Language-Oriented Business Applications

Helping End Users become Programmers

Markus Völter

voelter@acm.org
www.voelter.de
@markusvoelter
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
It’s what makes a business tick.
Distinguishes the business.

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

Contributed *not* by developers

... but is a typically implemented in *software*
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.
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 and modeling
There's no difference!
LWBs make Languages Easier

Blur the distinction between
programming and modeling.

Several different LWBs exist.

http://languageworkbenches.net
A Language Workbench – a tool for defining, composing and using ecosystems of languages.
Open Source
Apache 2.0
http://jetbrains.com/mps
V 3.2 is current
V 3.3 to be released Q4 2015
Comprehensive Support for many aspects of Language Definition.

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

Parsing

Projectional Editing

Concrete Syntax

Abstract Syntax Tree

Concrete Syntax

Abstract Syntax Tree
Projectional Editing

Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical
[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

```c
// 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];
    }
} aSummingFunction (function)
```

Mathematical

```latex
\begin{align*}
    \text{double \text{midnight2}(\text{int32 } a, \text{int32 } b, \text{int32 } c)} & = \frac{-b + \sqrt{b^2 - 4 \sum_{i=1}^{a} c}}{2a} \\
\end{align*}
```

```c
} \text{midnight2} \text{ (function)}
```

Tables

```c
define(int8 spd, int8 alt) {
    return
    | spd > 0 | spd > 100 | otherwise 0 |
    | alt < 0 | alt == 0 | alt > 0 | alt > 100 |
    | 1 | 10 | 30 | 50 |
    | 1 | 20 | 40 | 60 |
} define (function)
```

Graphical

```
Cst.Customer

Contract
starts: date
ends: date

Tariff
attributes

cust 1

trf 1
```
[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!
## Rigid Structures

<table>
<thead>
<tr>
<th>Rule Set Type</th>
<th>DemoRuleSetType</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business objects</strong></td>
<td></td>
</tr>
<tr>
<td>person : Person</td>
<td></td>
</tr>
<tr>
<td><strong>Variables:</strong></td>
<td><strong>Parent</strong></td>
</tr>
<tr>
<td>PRMI : int</td>
<td>&lt;no parent&gt;</td>
</tr>
<tr>
<td>FR : int</td>
<td></td>
</tr>
<tr>
<td>NN : int</td>
<td></td>
</tr>
<tr>
<td>TT : int</td>
<td>Libraries</td>
</tr>
<tr>
<td>J : int</td>
<td>Standard</td>
</tr>
<tr>
<td>A3 : int</td>
<td>Extra</td>
</tr>
<tr>
<td>G3 : int</td>
<td></td>
</tr>
<tr>
<td>ANUI : int</td>
<td></td>
</tr>
<tr>
<td>X : int</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule Set Type</th>
<th>DemoRuleSetType</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business objects</strong></td>
<td></td>
</tr>
<tr>
<td>&lt;no business objects&gt;</td>
<td></td>
</tr>
</tbody>
</table>

| **Variables:** | **Parent**     |
| <no variables> | <no parent>    |

<table>
<thead>
<tr>
<th>Libraries</th>
<th>&lt;no libraries&gt;</th>
</tr>
</thead>
</table>


Calculation Rules

**Rule Set** DemoRuleSet2 is of type DemoRuleSetType

- **EU0** : int  
  - save: false, print: false
- **CATEG** : string  
  - save: false, print: false
- **CATEG1** : double  
  - save: true, print: true

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

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

proxy for Customer.Customer

core data entity BillingRegion

code [key]: string
name: string
baseMinPrice: float
maxRebateFactor: float

entity Contract

starts: date
ends: date
customer: Customer 1
applicableTatiff: Tariff

entity Tariff

attributes: references:
# Tables for Reference Data

## Core Data Default Regions for entity BillingRegion

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

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

<table>
<thead>
<tr>
<th>value cust := entity.customer value pricingFactor :=</th>
<th>isLocal isLongDistance isRoaming otherwise 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>cust.isRebated</td>
<td>0.5</td>
</tr>
<tr>
<td>!cust.isRebated</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Business Rules, Math, Tooltips

**Calculations** CustomerBasic for Customer

**imports:** `TimeUnits`, `BusinessRequirements`

```plaintext
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.last.startTime.isOlderThan(30 day)
flag isRebated := magic of type boolean[T]
```

[A couple of statistics about the last month's activity]
```plaintext
value callsLastMonth := entity.calls.where(!it.startTime.isOlderThan(30 day))
flag activeThisMonth := !callsLastMonth.isEmpty
value devicesUsedLastMonth := callsLastMonth.select(it.sourceDevice).distinct
callsLastMonth.size
```
```plaintext
value totalPriceLastMonth := \[ \sum_{i = 0} \] callsLastMonth.at(i).price.value
```
```plaintext
value averageCallPriceLastMonth := \( \frac{\text{totalPriceLastMonth}}{\text{callsLastMonth.size}} \)
```

[Some random examples.]
```plaintext
value example := all[Call].first.customer.calls.first.startTime
```
Tests executed in the Editor

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

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

value 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>

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 Target
amountThisMonth :=
  c.isRebated | 40 |
  otherwise    | 50 |

conditional assign overrides BaseContract.amountThisMonth as of 20/8/2014 Target
amountThisMonth :=
  c.isRebated | 40 |
  otherwise    | 60 |
```
BDD-style Tests for Business Rules

```plaintext
rule checkStuff

given anything

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

Assessment: UnusedCode
query: unused code
sorted: x must be ok: x hide ok ones: 
last updated: Sep 18, 2014 (3 days ago) by markusvoelter

BaseContract
- x 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

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

3. The factor of points
   PointsFactor /functional: tags

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

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

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

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

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

int32 averageIntArray(int32[] arr, int32 size) {
    \[ \sum_{i = 0}^{\text{size}} \text{arr}[i] \]
    return \[ \frac{\sum_{i = 0}^{\text{size}} \text{arr}[i]}{\text{size}} \];
}
```
Explorability of the Language

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

int32 averageIntArray(int32[] arr, int32 size) {
    return \( \frac{\sum_{i=0}^{size} \text{arr}[i]}{size} \);
} averageIntArray (function)
```
Live Tests for Business Rules

Points you get for each trackpoint

InFlightPoints /functional: tags


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

calculation PointForATrackpoint: This rule computes the points awarded for a Trackpoint. It does so by taking into account the @alt and the @speed passed as arguments.

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

result = (BASEPOINTS * 1) *

<table>
<thead>
<tr>
<th>speed</th>
<th>alt &gt; 2000</th>
<th>alt &gt; 1000</th>
<th>otherwise 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>10</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

tests:
PointForATrackpoint(500, 46) == 0
PointForATrackpoint(1000, 165) == 210
PointForATrackpoint(2100, 140) == 100
PointForATrackpoint(2100, 200) == 300
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 = \[
\frac{10}{10 \times \text{BASEPOINTS}} \times 200 - 20
\]

<table>
<thead>
<tr>
<th>false</th>
<th>true</th>
<th>otherwise 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\text{alt} &gt; 2000]</td>
<td>[\text{alt} &gt; 1000]</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>165</td>
<td>speed &gt; 180</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>165</td>
<td>speed &gt; 130</td>
<td></td>
</tr>
</tbody>
</table>

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

! =

Formalization

Greek Letters
Analyses
Proofs
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
Guidance: +
Layout: Predefined *
Views: Clean
IDE/Tool: Clean
Learn/Effective: Powerful
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
Live Execution
„Visible Computation“
Document-Oriented

- Modeling Tools
- Spreadsheets
- Languages/IDEs
- Wikis
- Forms
- Applications
Why Version Control
Why Version Control

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

Test System

For Business People
Real-Like Data
May Have Bugs

Production System
Live for Customers
Real Data
Mission Critical

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

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

Non Functionals
Permissions, IP, Sharing

User Skills

Sep. of Concerns
Different Views

Educate,
Put results in context

Model Purpose
Analyze, Generate

Tool Capabilities
Notations, Editing, Scale

Software Engineering Practices

Refactor towards Structure

Get a better tool :-)

Educate,
The Language is not Enough
Requirements on the tool
Be a great LWB obviously.

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

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

Nobody wants to work with a sluggish tool
Scalability

Non-trivial languages and significant model sizes
Migration Support

Migrate existing models as the languages evolve.
Friendliness

Don’t overwhelm end users with too much “crufit”
Explorability

Ensure the language can be explored

```c
int32 sumArr(int32[] arr, int32 size) {
    size
    return \sum_{i=0} arr[i];
} sumUpIntArray (function)
```
A tool is not enough
Methodology = Process + Tool (+ Metrics)
Precision/Consistency refers to Artifacts and not to a rigid Process.
Discipline: do the right thing.

Define what is „right“
Force People?
Tool should makes the right thing easy.
Tool should 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
Different Processes
Higher Efficiency

- New Skills
- Role Change
- Job Loss

Automation
Focus on Engineering
Empower Business Ppl

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

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

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

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

Such an approach is completely pointless!!
Rule Language

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

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

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

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

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

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

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

What is the alternative? 

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

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

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

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

Immature Organization
-> never heard of unit test, CI and VCS? Bad sign!
How do you introduce this?
You never know how strong you are... until being strong is the only choice you have.
1. Agree this is the right way

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

2. Prototype it

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

3. Go for the real thing

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

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.
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
Thank you!