domain semantics and free from „technology stuff“
Today’s software is tomorrow’s legacy system.

No, it is not.
Grass Root
Excel/Access
A webcomic of romance, sarcasm, math, and language.

ALGORITHMS
by complexity

MORE COMPLEX
LEFTPAD  QUICKSORT  GIT  MERGE  SELF-DRIVING  CAR  GOOGLE  SEARCH  BACKEND

SPRAWLING EXCEL SPREADSHEET BUILT UP OVER 20 YEARS BY A CHURCH GROUP IN NEBRASKA TO COORDINATE THEIR SCHEDULING

http://xkcd.com/1667/
Various Departments interact with an opaque central IT system
Departments may build their own ad-hoc tools with Excel and Access
Possibly connected to CITS
Now: central IT system is an LWB with a couple of Languages
B1 only visible to users from department B

B1 potentially developed by devs associated with B

Department-specific functionality realized as language extensions
Non-CITS-connected systems remain as they are.

Department-specific functionality realized as language extensions

Language Workbench

B1 only visible to users from department B
B1 potentially developed by devs associated with B

Non-CITS-connected systems remain as they are.
5

Drawbacks
You need inhouse expertise for language engineering

or a very close and trusted vendor who does it for you.
You will bind yourself to a particular tool.

You can easily export models, but no portability for language definitions.
If you use this approach for real, you should have language engineering expertise in house.
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.
source
open
Thank you!