Language-Oriented Business Applications

Wir sind aber keine Programmierer!!

Oder doch?

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

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

voelter { ingenieurbüro für softwaretechnologie // itemis

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

Nothing teaches us better than our own experiences!

1st

Business Knowledge and Software

[Business Knowledge]

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

[Business Knowledge]

It‘s what makes a business tick.
Distinguishes the business.

Contributed **not** by developers

... but typically implemented
in software

[Business Knowledge]

SO HOW DOES IT GET INTO THE SOFTWARE?

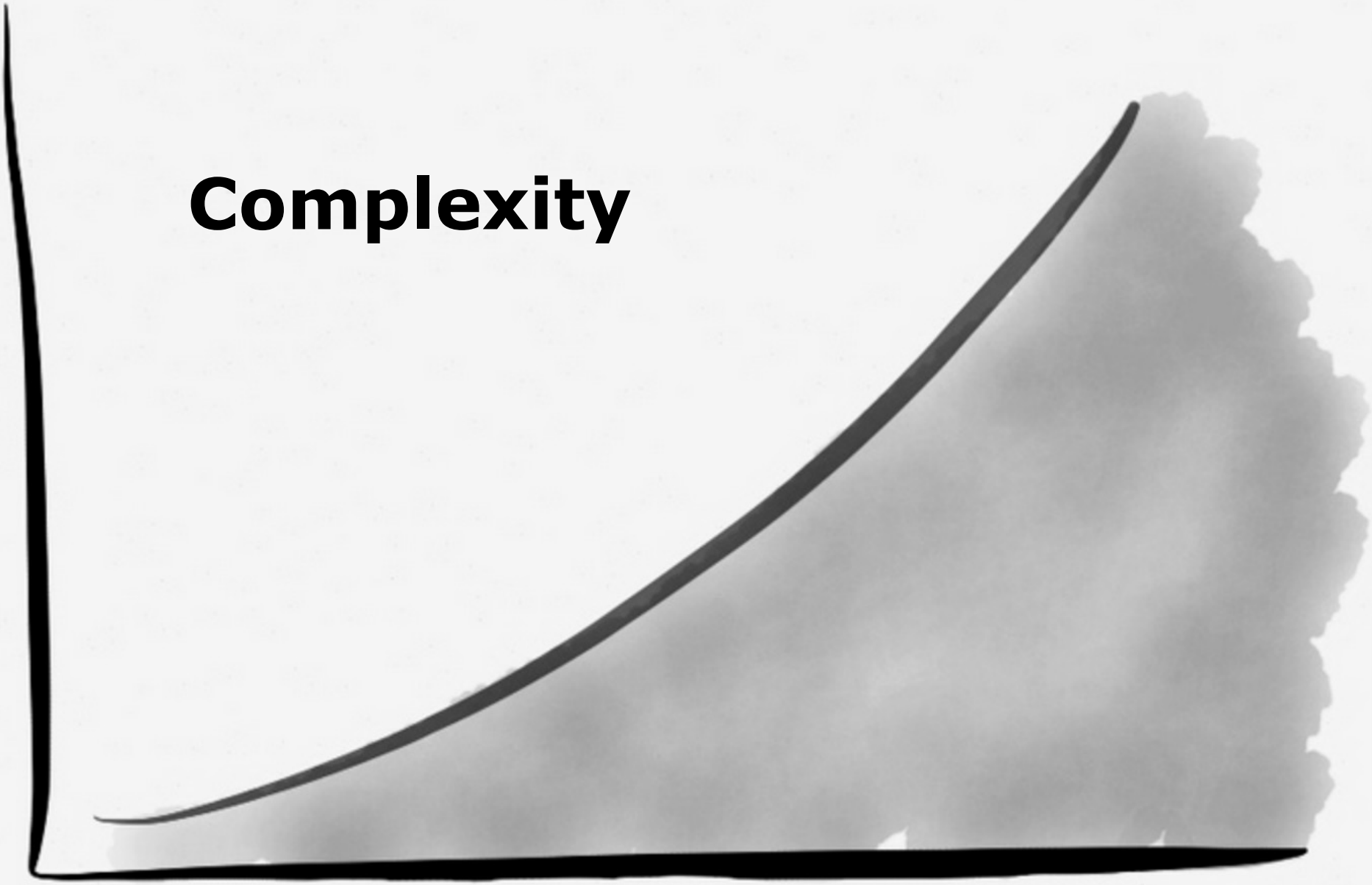
Contributed **not** by developers

... but typically implemented

in software

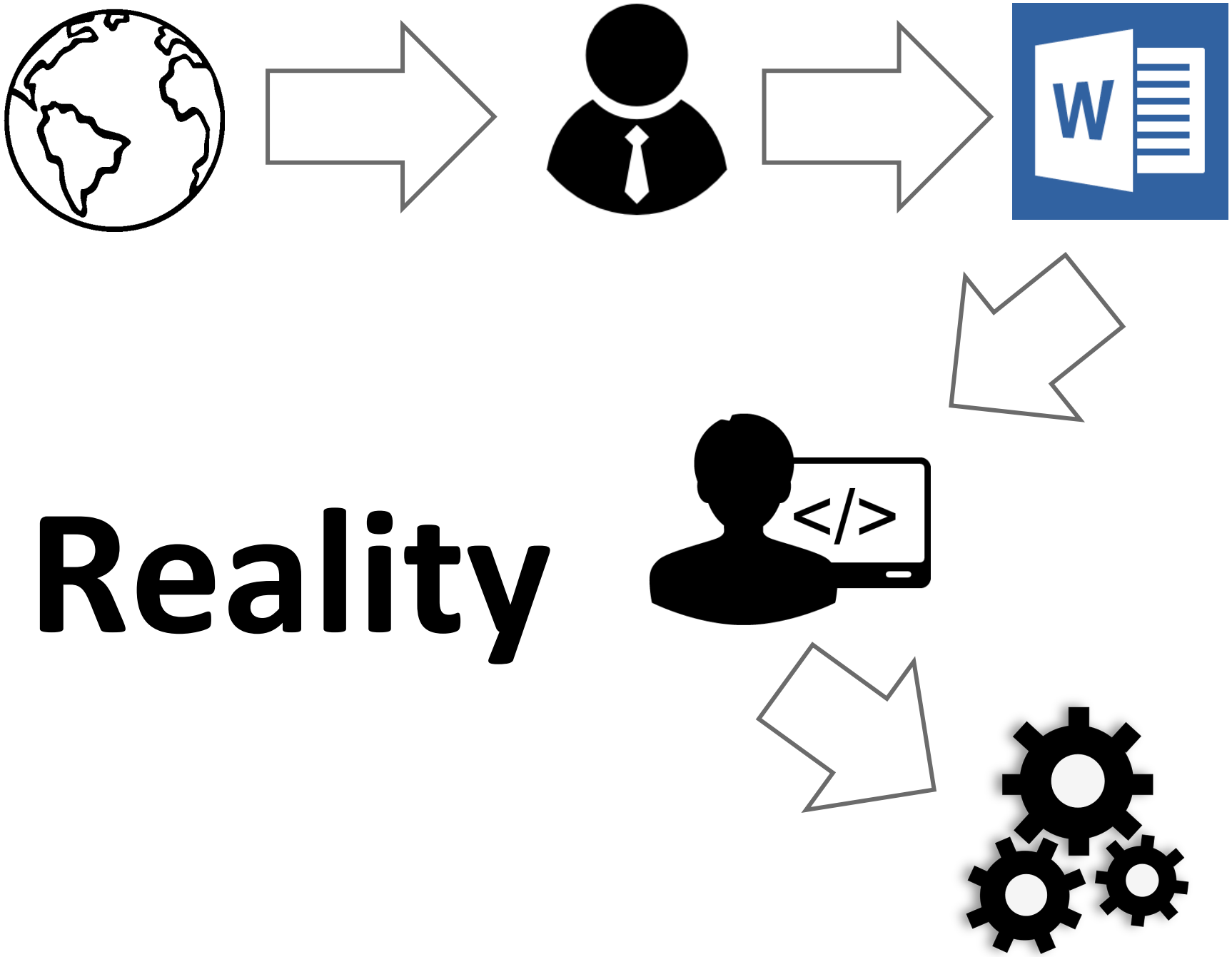
Trends/ Challenges

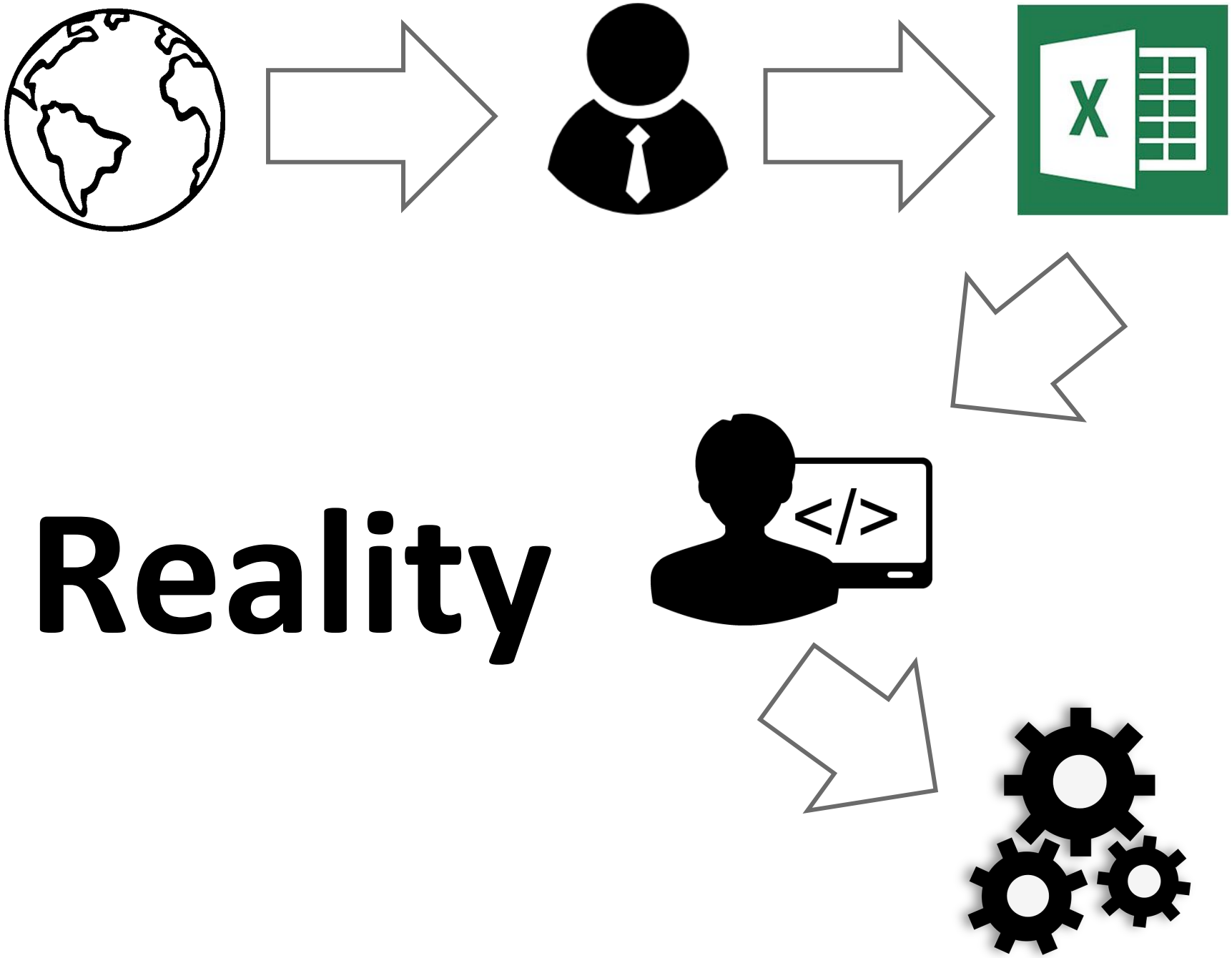
Complexity

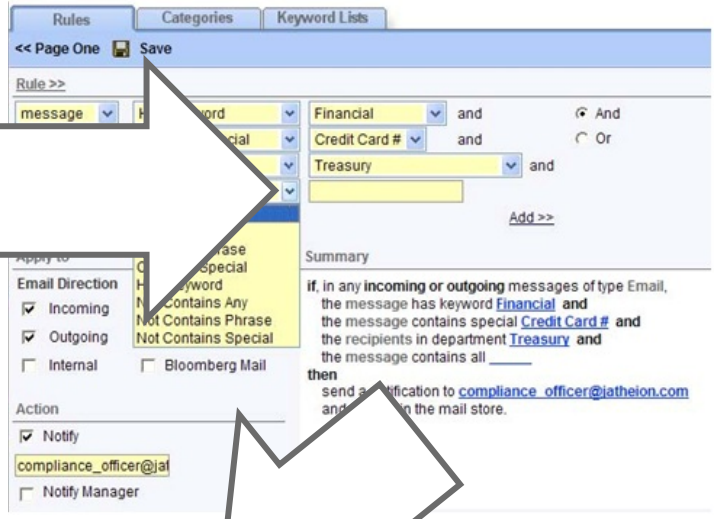


Mass Customization

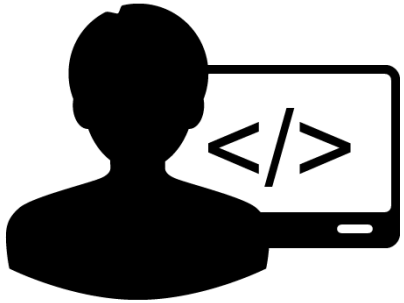
Time To Market

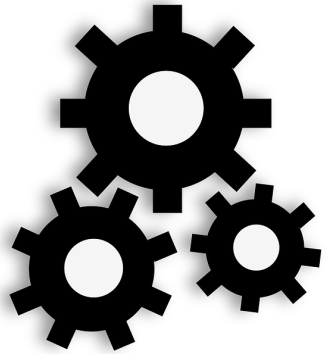

W
Reality
</>

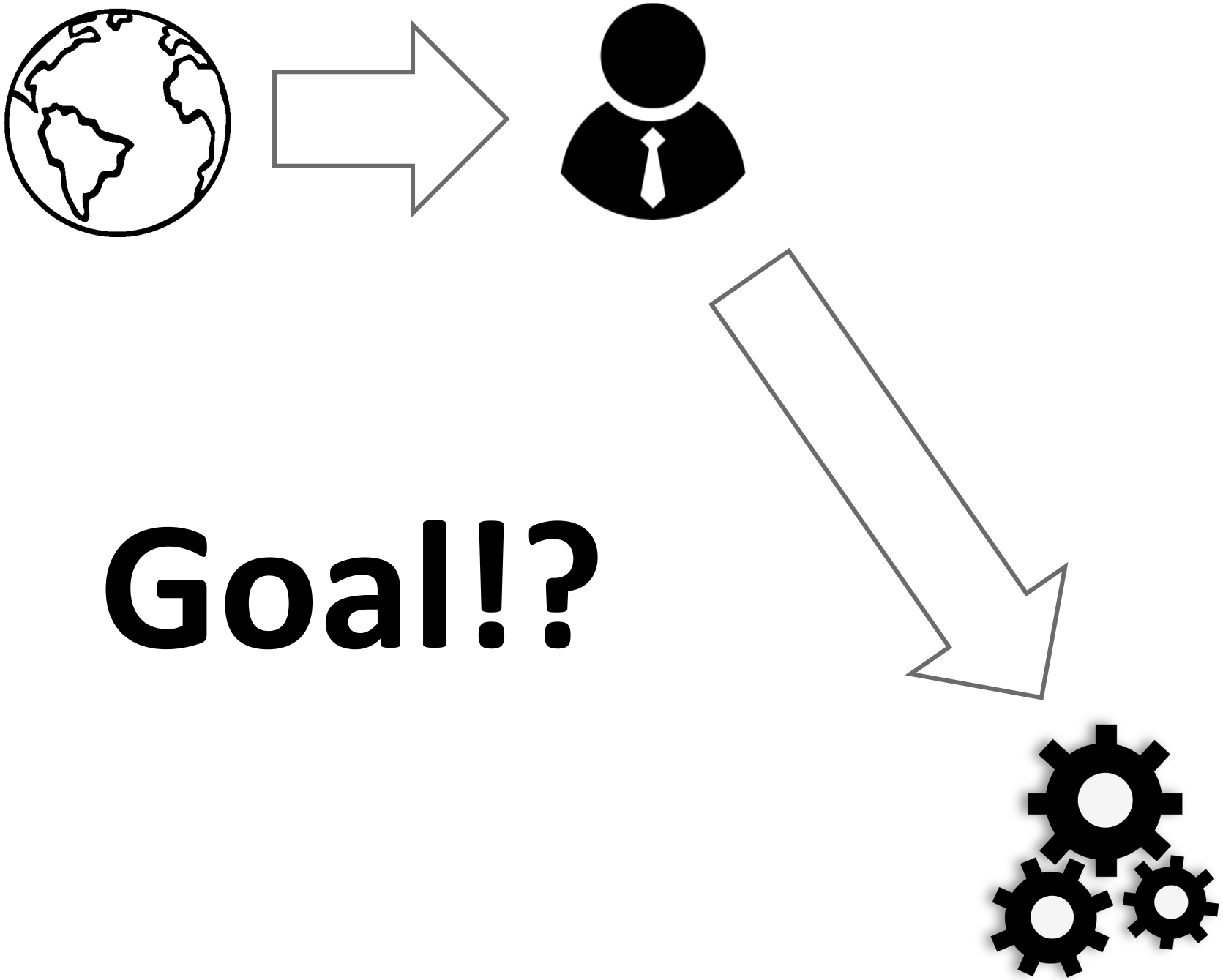

Reality
X
</>


Rules
Categories
Keyword Lists
<< Page One
Save
Rule >>
message
Financial
and
And
Credit Card #
and
Or
Treasury
and
Add >>
Summary
Email Direction
Incoming
Outgoing
Internal
Bloomberg Mail
Not Contains Any
Not Contains Phrase
Not Contains Special
if, in any incoming or outgoing messages of type Email,
the message has keyword Financial and
the message contains special Credit Card # and
the recipients in department Treasury and
the message contains all
then
Action
Notify
Notify Manager

Reality


</>




Goal!?

Let Business/Domain people contribute directly!

Give them expressive, productive tools to do so!

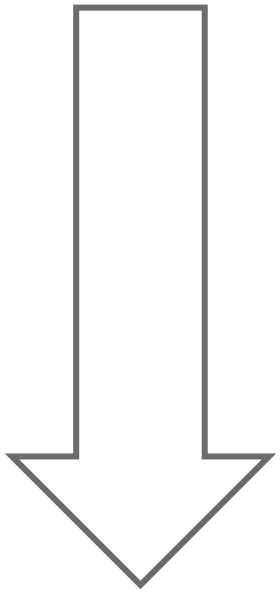
Expressivity for Core Domain Knowledge

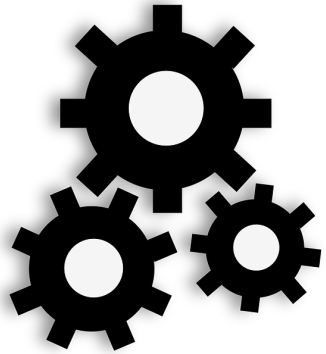
User-Friendly Notation
Great Tool/IDE

Testing

Meaningful Analyses

Synthesis of Software





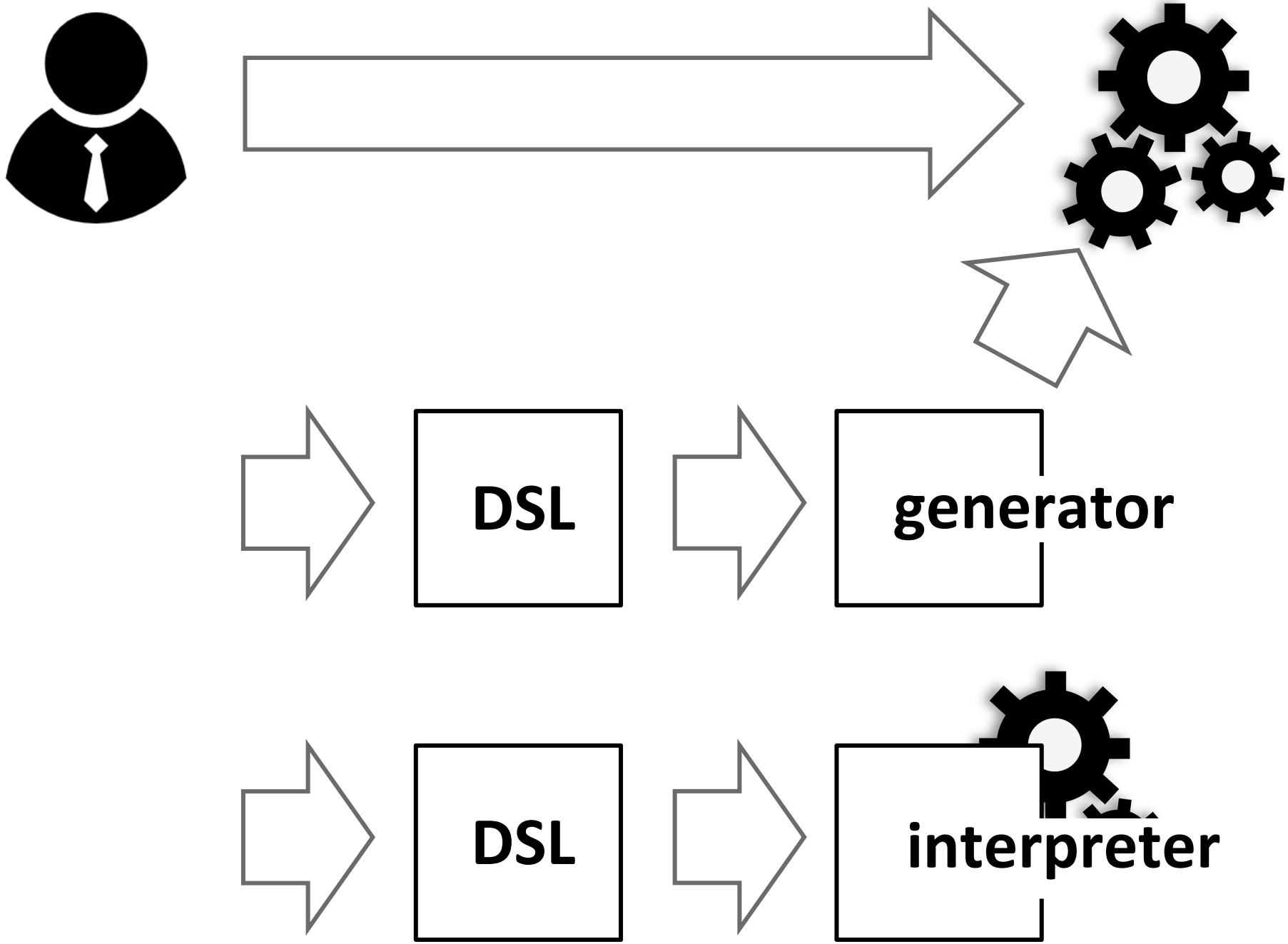
Not a software engineer.
Does not care about „software stuff“
But understands the domain very well.

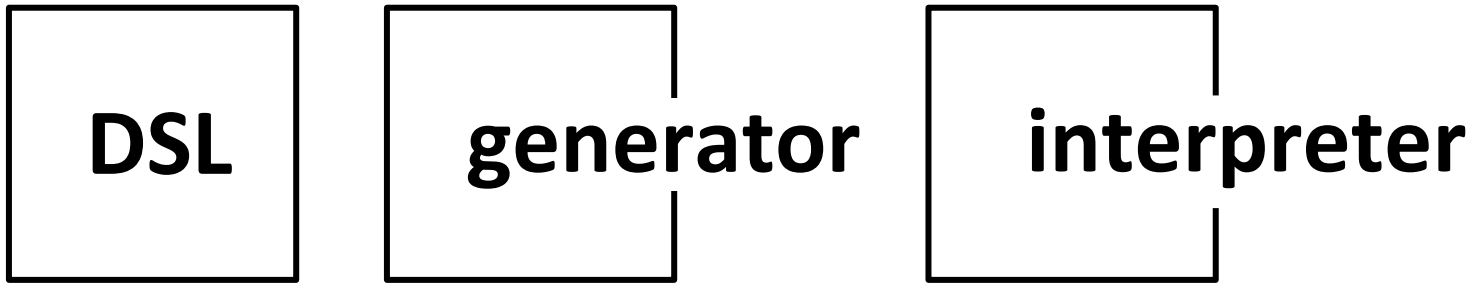
Not a software engineer.
Does not care about „software stuff“
But understands the domain very well.

He is a professional, not a „casual hacker“.

2

Language Workbenches


DSL
generator
DSL
interpreter



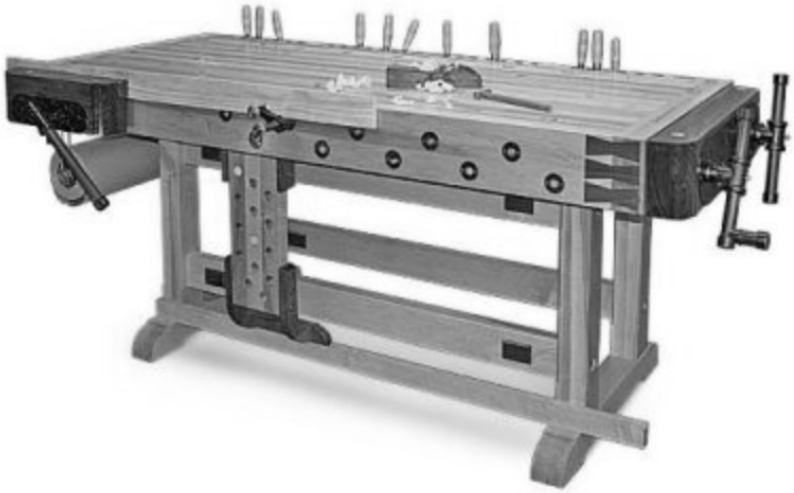
An old idea from the 1970s.

BUT...

Language Workbench

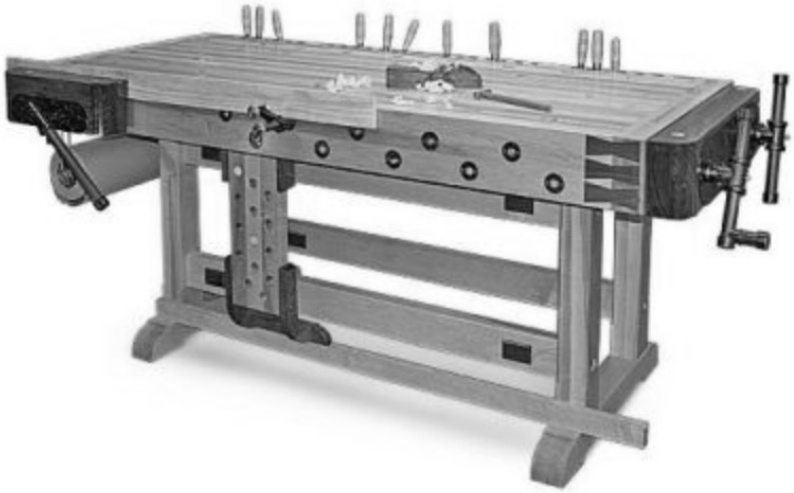
(Martin Fowler, 2004)

Freely
define
languages and
integrate
them



Language Workbench

(Martin Fowler, 2004)



powerful

editing

testing

refactoring

debugging

groupware

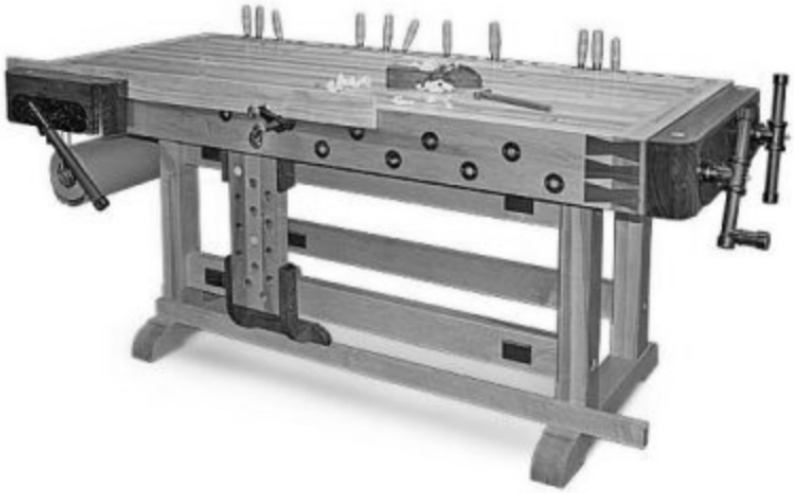
language definition

implies

IDE definition

Language Workbench

(Martin Fowler, 2004)



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://languageworkbenches.net

3

JetBrains MPS

A Language Workbench –

a tool for defining, composing and using ecosystems of languages.

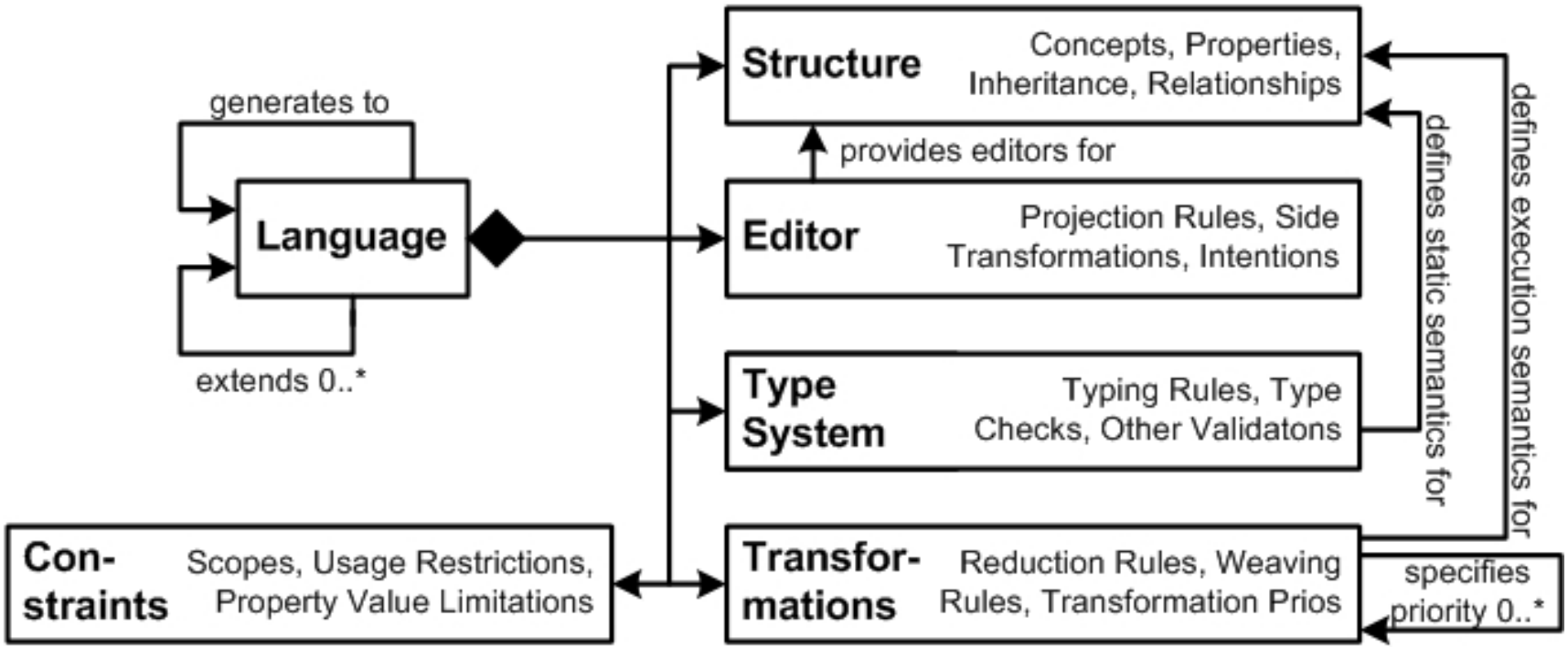
Open Source
Apache 2.0
http://jetbrains.com/mps

V 3.3 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, ...

SIEMENS

Mercedes-Benz

MPS

fortiss

OHB

BOSCH

HUAWEI

Whirlpool CORPORATIO

DYNAGEN power controls you can trust

McGill

UNIVERSITY OF WATERLOO

ERICSSON

SIOUX

Belastingdienst

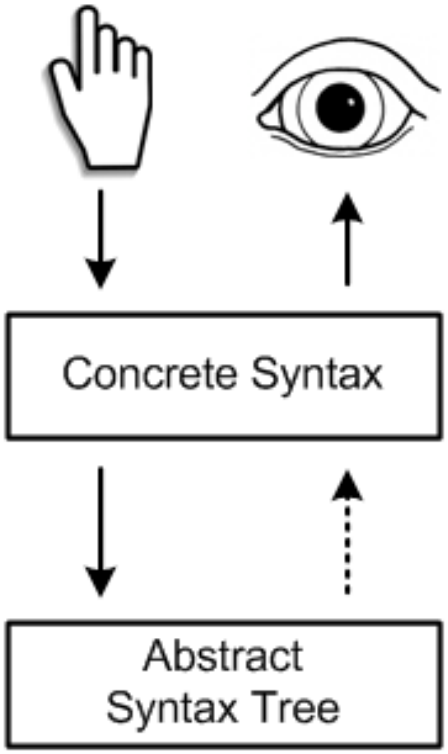
itemis

i2S INSURANCE KNOWLEDGE

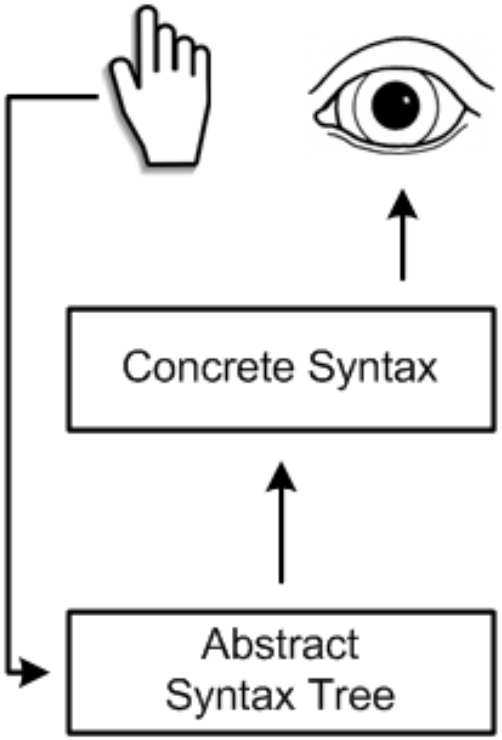
Audi

[Projectional Editing]

Parsing



Projectional Editing



[Projectional Editing]

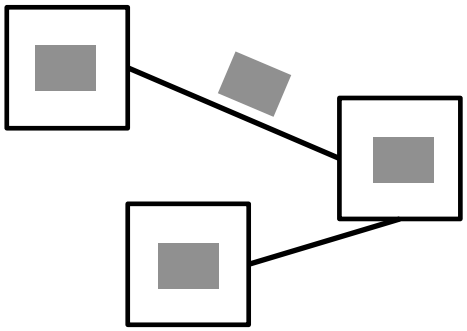
Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical



[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

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

Mathematical

```
double midnight2(int32 a, int32 b, int32 c) {
```

$$\text{return}\ \frac{-b + \sqrt{b^2 - \sum_{i=1}^{4} a * c}}{2 * a};$$

```
} midnight2 (function)
```

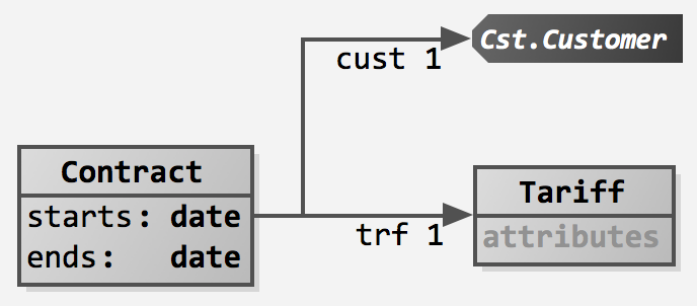
Tables

```
int16 decide(int8 spd, int8 alt) {
  return
```

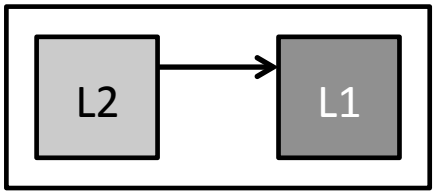
	spd > 0	spd > 100
alt < 0	1	1
alt == 0	10	20
alt > 0	30	40
alt > 100	50	60

```
otherwise 0;
} decide (function)
```

Graphical



[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.

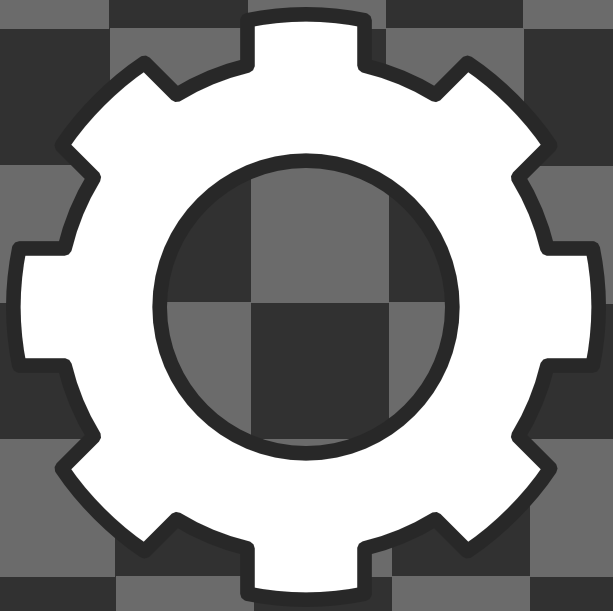
Usability Issues are mostly solved.

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



Rigid Structures

Rule Set Type DemoRuleSetType

Business objects

person : **Person**

Variables:

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

Parent

<no parent>

Libraries

Standard
Extra

Rule Set Type DemoRuleSetType

Business objects

<no business objects>

Variables:

<no variables>

Parent

<no parent>

Libraries

<no libraries>

Calculation Rules

```
rule set DemoRulseSet2 is of type DemoRuleSetType

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

Toggle Information

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

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

PREMIO = [ <always> => round(PREMIO * (1 + factacer), 0) ]
```

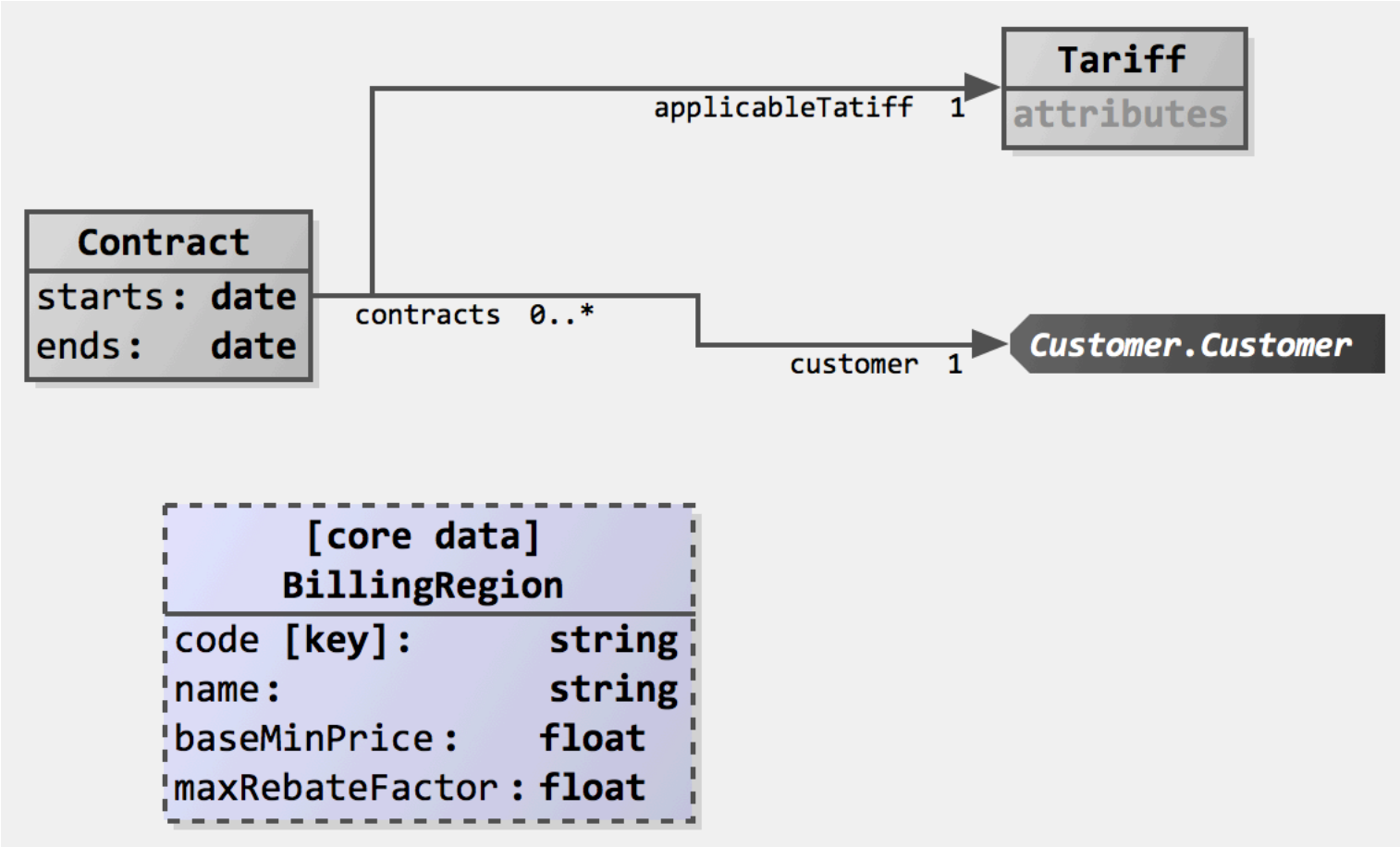
Prose-Like Language for Calc Rules

```
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 ongelijk het actuele voorkomen
    Einde declaratie
    ' dus: broer of zus (incl. erkend kind van ouder)
}

bloed- of aanverwanten in rechte lijn : lijst van Burgers  zijn gedefinieerd als {
    Een bloed- of aanverwant in rechte lijn is een Burger die
    bloedverwant in rechte lijn is of die
    aanverwant in rechte lijn is
    Einde declaratie
}
```

Textual Notation for Data Modeling

```
Data Contract
------------------------------------------------------------------------

proxy for Customer.Customer

core data entity BillingRegion
-------------------------------------------
code [key]:        string  |  references:
name:              string  |
baseMinPrice:      float   |
maxRebateFactor:   float   |

entity Contract
------------------------------------------------------------------------
starts: date  |  customer:          Customer 1 □  <--> contracts 0..*
ends:   date  |  applicableTatiff:  Tariff   1 □

entity Tariff
------------------------------
attributes:  |  references:
             |
```

Diagrams for Data Modeling



Tables for Reference Data

Core Data DefaultRegions **for entity** BillingRegion

Code	Name	Base Min Price	Max Rebate Factor
BW	Baden Württemberg	0.20	0.8
BY	Bayern	0.20	0.8
BE	Berlin	0.15	0.7
BB	Brandenburg	0.10	0.7
HB	Bremen	0.20	0.7
HH	Hamburg	0.15	0.7
HE	Hessen	0.15	0.7
MV	Mecklenburg-Vorpommern	0.10	0.7
NI	Niedersachsen	0.15	0.7
NW	Nordrhein-Westfalen	0.15	0.7
RP	Rheinland-Pfalz	0.15	0.7
SL	Saarland	0.15	0.7
SN	Sachsen	0.10	0.7
ST	Sachsen-Anhalt	0.10	0.7
SH	Schleswig-Holstein	0.15	0.7
TH	Thüringen	0.10	0.7

Word-Like Comments

```
Calculations CallCalculations for Call        imports: Σ CustomerBasic

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

value cust :=  entity.customer
value pricingFactor :=
```

	isLocal	isLongDistance	isRoaming
cust.isRebated	0.5	0.6	0.8
!cust.isRebated	0.8	0.9	1.0

```
otherwise 1
```

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

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

Business Rules, Math, Tooltips

```
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. ]
```

```
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
```

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

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

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

Tests executed in the Editor

```
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

                                  amount of calls - 1
  value current price :=          ∑                  ((all calls.at(i).price.value)) * discountFactor
                                  i = 0
    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 everageCallPrice :=   current price
                              ------------
                              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
```

Business Rules for Contracts

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

[final] assign callsThisMonth
callsThisMonth := c.callsLastMonth

assign amountThisMonth
amountThisMonth := 0

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

Business Rules for Contracts

```
contract FlatrateContract specializes BaseContract          imports: R BusinessRequirements
Context Objects:
  c: Customer

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

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

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

conditional assign overrides BaseContract.amountThisMonth as of 20/8/2014[T]
amountThisMonth := | c.isRebated  | 40  |
                   | otherwise    | 60  |
```

BDD-style Tests for Business Rules

```
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

Add Comment | Add Other Data | Add Child Requirement | Add Next Requirement | ⇩

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

workpackage inital **scope:** 1 **responsible:** peter **prio:** 1 **effort:** 1 **days**
[ ]

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

Add Comment | Add Other Data | Add Child Requirement | Add Next Requirement | ⇧ | ⇨ | ⇩

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

3 **The factor of points**
PointsFactor /functional: tags

Add Comment | Add Other Data | Add Child Requirement | Add Next Requirement | ⇧ | ⇨ | ⇩

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

Math Notations

```
vector<int16, 3> aVector = [1; 2; 3] * 512;

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

matrix<int16, 2x3> aMatrix = [1 + 2   2 * 7   42; 3   51   24];

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

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

Explorability of the Language

HelloMath ×

[C Extensions]

HelloMath **constraints**

model mbeddr.tutorial.main.math **imports** nothing

```
int32 sumUpIntArray(int32[] arr, int32 size) {

  return Σ_{i=0}^{size} arr[i];
} sumUpIntArray (function)

int32 averageIntArray(int32[] arr, int32 size) {
  return (Σ_{i=0}^{size} arr[i]) / size;
} averageIntArray (function)
```

Context Actions

Math Expressions

abs frac log pow product sqrt

sum

Context Sidebar

Context Actions

Prose for Describing Requirements

4 **Points you get for each trackpoint**
InFlightPoints /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. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.
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

Business Rules mixed with Prose

4 **Points you get for each trackpoint**
InFlightPoints /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. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.
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 = (BASEPOINTS * 1) *
```

	alt > 2000	alt > 1000
speed > 180	30	15
speed > 130	10	20

```
otherwise 0
```

```
  tests: ...
```

Tests for Business Rules

4 **Points you get for each trackpoint**

InFlightPoints /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. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.

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 = (BASEPOINTS * 1) *
```

	alt > 2000	alt > 1000	otherwise 0
speed > 180	30	15	
speed > 130	10	20	

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

Live Tests for Business Rules

4 **Points you get for each trackpoint**

InFlightPoints /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. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.

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 = (BASEPOINTS * 1) *
```

	alt > 2000	alt > 1000	otherwise 0
speed > 180	30	15	
speed > 130	10	20	

```
   tests: PointForATrackpoint(500, 46) == 0
          Point[illegible]) == 0
          PointForATrackpoint(1100, 165) == 210
          PointForATrackpoint(2100, 140) == 100
          PointForATrackpoint(2100, 200) == 300
```

Error: failed; expected 210, but was 200

Debugger for Business Rules

4 | Points you get for each trackpoint

InFlightPoints /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. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.

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

---

result = 200 / (10 / 10 | *BASEPOINTS* * 1) * 20 / (false / 1100 | alt > 2000 : (false / 165 | speed > 180 : 30 ; true / 165 | speed > 130 : 10) ; true / 1100 | alt > 1000 : (false / 165 | speed > 180 : 15 ; true / 165 | speed > 130 : 20)) **otherwise** 0

---

tests: PointForATrackpoint(500, 46) == 0
PointForATrackpoint(500, 1200) == 0
PointForATrackpoint(1100, 165) == 200 Clear ⇓ ⇑
PointForATrackpoint(2100, 140) == 100
PointForATrackpoint(2100, 200) == 300

4

Lessons Learned

How to make People precise?



Precision
!=
Programming

Precision != Programming

- Formulas, Rules
- Data Structures
- Tables
- Values

Precision != Programming

Formulas, Rules
Data Structures
Tables
Values

Performance
Scalability
Robustness
Deployment

Precision != Formalization

- Formulas, Rules
- Data Structures
- Tables
- Values

Precision

Formulas, Rules
Data Structures
Tables
Values

!=

Greek Letters
Analyses
Proofs

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

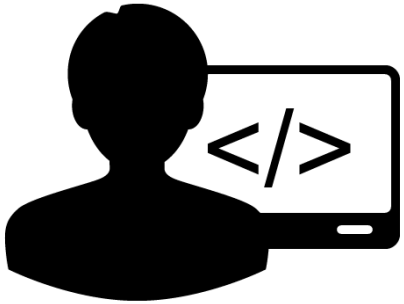
How to get business people to be precise

Technical People:
„It‘s not my job!“.

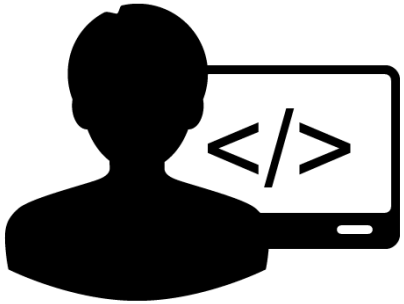
(and it really isn‘t)

Business L
vs. Programming L





Structure/Guid.	+	-
Notation	Mixed	Text
Views	*	1
IDE/Tool	Clean	Powerful
Learn/Effective	L	E



Structure/Guid.
Notation
Views
IDE/Tool
Learn/Effective

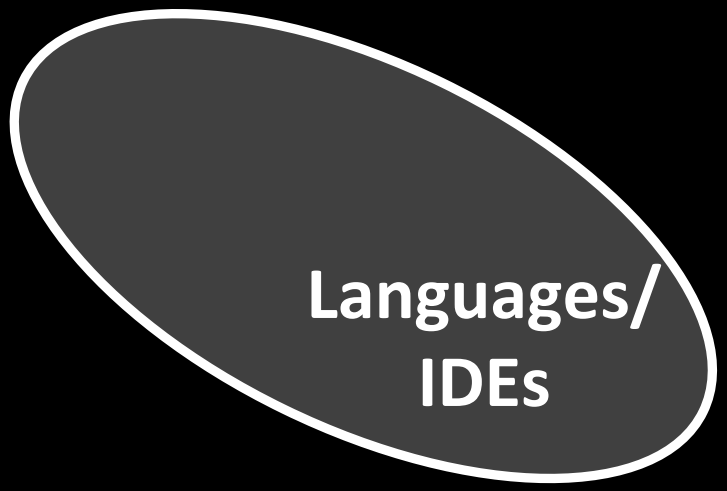
Business oriented languages are very different from what we have learned about languages for developers. LWBs let you build such languages.

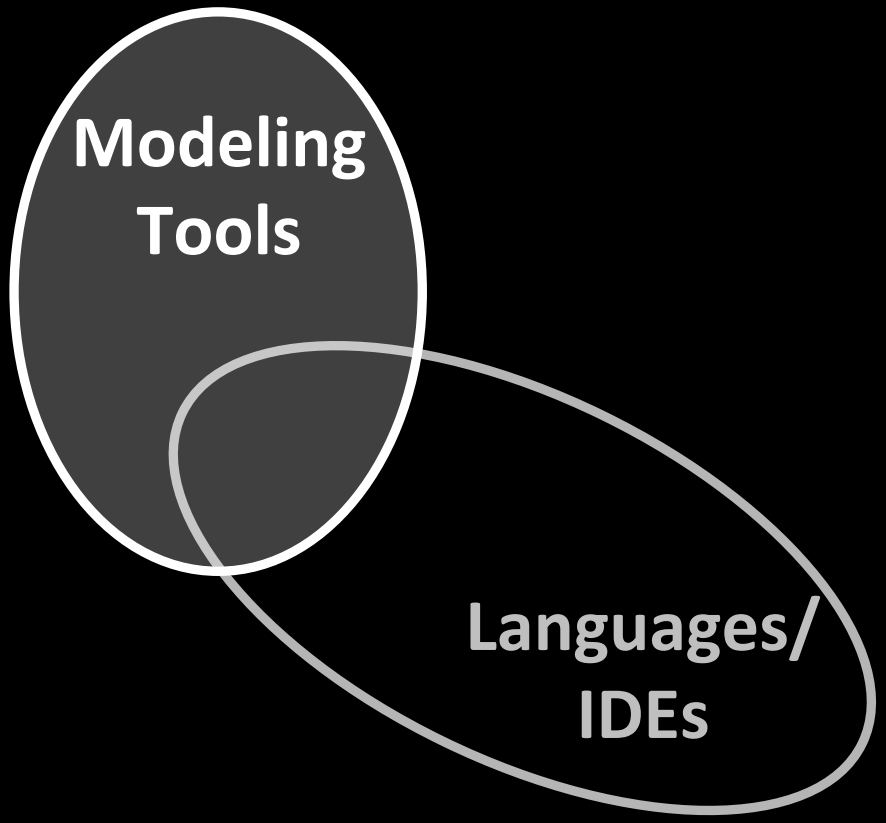
Language Workbenches enable developers to build really expressive tools for business people to work with data effectively.

A hybrid of many worlds

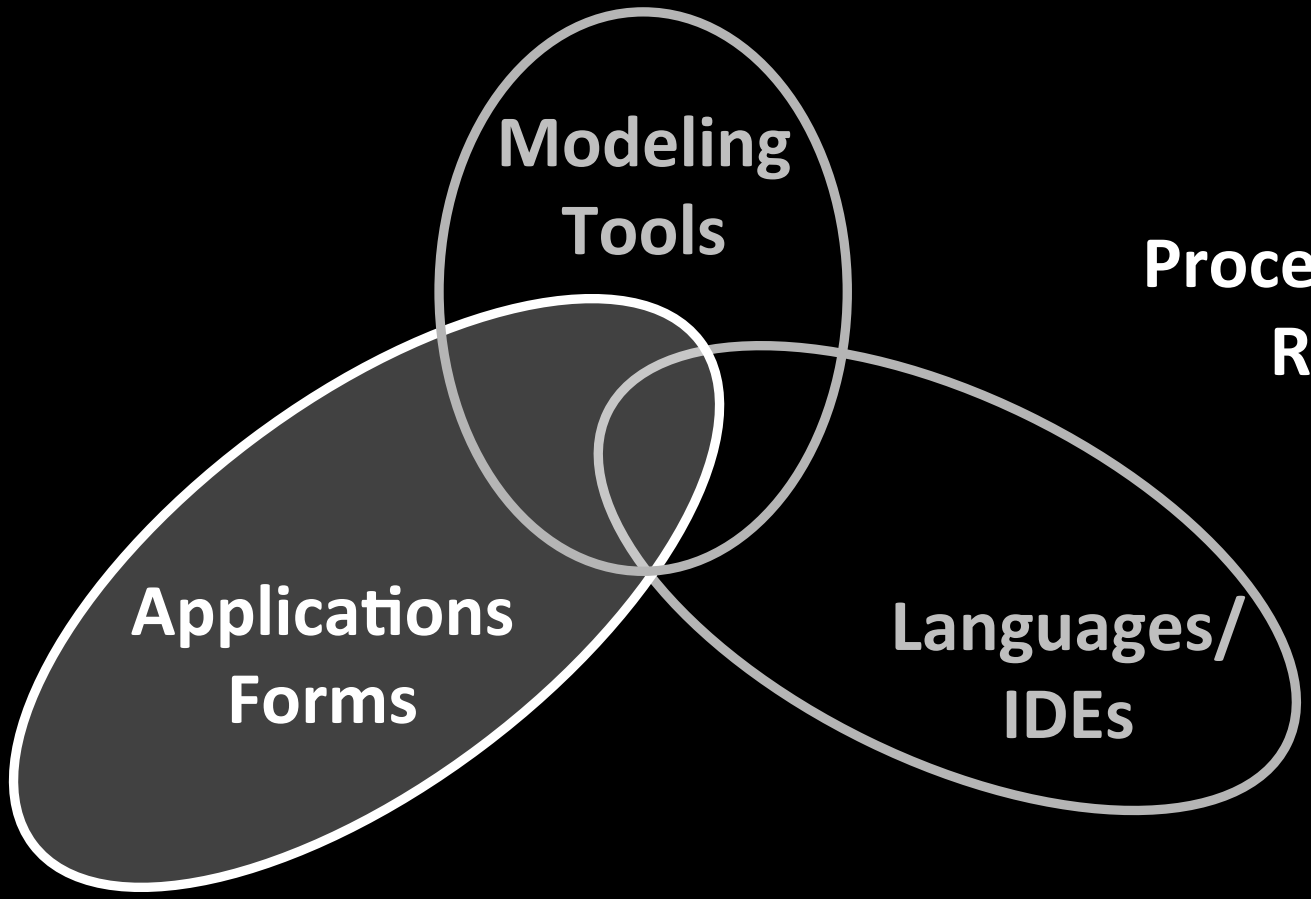


Expressions
Code Completion
Syntax Highlighting
Error Markup
Version Control
Refactoring
Debugging
Scalability
Code Reviews

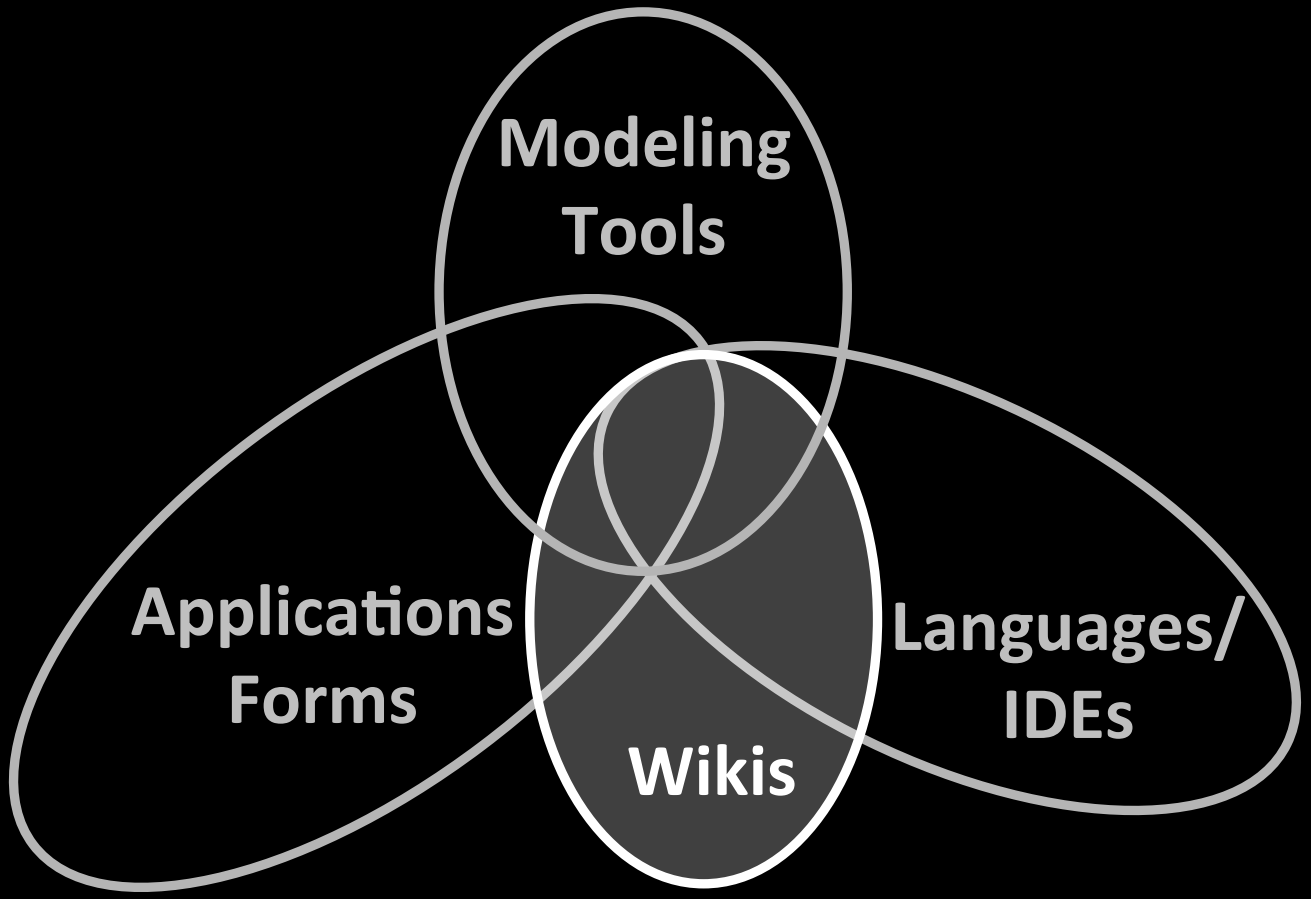




Abstraction Levels
Multiple Abstractions
Multiple Notations

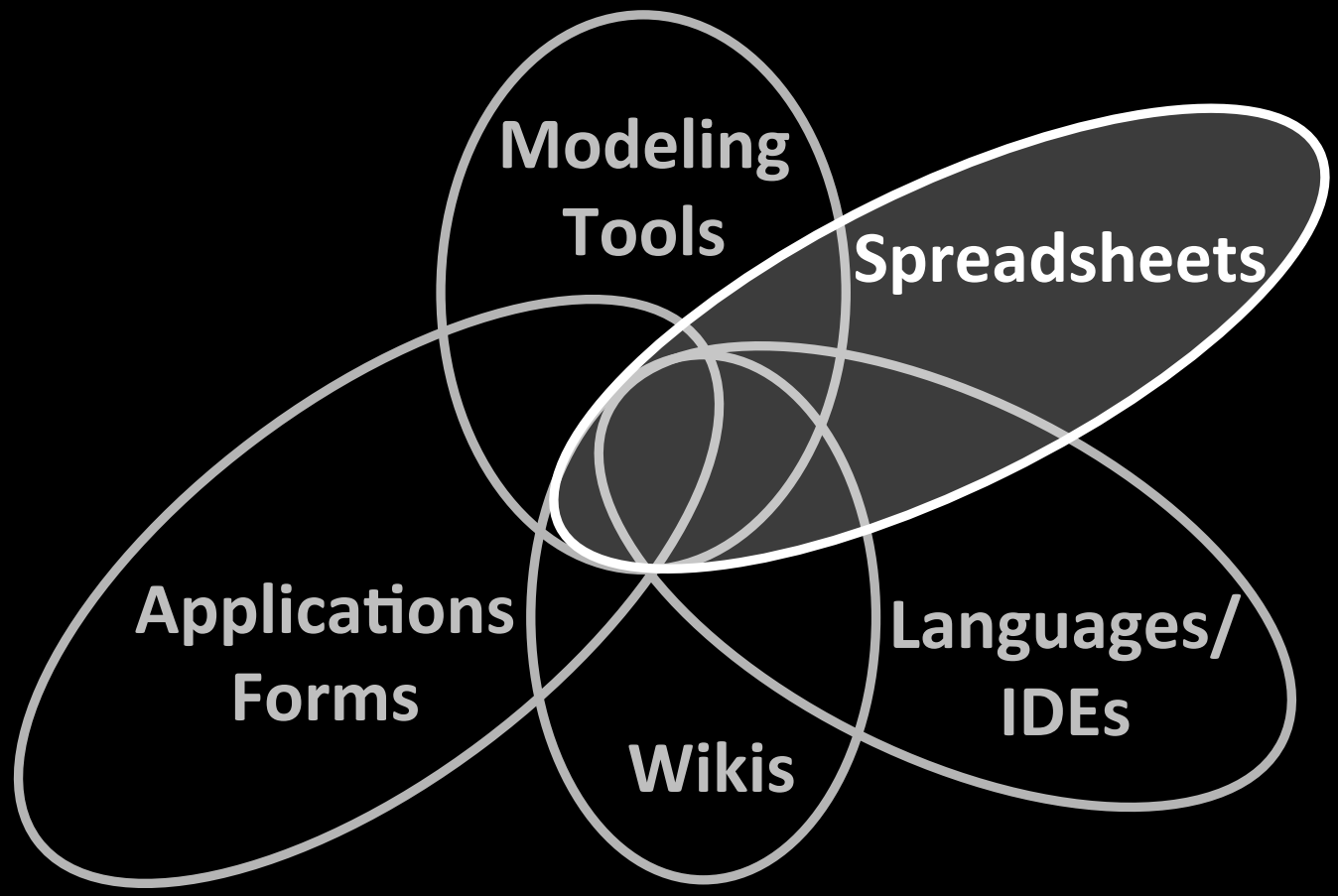


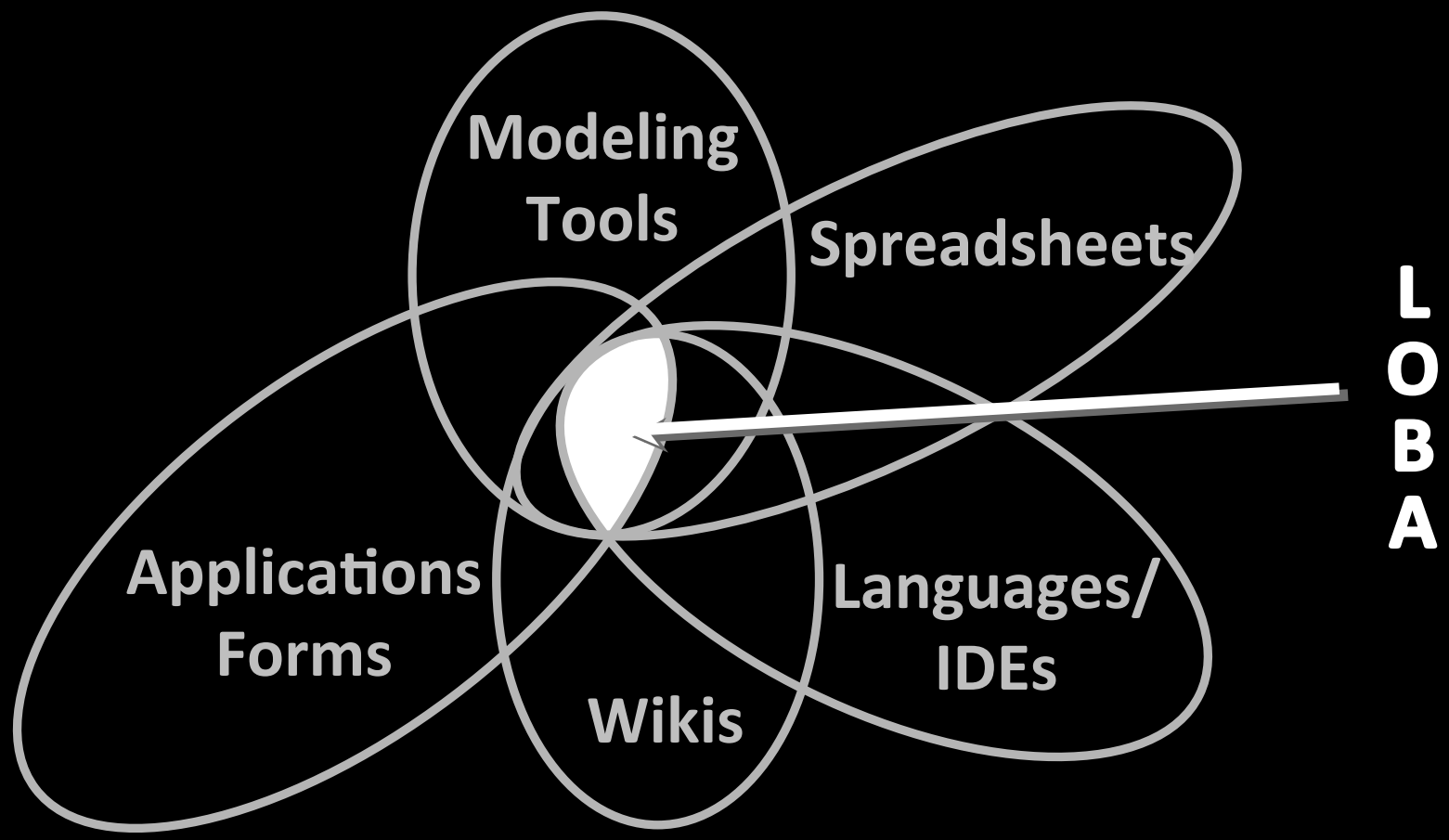
Process-Orientation
Rigid Structures
Visualizations
Tree Views
Guidance
Buttons



Prose Integration
Cross-Links
„Plugins“

Live Execution
„Visible Computation“
Document-Oriented




Modeling Tools
Spreadsheets
Applications Forms
Wikis
Languages/ IDEs
LOBA

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

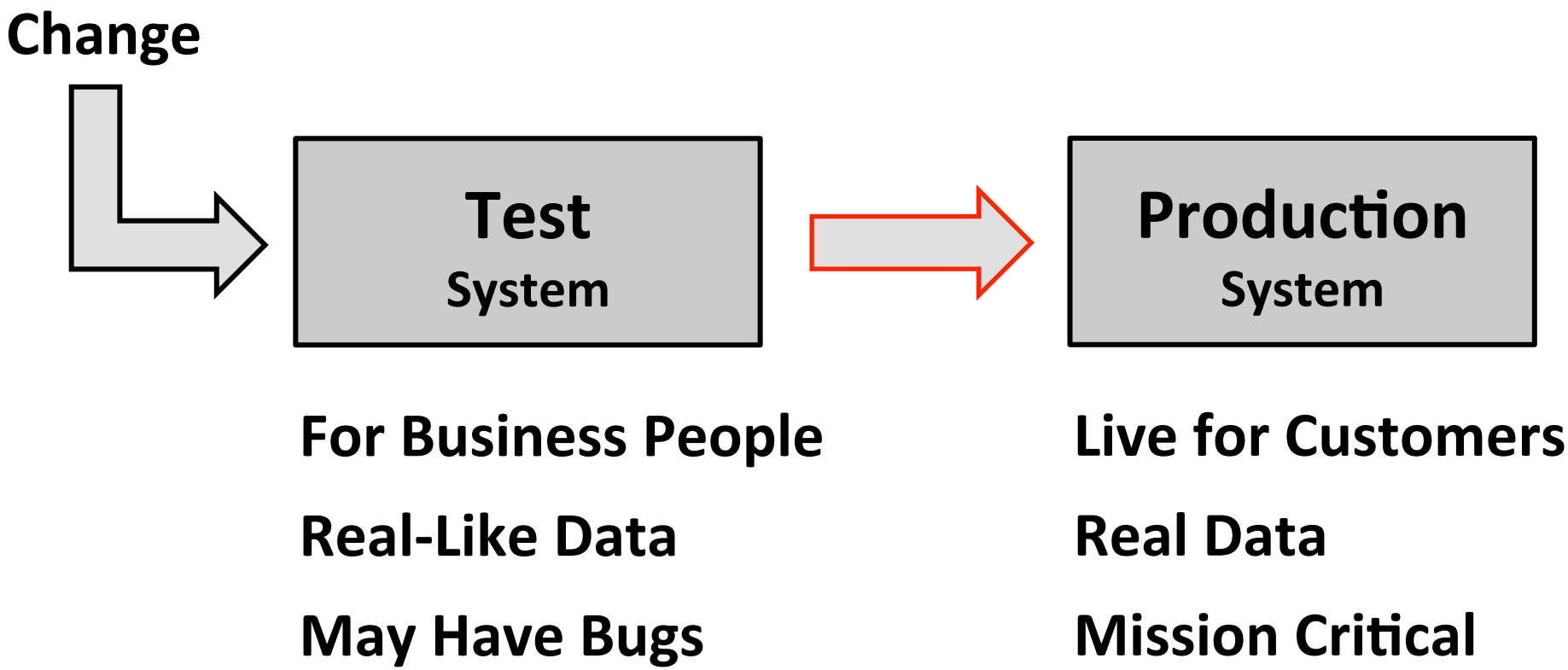
Consistency across Team
Development History
Time Machine
Branching (Feature, Version)

Why Version Control

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

Use Staging





Integration Tests
Simulations
Reviews

How do you achieve Consistency



LOCAL

CONSISTENCY.

CONSISTENCY.

CONSISTENCY.

Strict Language
Cross-References
Modularization and Reuse

Automatic Derivation based on rules (transformation, generation)

GLOBAL

CONSISTENCY.

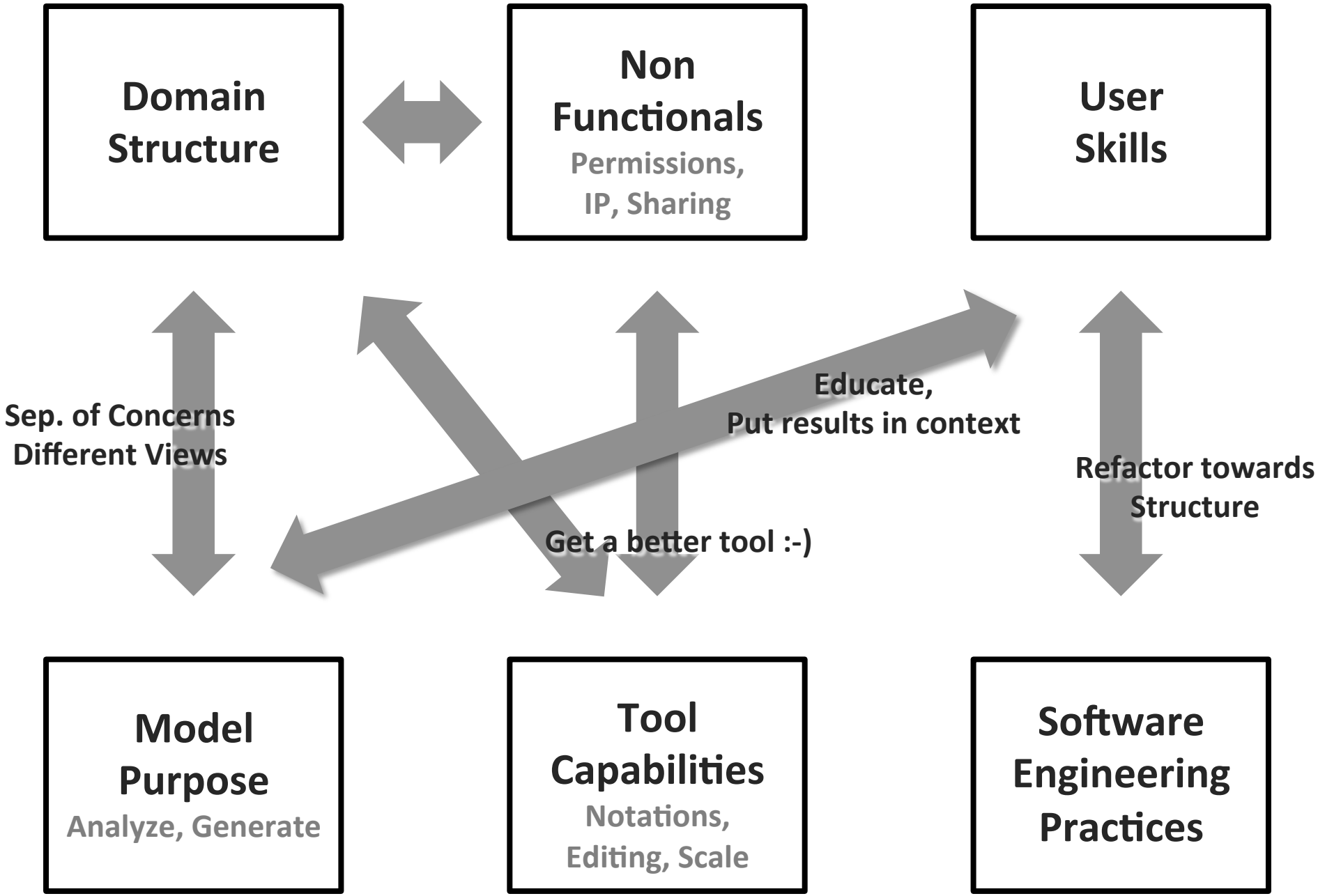
CONSISTENCY.

CONSISTENCY.

Common Respository
Version Control System
Periodic, Global Checks/Reports

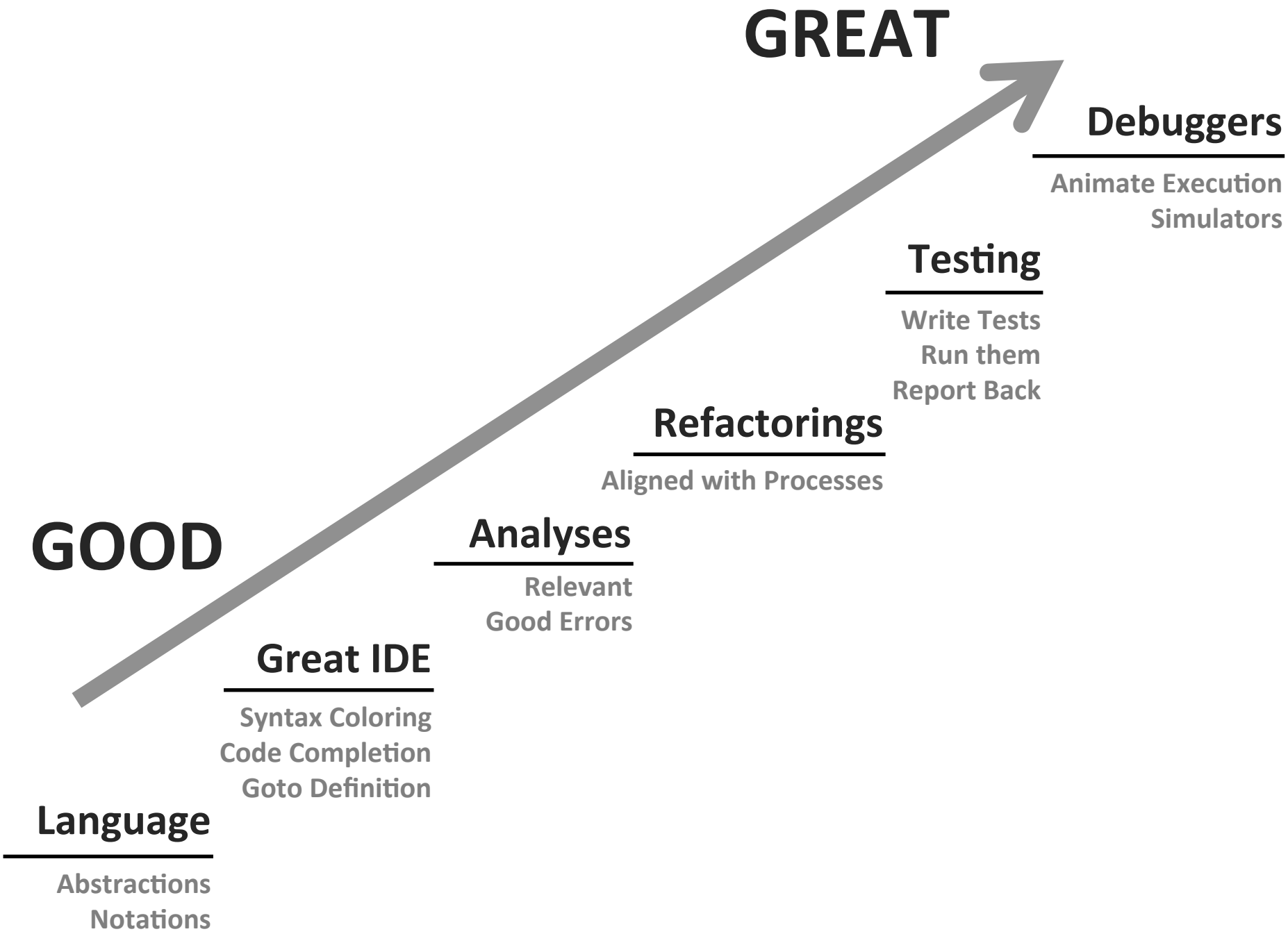
Influences on the Language




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

The Language is not Enough




GREAT
GOOD
Debuggers
Animate Execution
Simulators
Testing
Write Tests
Run them
Report Back
Refactorings
Aligned with Processes
Analyses
Relevant
Good Errors
Great IDE
Syntax Coloring
Code Completion
Goto Definition
Language
Abstractions
Notations

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

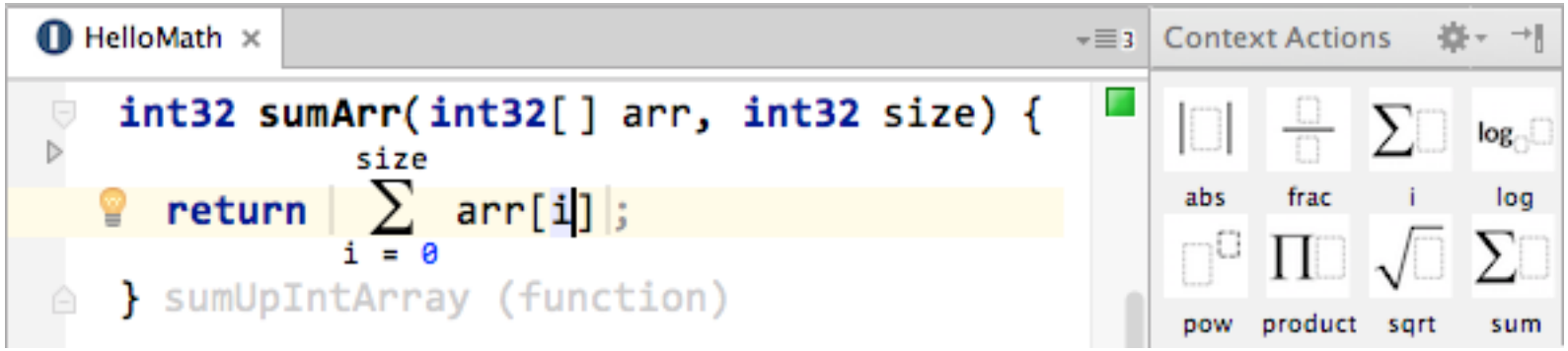
Migrate existing models
as the languages evolve.

Friendliness

Don‘t overwhelm end users with too much „cruft“

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.

Error Messages
Process-Guidance in the tool
Checklists to finish manual processes
Tool must fit the process!

Tool should makes the right thing easy.

Does this scale?

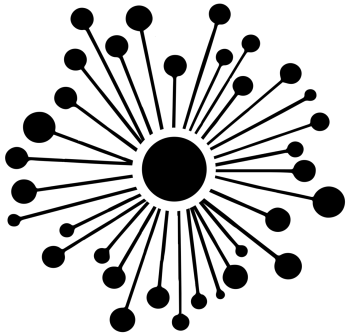


Does the approach scale?

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?

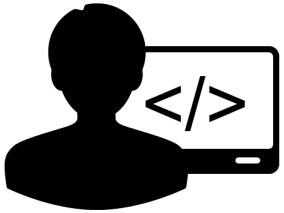
Yes, but it is a significant change, so:

- 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!

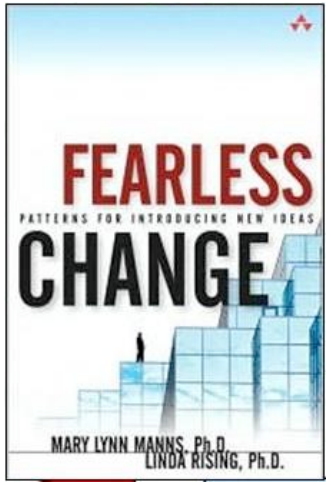


Precision and Formality	-> New Skills
Different Processes	-> Role Change
Higher Efficiency	-> Job Loss



Automation	-> Job Loss
Focus on Engineering	-> Role Change
Empower Business Ppl	-> Less Importance

Some people are afraid of this.
Take them seriously.



A change of

that must be managed!

We tried it before, and it failed.



MDA

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.

Contra-indications



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.

The Skunk Works®

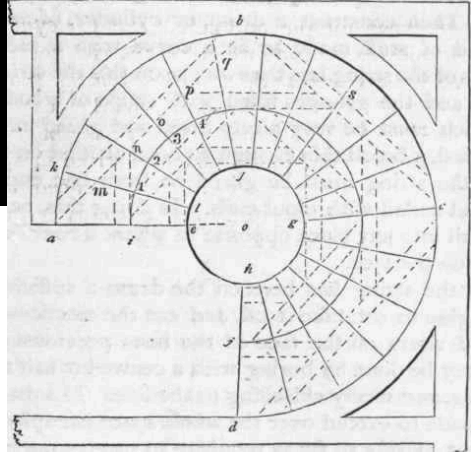
The Skunk Works

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

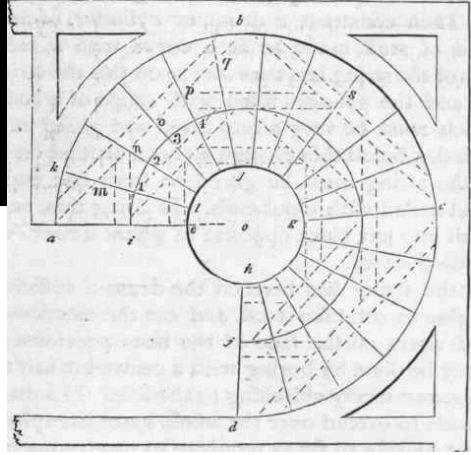
Vertical Slice through Domain, then expand

Step by Step 2

Increasing Levels of Formality

- Prose
- Prose + Glossary
- Prose + Glossary + Calculation Rules + Code Generation
- ...

Introducing the Approach



Step by Step 1

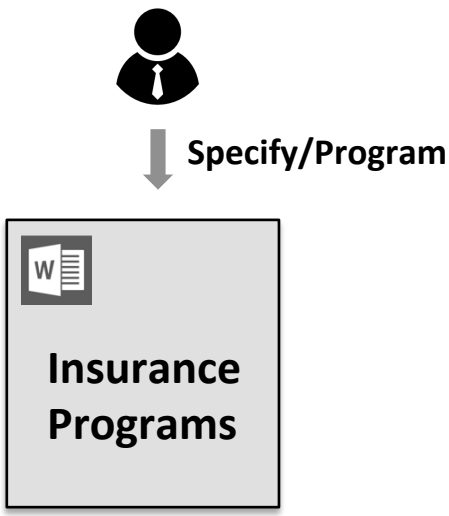
Vertical Slice through Domain, then expand

Step by Step 2

Increasing Levels of Formality

Keep the end goal (formalization, automation) in sight, othewise it is hard to justify „strange tools“ as opposed to a Wiki, e.g.

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 varibility

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 varibility

Formale Beschreibung

Funktion:	berbwvekFF		
Programmquelle:	vmsctfa1.c		
Produkt-Typ:	FONDS	**PK-Typ:**	KAPITAL-KONTO

	Name	Verw.	Entität
verwendete Attribute:	lkm_akt_param	E	PARAMETER
	lkm_faell_param	E	PARAMETER
	ber_zweck_param	E	PARAMETER
	kz_rzw_param	E	PARAMETER
	bwvek	A	RETURN

aufgerufene Funktionen: berbweinzelFF
VTRKermbtgfaellFF

Status: 18.1

Verarbeitungen

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** -1 bis **lkm_faell_param** – 1.

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

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 = *VTRKermbtgfaellFF*(lkm_faell_hilf)

Ende Falls **kz_rzw_param** = 12

bwvek = **bwvek**

+ kz_bf_hilf * *berbweinzelFF*(**lkm_akt_param**, lkm_faell_hilf - 1, **ber_zweck_param**)

Ende Schleife

return **bwvek**

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 varibility

Formale Beschreibung

Funktion: rg_kk_beta_satzTF
Programmquelle: vmscfo2.c
Produkt-Typ: FONDS, RSR **PK-Typ:** Kapital-Konto

verwendete Attribute:

Name	Verw.	Entität
fo_beta_satz	E	Kosten-Regeln
beta_satz	E	Rechnungsgrundlagen-KK
ko_ra_id	E	KOSTEN-RABATT
zmt_param	E	PARAMETER
kz_zus_gar	E	
zm	E	
beta_satz_fakt	E	VORGABEDATEN-KOSTENRABATT
zw	E	TVDKONTO_G
vtrk_zb	E	VTRK_BTG
kz_mandant	E	T_KK
satz_beta	A	Rückgabewert

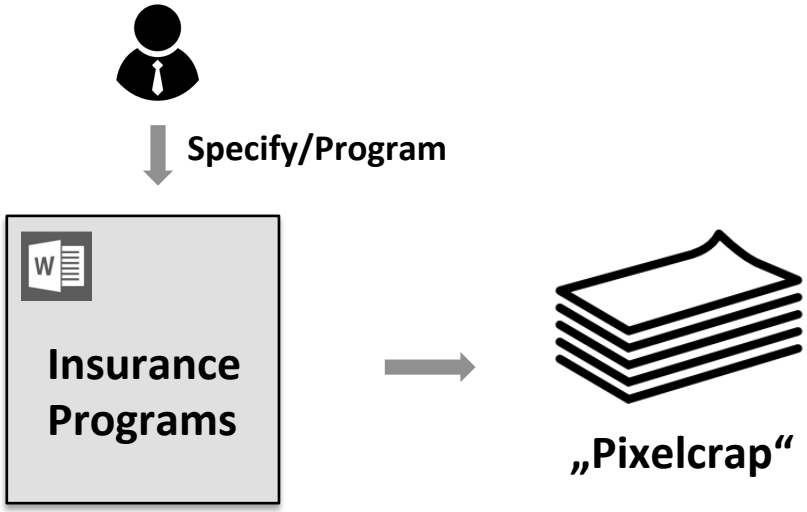
Aufgerufene Funktionen: rg_kk_beta_bp_satzTF

In dieser Funktion wird der Kostensatz β ermittelt.

Verarbeitung

fo_beta_satz	Berechnung **satz_beta**	Bemerkung
0	**satz_beta = beta_satz** Falls **zmt_param** <= 120 und **kz_zus_gar** = JA **satz_beta** = **satz_beta** * min(0,01 * max(**zmt_param** - 12; 0); 1) Ende (Falls **zmt_param** <= 120 und **kz_zus_gar** = JA)	Standard
1	**satz_beta** = **beta_satz** * min(0,01 * max(**zm** – 12; 0); 1)	PF
2	grenze = **vtrk_zb** * **zw** Falls grenze < 10000.0 satz_beta = **beta_satz** Sonst satz_beta = 0,074	GULPP

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 varibility

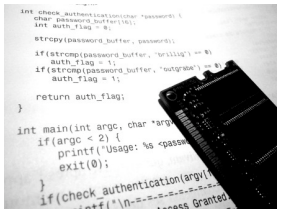
Printed, PDF

A real Example: Legacy

Specify/Program

„Pixelcrap“

Implement



C Code

Write formal code in a DSL
mixed with tables and text

No tool support whatsoever
No testing (except inspection)

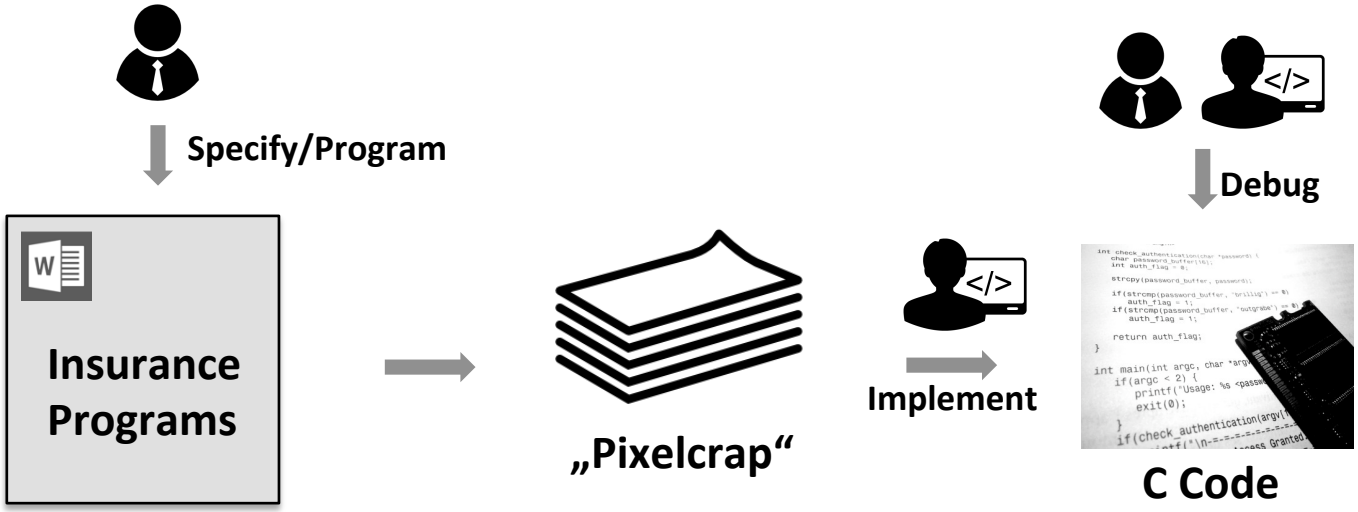
No reuse
No modularity
No varibility

Printed, PDF

Developer reads „spec“
Very idiomatic implementation

Dev acts as a human compiler and implements it in C

A real Example: Legacy



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 varibility

Printed, PDF

Developer reads „spec“
Very idiomatic implementation

Dev acts as a human compiler and implements it in C

A real Example: Current

Specify/Program/Test/Debug

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/Prog

Insurance Programs

Write formal code in a D
mixed with tables and te

Now with IDE support ar

The same nota

berbwvekFF (lkm_akt_param; lkm_faell_param; ber_zweck_param; kz_rzw_param)

Funktionenmodell berbwvekFF

Formale Beschreibung

Funktion: berbwvekFF
Programmquelle: vmsctfa1.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

VTRKermbtgfaellFF (a)
berbweinzelFF (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 - 1# bis #lkm_faell_param - 1#. Der Parameter @kz_rzw_param steuert die zu berücksichtigende Zahlweise des Zahlungsstroms. Möglich sind zur Zeit nur die Ausprägungen 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 = VTRKermbtgfaellFF (lkm_faell_hilf)
  Ende Falls kz_rzw_param = 12
  bwvek = bwvek + kz_bf_hilf * berbweinzelFF (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

berbwvekFF (lkm_akt_param; lkm_faell_param; ber_zweck_param; kz_rzw_param)

Funktionenmodell berbwvekFF

Formale Beschreibung

Funktion: berbwvekFF
Programmquelle: vmsctfa1.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

VTRKermbtgfaellFF (a)
berbweinzelFF (a; b; c)

Beschreibung

Die Funktion liefert den Barwert per @lkm_akt_param des vorschüssigen Zahlungss
@lkm_akt_param bis @lkm_faell_param – jeweils einschließlich. Zahlungszeitpunkt
#lkm_akt_param – 1# bis #lkm_faell_param – 1#. Der Parameter @kz_rzw_param steu
Zahlweise des Zahlungsstroms. Möglich sind zur Zeit nur die Ausprägungen 0 (Zahl
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 = VTRKermbtgfaellFF (lkm_faell_hilf)
  Ende Falls kz_rzw_param = 12
  bwvek = bwvek + kz_bf_hilf * berbweinzelFF (lkm_akt_param; lkm_faell_hilf - 1
Ende Schleife über lkm_akt_param bis lkm_faell_param

return bwvek
```

Formale Beschreibung

Funktion:	berbwvekFF		
Programmquelle:	vmsctfa1.c		
Produkt-Typ:	FONDS	**PK-Typ:**	KAPITAL-KONTO
	Name	Verw.	Entität
verwendete Attribute:	lkm_akt_param	E	PARAMETER
	lkm_faell_param	E	PARAMETER
	ber_zweck_param	E	PARAMETER
	kz_rzw_param	E	PARAMETER
	bwvek	A	RETURN
aufgerufene Funktionen:	berbweinzelFF		
	VTRKermbtgfaellFF		
Status:	18.1		

Verarbeitungen

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** -1 bis **lkm_faell_param** – 1.

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

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 = *VTRKermbtgfaellFF*(lkm_faell_hilf)

Ende Falls **kz_rzw_param** = 12

bwvek = **bwvek**

+ kz_bf_hilf * *berbweinzelFF*(**lkm_akt_param**, lkm_faell_hilf - 1, **ber_zweck_param**)

Ende Schleife

return **bwvek**

A real Exam

Specify/Program/Test

Insurance Programs

Write formal code in a DSL mixed with tables and text

Now with IDE support and execu

The same notation!

rg_kk_beta_satzTF (zmt_param)

Formale Beschreibung

Funktion: rg_kk_beta_satzTF
Programmquelle: vmscfo2.c
Produkt-Typ: Fonds, RSR **PK-Typ:** Kapital-Konto
Status: 18.1

Parameter-Attribute
zmt_param

Verwendete VADM-Attribute
rg_kk.fo_beta_satz E
rg_kk.beta_satz E
rg_kk.kz_zus_gar E
rg_kk.zm E
rg_kk.vtrk_zb E
rg_kk.zw E
rg_kk.ko_ra_id E
rg_kk.kz_mandant E
rg_kk.beta_satz_fakt E

Rückgabe-Attribut
satz_beta

aufgerufene Funktionen
Kommazahl MIN (Kommazahl a; Kommazahl b)
Kommazahl MAX (Kommazahl a; Kommazahl b)
rg_kk_beta_bp_satzTF ( )
rg_kk_beta_ap_satzTF ( )

Beschreibung
In dieser Funktion wird der Kostensatz β ermittelt.

Hilfsvariablen
grenze
fak_beta
beta_bp_satz_hilf
beta_ap_satz_hilf

Verarbeitungen

rg_kk.fo_beta_satz	Berechnung @satz_beta	Bemerkung
0	*satz_beta* = rg_kk.beta_satz Falls zmt_param <= 120 und rg_kk.kz_zus_gar = JA *satz_beta* = *satz_beta* * MIN (0,01 * MAX (zmt_param - 12; 0); 1) sonst Verarbeitung hinzufügen Ende Falls zmt_param <= 120 und rg_kk.kz_zus_gar = JA	Standard
1	*satz_beta* = rg_kk.beta_satz * MIN (0,01 * MAX (rg_kk.zm - 12; 0); 1)	PF
2	grenze = rg_kk.vtrk_zb * rg_kk.zw Falls grenze < 10000,0 *satz_beta* = rg_kk.beta_satz sonst *satz_beta* = 0,074 Ende Falls grenze < 10000,0	GULPP
3	*satz_beta* = rg_kk.beta_satz Falls zmt_param <= 156 und rg_kk.kz_zus_gar = JA fak_beta = 0,05 * MIN (zmt_param / 12 - 1; 1) +	FV Standard Z-D ab TV 8

A real Example: Current

Specify/Pro

Insurance Programs

Write formal code in a
mixed with tables and

Now with IDE support a

The same not

GeometrischesMittel () ×

Funktionenmodell GeometrischesMittel

Formale Beschreibung

```
Funktion:        GeometrischesMittel
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

```
result = √-1
result = √a * b
return re
```

```
result                                  ^returnParameter (t.c.z.v.m.f.f.demo.sprint2.GeometrischesMittel)
HOCHR_VK.REN_A                          ^literals (t.c.z.v.m.f.f.demo.sprint2.HOCHR_VK)
HOCHR_VK.REN_S                          ^literals (t.c.z.v.m.f.f.demo.sprint2.HOCHR_VK)
HOCHR_VK.REN_U                          ^literals (t.c.z.v.m.f.f.demo.sprint2.HOCHR_VK)
PK_TYP_ID.GRUPPIERUNG_RECHNUNGSLEGUNG   ^literals (t.c.z.v.m.f.f.demo.sprint2.PK_TYP_ID)
PK_TYP_ID.REGULIERUNG                   ^literals (t.c.z.v.m.f.f.demo.sprint2.PK_TYP_ID)
tarif.reg.fo_aktiv                      ^members (t.c.z.v.m.f.f.demo.sprint2.tarif)
tarif.reg.fo_bil_dir                    ^members (t.c.z.v.m.f.f.demo.sprint2.tarif)
tarif.reg.fo_bs                         ^members (t.c.z.v.m.f.f.demo.sprint2.tarif)
tarif.rund.rd_la2_rel                   ^members (t.c.z.v.m.f.f.demo.sprint2.tarif)
```

Context Actions

Attribute
- a
- b
- result

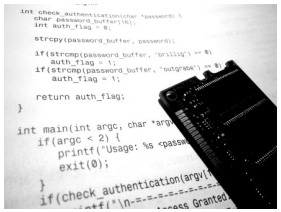
Aufzählungen
- HOCHR_VK.BUZB
- HOCHR_VK.BUZR
- HOCHR_VK.FLV
- HOCHR_VK.GFV
- HOCHR_VK.HZV_A
- HOCHR_VK.HZV_S
- HOCHR_VK.KAP
- HOCHR_VK.REN_A
- HOCHR_VK.REN_S
- HOCHR_VK.REN_U
- HOCHR_VK.RIS
- HOCHR_VK.RIZ
- HOCHR_VK.RSR
- HOCHR_VK.SBU
- HOCHR_VK.SEU
- PK_TYP_ID.AUSGANGS_KOMPONENTE
- PK_TYP_ID.BANK_KOMPONENTE
- PK_TYP_ID.DYNAMIK
- PK_TYP_ID.EXTERN_VERWALTETER_TARIF
- PK_TYP_ID.FOERDERKENNZEICHEN
- PK_TYP_ID.GRUPPIERUNG_RECHNUNGSLEGUNG
- PK_TYP_ID.KAPITAL_KONTO
- PK_TYP_ID.LV_PRODUKT_VT_PARAMETER
- PK_TYP_ID.LV_TARIF
- PK_TYP_ID.MANDANT
- PK_TYP_ID.OPTION
- PK_TYP_ID.PROZESS
- PK_TYP_ID.REGULIERUNG
- PK_TYP_ID.RISIKO_PRUEFUNG
- ECK_VERSICHERUNG
- ILAUSZAHLUNG
- RGUETUNGSGRUNDLAGE
- RTRIEBSWEG
- AEHRUNG
- HLWEG
- HLWEISE
- D
- KUNFT
- MY
- SERVICE.ERH

A real Example: Current

Specify/Program/Test/Debug

Insurance Programs

Generate



C Code

Write formal code in a DSL
mixed with tables and text

Now with IDE support and executable tests

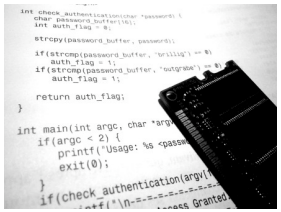
The same notation!

Exactly the same C code.

A real Example: Future

Specify/Program/Test/Debug

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 do do it.

Is this the next legacy system?



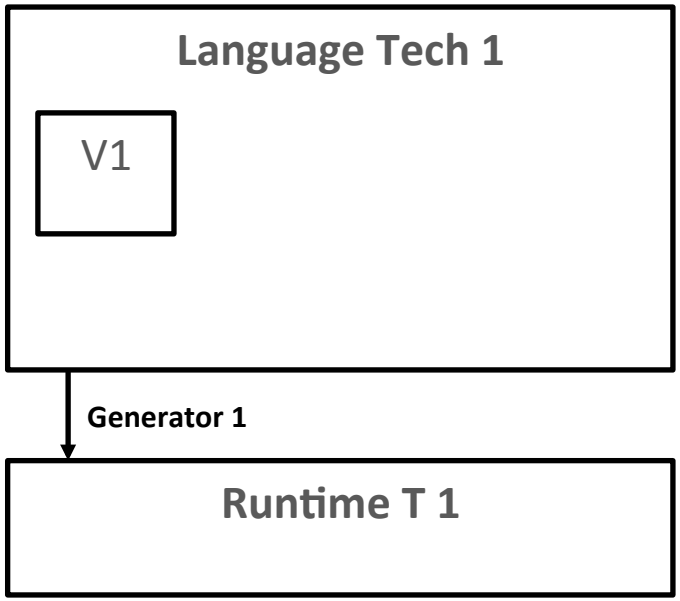
“Today‘s software is tomorrow‘s legacy system.”

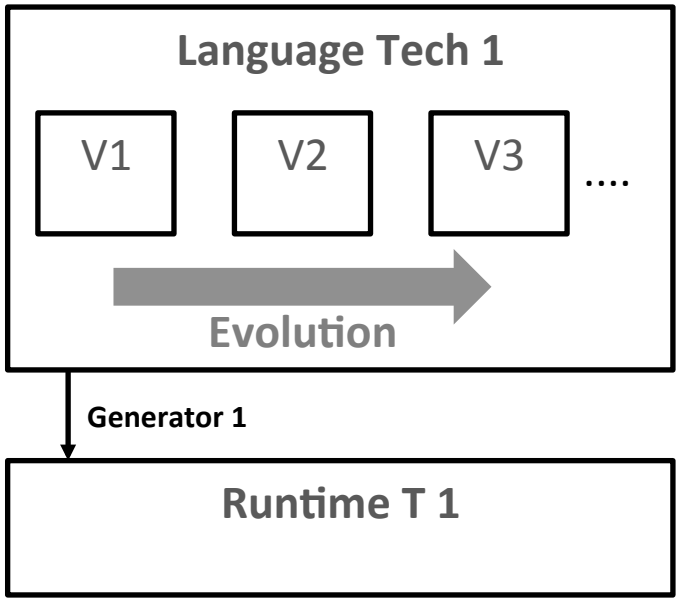
Or is it?

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“

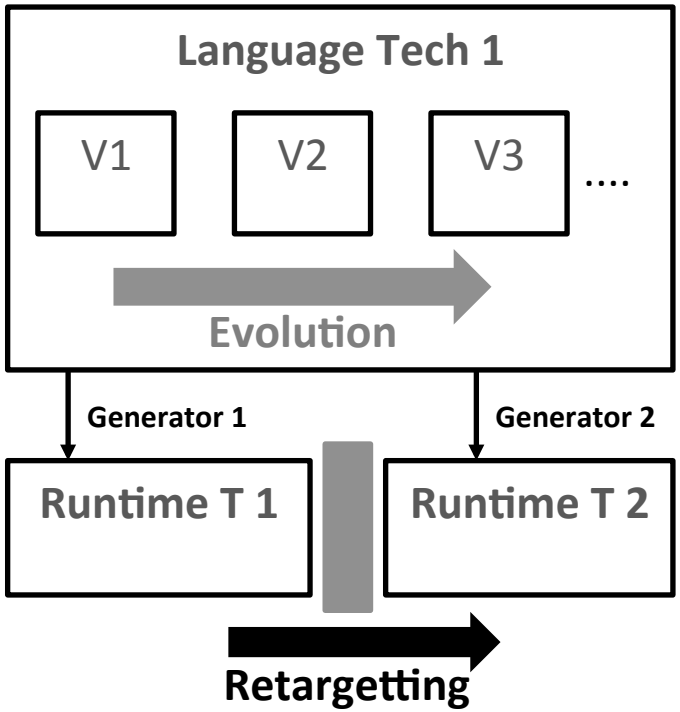

Language Tech 1
V1
Generator 1
Runtime T 1



Existing models become incompatble with new language

⇒ **Language Versions**

Migration Scripts

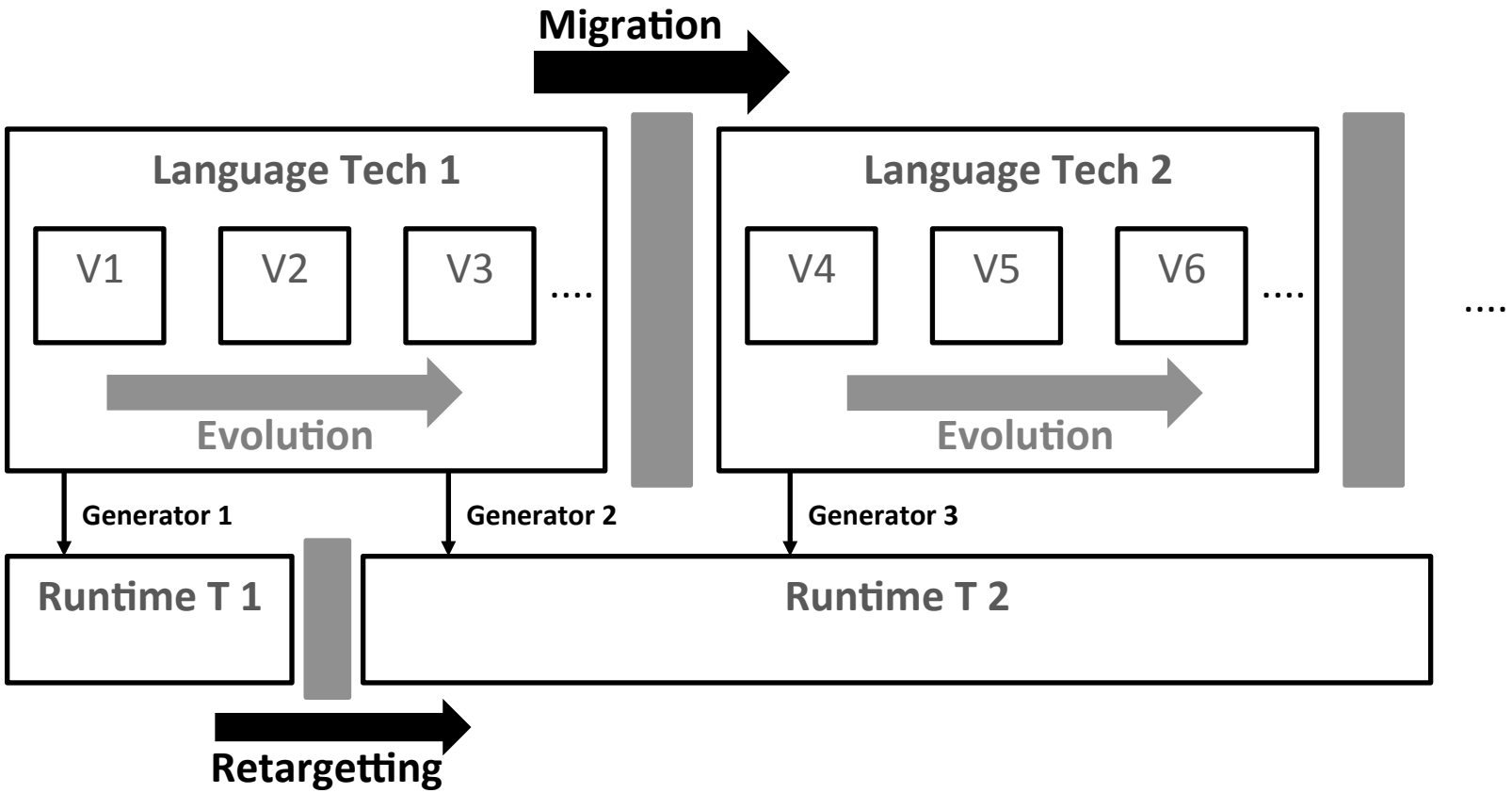


Runtime Tech outdated, uncool or slow

⇒ **Keep Lang Technology**

Keep Models

Build new Generator



Language Tech outdated, uncool

⇒ **Build new Tool**

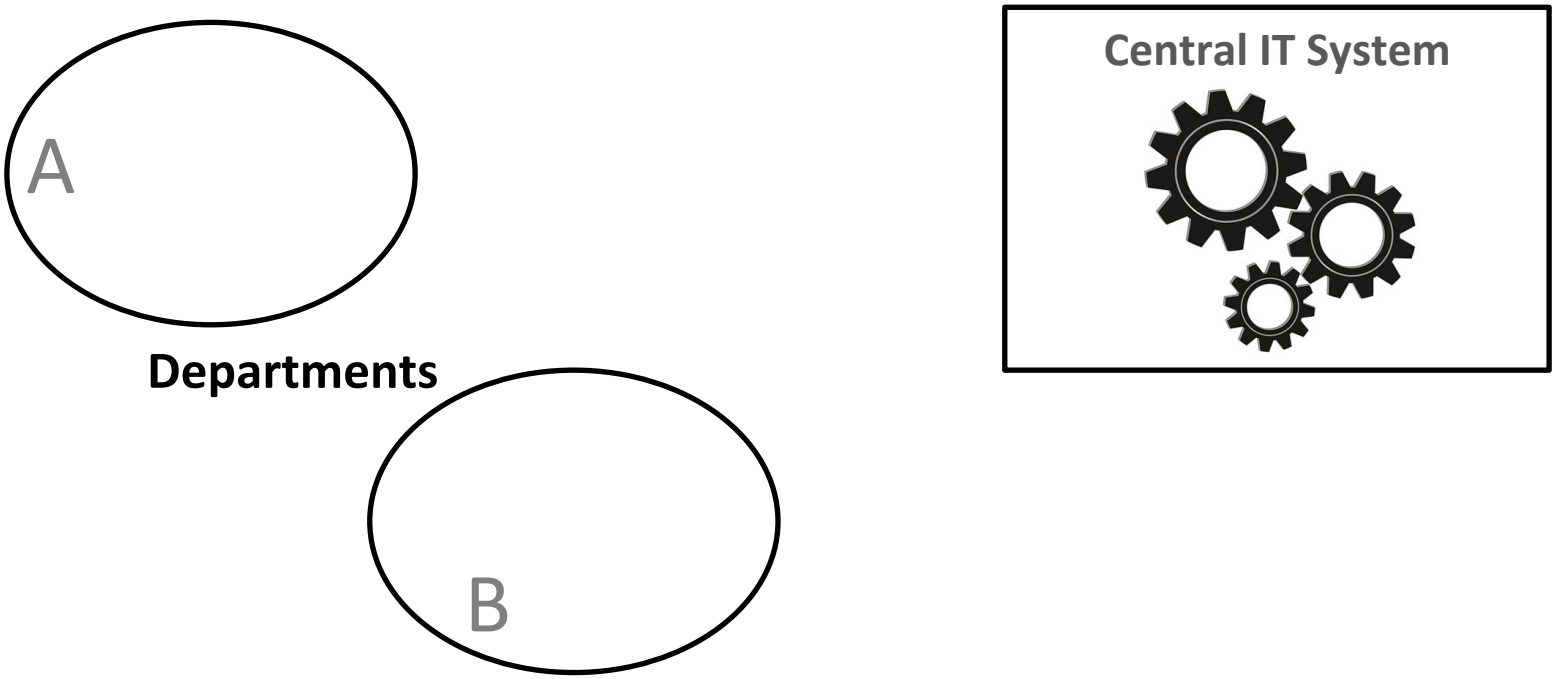
Migrate Data Simple, because it well-defined domain semantics and free from „technology stuff“

“Today‘s software is tomorrow‘s legacy system.”

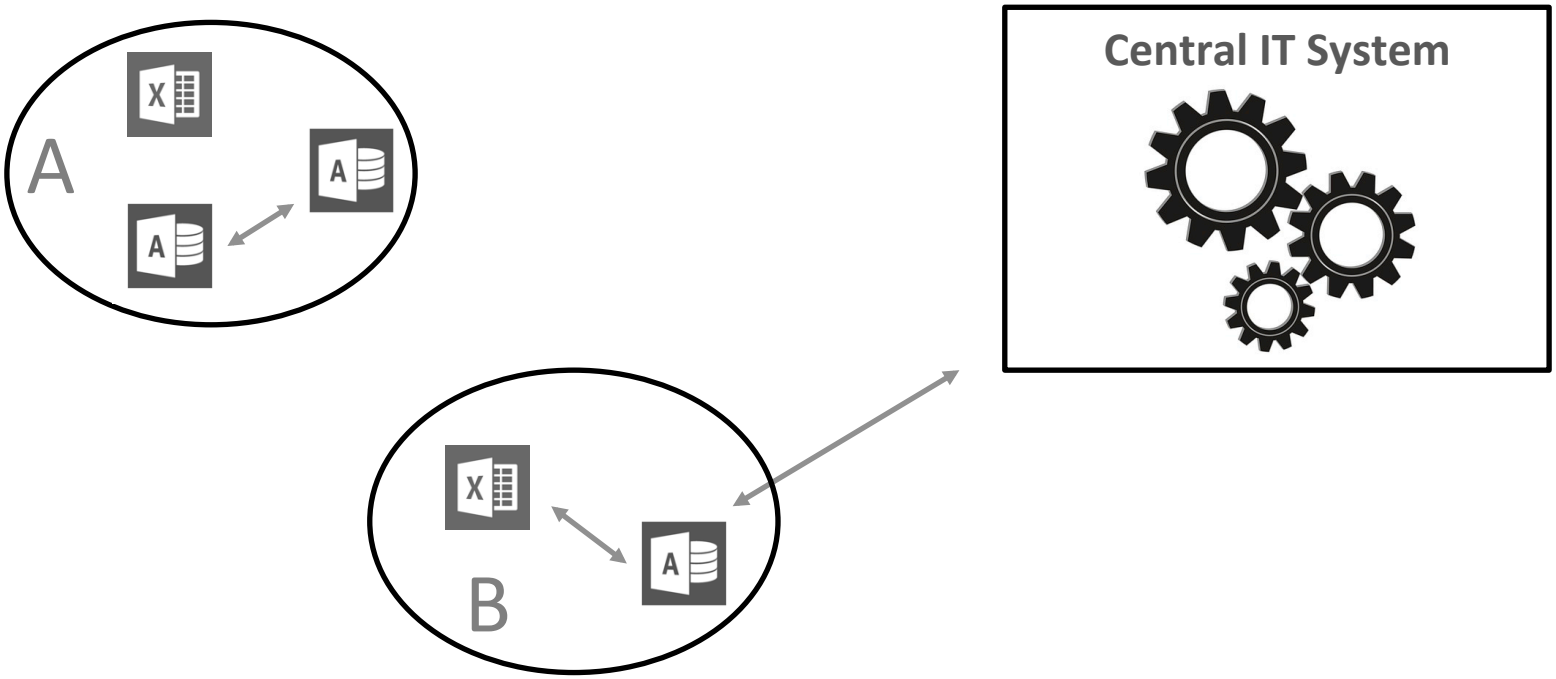
No, it is not.

Grass Root Excel/Access



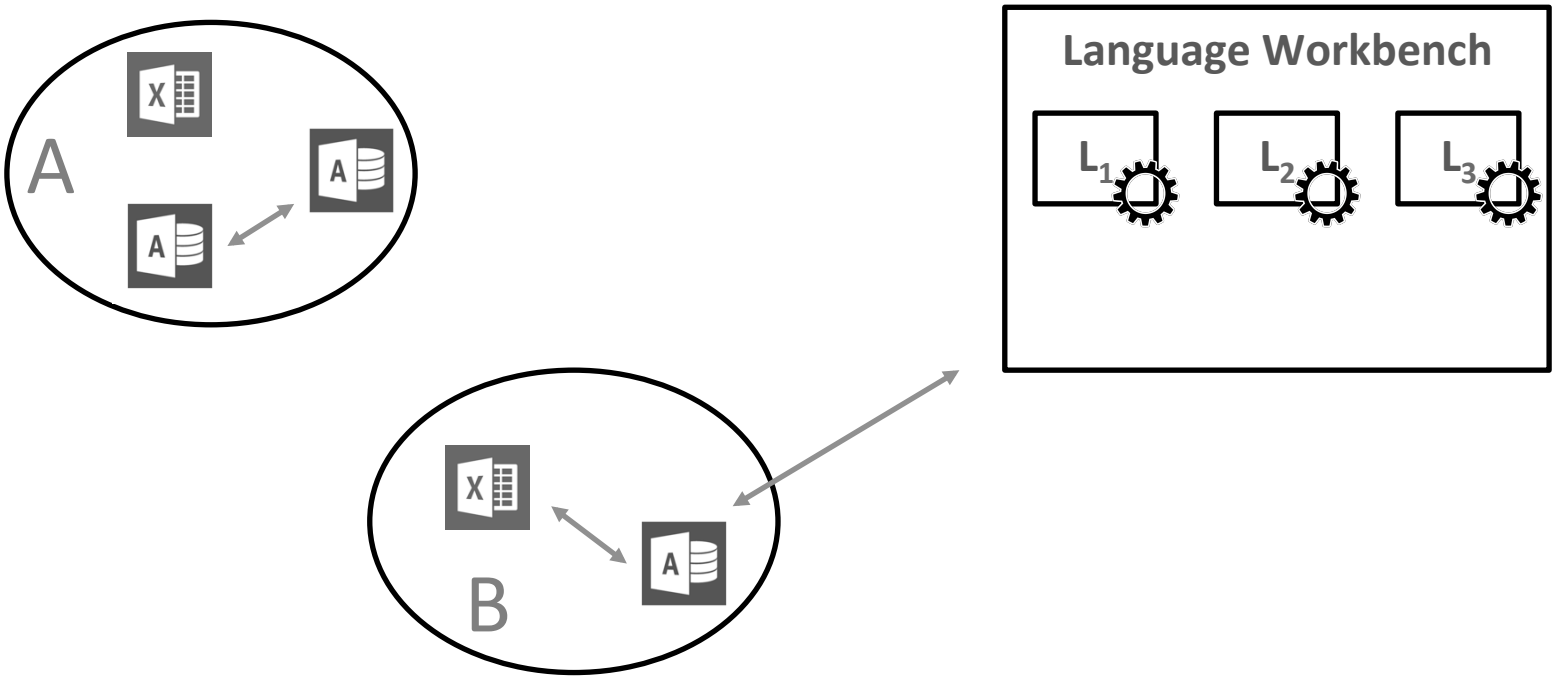


Various Departments interact with an opaque central IT system

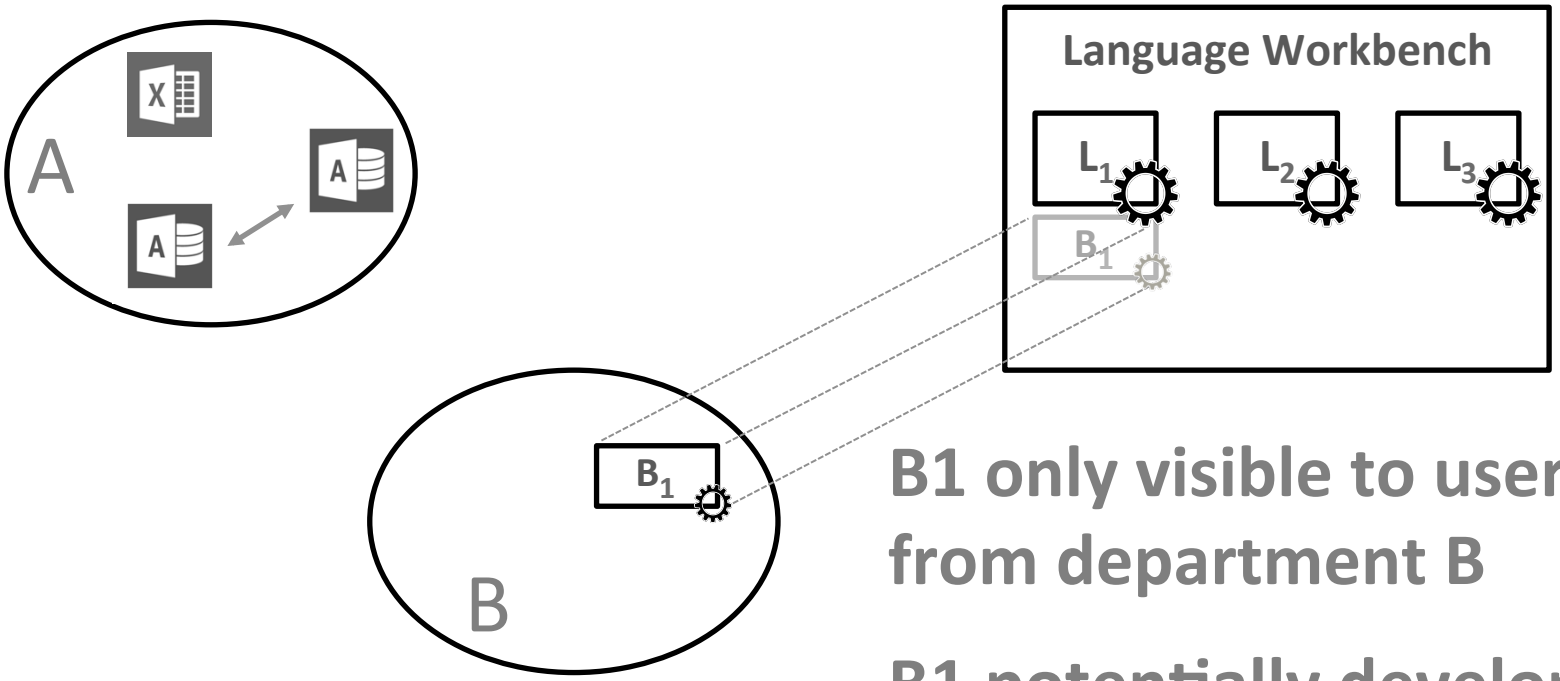


Departments may build their own ad-hoc tools with Excel and Access

Possibly connected to CITS



Now: central IT system is an LWB with a couple of Languages

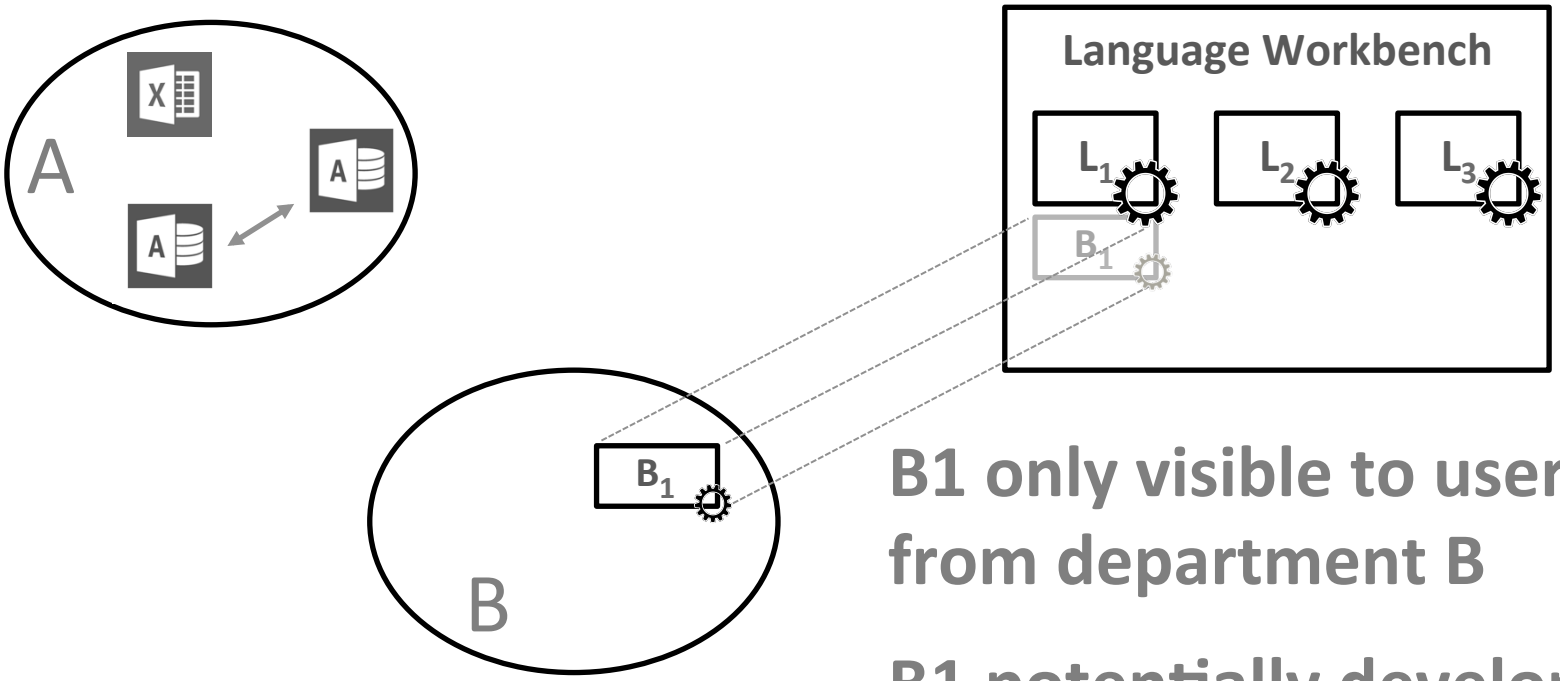


B1 only visible to users from department B

B1 potentially developed by devs associated with B

Department-specific functionality realized as language extensions

Non-CITS-connected systems remain as they are.



B1 only visible to users from department B

B1 potentially developed by devs associated with B

Department-specific functionality realized as language extensions

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 defnitions.

If you use this approach for real,

you should have language engineering expertise in house.

6

Summary

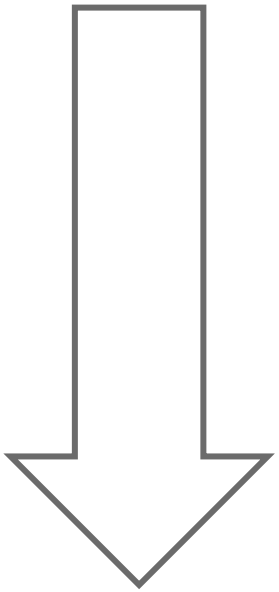
Expressivity for Core Domain Knowledge

Build Language for Domain!

User-Friendly Notation

Great Tool/IDE

You‘ve seen the demos.

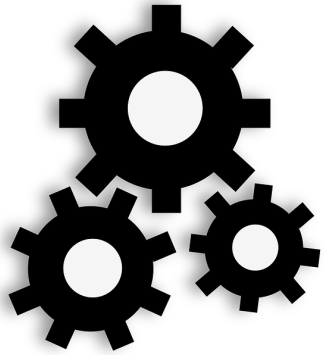


Testing

An integrated DSL for testing.

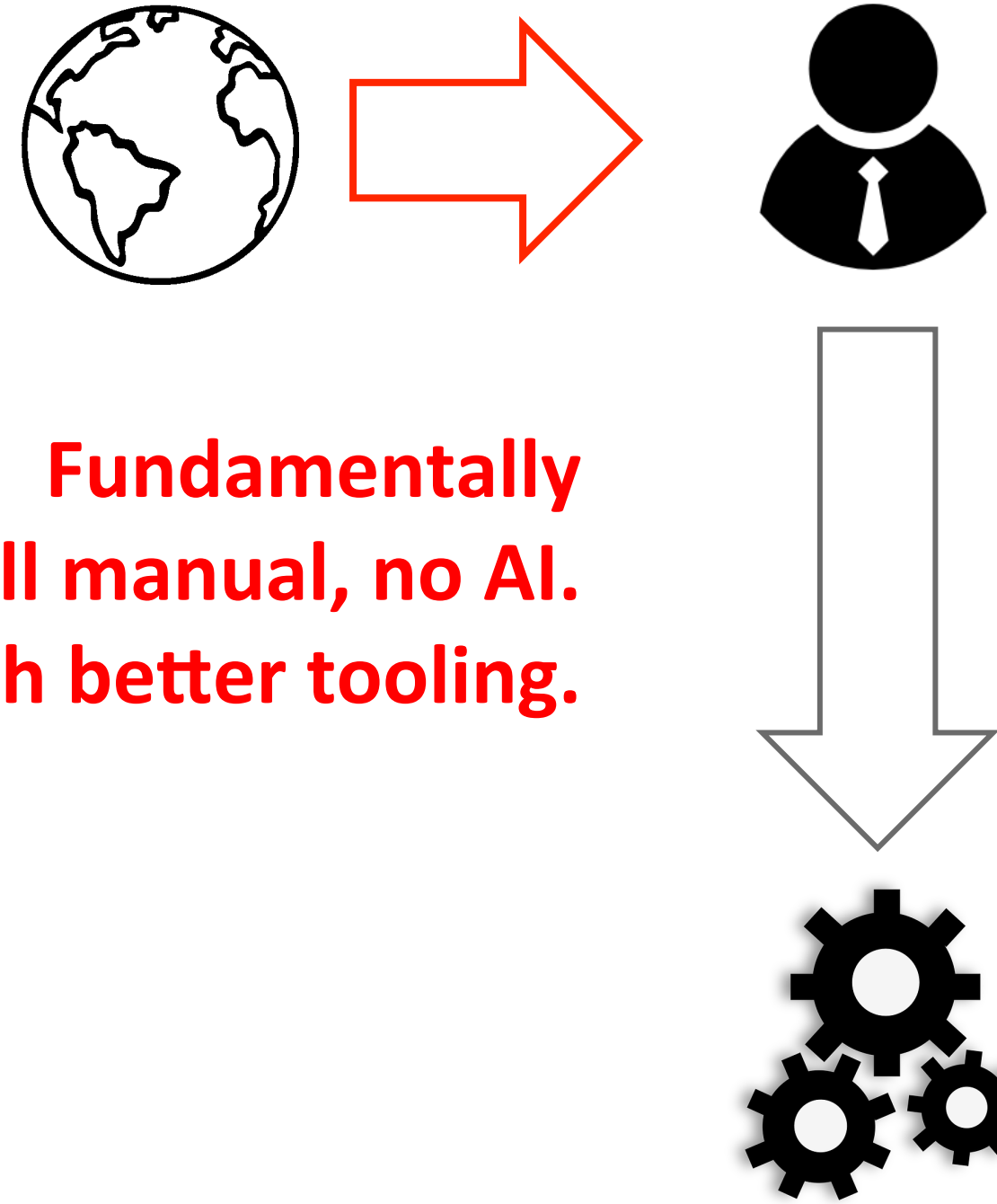
Meaningful Analyses

Types, Consistency, Checking

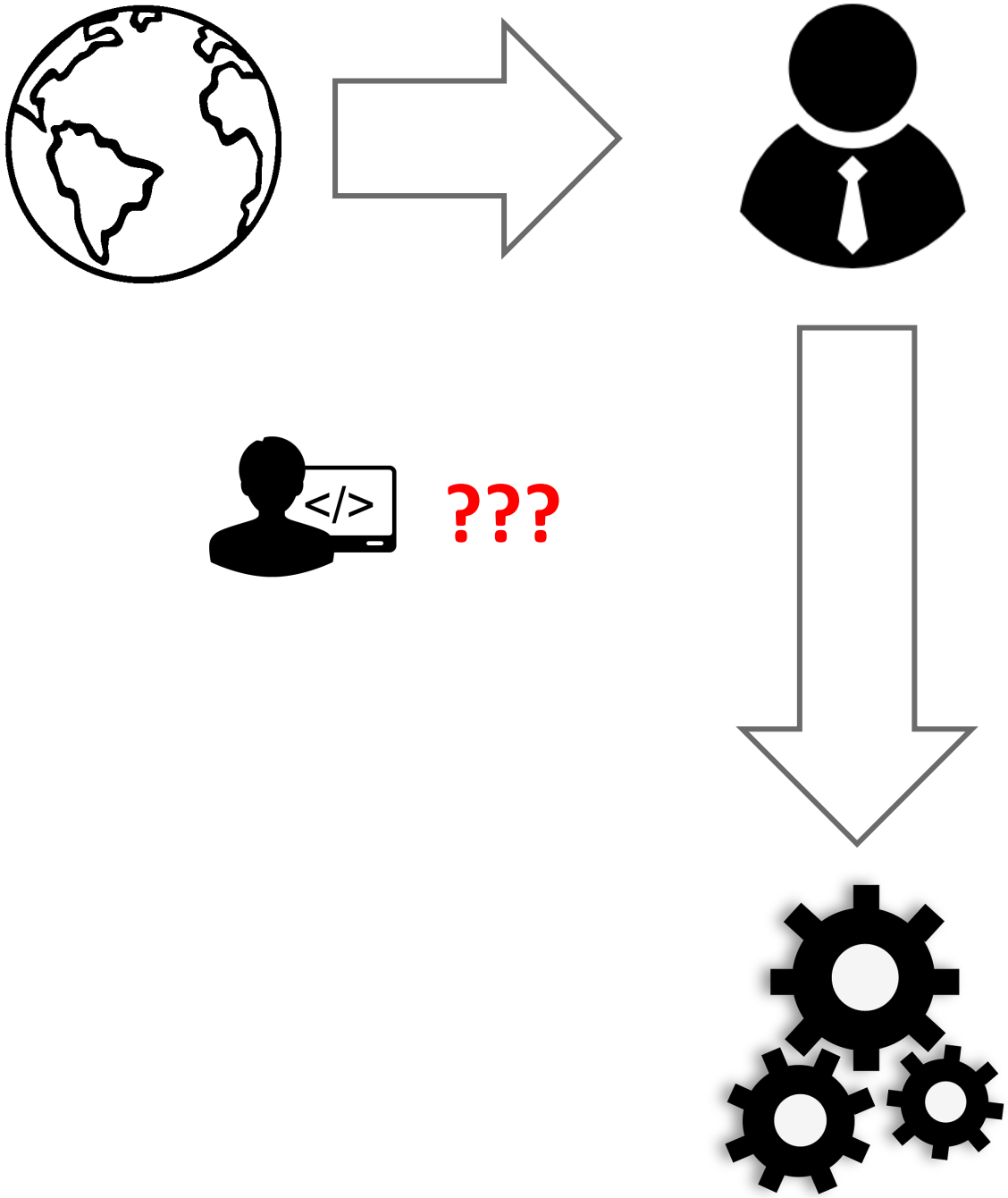


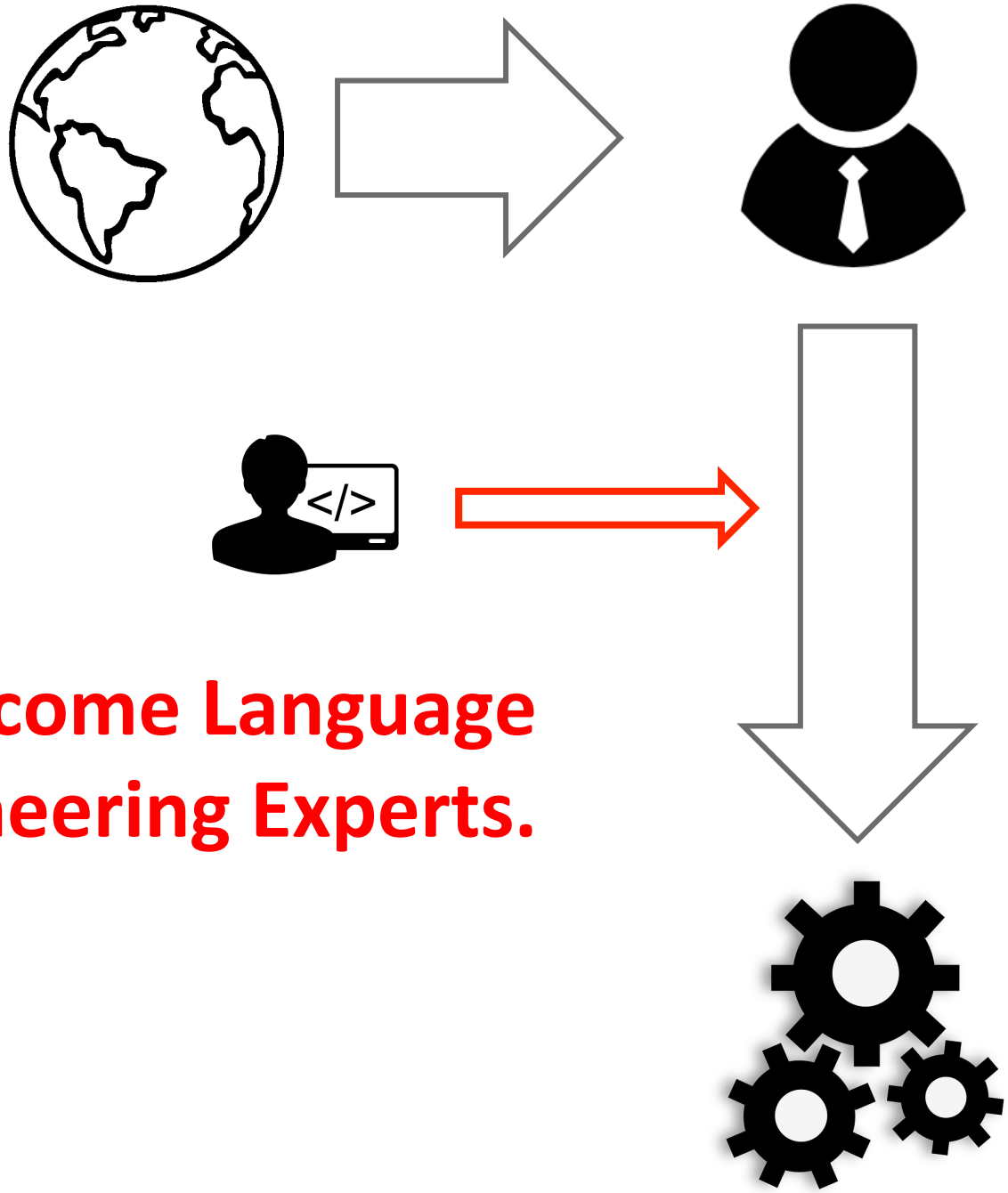
Synthesis of Software

Code Generation.

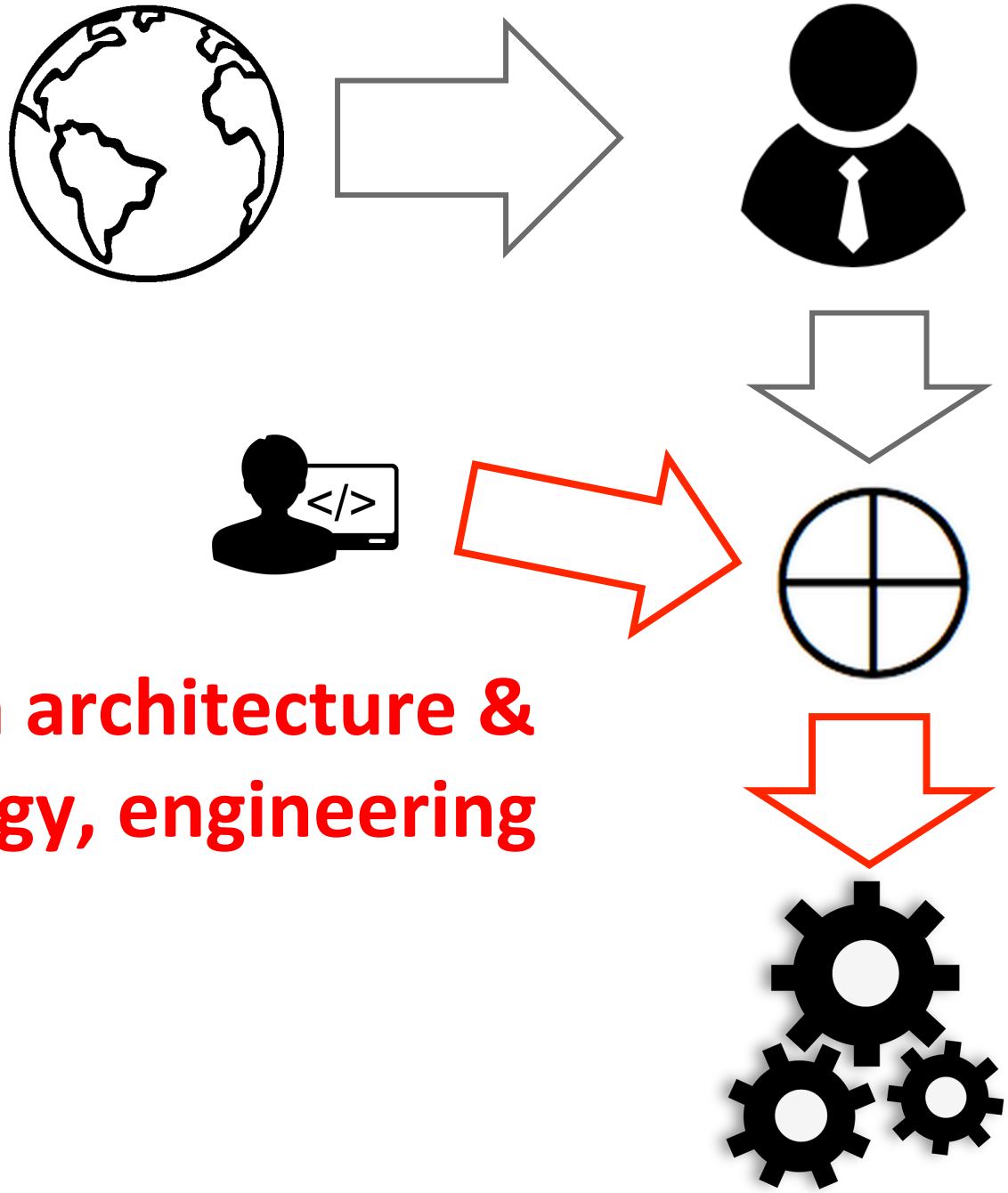


Fundamentally
still manual, no AI.
But much better tooling.





Become Language Engineering Experts.



Focus on architecture & technology, engineering

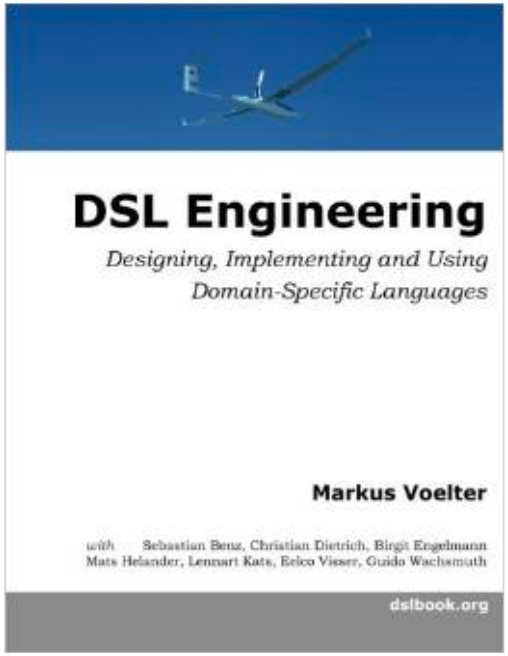
If you have to build a business app,

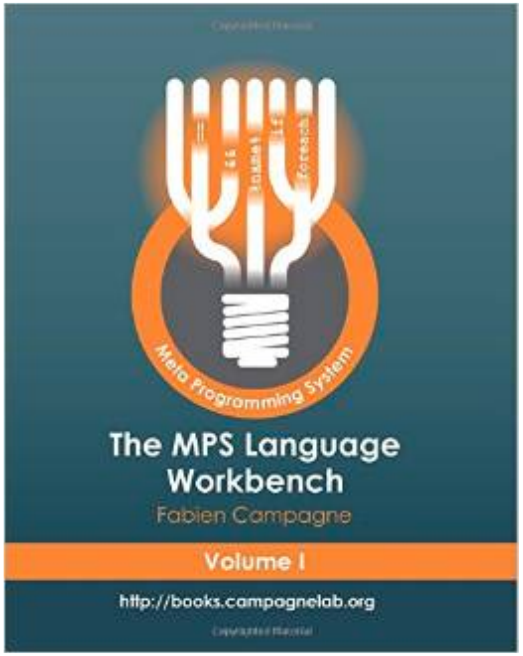
consider using an LWB as the foundation,

and recasting the „application“ as a set of languages.

[open

source





mbeddr

MPS Meta Programming System

voelter { ingenieurbüro für softwaretechnologie // itemis

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