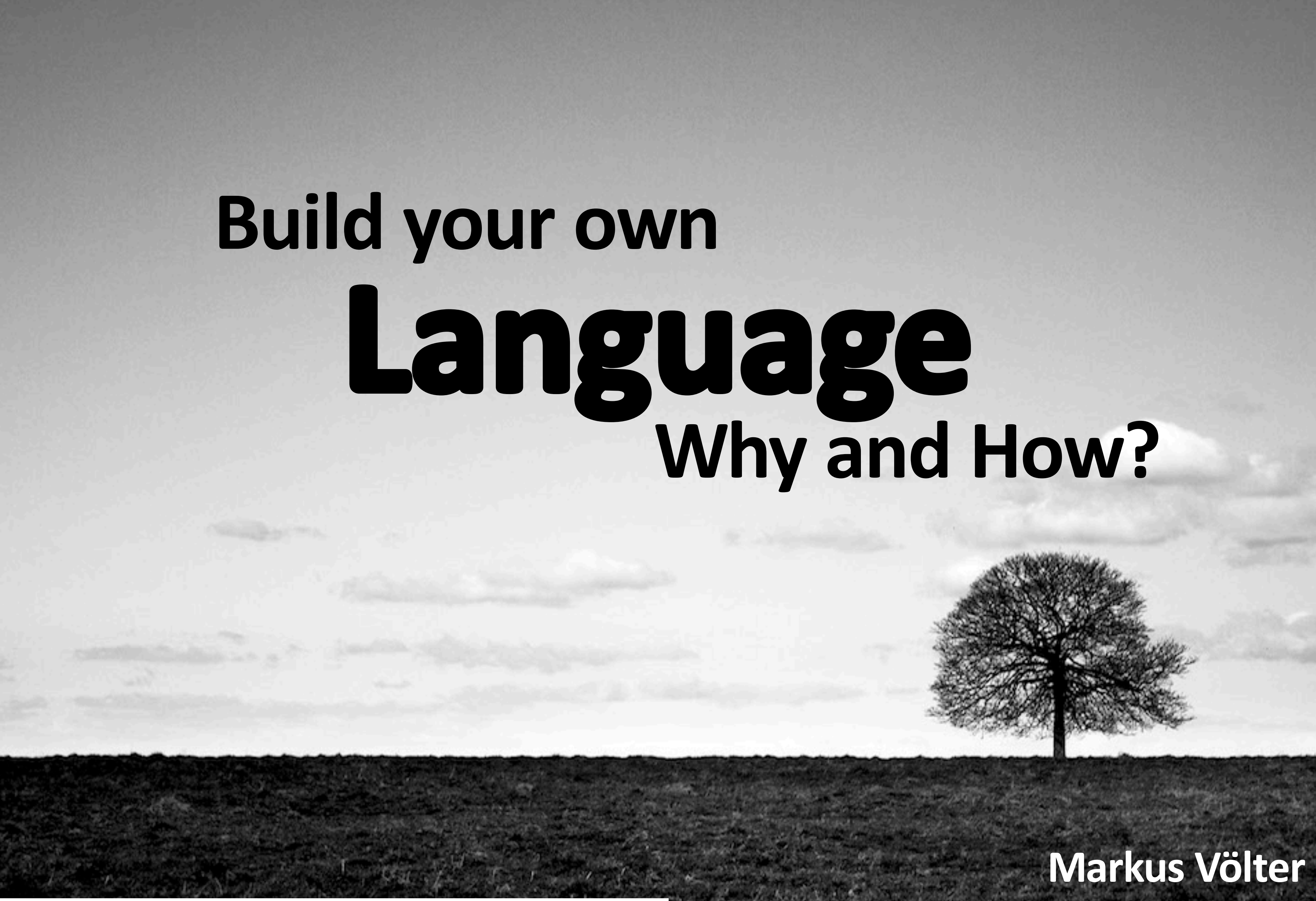


# Build your own Language

## Why and How?

**Markus Völter**

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

voelter {ingenieurbüro für softwaretechnologie //itemis

1

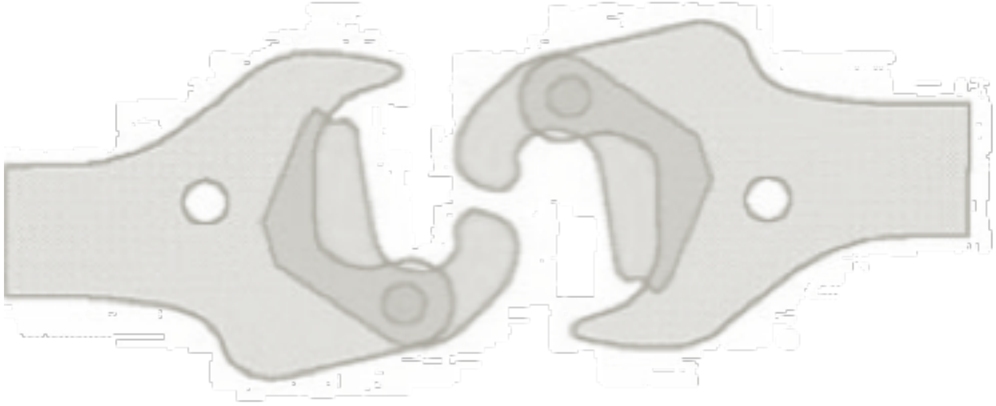
# Motivation

**I just finished v3 of the requirements document. But I am sure it will take another two months of ping-pong with IT to get the damn thing to run.**

**Aargh, another half-baked requirements document. Those guys always rely on us to “debug“ it and make it work.**

The IT guys have decided to port the system to mobile phones. We have to do another re-write/-understand of all the Fachlichkeit. Again!!

Well, yes, but we have to keep up with the evolving technologies and new platforms. No way around it!



**Decouple Fachlichkeit and Technology!**
so you can evolve both independently.

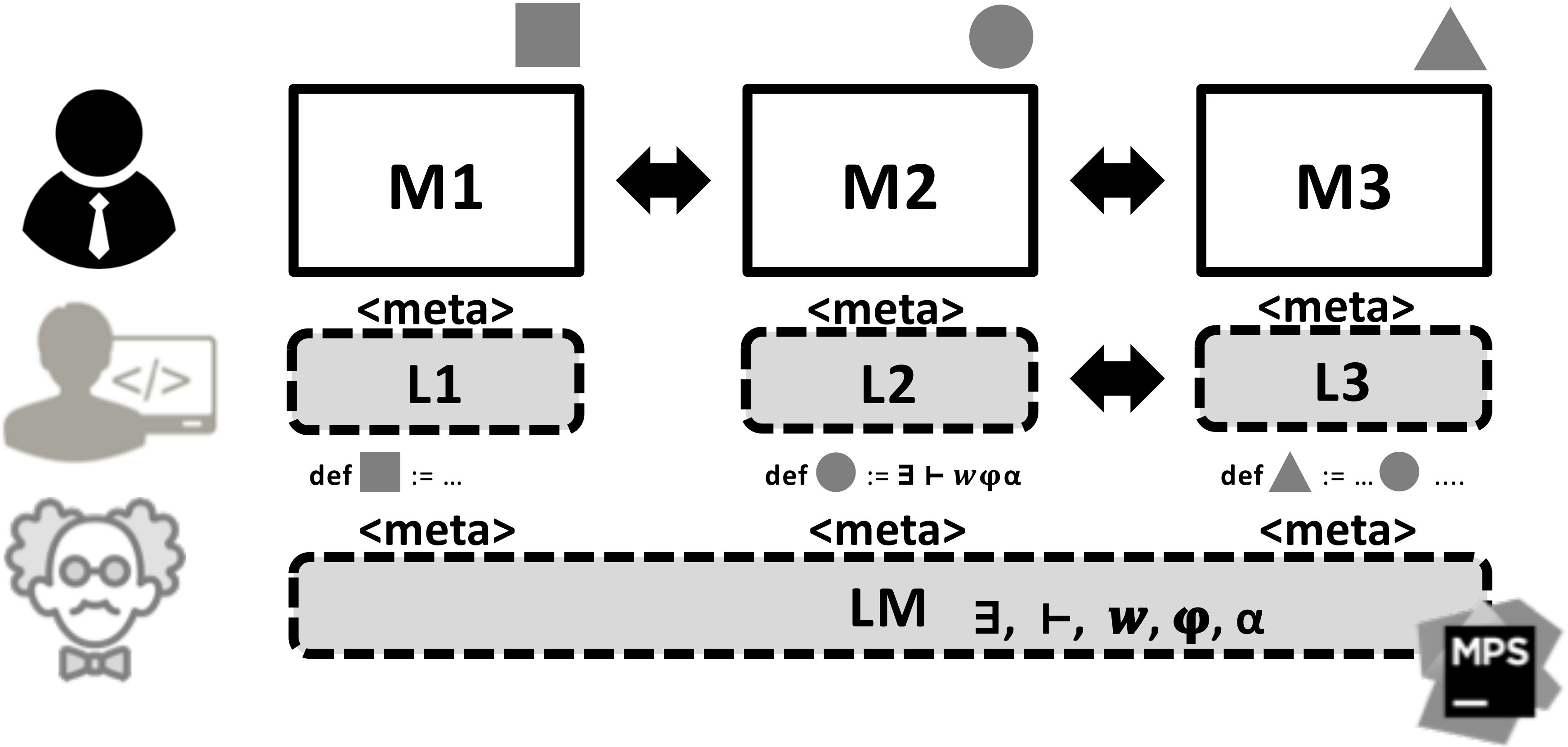
**Represent Fachlickeit precisely/formally,**
so you can analyze, test, simulate.

**Use “friendly“ languages,**
so domain experts can contribute directly.

Formal Language.
Checkable.
Understandable.

# DSL

Domain
Specific
Language



M1 ■

M2 ●

M3 ▲

<meta> L1

<meta> L2

<meta> L3

def ■ := …

def ● := ∃ ⊢ $w$φα

def ▲ := … ● ….

<meta> <meta> <meta>

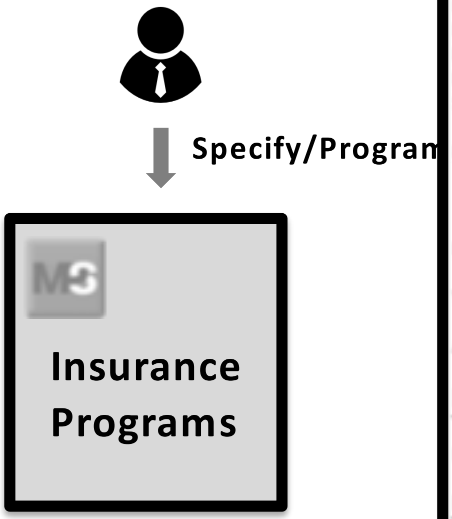
LM ∃, ⊢, $w$, φ, α

MPS

# 2

# Examples

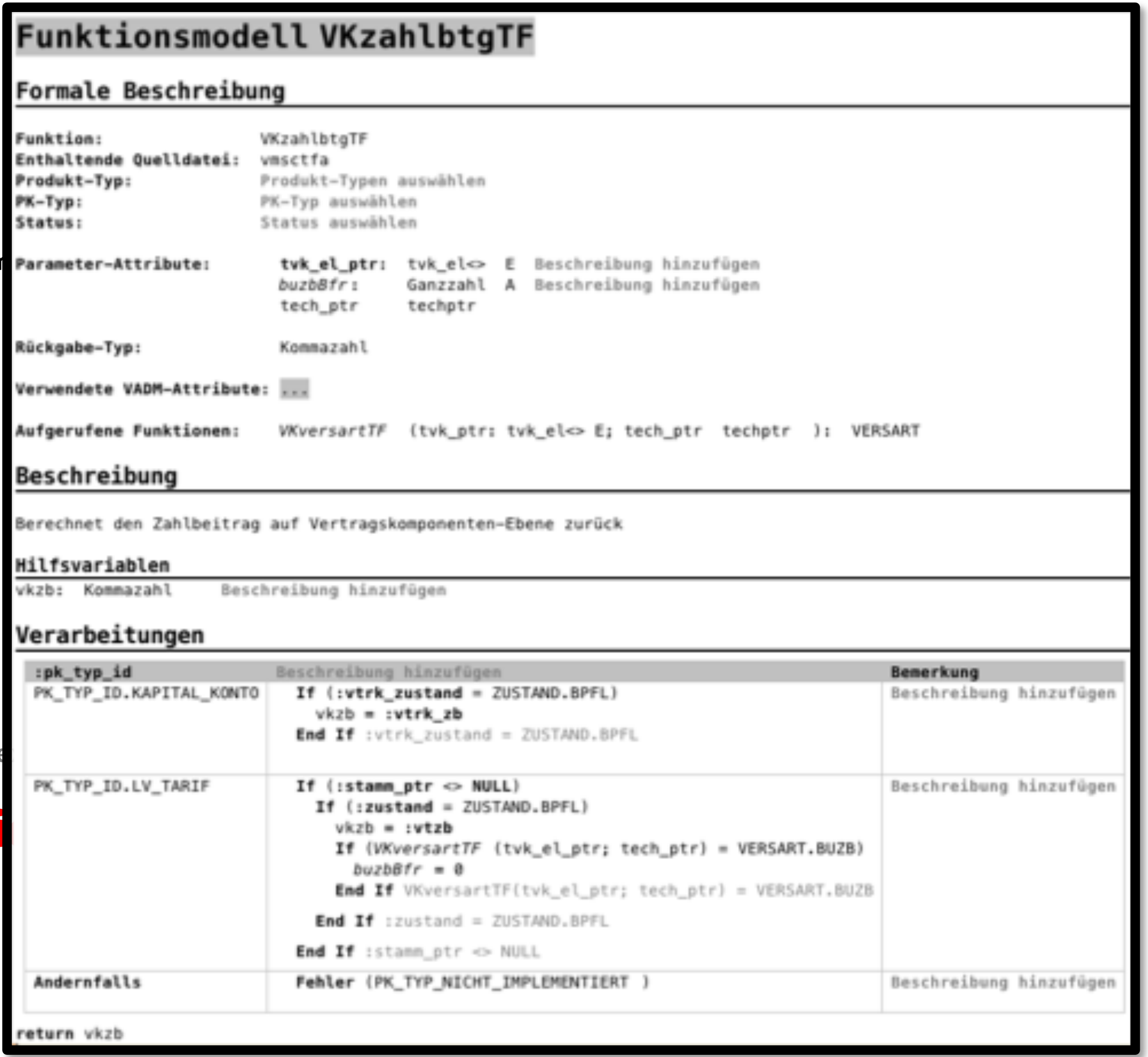
# Insurance Contracts



Write formal code in a DSL mixed with tables and text

Now with IDE support and e

**The same notati**



## 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
                            buzbBfr:     Ganzzahl   A  Beschreibung hinzufügen
                            tech_ptr     techptr

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

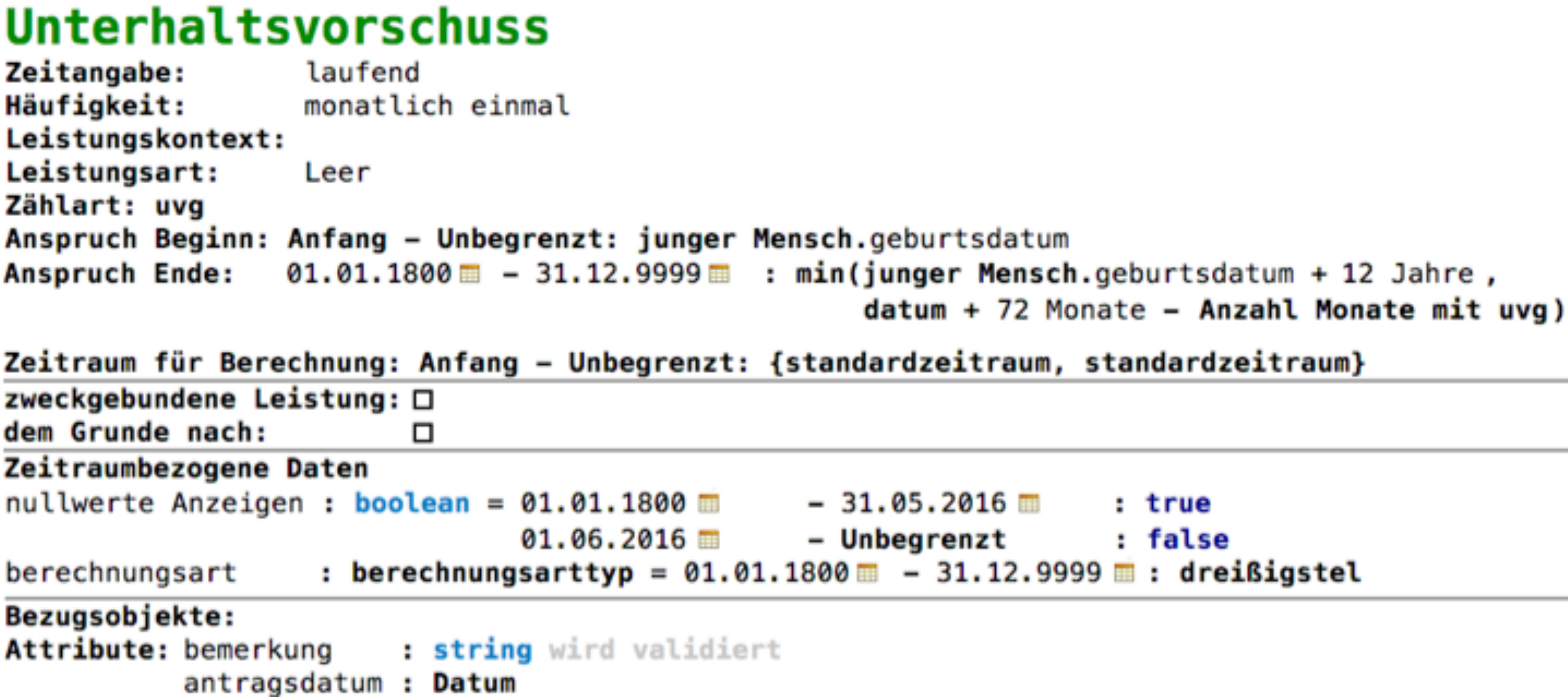
```
vkzb:  Kommazahl    Beschreibung hinzufügen
```

### Verarbeitungen

| :pk_typ_id | Beschreibung hinzufügen | Bemerkung |
|---|---|---|
| PK_TYP_ID.KAPITAL_KONTO | If (:vtrk_zustand = ZUSTAND.BPFL)<br>  vkzb = :vtrk_zb<br>End If :vtrk_zustand = ZUSTAND.BPFL | Beschreibung hinzufügen |
| PK_TYP_ID.LV_TARIF | If (:stamm_ptr <> NULL)<br>  If (:zustand = ZUSTAND.BPFL)<br>    vkzb = :vtzb<br>    If (VKversartTF (tvk_el_ptr; tech_ptr) = VERSART.BUZB)<br>      buzbBfr = 0<br>    End If VKversartTF(tvk_el_ptr; tech_ptr) = VERSART.BUZB<br>  End If :zustand = ZUSTAND.BPFL<br>End If :stamm_ptr <> NULL | Beschreibung hinzufügen |
| Andernfalls | Fehler (PK_TYP_NICHT_IMPLEMENTIERT ) | Beschreibung hinzufügen |

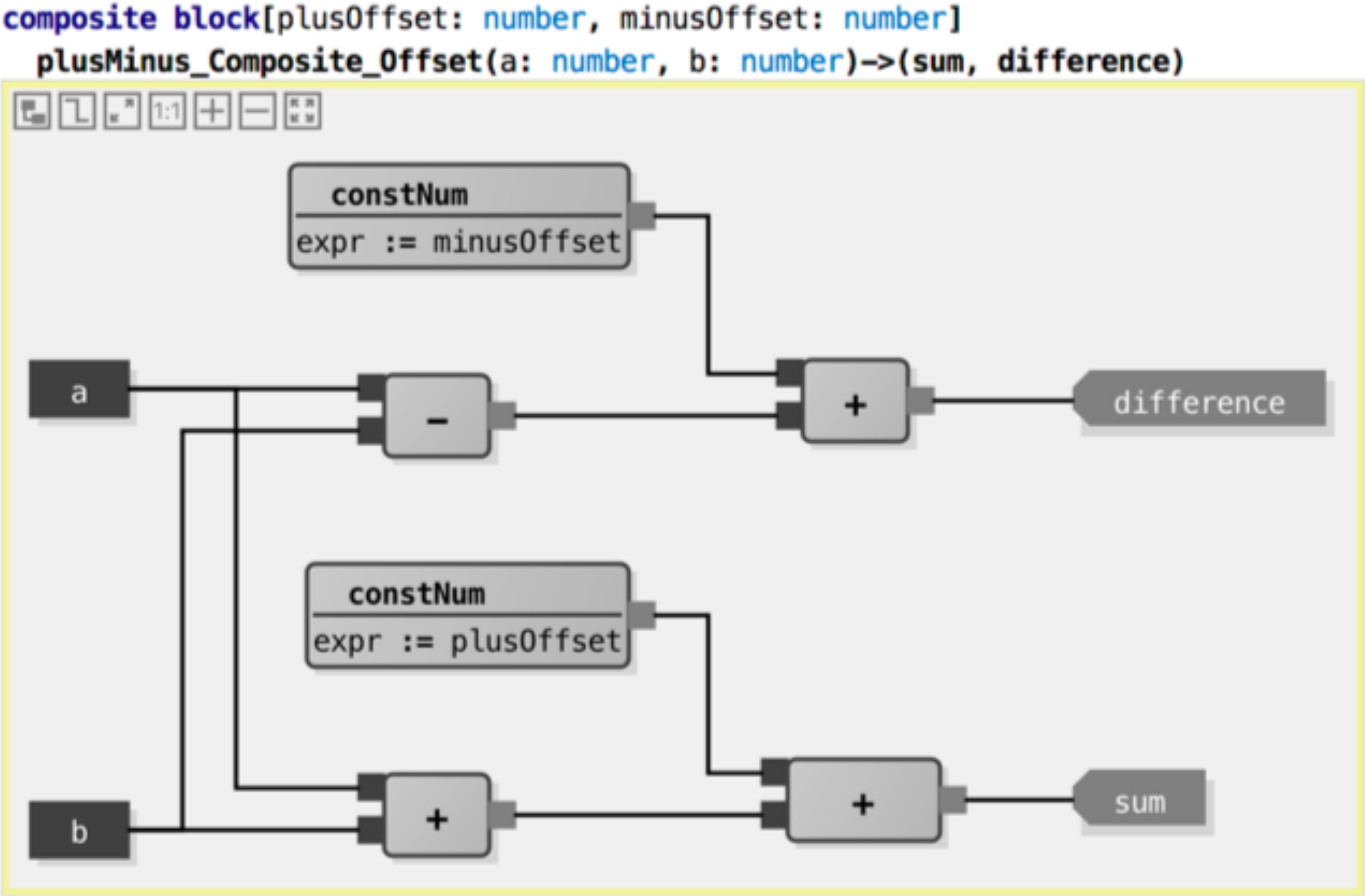
```
return vkzb
```

# Public Benefits



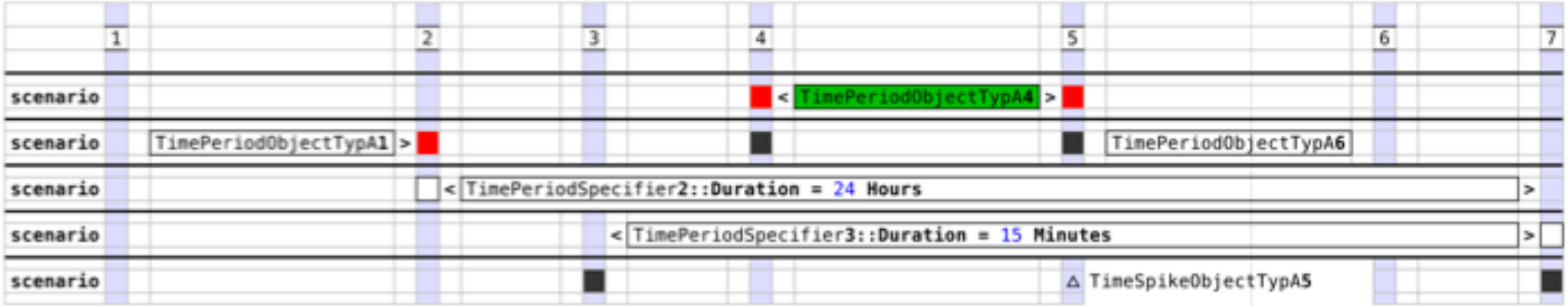
```
Unterhaltsvorschuss
Zeitangabe:        laufend
Häufigkeit:        monatlich einmal
Leistungskontext:
Leistungsart:      Leer
Zählart: uvg
Anspruch Beginn: Anfang - Unbegrenzt: junger Mensch.geburtsdatum
Anspruch Ende:   01.01.1800▦ - 31.12.9999▦  : min(junger Mensch.geburtsdatum + 12 Jahre ,
                                                  datum + 72 Monate - 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
Bezugsobjekte:
Attribute: bemerkung     : string wird validiert
           antragsdatum  : Datum
```

# Data Flow Programming



```
composite block[plusOffset: number, minusOffset: number]
  plusMinus_Composite_Offset(a: number, b: number)->(sum, difference)
```

# Tachograph Rules



1 2 3 4 5 6 7

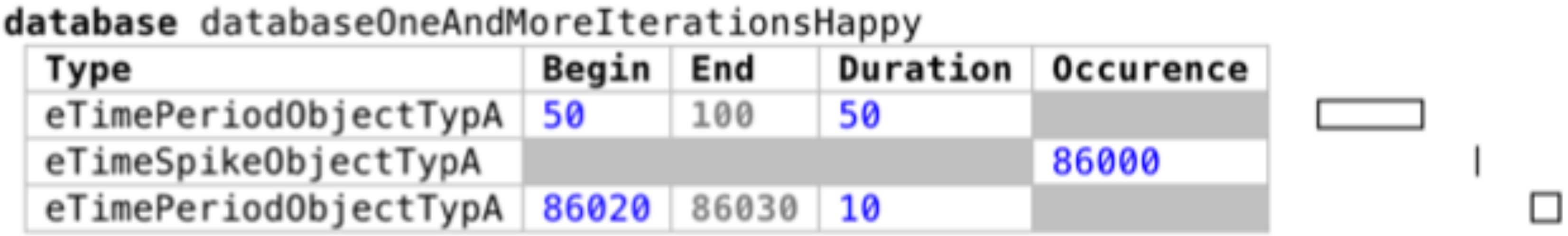
scenario 🟥 < TimePeriodObjectTypA4 > 🟥

scenario TimePeriodObjectTypA1 > 🟥 ■ ■ TimePeriodObjectTypA6

scenario □ < TimePeriodSpecifier2::Duration = 24 Hours >

scenario < TimePeriodSpecifier3::Duration = 15 Minutes > □

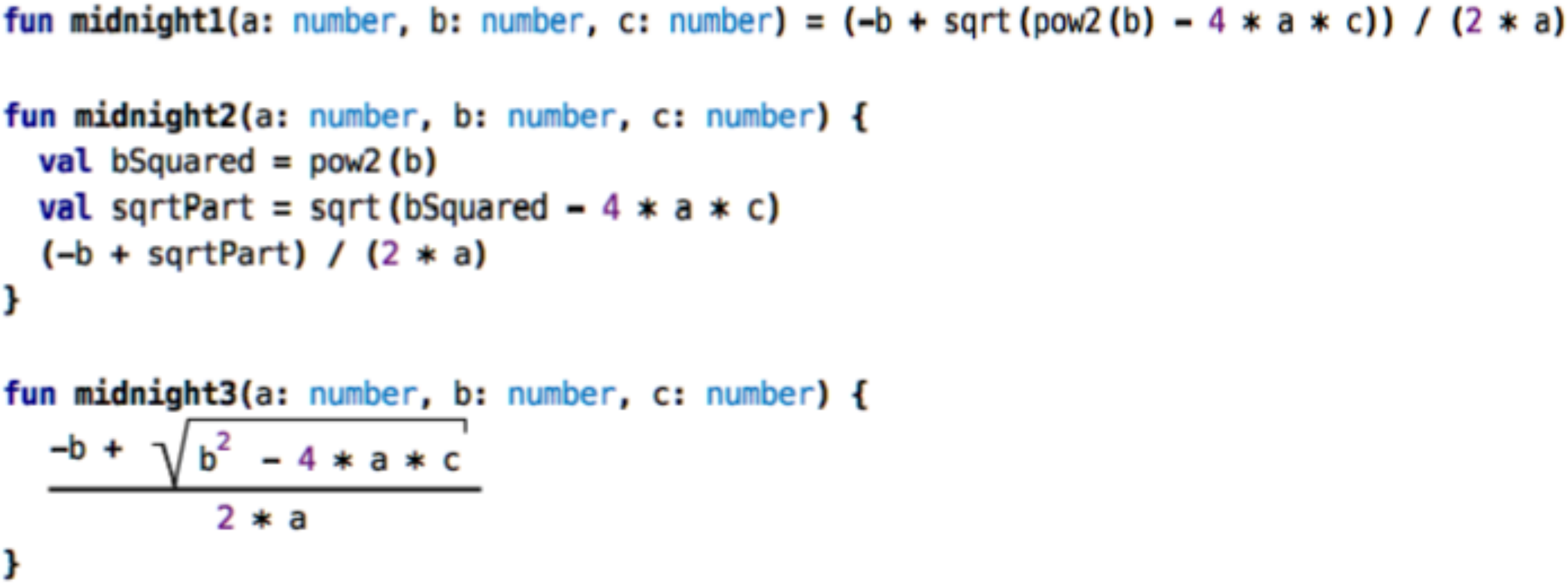
scenario ■ △ TimeSpikeObjectTypA5 ■



**database** databaseOneAndMoreIterationsHappy

| Type | Begin | End | Duration | Occurence |
|---|---|---|---|---|
| eTimePeriodObjectTypA | 50 | 100 | 50 | |
| eTimeSpikeObjectTypA | | | | 86000 |
| eTimePeriodObjectTypA | 86020 | 86030 | 10 | |

# Math

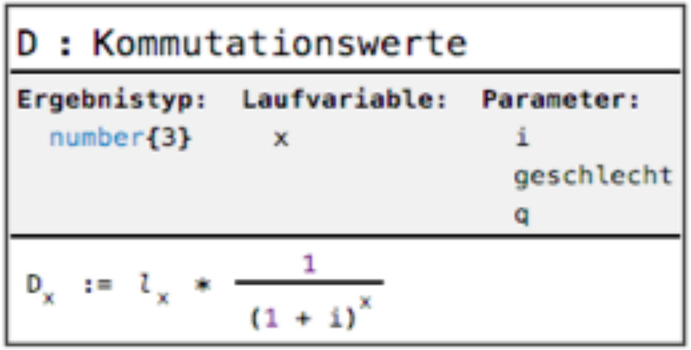


```
fun midnight1(a: number, b: number, c: number) = (-b + sqrt(pow2(b) - 4 * a * c)) / (2 * a)

fun midnight2(a: number, b: number, c: number) {
  val bSquared = pow2(b)
  val sqrtPart = sqrt(bSquared - 4 * a * c)
  (-b + sqrtPart) / (2 * a)
}

fun midnight3(a: number, b: number, c: number) {
```

$$\frac{-b + \sqrt{b^2 - 4 * a * c}}{2 * a}$$

```
}
```

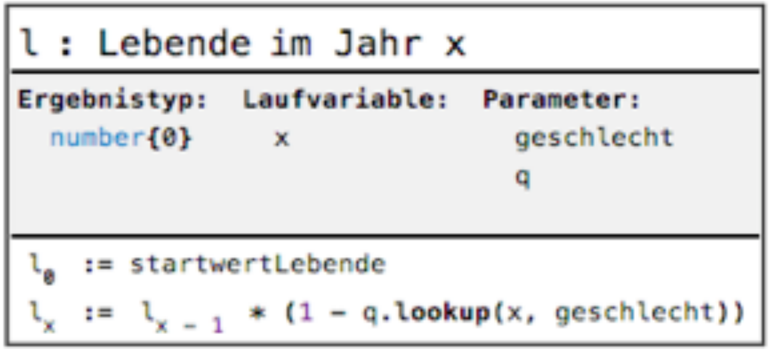
# Insurance Math



D : Kommutationswerte

| Ergebnistyp: | Laufvariable: | Parameter: |
|---|---|---|
| number{3} | x | i<br>geschlecht<br>q |

$D_x := l_x * \frac{1}{(1 + i)^x}$



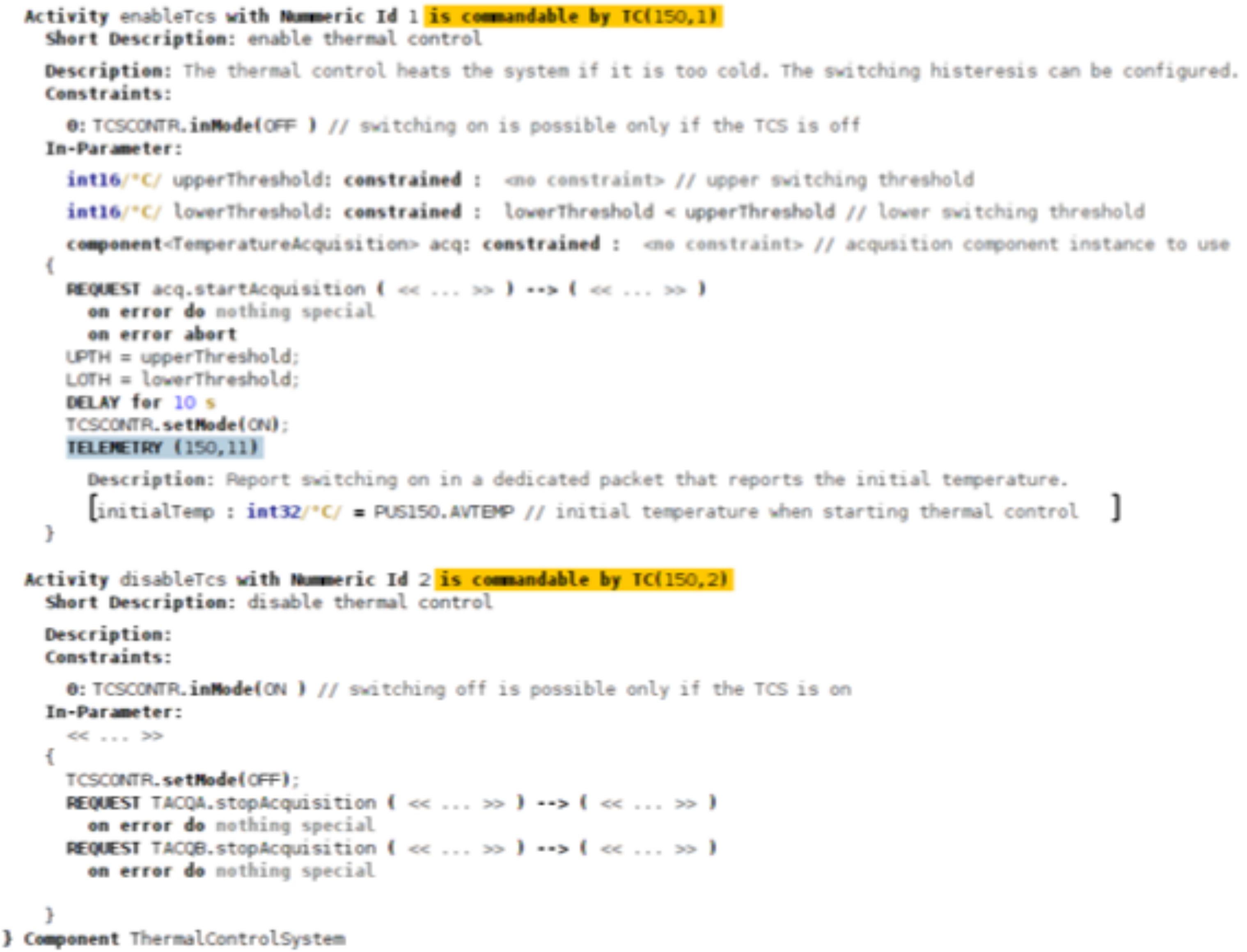
l : Lebende im Jahr x

| Ergebnistyp: | Laufvariable: | Parameter: |
|---|---|---|
| number{0} | x | geschlecht<br>q |

$l_0$ := startwertLebende

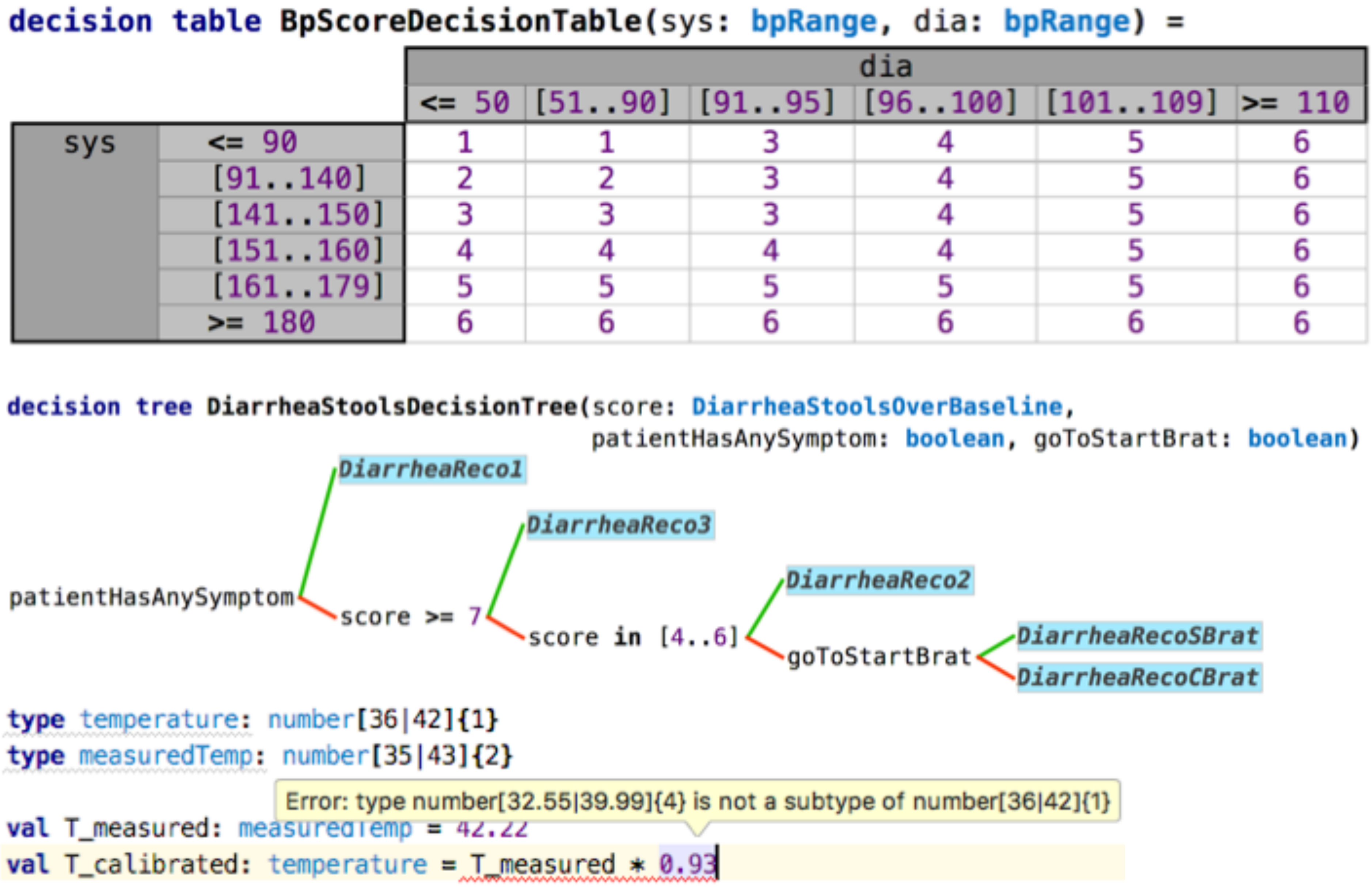
$l_x := l_{x - 1} * (1 - q.\mathbf{lookup}(x, geschlecht))$

# Satellite Software



```
  Activity enableTcs with Nummeric 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 histeresis can be configured.
    Constraints:

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

      int16/°C/ upperThreshold: constrained :  <no constraint> // upper switching threshold

      int16/°C/ lowerThreshold: constrained :  lowerThreshold < upperThreshold // lower switching threshold

      component<TemperatureAcquisition> acq: constrained :  <no constraint> // acqusition component instance to use
    {
      REQUEST acq.startAcquisition ( << ... >> ) --> ( << ... >> )
        on error do nothing special
        on error abort
      UPTH = upperThreshold;
      LOTH = lowerThreshold;
      DELAY for 10 s
      TCSCONTR.setMode(ON);
      TELEMETRY (150,11)

        Description: Report switching on in a dedicated packet that reports the initial temperature.
        [initialTemp : int32/°C/ = PUS150.AVTEMP // initial temperature when starting thermal control   ]
    }

  Activity disableTcs with Nummeric 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:
      << ... >>
    {
      TCSCONTR.setMode(OFF);
      REQUEST TACQA.stopAcquisition ( << ... >> ) --> ( << ... >> )
        on error do nothing special
      REQUEST TACQB.stopAcquisition ( << ... >> ) --> ( << ... >> )
        on error do nothing special

    }
} Component ThermalControlSystem
```

# Healthcare



```
decision table BpScoreDecisionTable(sys: bpRange, dia: bpRange) =
```

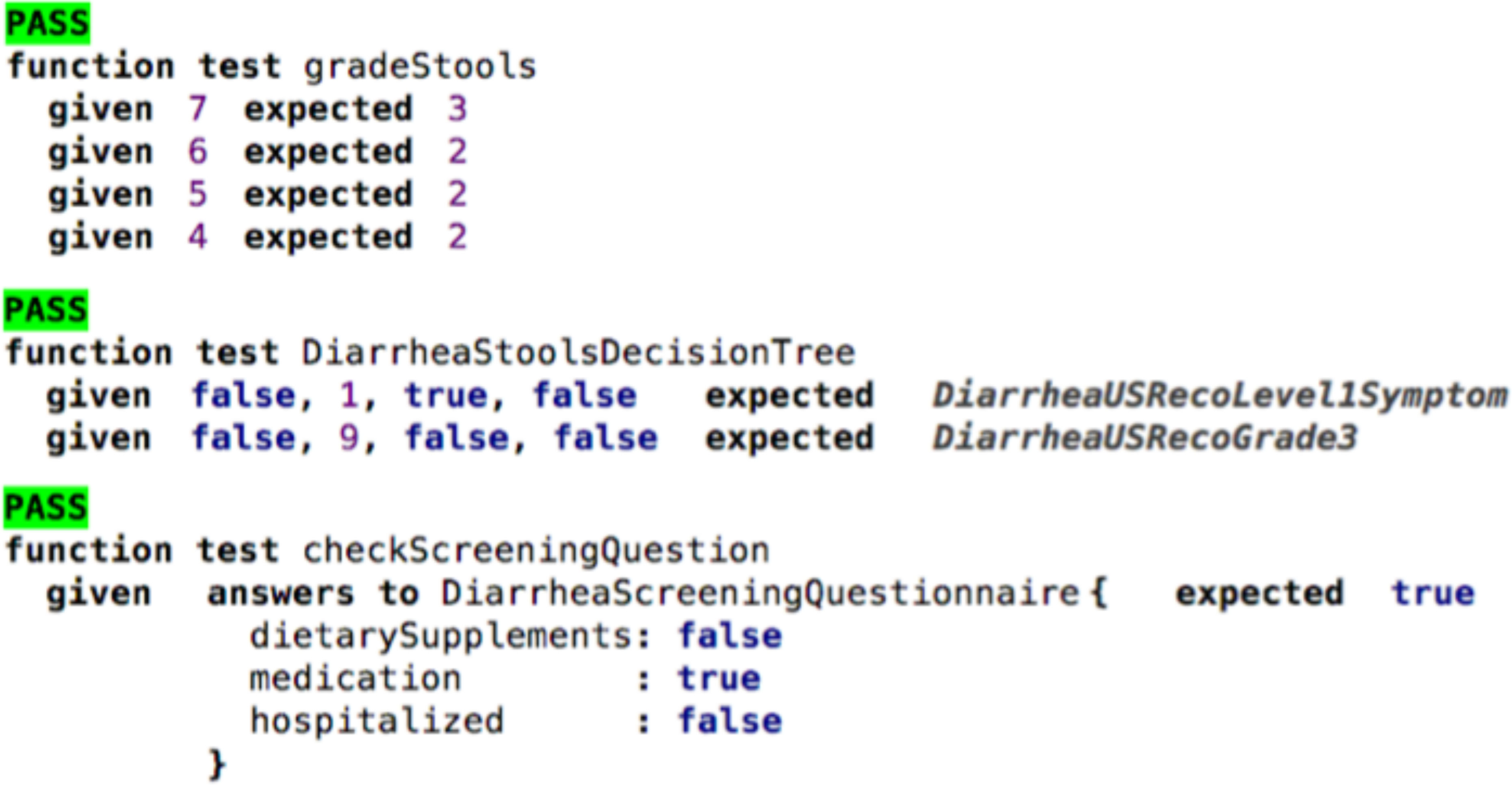
| | | dia | | | | | |
|---|---|---|---|---|---|---|---|
| | | <= 50 | [51..90] | [91..95] | [96..100] | [101..109] | >= 110 |
| sys | <= 90 | 1 | 1 | 3 | 4 | 5 | 6 |
| | [91..140] | 2 | 2 | 3 | 4 | 5 | 6 |
| | [141..150] | 3 | 3 | 3 | 4 | 5 | 6 |
| | [151..160] | 4 | 4 | 4 | 4 | 5 | 6 |
| | [161..179] | 5 | 5 | 5 | 5 | 5 | 6 |
| | >= 180 | 6 | 6 | 6 | 6 | 6 | 6 |

```
decision tree DiarrheaStoolsDecisionTree(score: DiarrheaStoolsOverBaseline,
                                          patientHasAnySymptom: boolean, goToStartBrat: boolean)
```

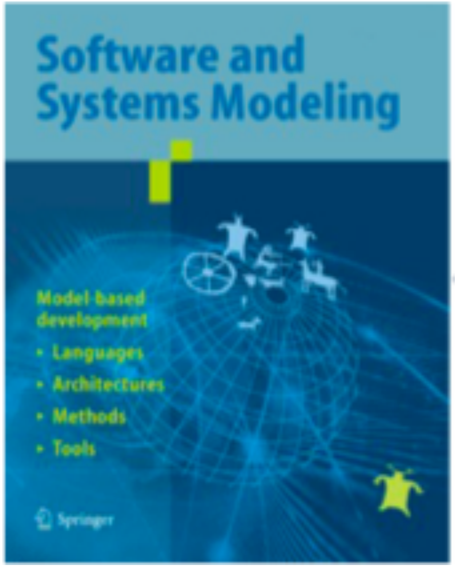
```
type temperature: number[36|42]{1}
type measuredTemp: number[35|43]{2}

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

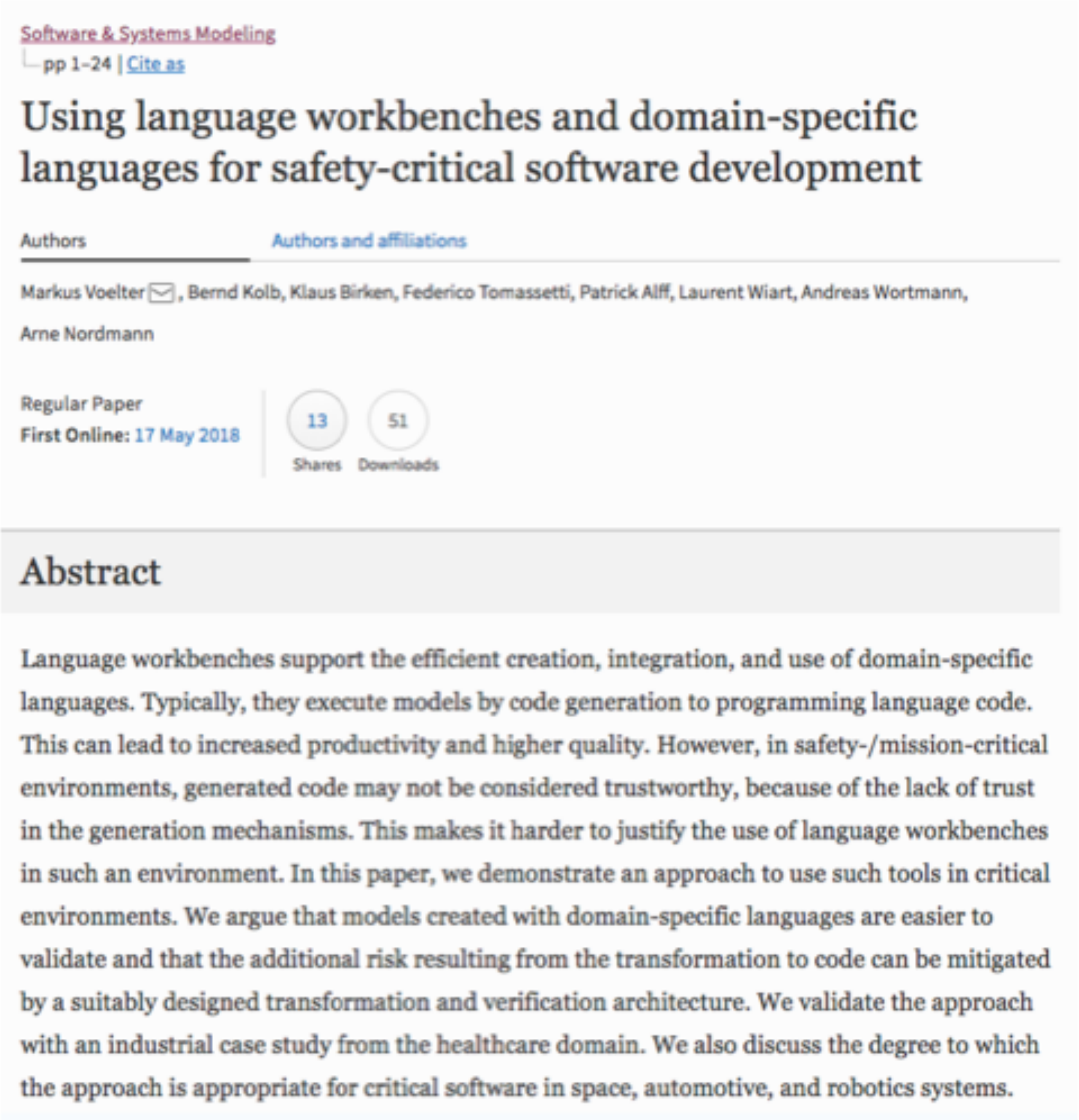
Error: type number[32.55|39.99]{4} is not a subtype of number[36|42]{1}

# Healthcare



```
PASS
function test gradeStools
  given 7  expected  3
  given 6  expected  2
  given 5  expected  2
  given 4  expected  2

PASS
function test DiarrheaStoolsDecisionTree
  given  false, 1, true, false    expected   DiarrheaUSRecoLevel1Symptom
  given  false, 9, false, false   expected   DiarrheaUSRecoGrade3

PASS
function test checkScreeningQuestion
  given   answers to DiarrheaScreeningQuestionnaire{   expected  true
            dietarySupplements: false
            medication        : true
            hospitalized      : false
          }
```





Software & Systems Modeling
pp 1–24 | Cite as

# Using language workbenches and domain-specific languages for safety-critical software development

Authors | Authors and affiliations

Markus Voelter ✉, Bernd Kolb, Klaus Birken, Federico Tomassetti, Patrick Alff, Laurent Wiart, Andreas Wortmann, Arne Nordmann

Regular Paper
**First Online:** 17 May 2018

13 Shares | 51 Downloads

## 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.

**http://voelter.de/data/pub/MPS-in-Safety-1.0.pdf**

# 3

# MPS Demo

# (Meta-) Tooling

Language Workbench

Open Source, by Jetbrains

Very Powerful

Used for years by itemis and others

Vast Experience

# (Meta-) Tooling

MPS

Language Workbench

Open Source, by Jetbrains

Very Powerful

Used for years by itemis and others

Vast Experience

JET BRAINS

Audi

OHB

SIEMENS

océ

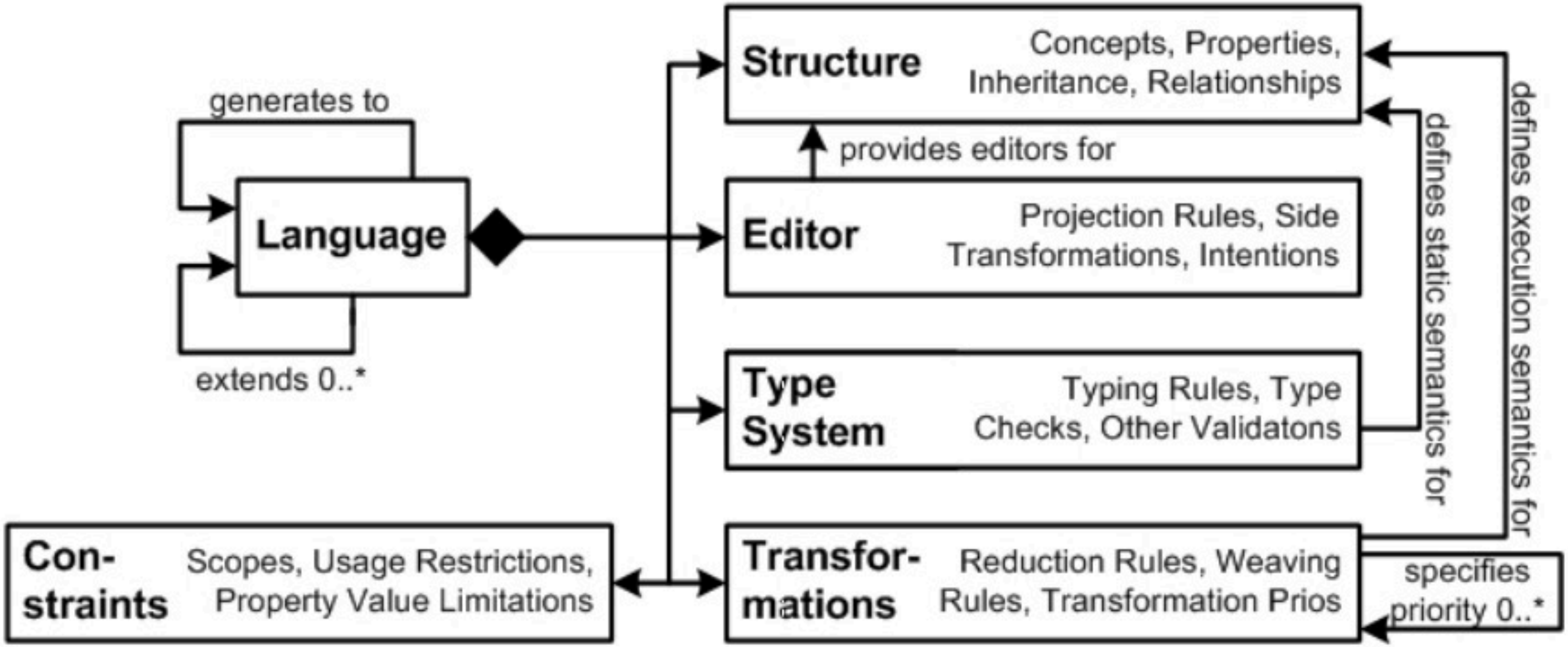
workday.

ZURICH

VOLUNTIS

Belastingdienst

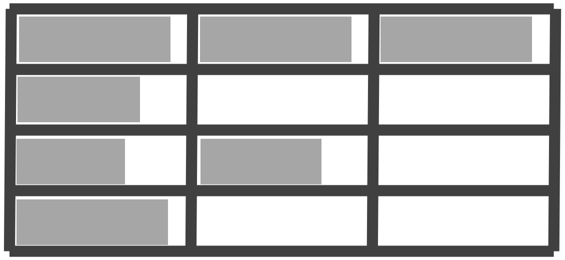
# MPS: Language Toolkit

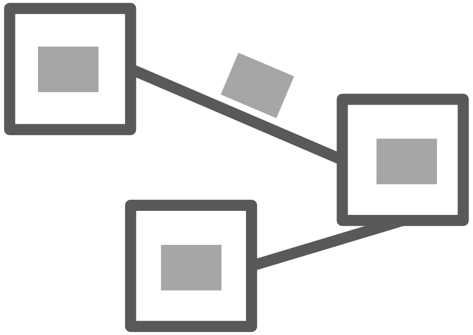


generates to
Language
extends 0..*
Structure
Concepts, Properties, Inheritance, Relationships
provides editors for
Editor
Projection Rules, Side Transformations, Intentions
Type System
Typing Rules, Type Checks, Other Validatons
Con-straints
Scopes, Usage Restrictions, Property Value Limitations
Transfor-mations
Reduction Rules, Weaving Rules, Transformation Prios
defines static semantics for
defines execution semantics for
specifies priority 0..*

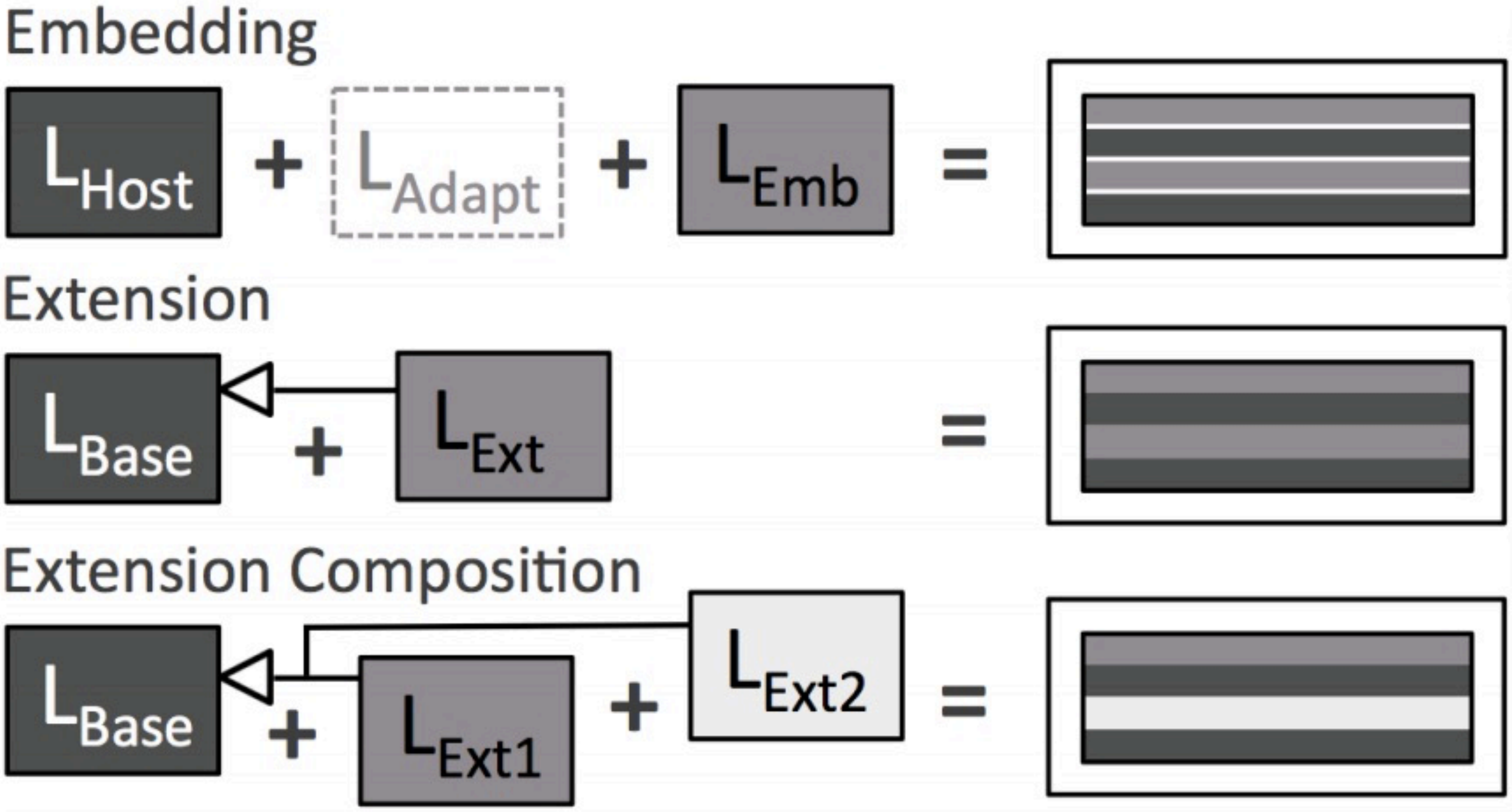
**+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...**

# MPS: Notational Freedom



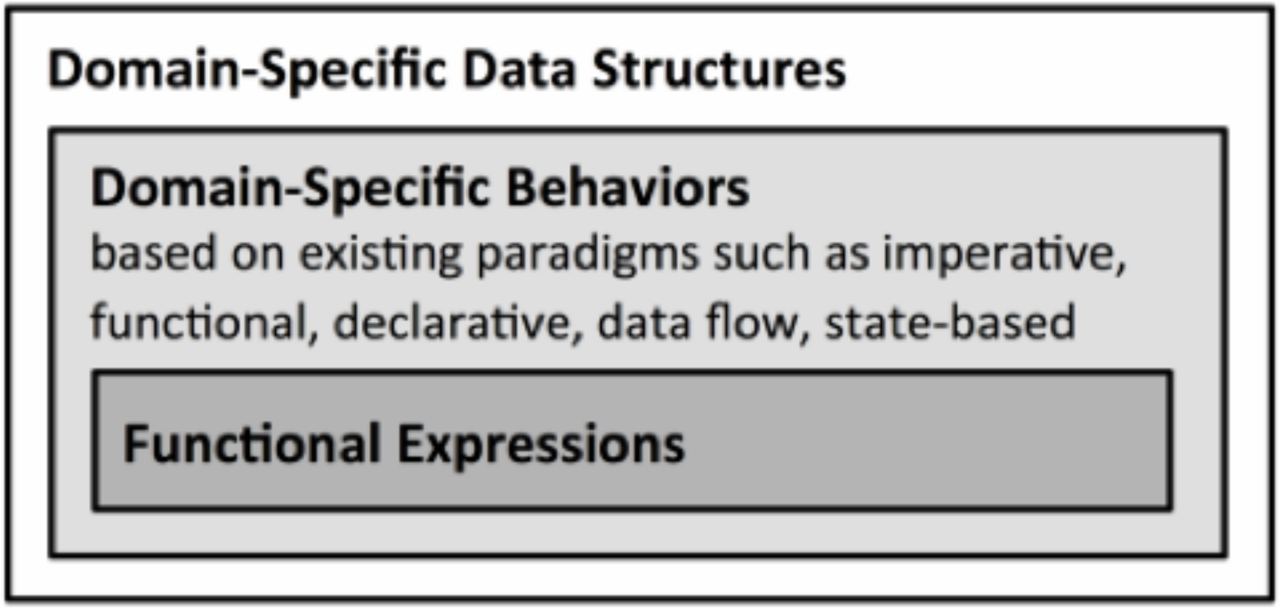


# MPS: Language Composition


Embedding
$L_{Host}$ + $L_{Adapt}$ + $L_{Emb}$ =
Extension
$L_{Base}$ + $L_{Ext}$ =
Extension Composition
$L_{Base}$ + $L_{Ext1}$ + $L_{Ext2}$ =

# MPS: Language Composition

**Embedding/Extending the KernelF functional language is key to DSL development productivity.**



Domain-Specific Data Structures

Domain-Specific Behaviors

based on existing paradigms such as imperative, functional, declarative, data flow, state-based

Functional Expressions

# Other Language Workbenches

| | |
|---|---|
| {s} spoofax | TU Delft |
| xtext | itemis/Typefox |
| Rascal | CWI Amsterdam |
| The Whole Platform | Solmi/Persiani |

ELSEVIER

Computer Languages, Systems & Structures

Volume 44, Part A, December 2015, Pages 24–47

Special issue on the 6th and 7th International Conference on Software Language Engineering (SLE 2013 and SLE 2014)

# Evaluating and Comparing Language Workbenches

## *Existing Results and Benchmarks for the Future*

Sebastian Erdweg[d], Tijs van der Storm[a], Markus Völter[e], Laurence Tratt[b], Remi Bosman[f], William R. Cook[c], Albert Gerritsen[f], Angelo Hulshout[g], Steven Kelly[h], Alex Loh[c], Gabriël Konat[1], Pedro J. Molina[j], Martin Palatnik[f], Risto Pohjonen[h], Eugen Schindler[f], Klemens Schindler[f], Riccardo Solmi[1], Vlad Vergu[1], Eelco Visser[1], Kevin van der Vlist[k], Guido Wachsmuth[1], Jimi van der Woning[l]

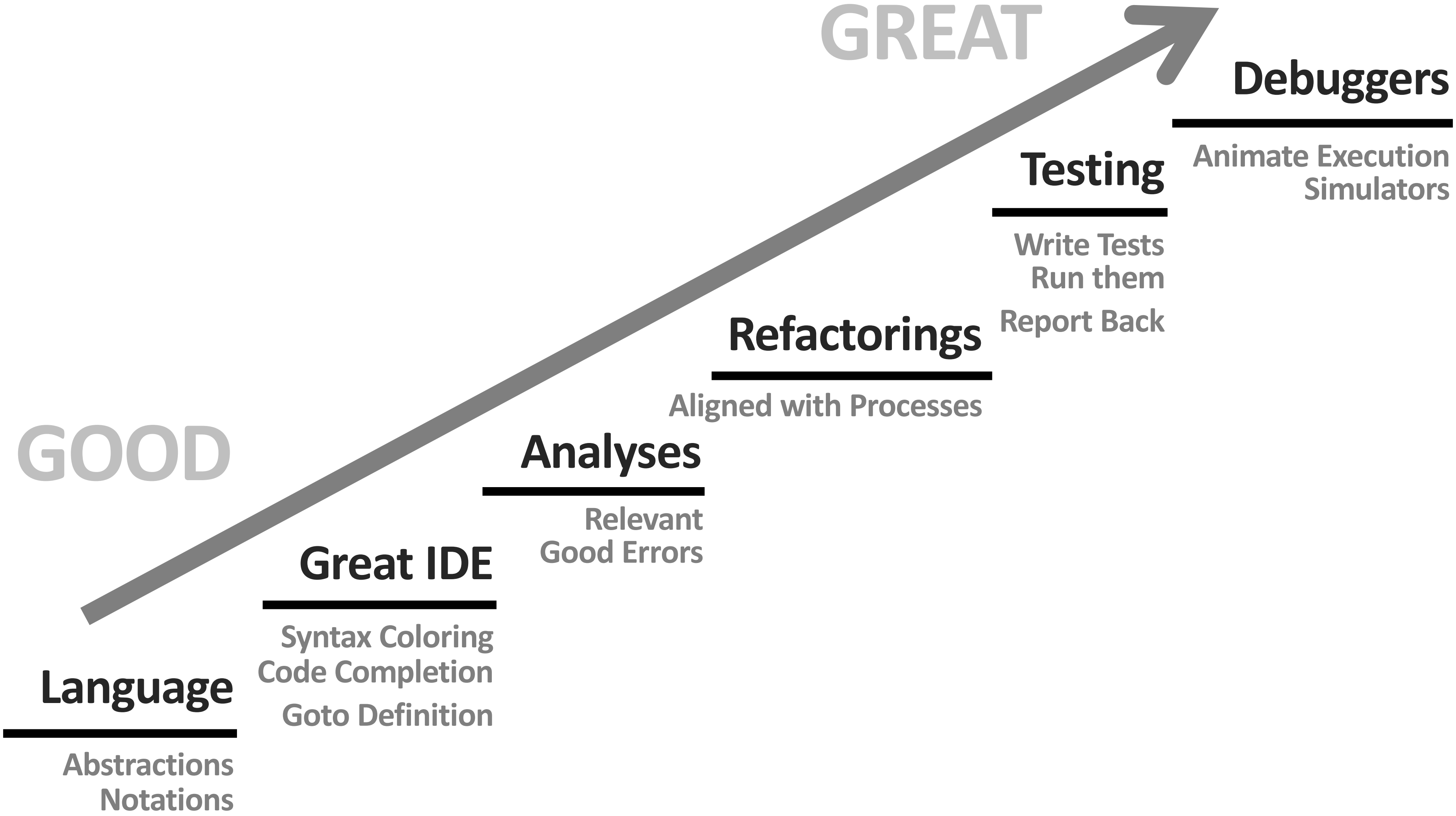
[a] *CWI, The Netherlands*
[b] *King's College London, UK*
[c] *University of Texas at Austin, US*
[d] *TU Darmstadt, Germany*
[e] *voelter.de, Stuttgart, Germany*
[f] *Sioux, Eindhoven, The Netherlands*
[g] *Delphino Consultancy*
[h] *MetaCase, Jyväskylä, Finland*
[i] *TU Delft, The Netherlands*
[j] *Icinetic, Sevilla, Spain*
[k] *Sogyo, De Bilt, The Netherlands*
[l] *Young Colfield, Amsterdam, The Netherlands*

http://voelter.de/data/pub/LWB-ResultsAndBenchmarks.pdf

# 4

# Lessons Learned

# A Language
# is not Enough



GOOD

**Language**

Abstractions
Notations

**Great IDE**

Syntax Coloring
Code Completion
Goto Definition

**Analyses**

Relevant
Good Errors

**Refactorings**

Aligned with Processes

**Testing**

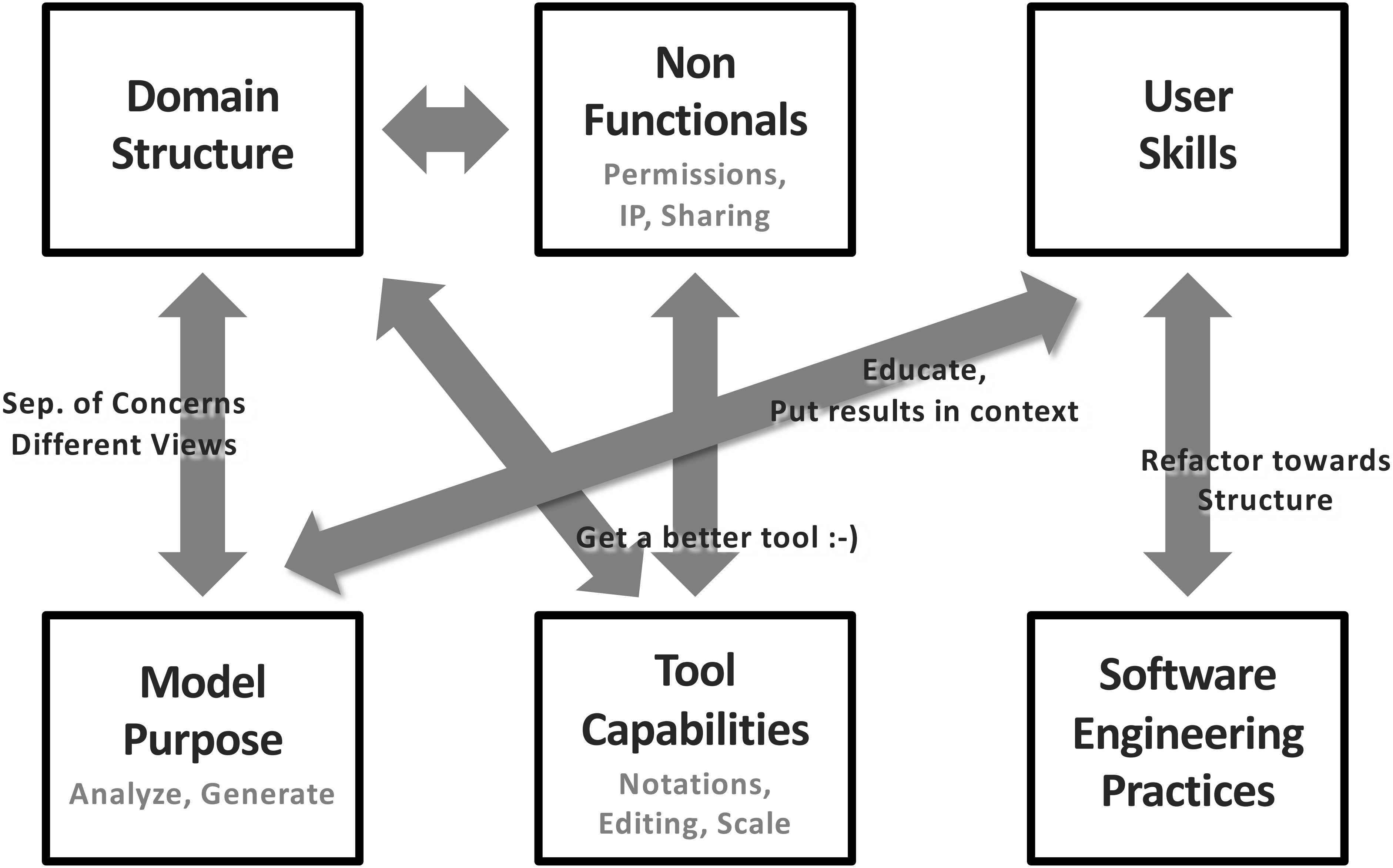
Write Tests
Run them
Report Back

**Debuggers**

Animate Execution
Simulators

GREAT

# Influences on the Language



Domain Structure

Non Functionals

Permissions, IP, Sharing

User Skills

Sep. of Concerns Different Views

Educate, Put results in context

Refactor towards Structure

Get a better tool :-)

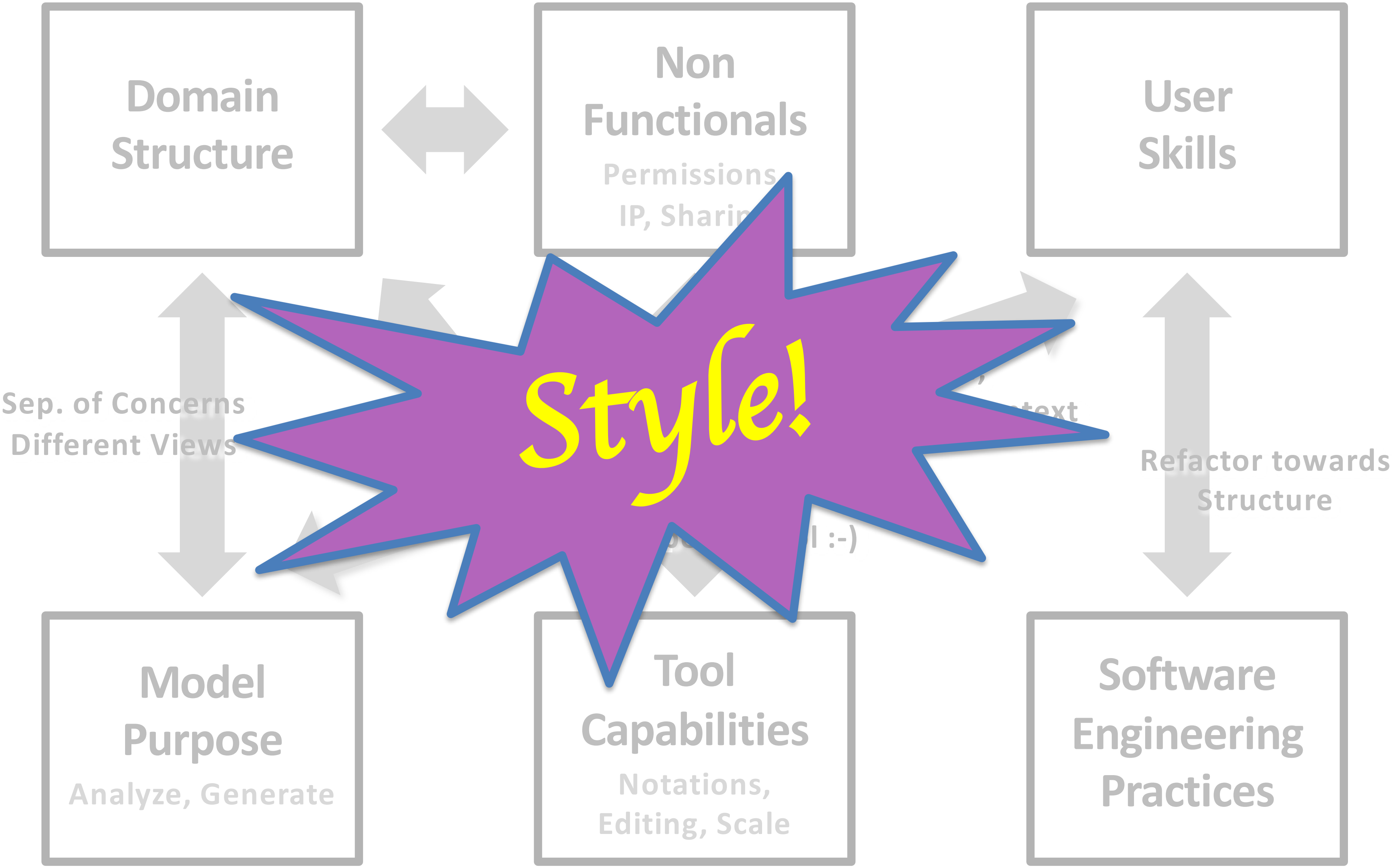
Model Purpose

Analyze, Generate

Tool Capabilities

Notations, Editing, Scale

Software Engineering Practices



Domain Structure

Non Functionals

Permissions

IP, Sharin

User Skills

Sep. of Concerns

Different Views

Style!

text

Refactor towards Structure

l :-)

Model Purpose

Analyze, Generate

Tool Capabilities

Notations, Editing, Scale

Software Engineering Practices

# How to make People precise?

Formulas, Rules
Data Structures
Tables
Values

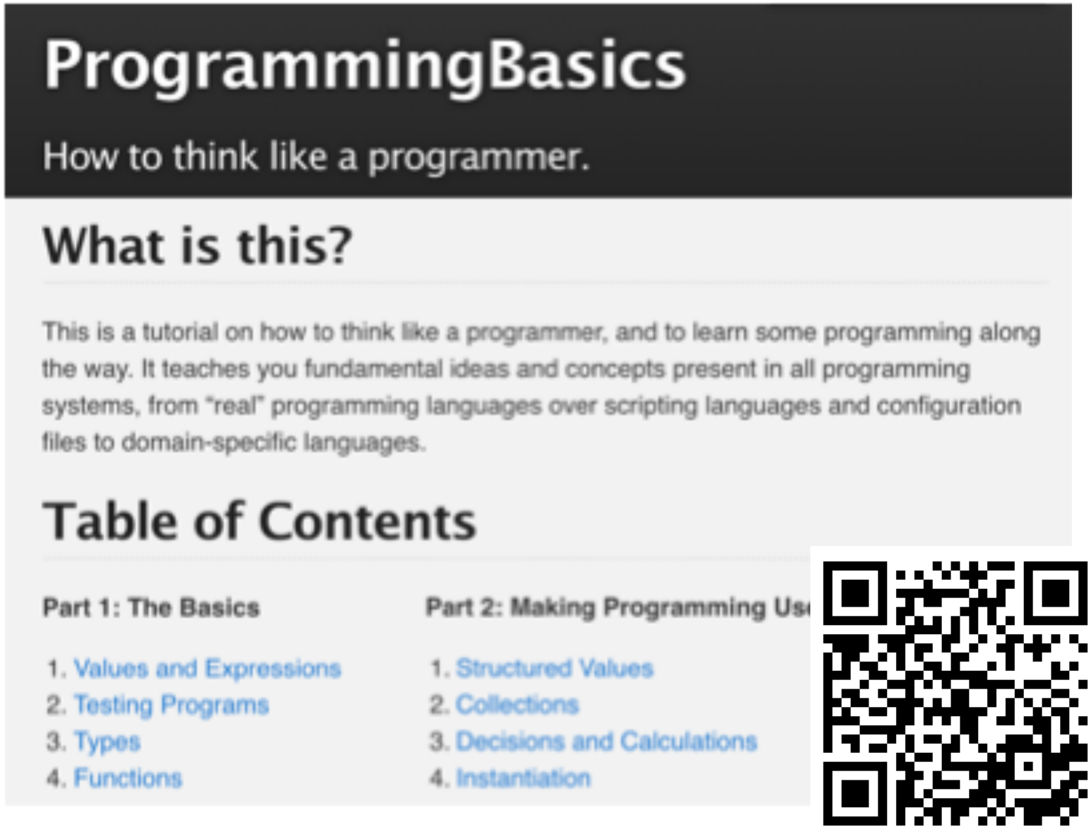
Precision

!=

Programming

Performance
Scalability
Robustness
Deployment

# Training is required.



## ProgrammingBasics

How to think like a programmer.

### What is this?

This is a tutorial on how to think like a programmer, and to learn some programming along the way. It teaches you fundamental ideas and concepts present in all programming systems, from "real" programming languages over scripting languages and configuration files to domain-specific languages.

### Table of Contents

Part 1: The Basics

Part 2: Making Programming Us

# Skills?

“Organizations do not have the necessary skills. True. But... AI Big Data Agile ☁ REST”

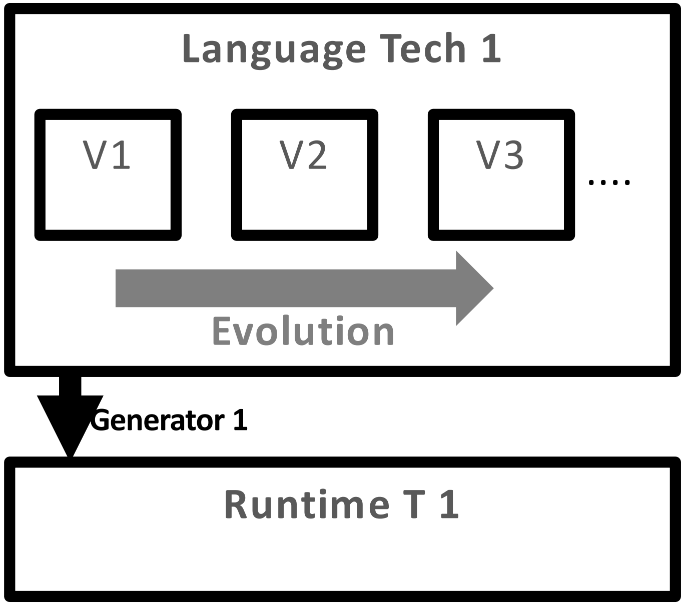
So build it. Evolve. Hire. Buy.

I LIKE TO CON PEOPLE. AND I LIKE TO INSULT PEOPLE.
IF YOU COMBINE CON AND INSULT, YOU GET "CONSULT."
I'M HERE TO CONSULT YOU.
IT SOUNDS EXPENSIVE AND DEMEANING. ... OKAY.
5/21/90 © 1990 United Feature Syndicate, Inc.

# Is this the next legacy system?

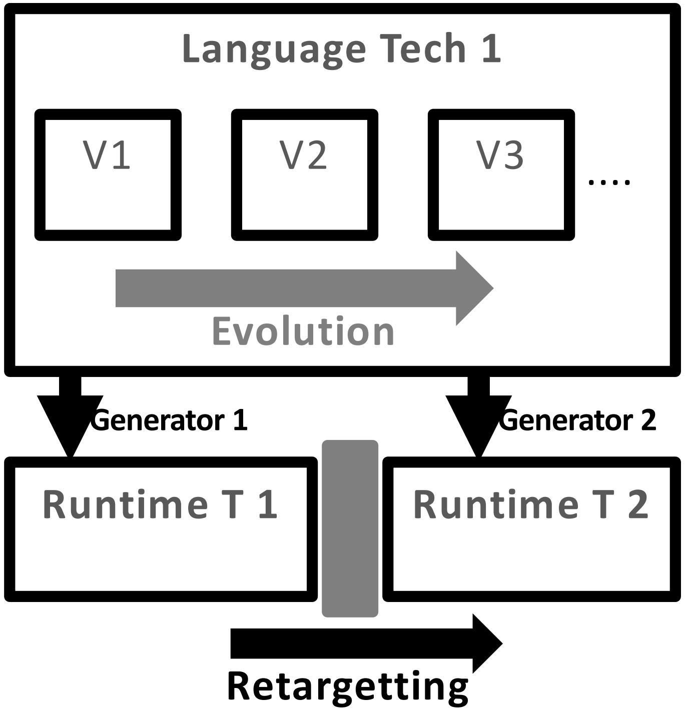
> Today‘s software is tomorrow‘s legacy system.

**Or is it?**



Existing models become incompatble with new language

⇒ **Language Versions**
**Migration Scripts**

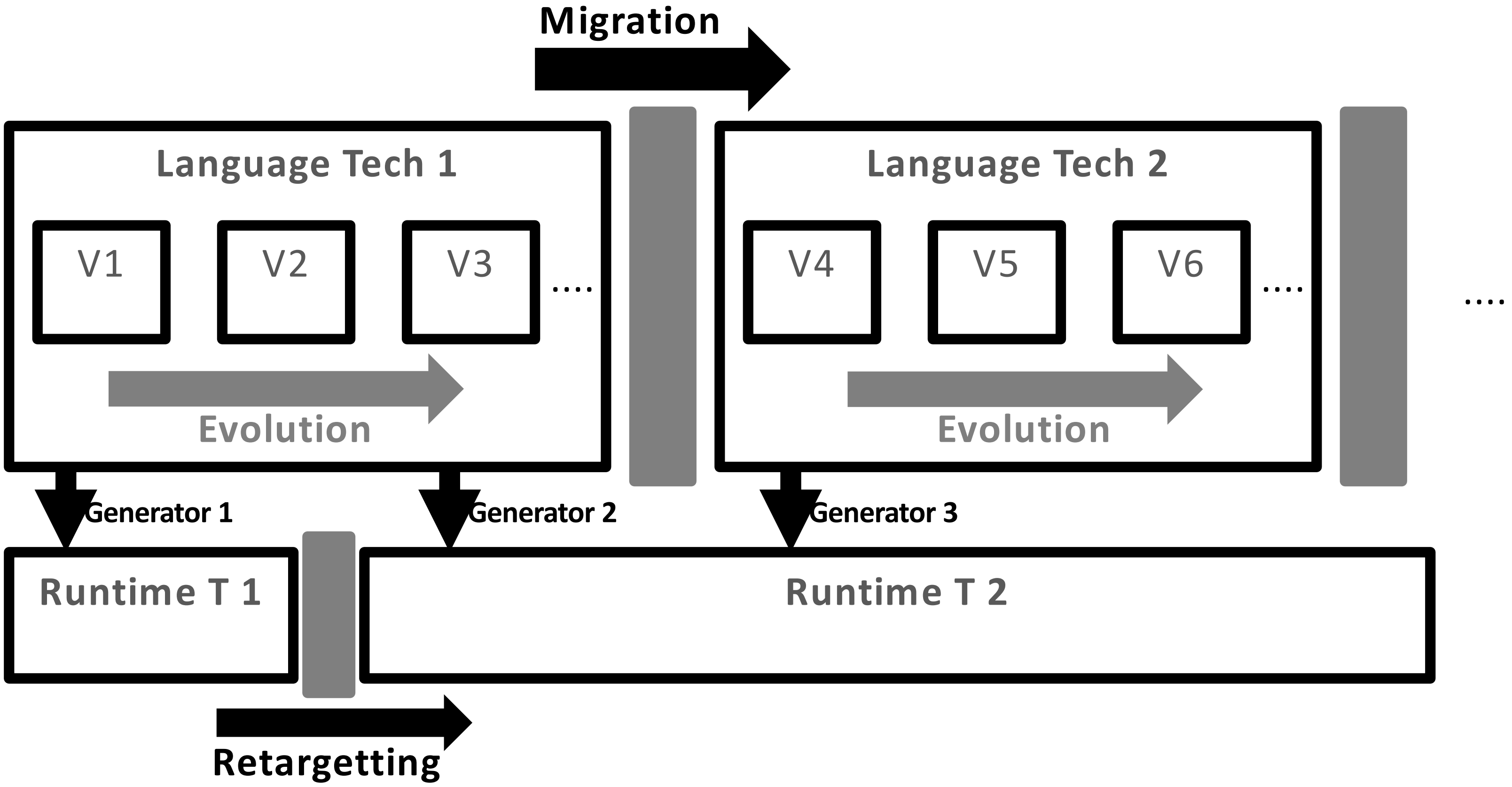


Runtime Tech outdated, uncool or slow

⇒ **Keep Lang Technology**

**Keep Models**

**Build new Generator**



Language Tech outdated, uncool

⇒ **Build new Tool**

**Migrate Data**

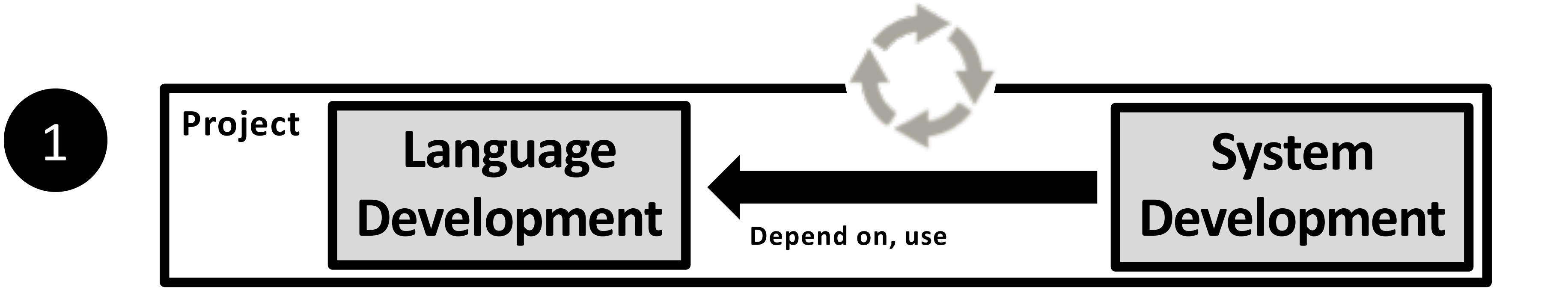
**Feasible, because it well-defined domain semantics and free from „technology stuff"**

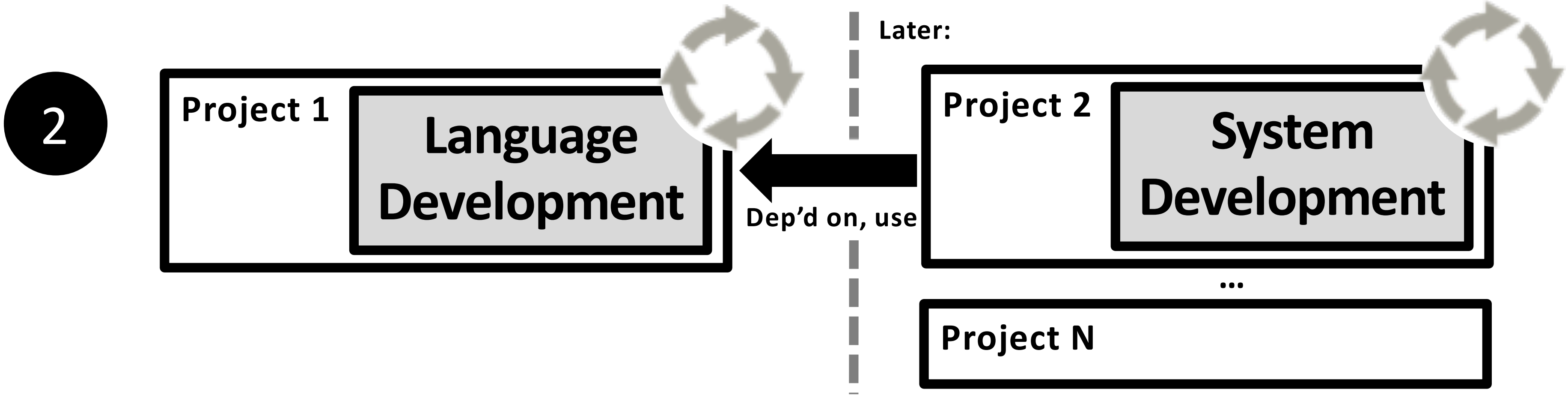
> Today‘s software is tomorrow‘s legacy system.

**No, it is not.**

# In conflict with Agile?

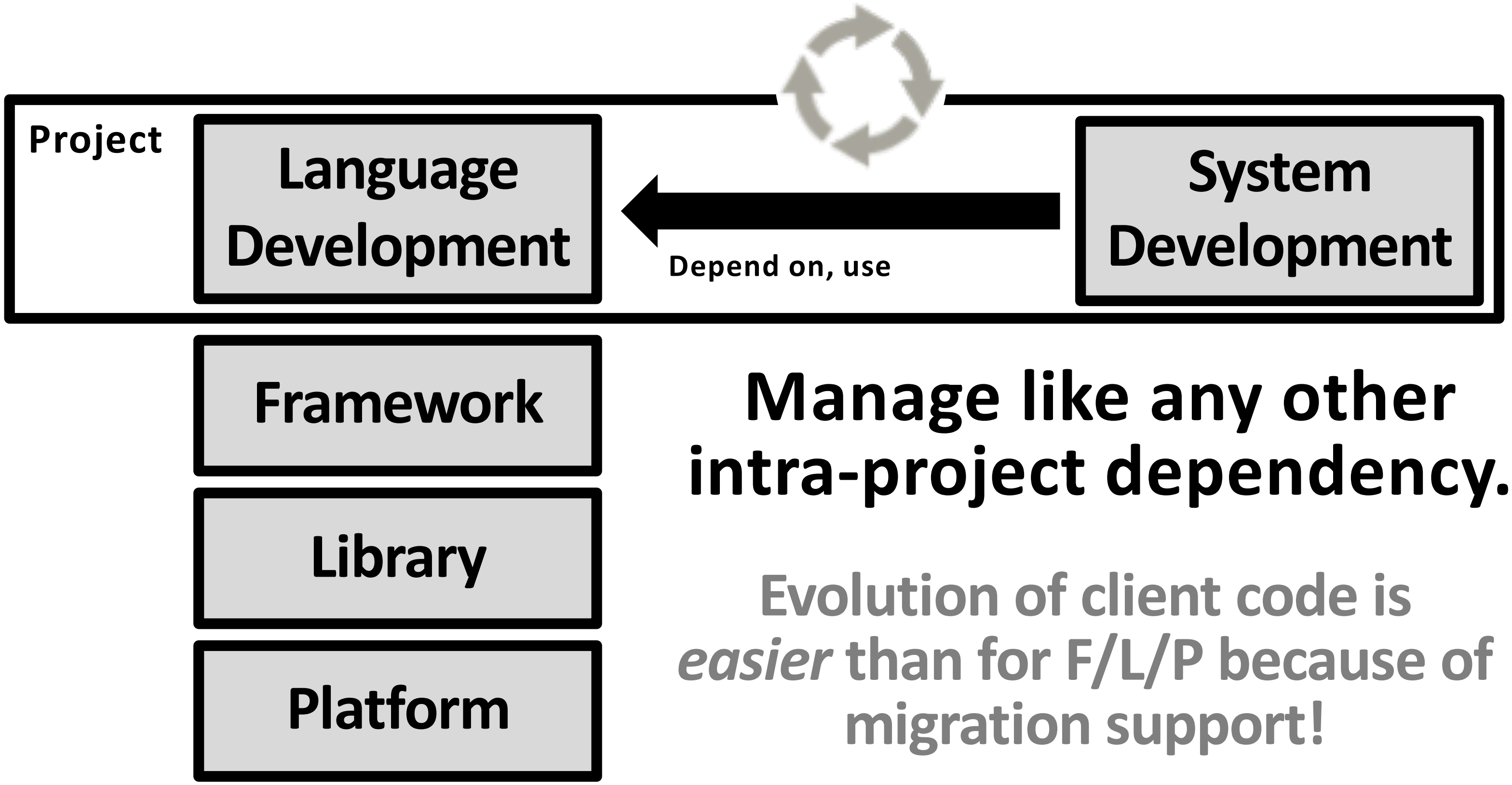
# „MD* and Agile is in Conflict.“





# „MD* and Agile is in Conflict.“

1

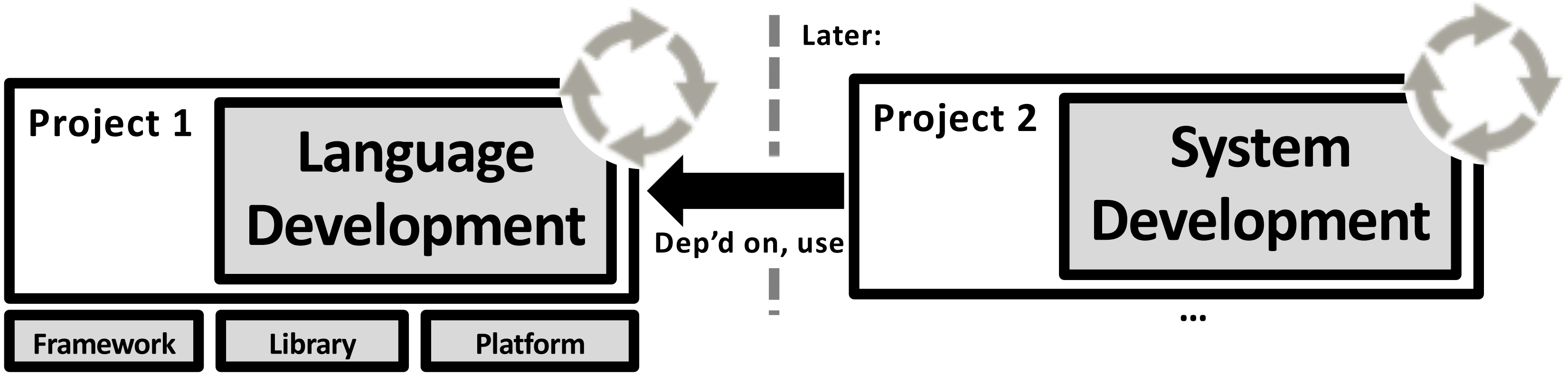


**Manage like any other intra-project dependency.**

Evolution of client code is *easier* than for F/L/P because of migration support!

# „MD* and Agile is in Conflict.“

2



Project 1

Language Development

Framework

Library

Platform

Later:

Dep’d on, use

Project 2

System Development

…

**Manage like any other 3rd party depencency:**

Development Roadmap

Issue Tracker

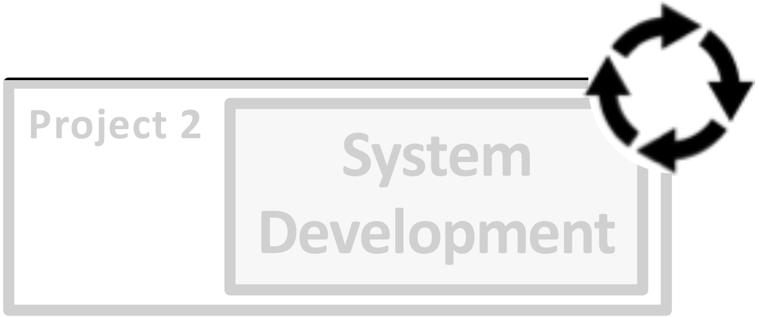
Release Notes

…

# „MD* and Agile is in Conflict.“

WTF?

3



**Models and DSLs are an Enabler for Agility:**

Integration of Domain Experts

„Living“ Requirements

Decoupled Fachlichkeit & Technik

# „MD* and Agile is in Conflict.“

4

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

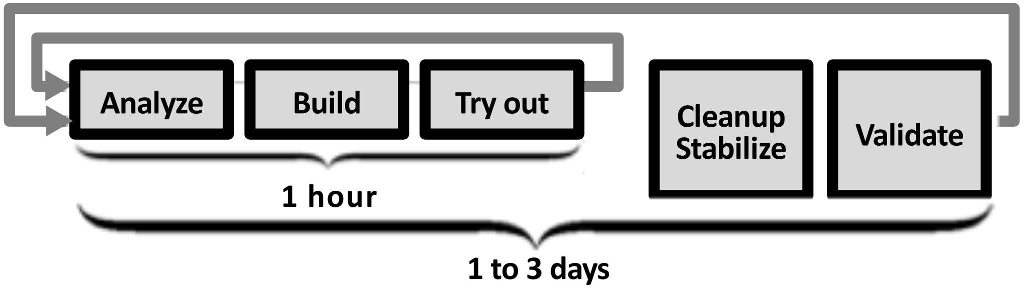
I've looked at the implementation of the language in MPS, but I didn't find much. Is this all there is? Where's the magic?

**[Customer]**

# „MD* and Agile is in Conflict.“

4

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

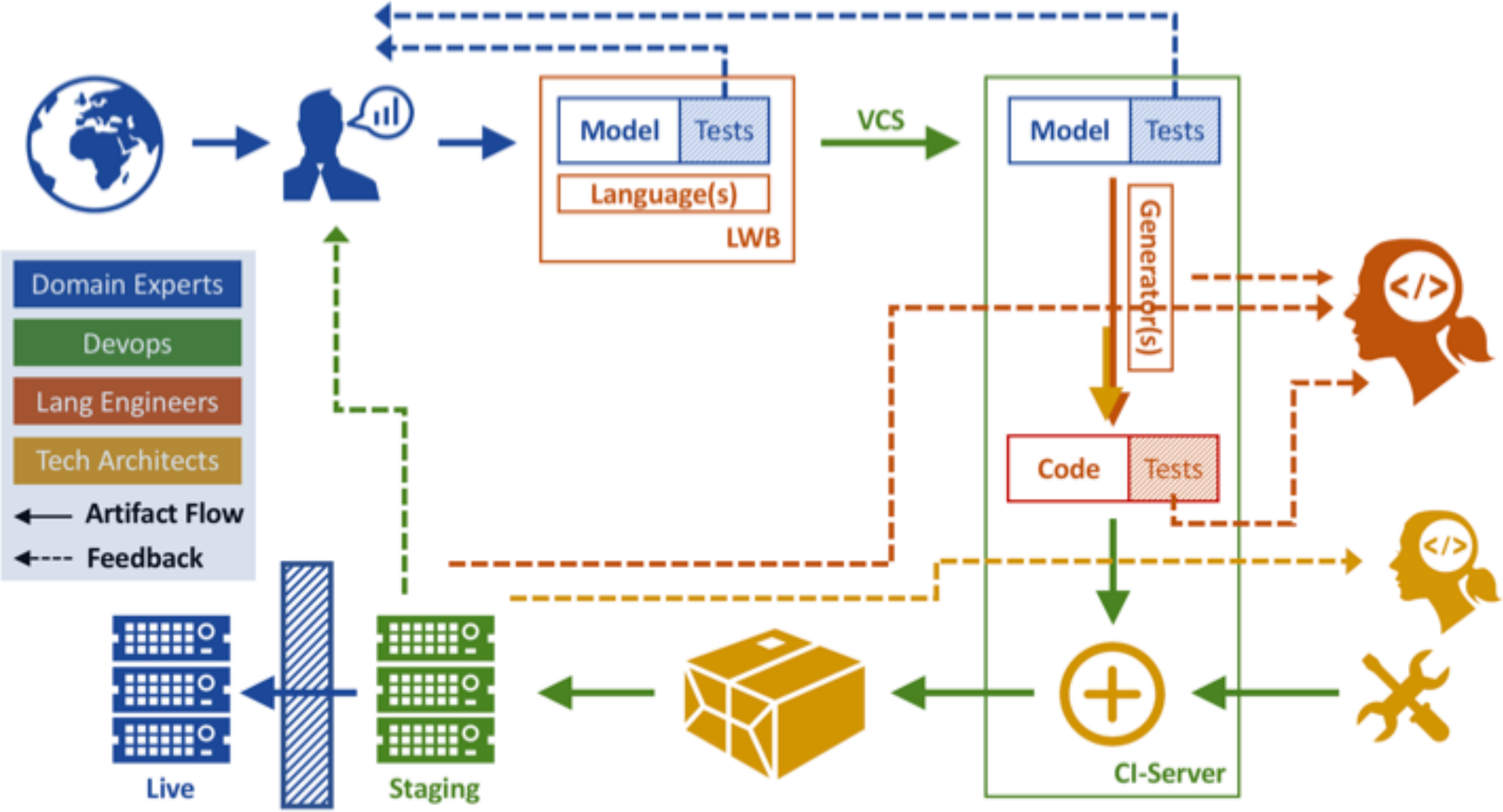


I've looked at the implementation of the language in MPS, but I didn't find much. Is this all there is? Where's the magic?

**[Customer]**

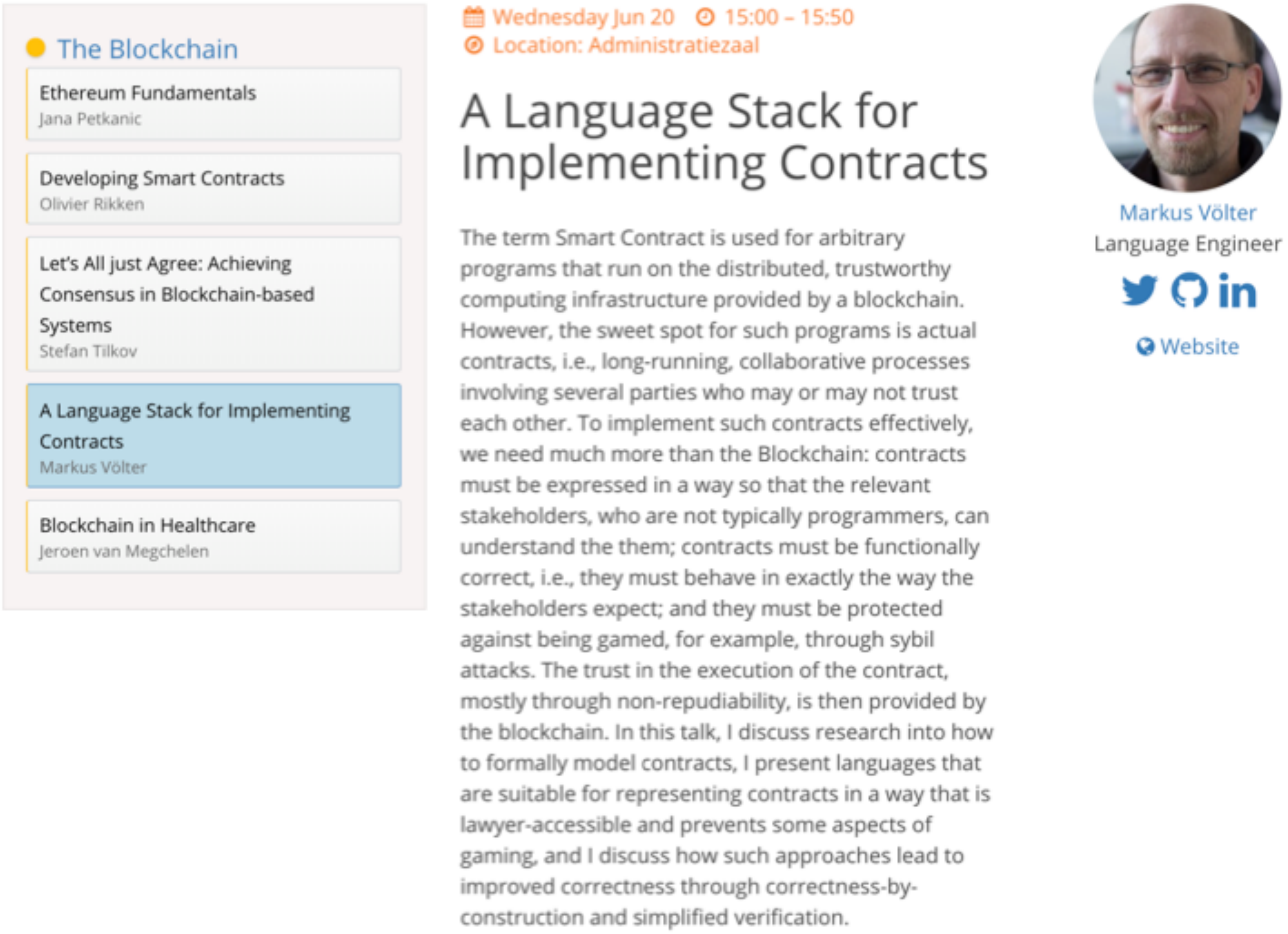
# What about CI?

# You integrate like any other automatable CI step.



5

# Wrap Up



● The Blockchain

Ethereum Fundamentals
Jana Petkanic

Developing Smart Contracts
Olivier Rikken

Let's All just Agree: Achieving Consensus in Blockchain-based Systems
Stefan Tilkov

A Language Stack for Implementing Contracts
Markus Völter

Blockchain in Healthcare
Jeroen van Megchelen

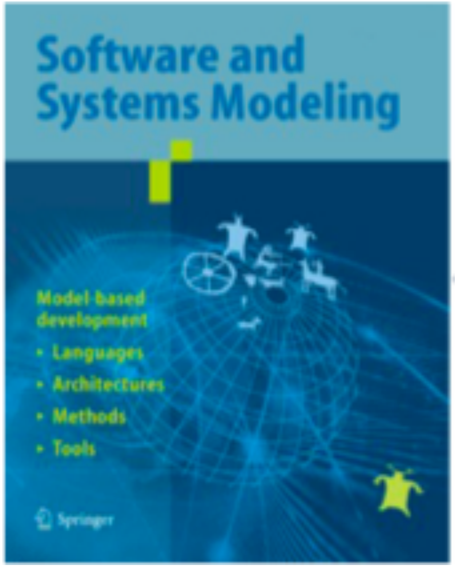
Wednesday Jun 20 15:00 – 15:50
Location: Administratiezaal

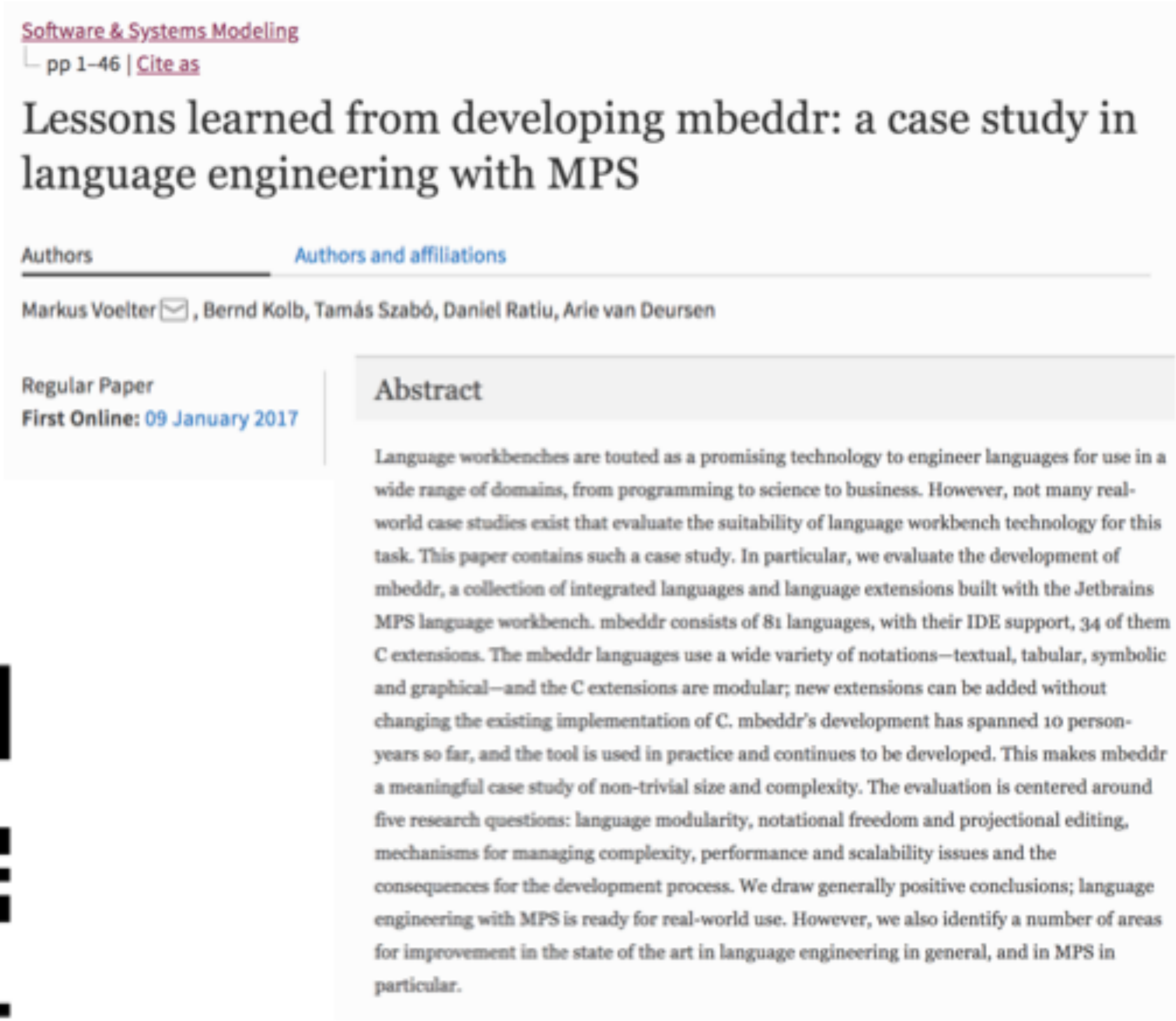
# A Language Stack for Implementing Contracts

The term Smart Contract is used for arbitrary programs that run on the distributed, trustworthy computing infrastructure provided by a blockchain. However, the sweet spot for such programs is actual contracts, i.e., long-running, collaborative processes involving several parties who may or may not trust each other. To implement such contracts effectively, we need much more than the Blockchain: contracts must be expressed in a way so that the relevant stakeholders, who are not typically programmers, can understand the them; contracts must be functionally correct, i.e., they must behave in exactly the way the stakeholders expect; and they must be protected against being gamed, for example, through sybil attacks. The trust in the execution of the contract, mostly through non-repudiability, is then provided by the blockchain. In this talk, I discuss research into how to formally model contracts, I present languages that are suitable for representing contracts in a way that is lawyer-accessible and prevents some aspects of gaming, and I discuss how such approaches lead to improved correctness through correctness-by-construction and simplified verification.

Markus Völter
Language Engineer

Website





Software & Systems Modeling
pp 1–46 | Cite as

# Lessons learned from developing mbeddr: a case study in language engineering with MPS

Authors    Authors and affiliations

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

Regular Paper
**First Online:** 09 January 2017

## 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.

**http://voelter.de/data/pub/voelterEtAl2017-buildingMbeddr.pdf**

## Separation of concerns is key

to avoid the legacy trap

## DSLs can isolate business logic

completely from technical concerns

## DSLs can help integrate domain experts

with communication/review or even coding

## Language Workbenches enable DSLs

by reducing effort to build, compose and maintain them

## DSLs are not in conflict with Agile

... to the contrary, DSLs are a powerful enabler!