

Language-Oriented Business Applications

Helping **End Users**
become
Programmers

Markus Völter

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

1



**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

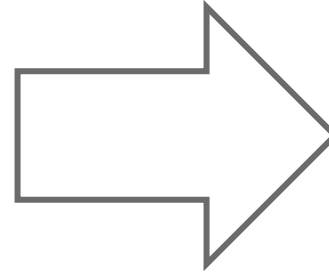
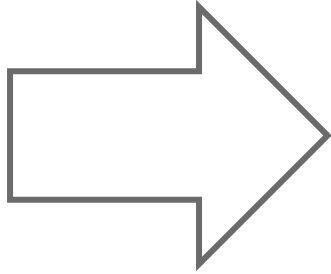
... but is a typically
implemented in **software**

[Business Knowledge]

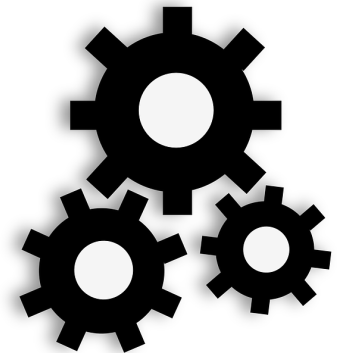
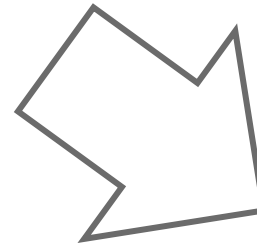
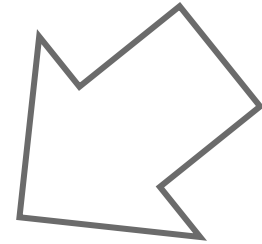
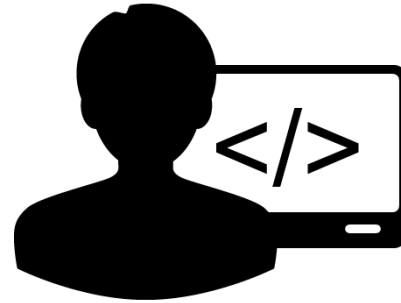
**SO HOW DOES IT GET
INTO THE SOFTWARE?**

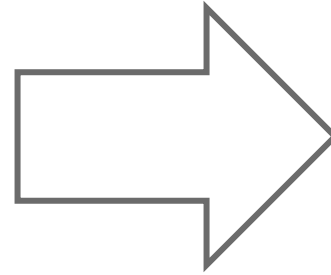
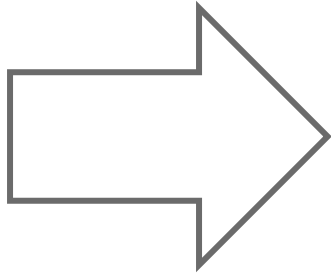
Contributed **not** by developers

... but is a typically
implemented in **software**

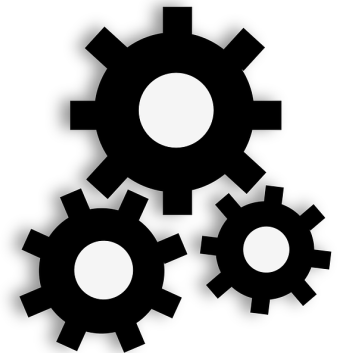
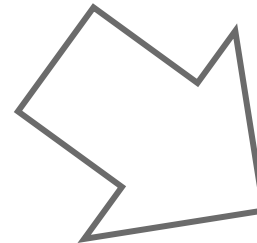
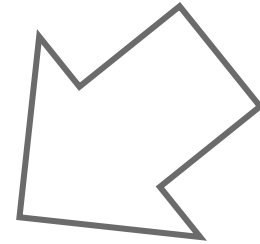
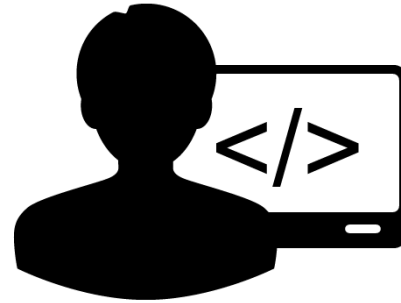


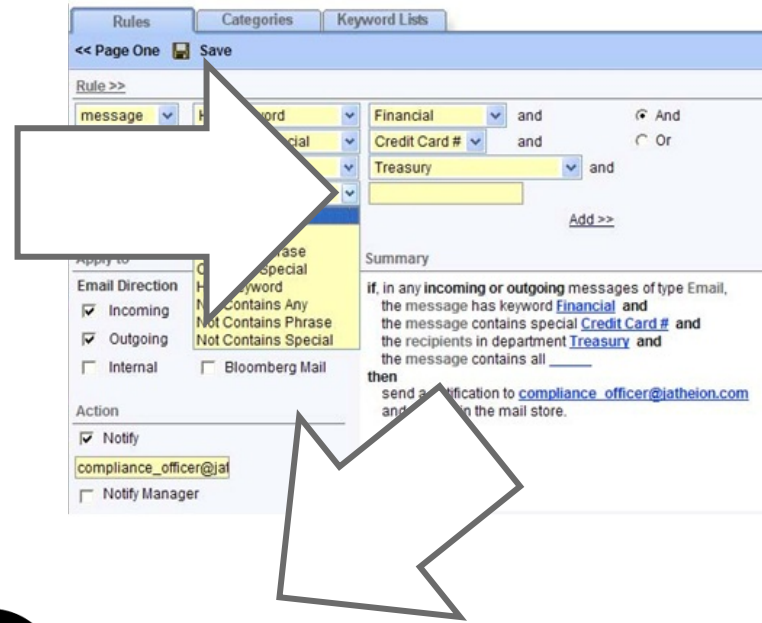
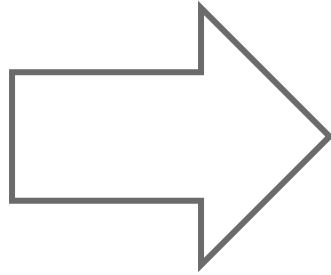
Reality



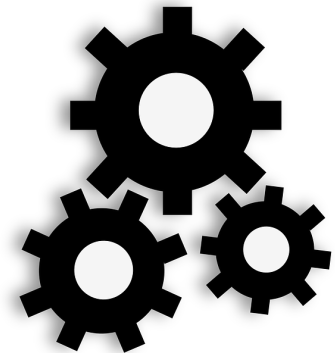
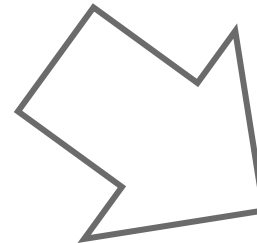
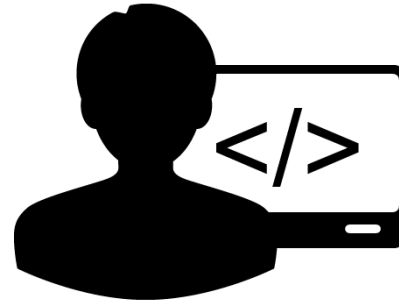


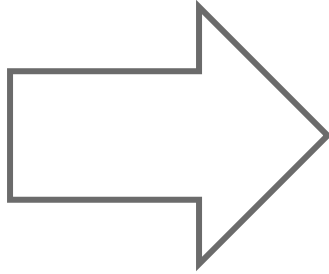
Reality



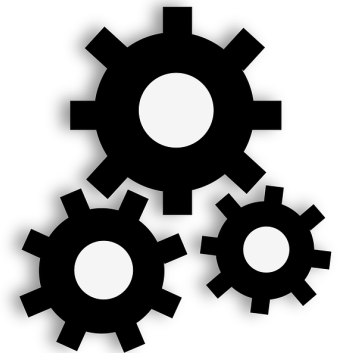
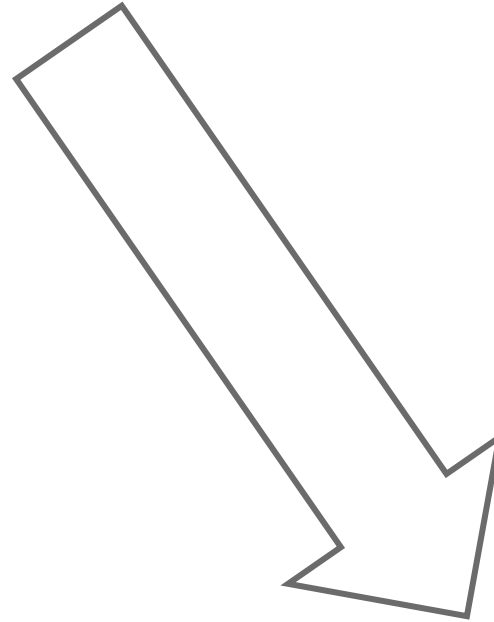


Reality





Goal!?





**Let Business/Domain
people contribute directly!**

**Give them expressive,
productive tools to do so!**

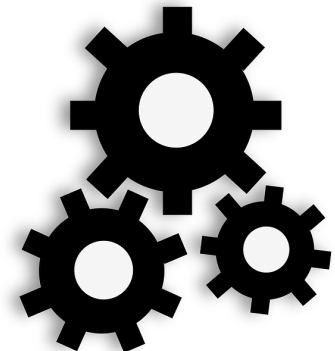
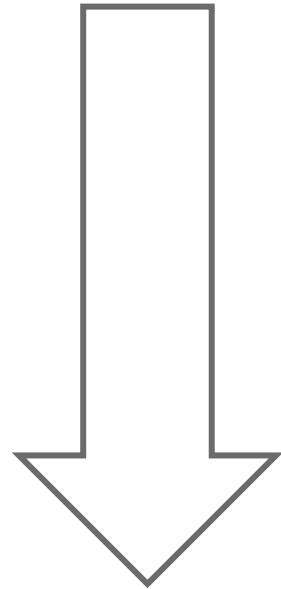
**Expressivity for Core
Domain Knowledge**

**User-Friendly Notation
Great Tool/IDE**

Testing

Meaningful Analyses

Synthesis of Software







Not a software engineer.

Does not care about „software stuff“

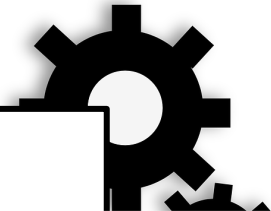
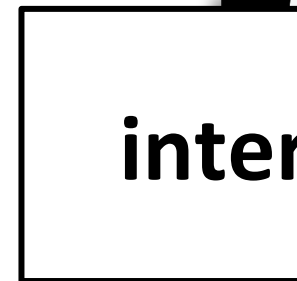
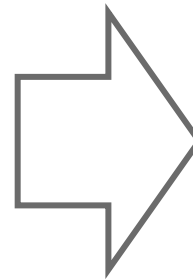
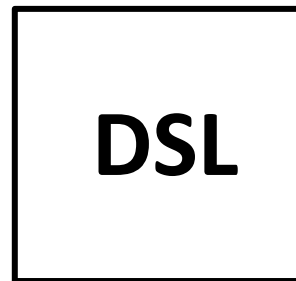
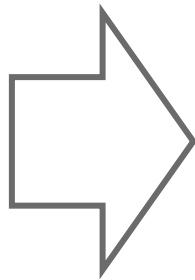
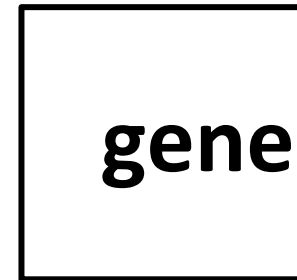
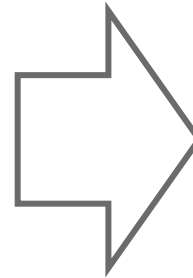
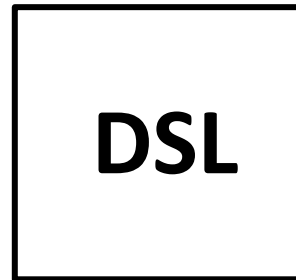
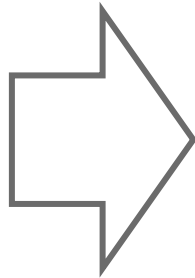
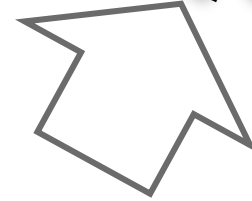
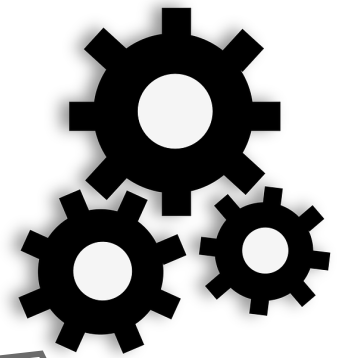
But understands the domain very well.

He is a professional, not a casual hacker.

2



Language Workbenches



interpreter

DSL

generator

interpreter

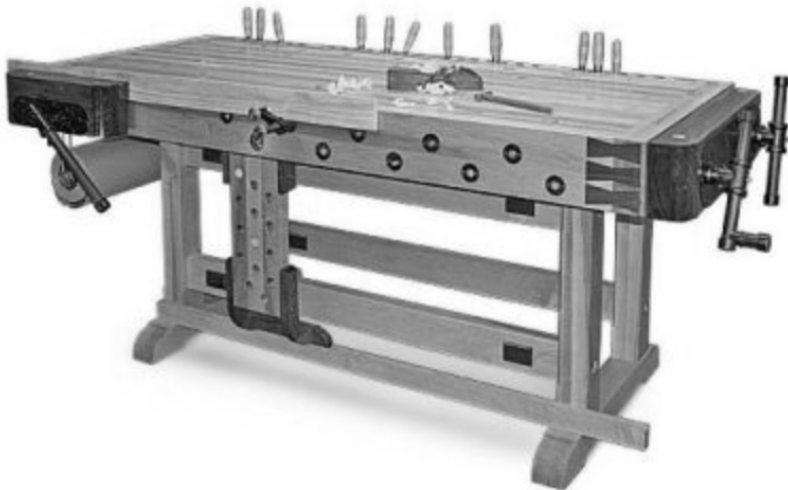
**An old idea
from the 1970s.**

BUT...

Language Workbench

(Martin Fowler, 2004)

Freely
define
languages and
integrate
them

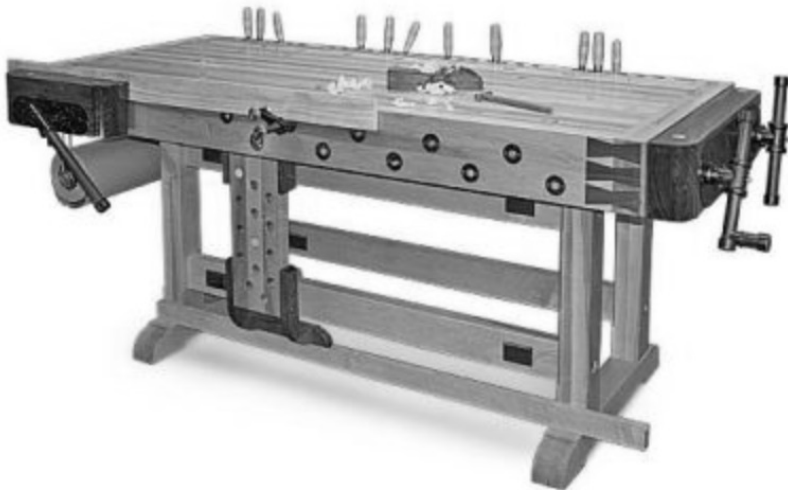


Language Workbench

(Martin Fowler, 2004)

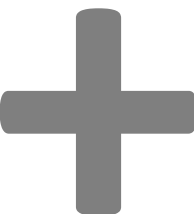
powerful
editing +
testing
refactoring
debugging
groupware

language definition
implies
IDE definition

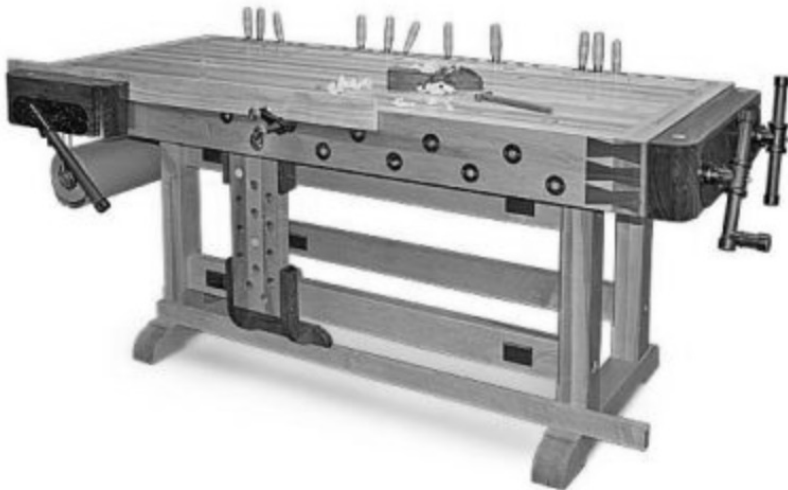


Language Workbench

(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>

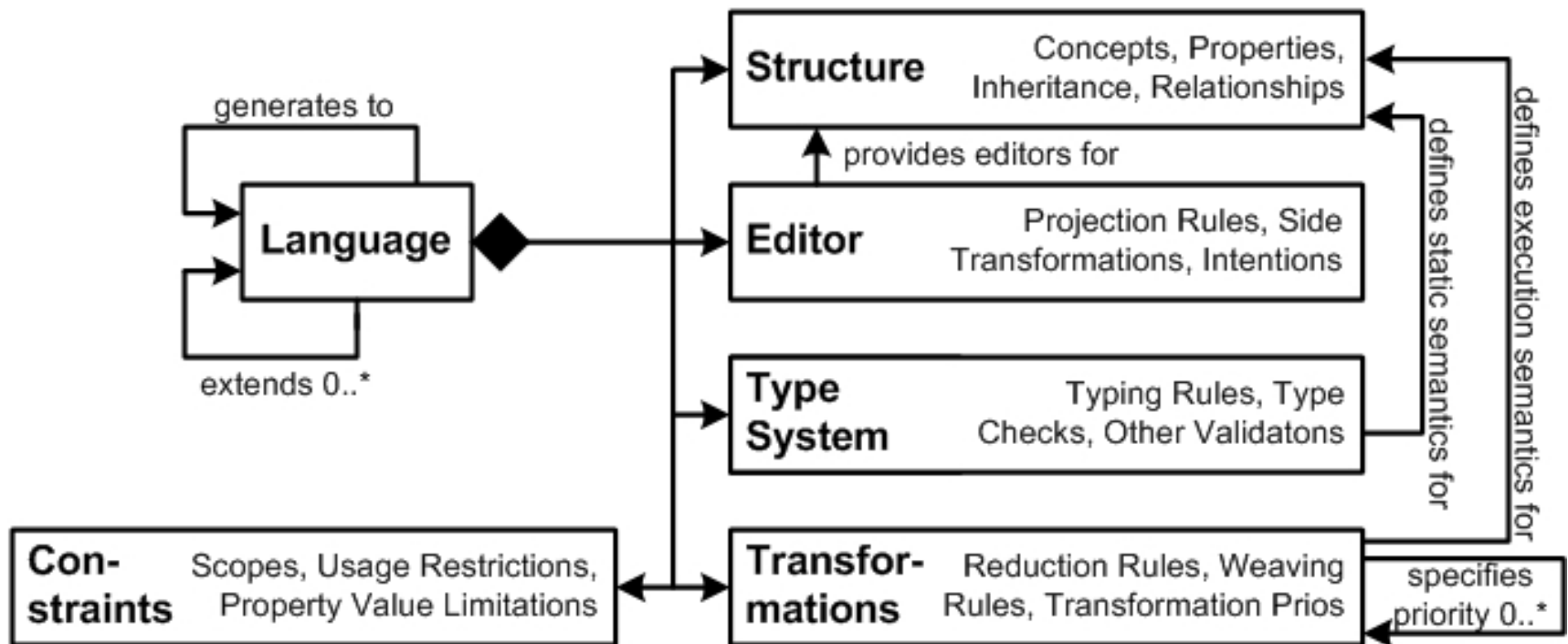


V 3.2 is current

V 3.3 to be released Q4 2015

[Language Workbench]

Comprehensive Support for many aspects of Language Definition.



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



SIEMENS

fortiss



**UNIVERSITY OF
WATERLOO**



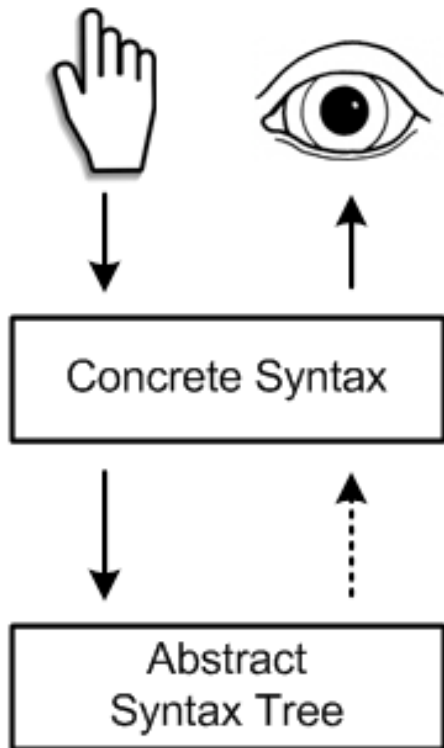
Belastingdienst

itemis

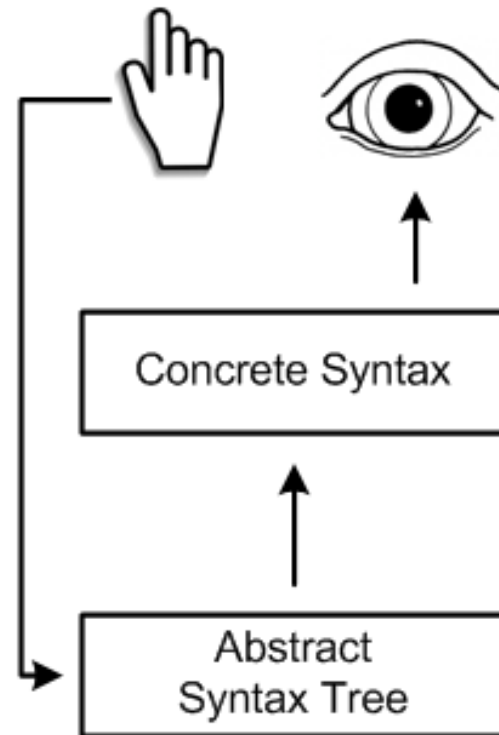


[Projectional Editing]

Parsing



Projectional Editing



[Projectional Editing]

Syntactic Flexibility

Regular Code/Text

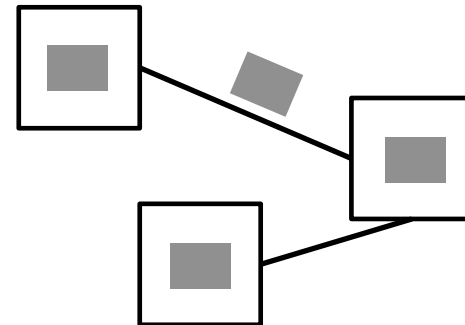


Mathematical



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)
```

Tables

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

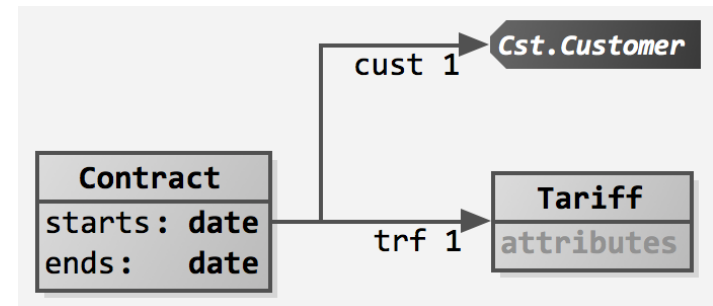
|           | spd > 0 | spd > 100 |
|-----------|---------|-----------|
| alt < 0   | 1       | 1         |
| alt == 0  | 10      | 20        |
| alt > 0   | 30      | 40        |
| alt > 100 | 50      | 60        |

 otherwise 0;  
} decide (function)
```

Mathematical

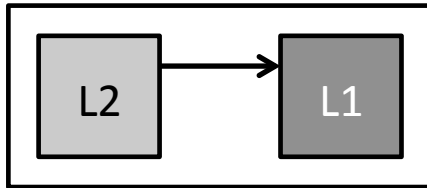
```
double midnight2(int32 a, int32 b, int32 c) {  
    
$$\text{return } \frac{-b + \sqrt{b^2 - \sum_{i=1}^4 a * c}}{2 * a};$$
  
} midnight2 (function)
```

Graphical



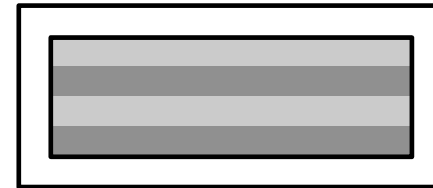
[Projectional Editing]

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.

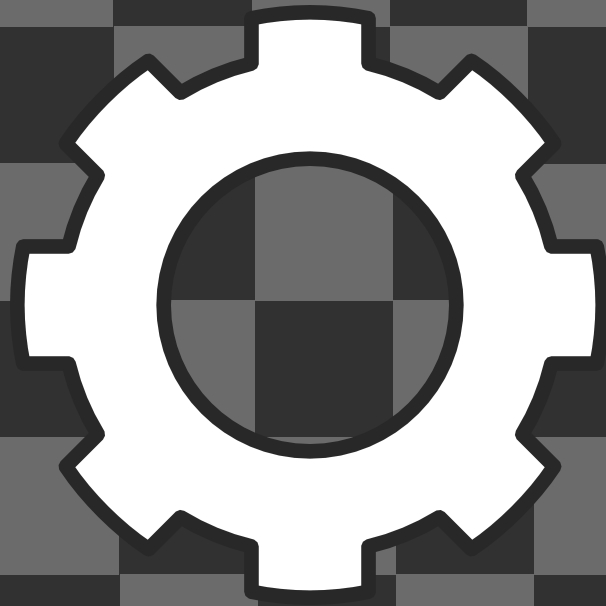
MPS is great, but alternatives exist.



**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

$$\text{PREMIO} = \left[\begin{array}{l} \text{A1} > 10 \Rightarrow \text{EU0} \\ \text{<always>} \Rightarrow \text{FLAG} \end{array} \right]$$
$$\text{FLAG} = \left[\begin{array}{l} \text{CATEG1 equals } 60 \text{ or CATEG1 equals } 63 \text{ or CATEG1 equals } 64 \Rightarrow 160 \\ \text{PREMIO equals } 0 \Rightarrow 162 \\ \text{CATEG1} > 0 \text{ or substr(inga[4], 1, 1) equals "v" \Rightarrow 163} \\ \text{<always>} \Rightarrow \text{PREMIO} + \text{FLAG} \end{array} \right]$$
$$\text{PREMIO} = \left[\text{<always>} \Rightarrow \text{round}(\text{PREMIO} * (1 + \text{factacer}), 0) \right]$$

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		

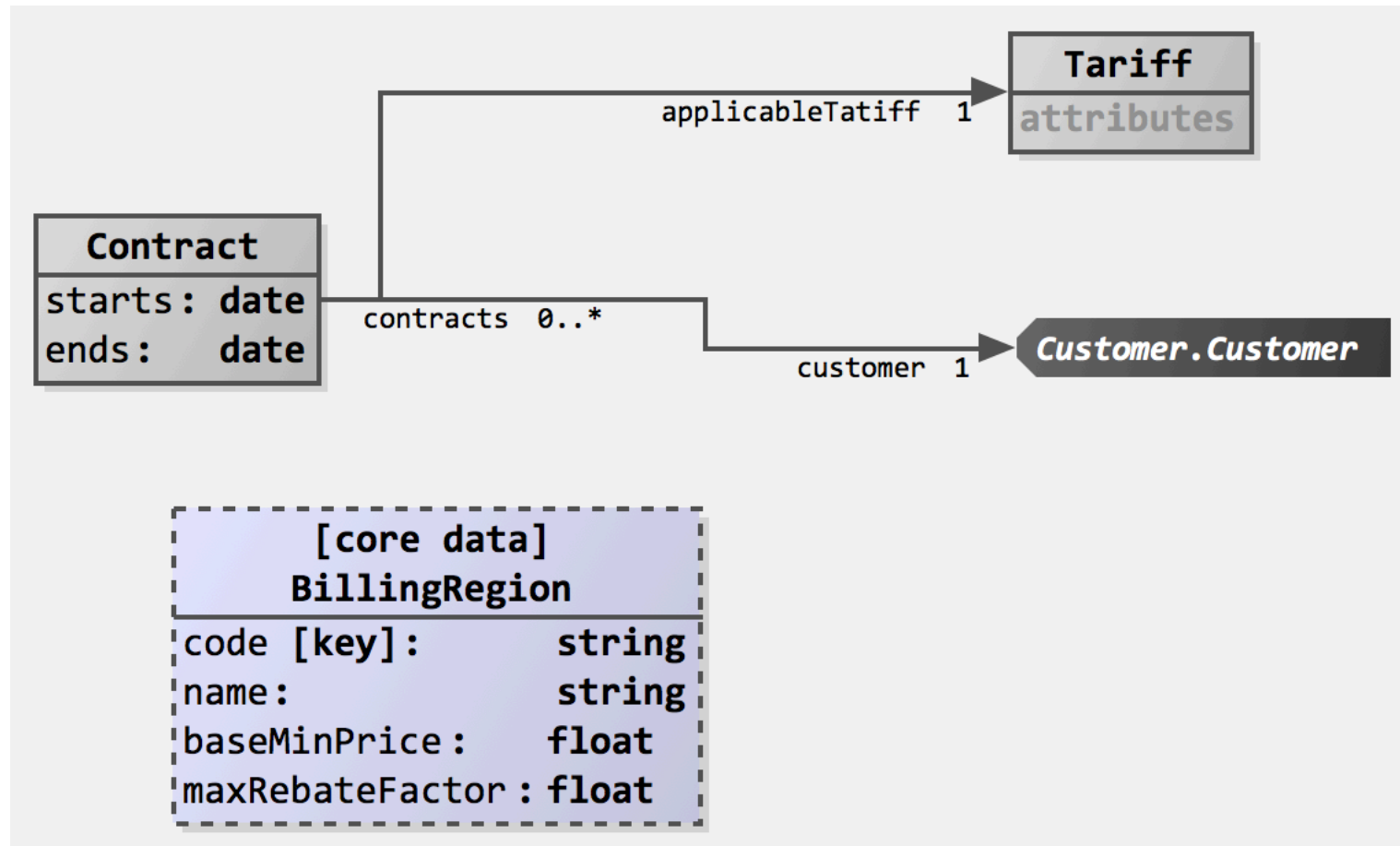
entity Contract

starts: date		customer:	Customer	1	□	<-->	contracts	0..*
ends: date		applicableTariff:	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:  $\Sigma$  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 otherwise 1
    cust.isRebated 0.5 0.6 0.8
    !cust.isRebated 0.8 0.9 1.0
```


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 isRecentlyActive := !entity.calls.first.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 :=
$$\sum_{i = 0}^{\text{callsLastMonth.size}} \text{callsLastMonth.at}(i).\text{price.value}$$

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

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 :=  $\sum_{i=0}^{\text{amount of calls} - 1} ((\text{all calls.at}(i).\text{price.value})) * \text{discountFactor}$ 
  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 everageCallPrice :=  $\frac{\text{current price}}{\text{amount of calls}}$ 
  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
```

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:**  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 initial 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 =  $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$  * 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 = aMatrixT ;
```

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

Explorability of the Language

The screenshot shows a code editor window titled "HelloMath" with a tab icon. The code is written in C and defines two functions: `sumUpIntArray` and `averageIntArray`. The `sumUpIntArray` function takes an array `arr` and its size `size` and returns the sum of its elements. The `averageIntArray` function takes the same inputs and returns the average of the elements. The code is as follows:

```
★ [C Extensions]
① HelloMath
model mbeddr.tutorial.main.math constraints imports nothing

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)
```

On the right side, there is a sidebar titled "Context Actions" with a search bar. Below the search bar, there is a section titled "Math Expressions" containing several icons for mathematical operations: `abs`, `frac`, `log`, `pow`, `product`, `sqrt`, and `sum`. The sidebar also has a vertical label "Context Sidebar" and a "Context Actions" button at the bottom.

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
PointForATrackpoint(1100, 165) == 210
PointForATrackpoint(2100, 140) == 100
PointForATrackpoint(2100, 200) == 300

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

result =

200

10

10|BASEPOINTS * 1

false

1100|alt > 2000

true

1100|alt > 1000

otherwise 0

30

15

10

20

	false	true	
<div><div><div>165 speed > 180</div><div>true</div><div>165 speed > 130</div></div></div>	<div><div>30</div></div>	<div><div>15</div><div>20</div></div>	

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

4



**Lessons
Learned**

How to make People precise?





Precision

!=

Formulas, Rules
Data Structures
Tables
Values

Performance
Scalability
Robustness
Deployment

Programming



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**

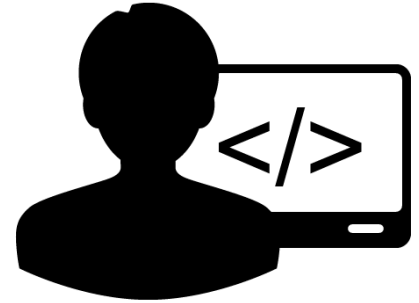


Technical People:
„It's not my job!“.

(and it really isn't)

Business L vs. Programming L





Structure

+

-

Notation

Mixed

Text

Guidance

+

-

Layout

Predefined

Custom

Views

*

1

IDE/Tool

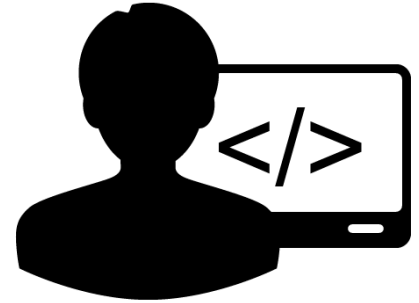
Clean

Powerful

Learn/Effective

L

E



Structure

Notation

Guidance

Layout

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

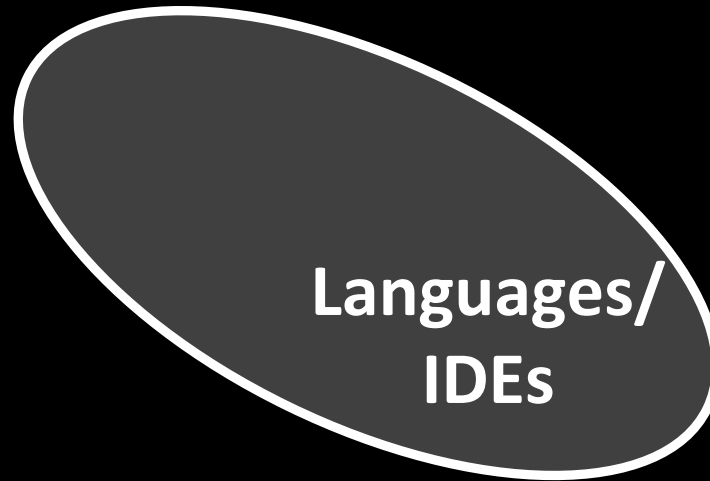
Version Control

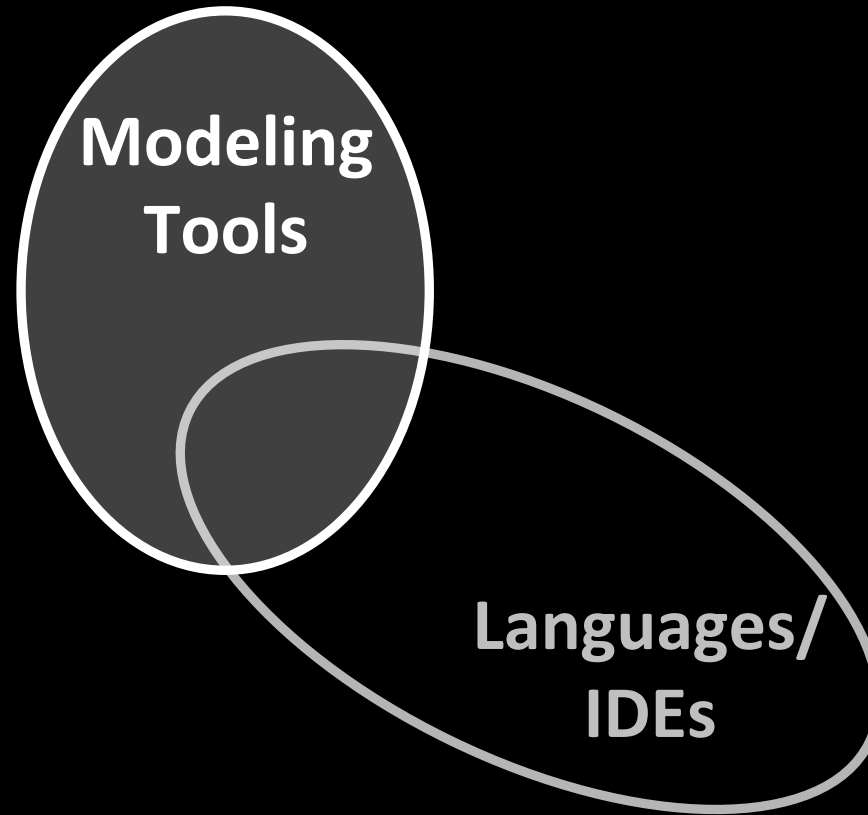
Refactoring

Debugging

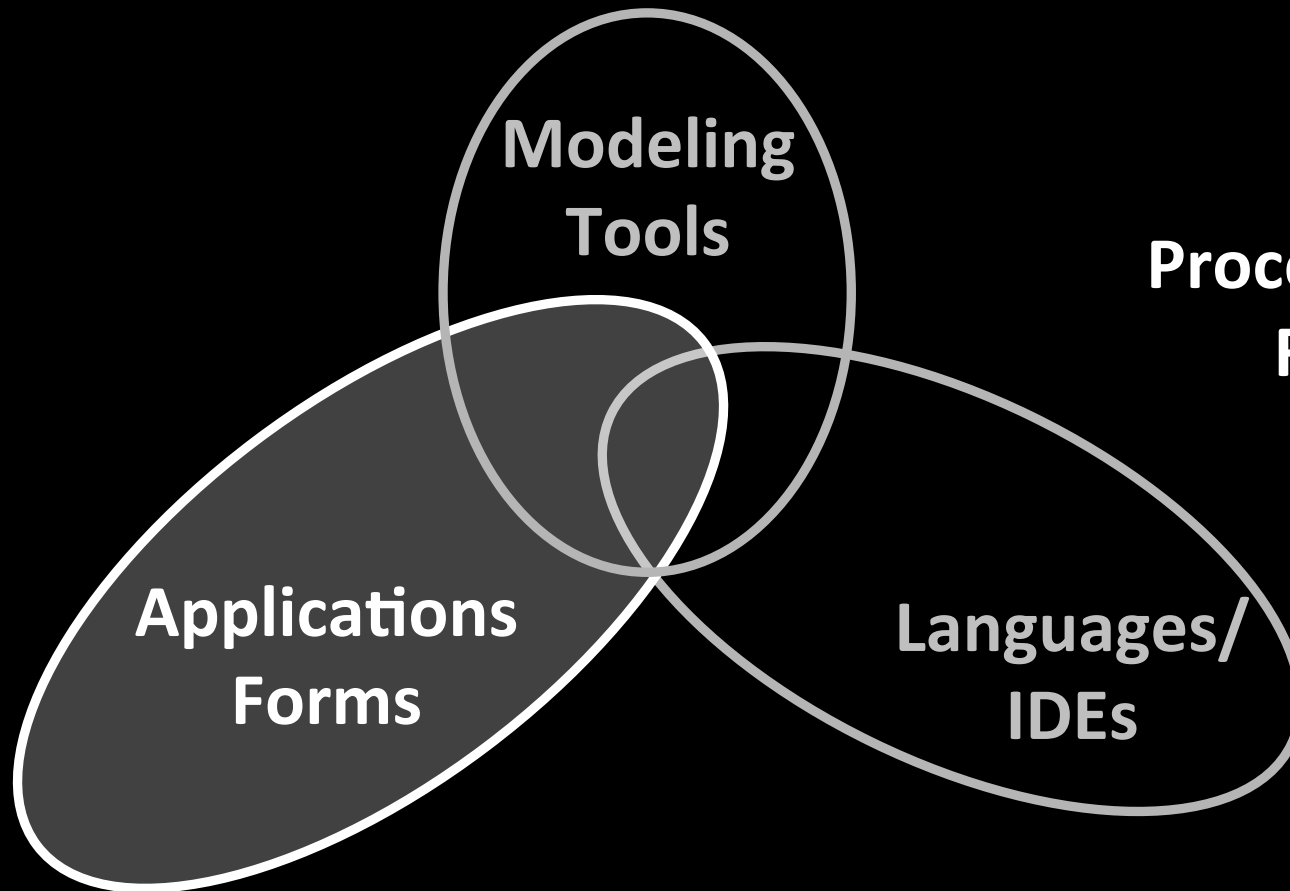
Scalability

Code Reviews

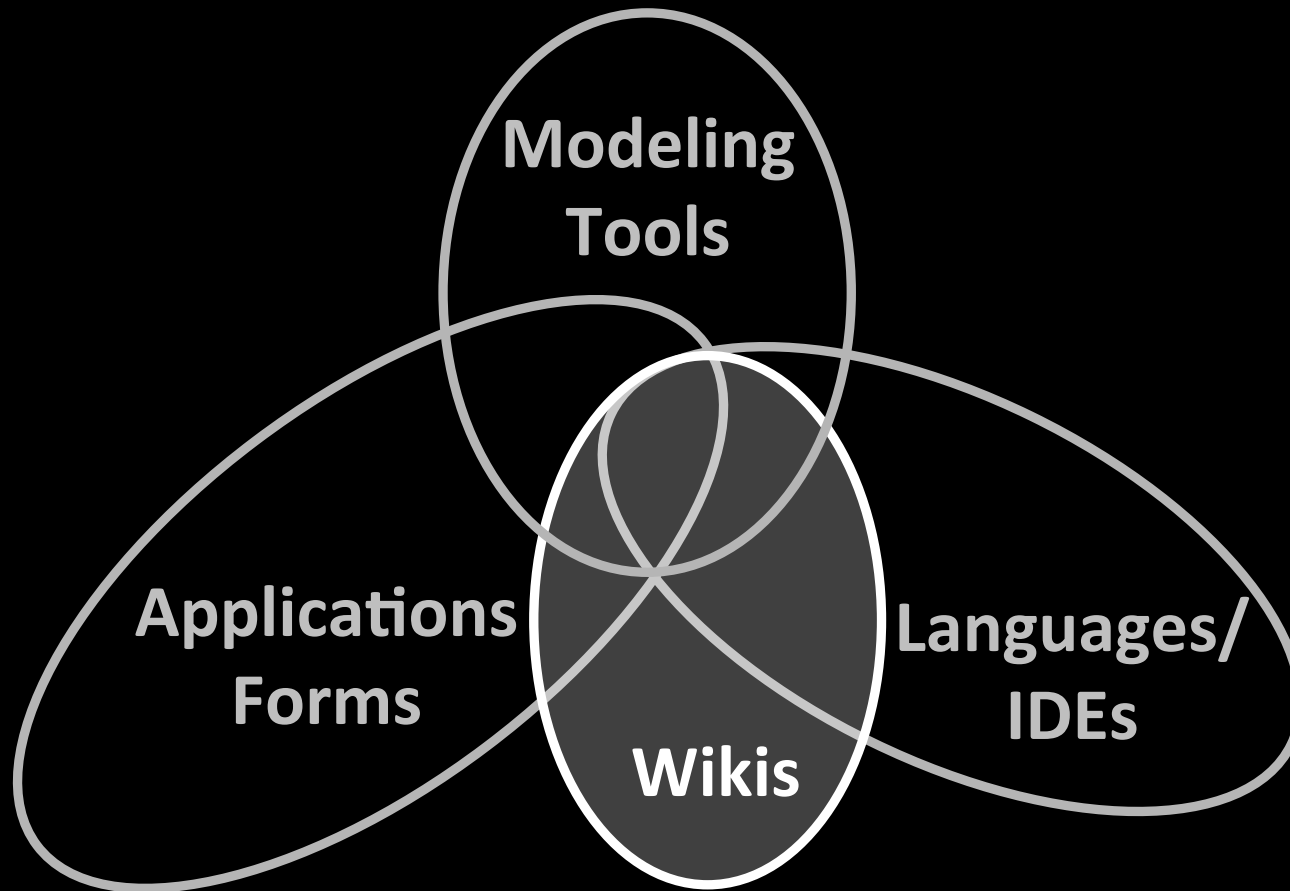




Abstraction Levels
Multiple Abstractions
Multiple Notations

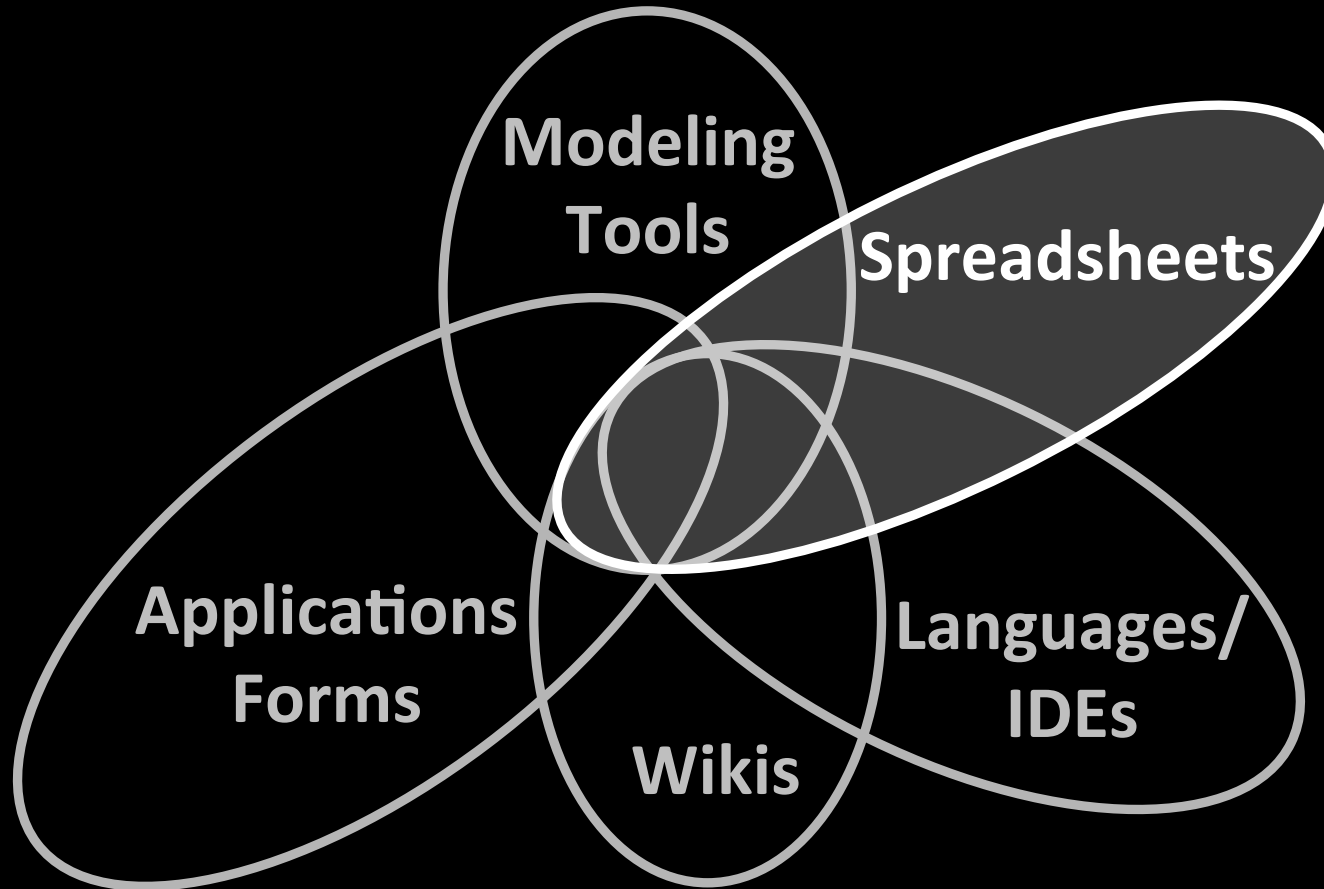


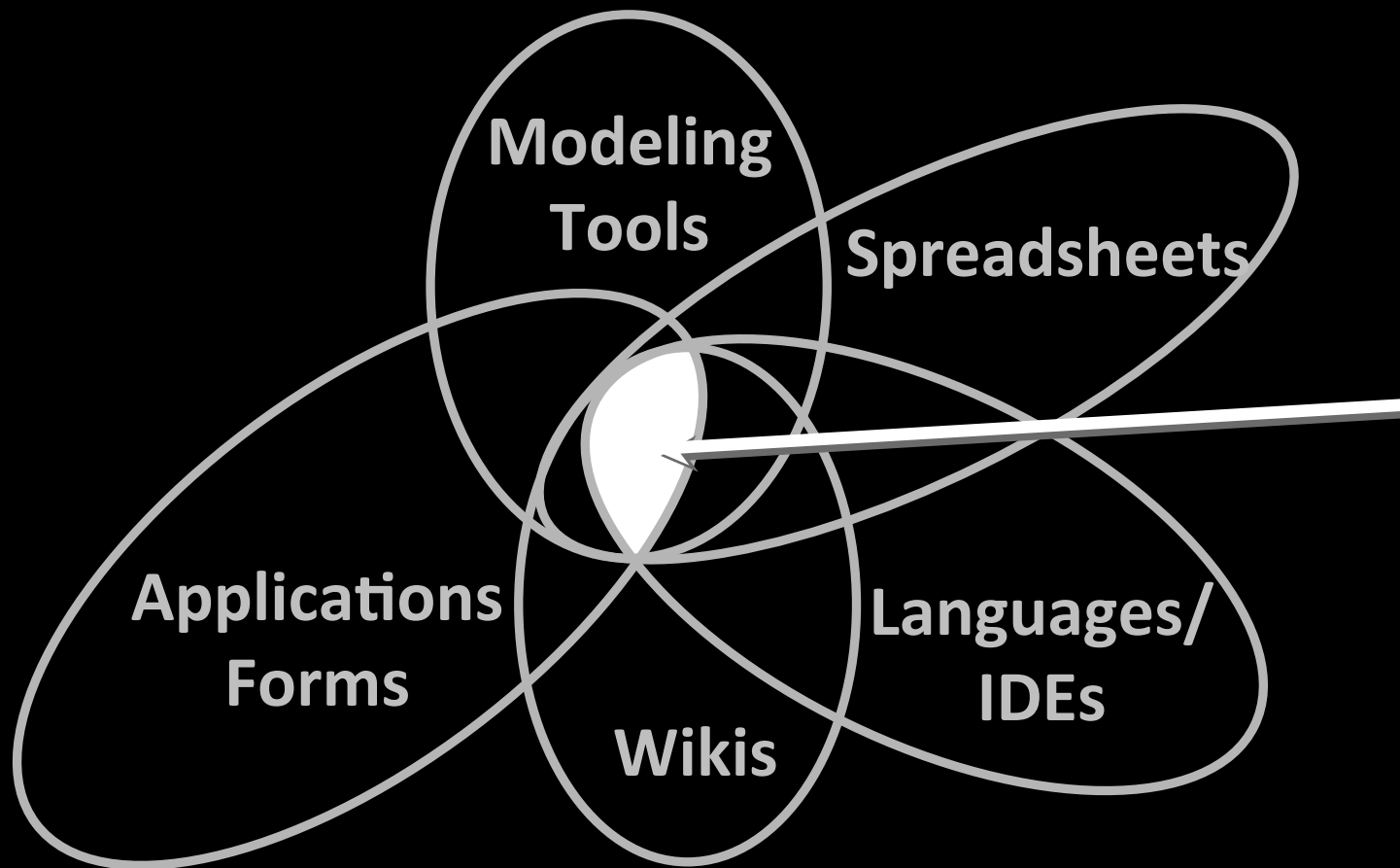
Process-Orientation
Rigid Structures
Visualizations
Tree Views
Guidance
Buttons



**Prose Integration
Cross-Links
„Plugins“**

Live Execution
„Visible Computation“
Document-Oriented





**L
O
B
A**

Why Version Control



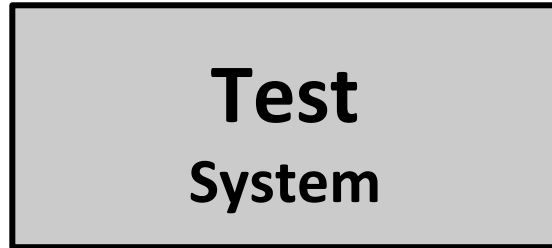
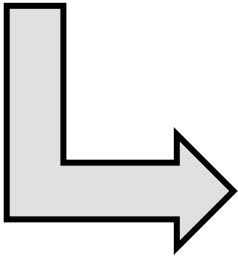
Why Version Control

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

Use Staging



Change



For Business People

Real-Like Data

May Have Bugs

Live for Customers

Real Data

Mission Critical

Integration Tests
Simulations
Reviews

How do you achieve Consistency



LOCAL

CONSISTENCY.

CONSISTENCY.

Strict Language

Cross-References

Modularization and Reuse

**Automatic Derivation based on rules
(transformation, generation)**



GLOBAL



**Common Respository
Version Control System
Periodic, Global Checks/Reports**

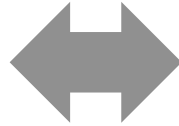
Influences on the Language



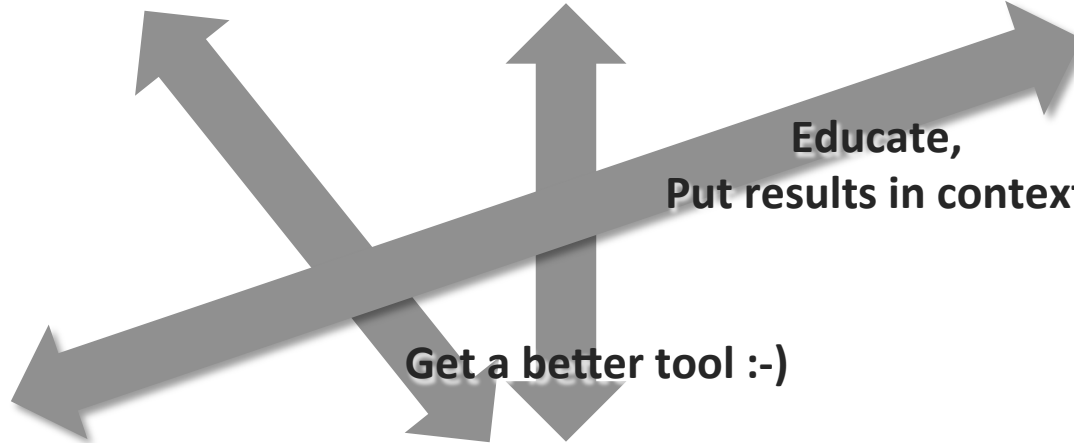
**Domain
Structure**

**Non
Functionals**
Permissions,
IP, Sharing

**User
Skills**



Sep. of Concerns
Different Views



Refactor towards
Structure

**Model
Purpose**
Analyze, Generate

**Tool
Capabilities**
Notations,
Editing, Scale

**Software
Engineering
Practices**

The Language is not Enough



GREAT

Debuggers

Animate Execution
Simulators

Testing

Write Tests
Run them
Report Back

Refactorings

Aligned with Processes

Analyses

Relevant
Good Errors

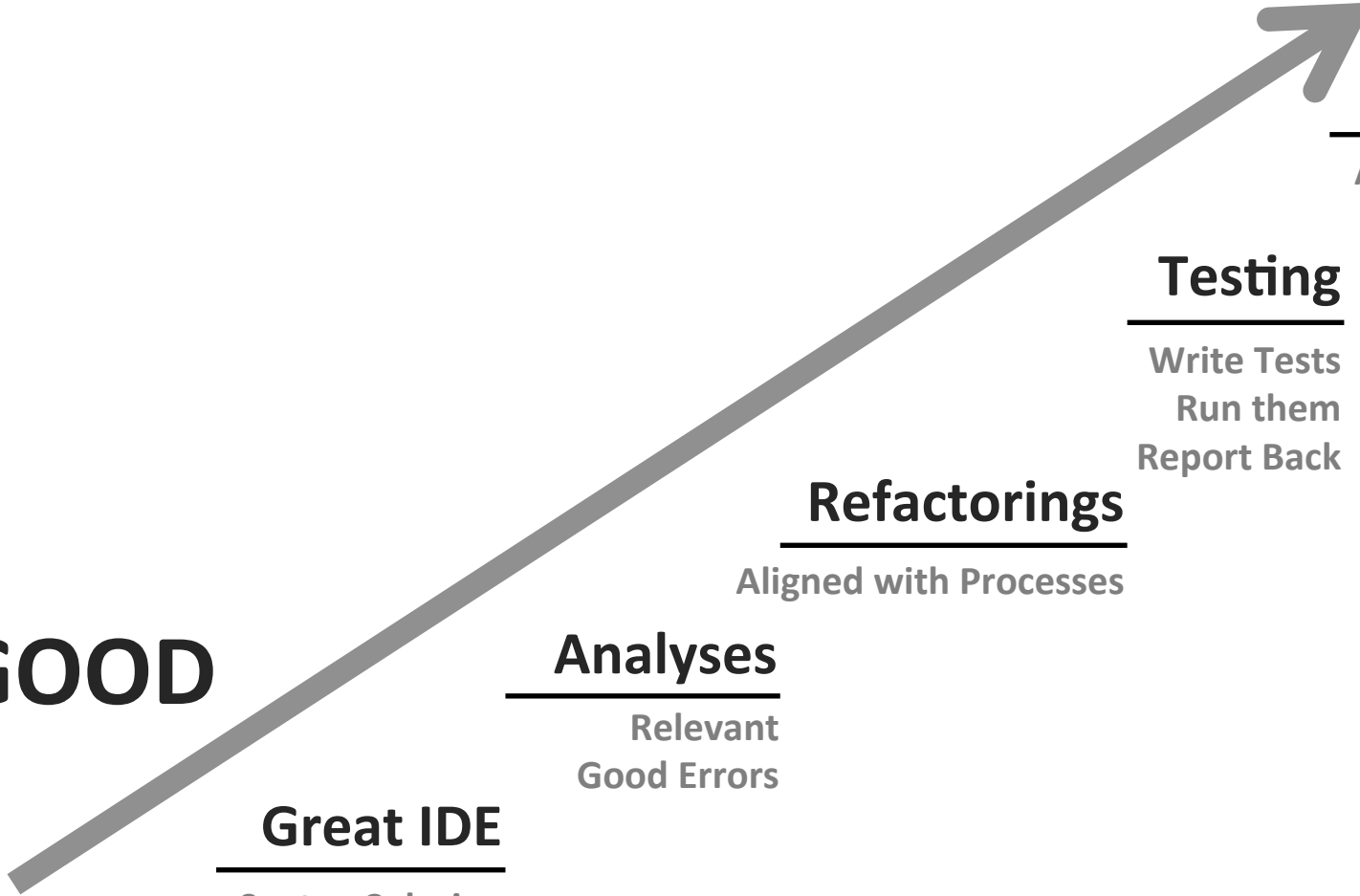
Great IDE

Syntax Coloring
Code Completion
Goto Definition

GOOD

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



Migration 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

The screenshot shows a code editor with a tab labeled 'HelloMath'. The code defines a function `sumArr` that takes an `int32[]` array and an `int32` size, and returns the sum of the array elements. The return statement is highlighted, showing a lightbulb icon and a summation symbol \sum with the index `i` ranging from 0 to `size`. A context menu is open on the right, displaying various mathematical symbols and functions such as `abs`, `frac`, `log`, `pow`, `product`, `sqrt`, and `sum`.

```
int32 sumArr(int32[] arr, int32 size) {  
    return  $\sum_{i=0}^{size} arr[i]$ ;  
}
```

Context Actions

$ x $	$\frac{a}{b}$	\sum	\log
abs	frac	i	log
x^y	\prod	\sqrt{x}	\sum
pow	product	sqrt	sum

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?

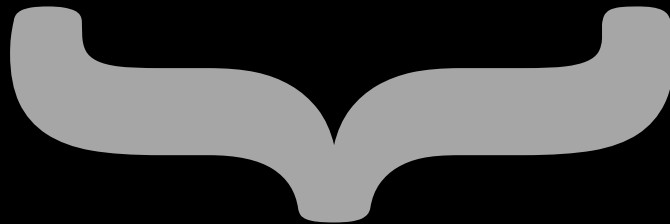
Tool should makes the right thing easy.

Error Messages

Process-Guidance in the tool

Checklists to finish manual processes

Tool must fit the process!



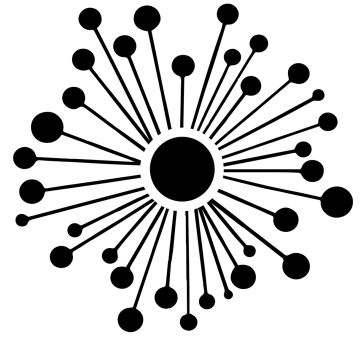
Tool should makes the right thing easy.



Does this scale?



Does the approach scale?



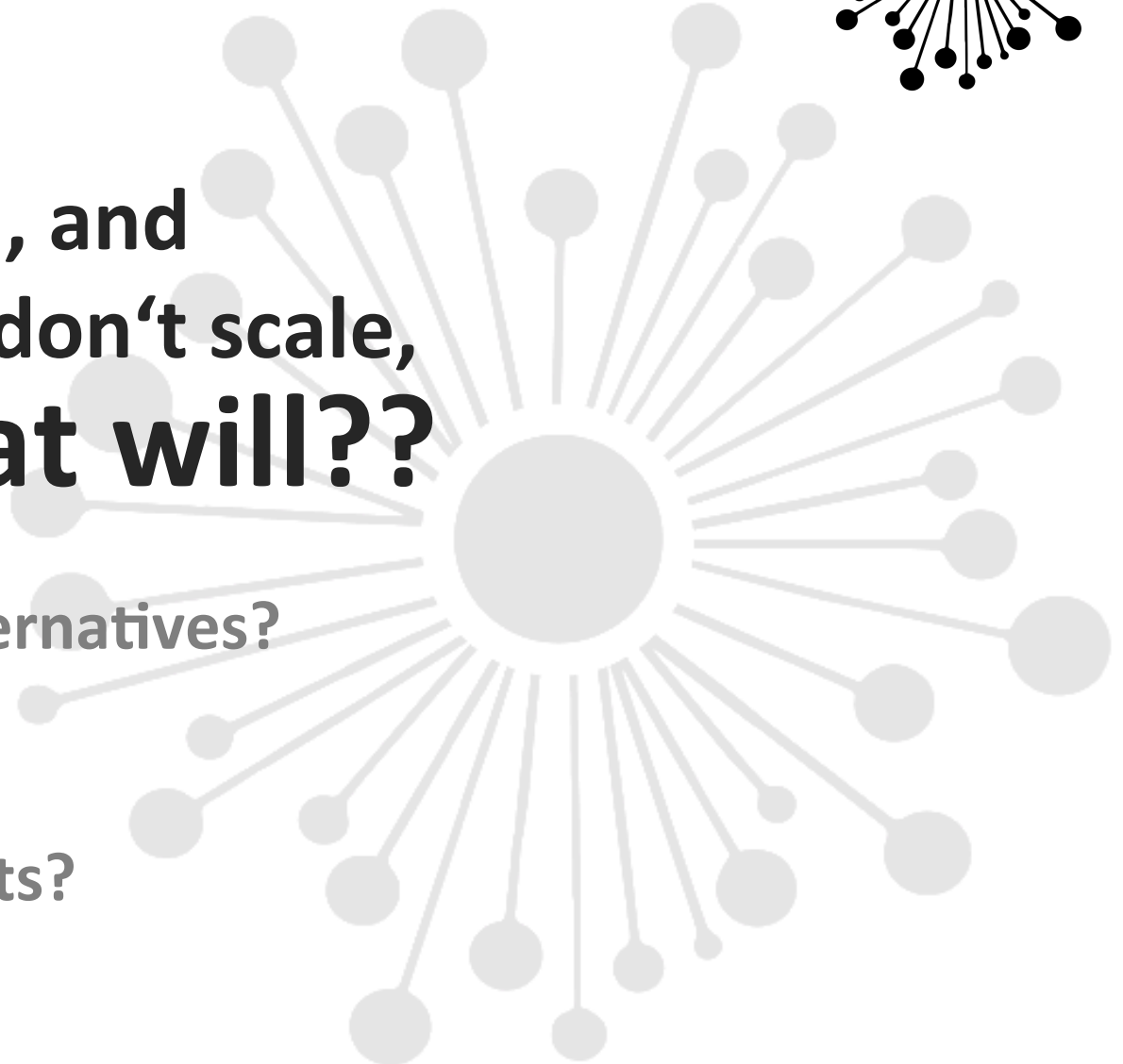
If **structure**,
formalization, and
tool support don't scale,
then what will??

What are the alternatives?

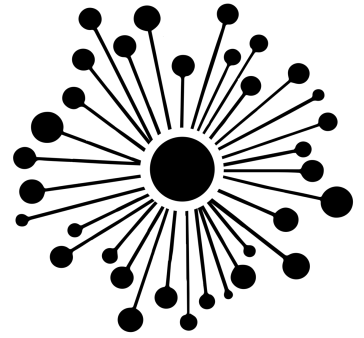
Excel?

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.

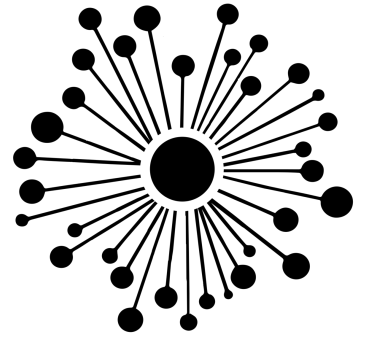
In terms of team size?

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.



THREAT.

This is a threat!

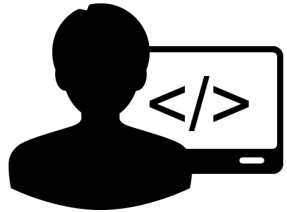




Precision and Formality
Different Processes
Higher Efficiency



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

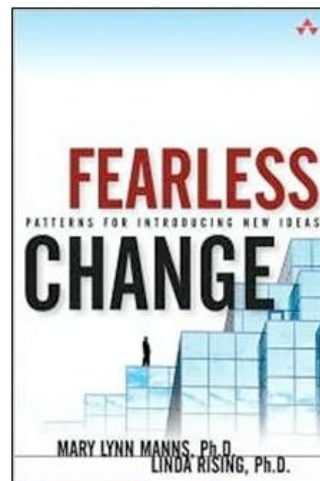


Automation
Focus on Engineering
Empower Business Ppl

- > Job Loss
- > Role Change
- > Less Importance

THREAT

**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.

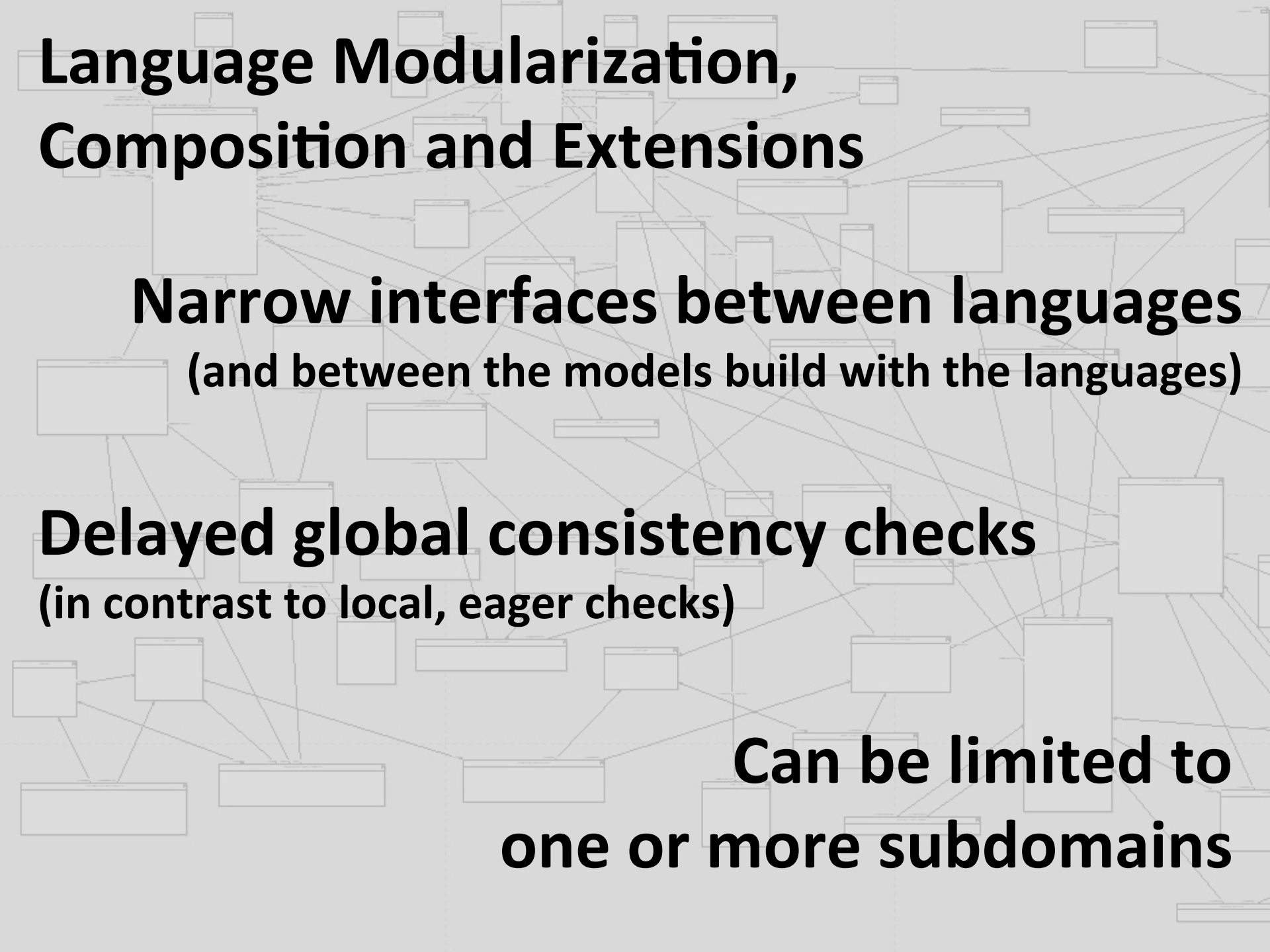
How is this not an EDM?





FAILURE

**Requires coordination with
the whole enterprise
– never works.**

The background of the slide features a complex, light gray network diagram. It consists of numerous rectangular boxes of varying sizes, some of which are stylized to look like computer windows or documents. These boxes are interconnected by a dense web of thin, gray lines, creating a sense of a large-scale system or a distributed network. The overall aesthetic is technical and modern.

Language Modularization, Composition and Extensions

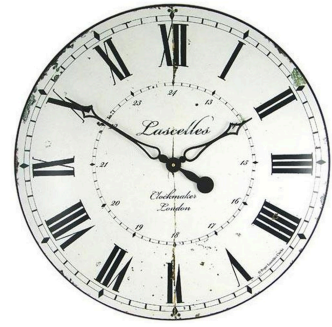
Narrow interfaces between languages
(and between the models build with the languages)

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***

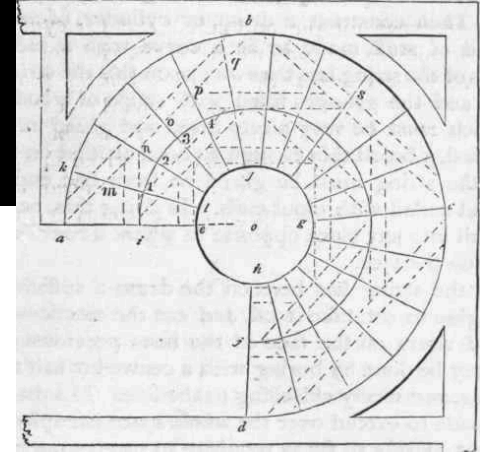


Create a dedicated team/organization whose goal it is to be successful with the approach.

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

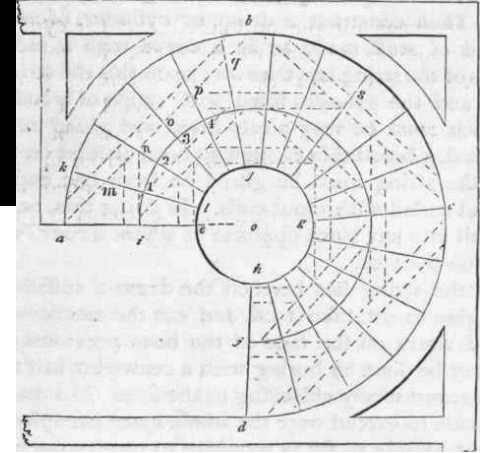
Prose

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.

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.

5



Summary

**Expressivity for Core
Domain Knowledge**

Build Language for Domain!

User-Friendly Notation

Great Tool/IDE

You've seen the demos.

Testing

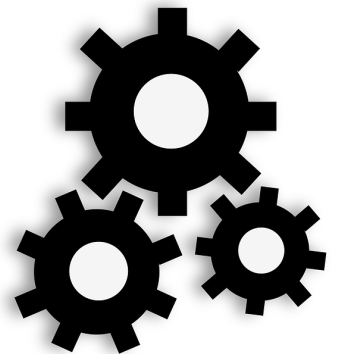
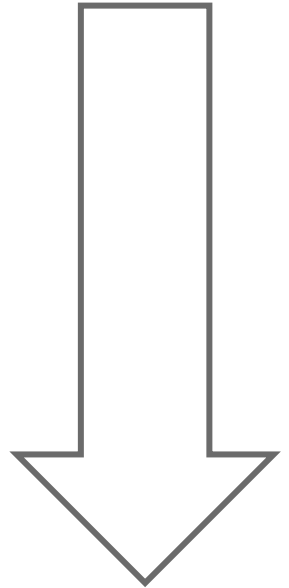
An integrated DSL for testing.

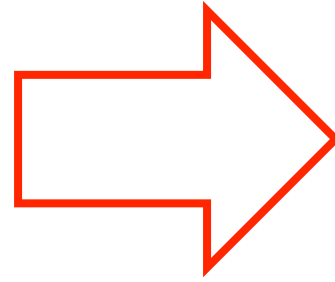
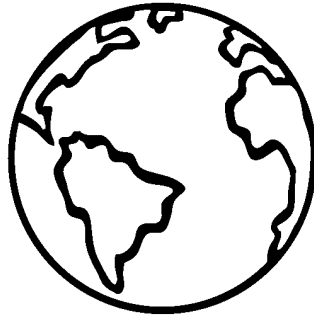
Meaningful Analyses

Types, Consistency, Checking

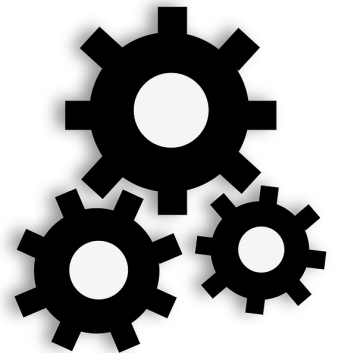
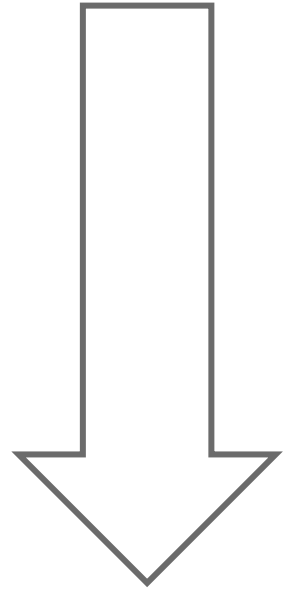
Synthesis of Software

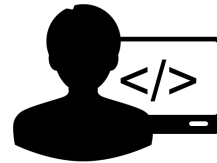
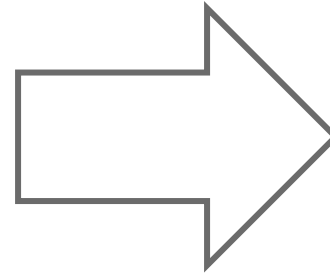
Code Generation.



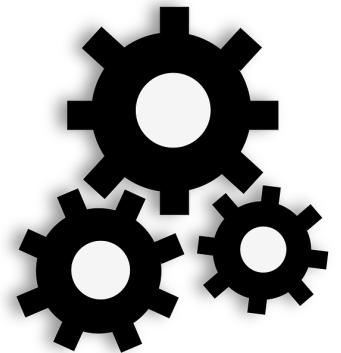
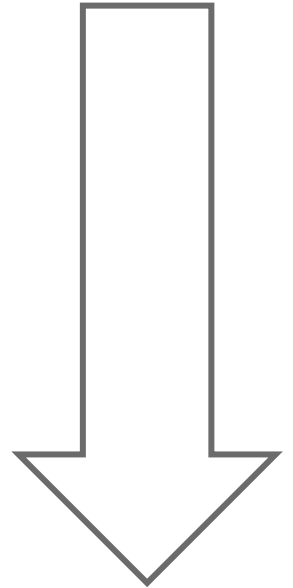


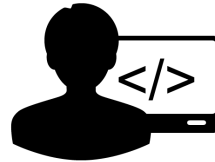
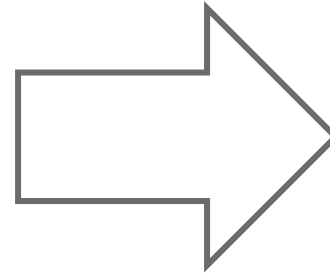
**Fundamentally
still manual, no AI.
But much better tooling.**



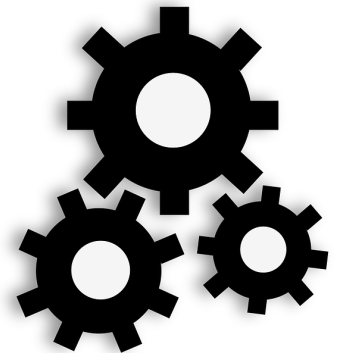
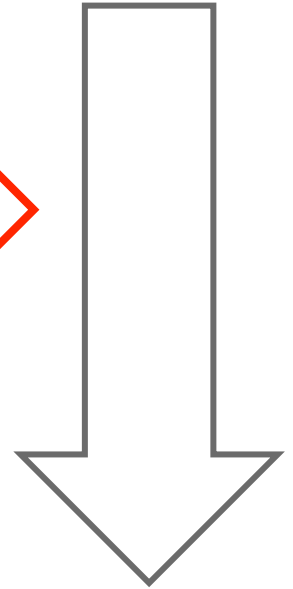


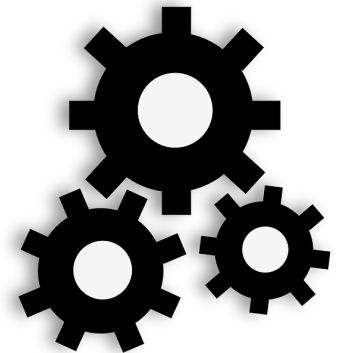
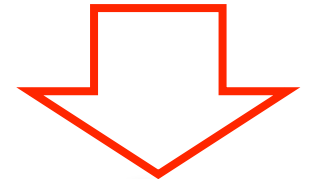
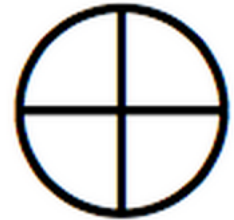
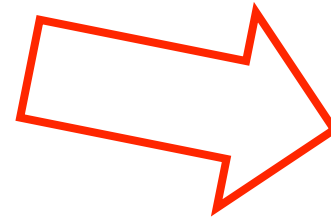
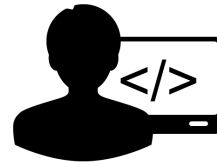
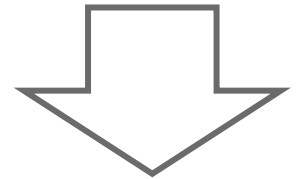
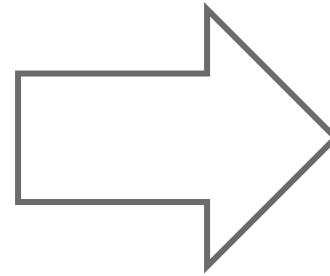
???





**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

