
from Programming
to Modeling
Markus Voelter
Independent/itemis
voelter@acm.org
itemis


from Programming
to Modeling
...and back again
Markus Voelter
Independent/itemis
voelter@acm.org
itemis


„Houston,
we have a
Problem!“


Programming
Languages
30
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C


Programming
Languages
are not
expressive
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30

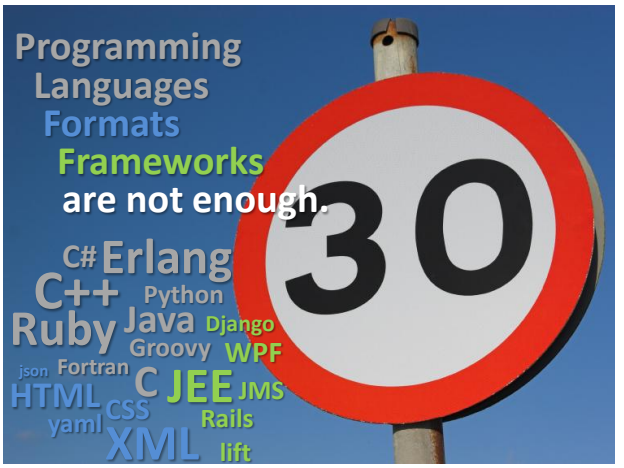

Programming
Languages
are not
high-level
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


Programming
Languages
are not
abstract
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


Programming
Languages
are not
domain-specific
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


Programming
Languages
are not
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


Programming
Languages
Formats
C# Erlang
C++ Python
Ruby Java
Groovy
json Fortran C
HTML CSS
yaml XML
30



So?

Modeling

Modeling

... Higher Level

... Domain Specific
Concepts & Notations

... Code Generation
Interpretation

Modeling

... Higher Level

... Domain Specific
Concepts & Notations

... Code Generation
Interpretation

Solves the Problem!

But:


Different Worlds
Programming Tools
!=
Modeling Tools

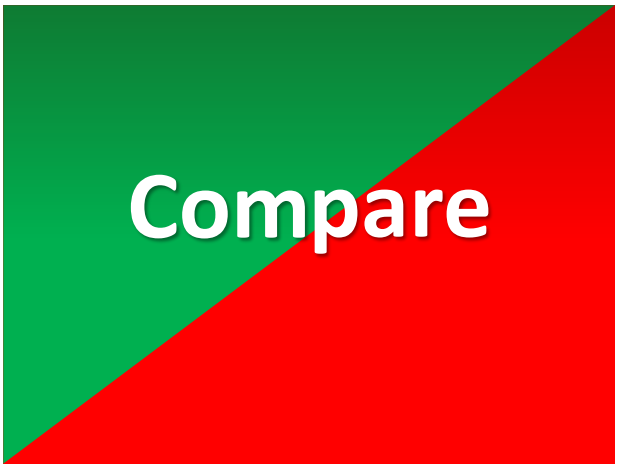

Different Worlds
Modeling Tool
!=
Modeling Tool


Different Worlds
Mix Models and Programs


Different Worlds
Mix Models and Programs
AST Navigation & Query


Different Worlds
Mix Models and Programs
AST Navigation & Query
Integration of 3GL code


Different Worlds
Mix Models and Programs
AST Navigation & Query
Integration of 3GL code
Code Constraints


Compare


Compare


Domain Specific
Notations
and
Abstractions
Flexible!
Limited!
Frameworks
Libraries
(Fluent) APIs


Graphical
Textual
Forms
Tables
Flexible!
Limited!
Textual
Trees


Customize
Generator
or
Interpreter
Flexible!
Limited!
Reflection
Meta Programs
Open Compilers

Define custom
Query
or Navigate
or Transform

Flexible!

Limited!

AST APIs
Static Analysis
Regex

Custom
Validation
or Error Checks

Flexible!

Limited!

IDE plugins
Static Analysis
Open Compilers

Different
Representations
and Projections

Flexible!

Limited!

Text is Text
Code Folding
Tree Views
Visualizations

Mixing
and
Composing
Languages

Flexible?

Limited!

Python-to-C-like
Internal DSLs
Embed-As-String
Specific: LINQ

Scalable
Usable
IDE Support

Brittle!

Mature!

Modeling Tools…!?

Debugging
Refactoring
Testing

Brittle!

Mature!

?

Brittle!

for some tools…

Versioning
Diff, Merge
Branching

Mature!

Gets some Jobs done.

some people doubt that…

Gets the Job Done!

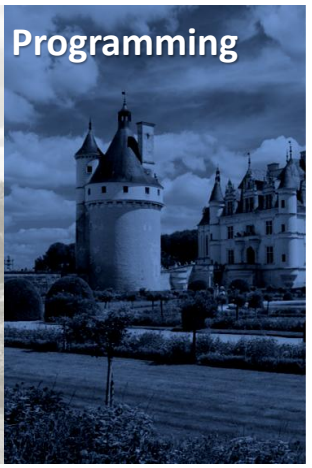
everybody agrees…

Why
the difference?

History?

Modeling

Programming



Modeling

Programming

… (Mostly) Textual Notations

… Concrete Syntax Storage

… (Fancy) ASCII Editors

… Read-Only Visualizations

Modeling

… (Mostly) Graphical Notations

… Abstract Syntax Storage

… Projecting Editors

… Different editable views for model

Programming

… (Mostly) Textual Notations

… Concrete Syntax Storage

… (Fancy) ASCII Editors

… Read-Only Visualizations

Why

the difference?

It is time for …

… a Different Perspective

Programming the way we do Modeling?

Modeling the way we do Programming?

Modeling == Programming

Programming == Modeling

Where
do we go
from here?

We don‘t want to
model,
we want to
program!

We don‘t want to
model,
we want to
program!

… at different levels of abstaction
… from different viewpoints
… integrated!

We don‘t want to
model,
we want to
program!

… with different degrees of domain-specificity
… with suitable notations
… with suitable expressiveness

We don't want to
model,
we want to
program!

And always:
precise and tool processable

1. Enabling Technologies

1. Enabling Technologies
2. Available Tooling

1. Enabling Technologies
2. Available Tooling
3. A vision for programming

1 Enabling Technologies

1 Enabling Technologies

Advanced
Parser Generators

Modeling as Programmig

... (Mostly) Textual Notations
... Concrete Syntax Storage
... (Fancy) ASCII Editors
... Read-Only Visualizations

Custom
Syntax

Graphical
Textual
Symbolic++

IDE
Support

Teamwork
Debugging
Custom Editors

Complete
Symbolic
Integration

Goto Def
Find Refs
Refactoring

Xtext-like Tools

provide editor support

Custom Editors
Teamwork
Goto Def
Find Refs
Refactoring

Limited to Unicode

how to handle
non-character symbols

Graphics != Text

two worlds…

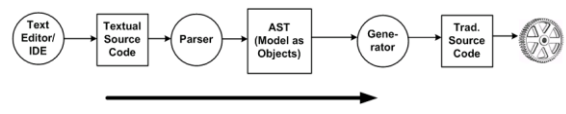
separate editors
… per syntax/viewpoint
… models can still be ref integrated

2 Enabling Technologies

Projectional Editing

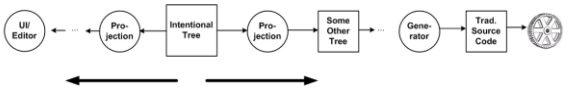
Parser-based

text
… to tree
… to text



Projectional

tree
... to text-lookalike (editor)
... to other trees ... [*]
... to text



Programming as Modeling

... (Mostly) Graphical Notations
... Abstract Syntax Storage
... Projecting Editors
... Different editable views for model

Programming as Modeling

... ~~(Mostly) Graphical~~ Any kind of Notations
... Abstract Syntax Storage
... Projecting Editors
... Different editable views for model

Language Composition

There's no parsing.
Unique Language Element Identity.
Unlimted language composition.

Flexible Notations

Textual
like ASCII
Graphical
box & line
Semi-Graphical
mathematical

} treated the same
can be mixed

Automatic IDE Extension

tool support is inherent
for languages build with
projectional tools

language definition
implies
IDE definition

Multiple
Notations

... for the same concepts

e.g. in different contexts
or for different tasks

Partial
Projections

... different views
... for different roles/people
... only a particular variant

Storage
!= Schema

... store arbitraty meta data
- change log
- conflicting information
- variability annotations

... independent of language schema!
... „aspects“, overlay

Live
Programs

think: spreadsheet

a change to one part of program can lead to (dependent) changes in other parts

Tree Editing

... is different from editing text

... try to make it feel like text

... takes some getting used to

but: for more flexible notations
a more general editing paradigm
is needed

Infrastructure
Integration

... storage is not text

... diff/merge must be in tool

... existing text tools don‘t work

Proprietary Tools

... no standards

... no interop

[redacted]

2 Available Tooling

2 Available Tooling

Eclipse Xtext

eclipse modeling PROJECT

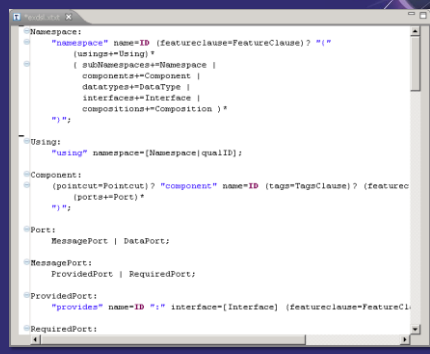
http://eclipse.org/modeling

xtext

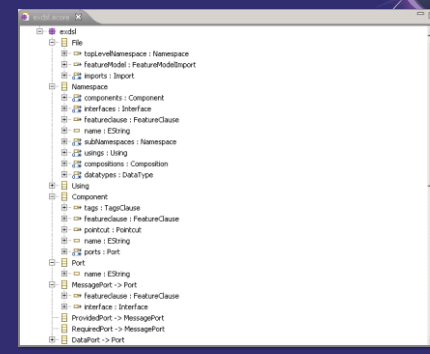
eclipse modeling PROJECT

http://eclipse.org/xtext

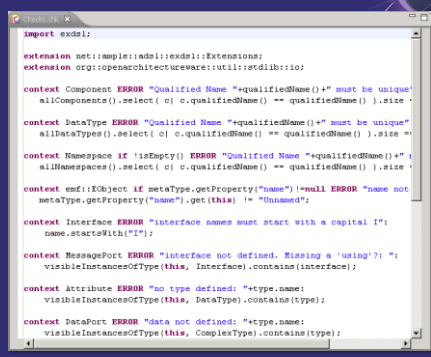
Xtext: Specify Grammar



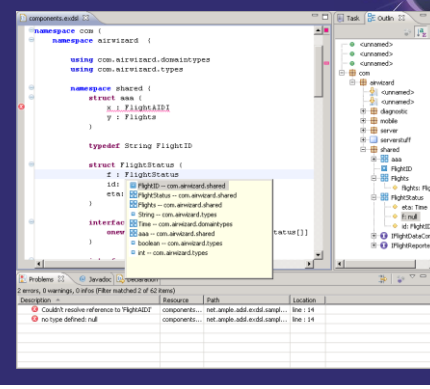
Xtext: Gen. Meta Model



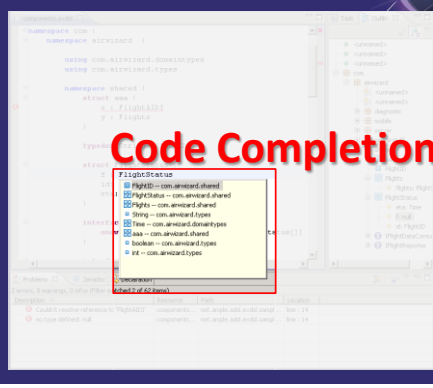
Xtext: Constraints



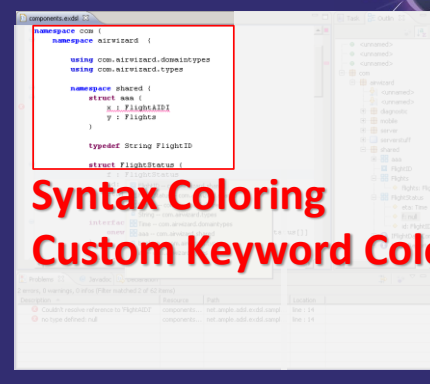
Xtext: Generated Editor



Xtext: Generated Editor



Xtext: Generated Editor



Xtext: Generated Editor

eclipse modeling PROJECT

Realtime Constraint Validation

Xtext: Generated Editor

eclipse modeling PROJECT

Customizable Outlines

Xtext: Generated Editor

eclipse modeling PROJECT

Code Folding

Xtext: Generated Editor

eclipse modeling PROJECT

Goto Definition
Find References
Cross-File References
Model as EMF

Xtext: Generated Editor

eclipse modeling PROJECT

DEMO I

Building DSLs with
Eclipse Xtext

2 Available Tooling

2 Available Tooling

Jetbrains'
Meta Programming System

also do...
IntelliJ IDEA
Resharper

released in
Q3 2009

currently
1.1 RC1

licensed under

Apache 2.0

Build new standalone DSLs

Build new standalone DSLs
Build DSLs that reuse parts of other languages

Build new standalone DSLs
Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage) Java++

extend base language

Build new standalone DSLs
Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage) Java++

extend base language
build DSLs that reuse parts of BaseLanguage

Language Extension Example

Language Extension Example

Old

```
ReadWriteLock l = …
l.readLock().lock();
try {
  //code
} finally {
  l.readLock().unlock();
}
```

Language Extension Example

Old

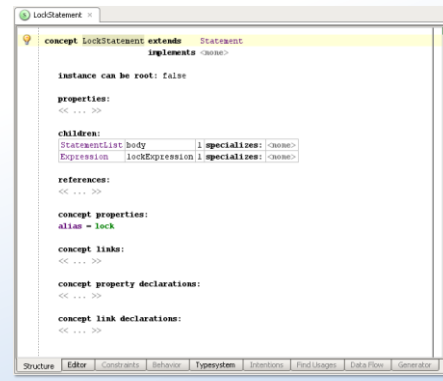
```
ReadWriteLock l = …
l.readLock().lock();
try {
  //code
} finally {
  l.readLock().unlock();
}
```

New

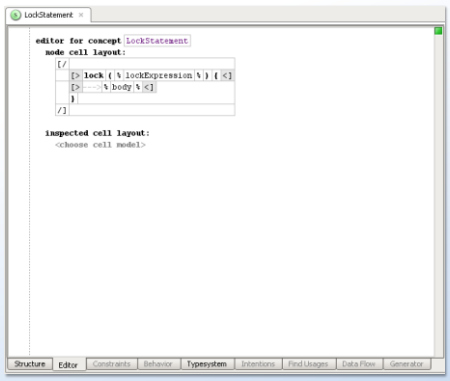
```
ReadWriteLock l = …
lock (l) {
  //code
}
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

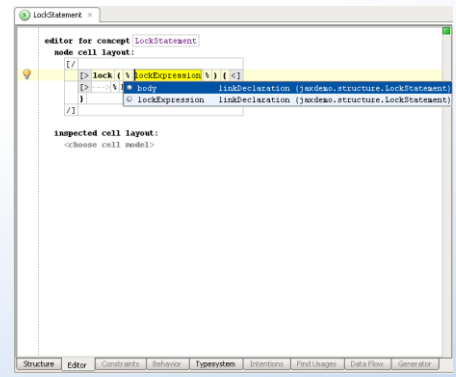
Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator

```
LockStatement | typeof_LockStatement
rule typeof_LockStatement (
  applicable for concept = LockStatement as lockStatement
  overrides false
  child type restrictions << ... >>

  do {
    typeof(lockStatement.lockExpression) :<<=: lock ;
  }
}
```

```
Lock          : interface (j.u.concurrent.locks@)
LockSupport   : class (j.u.concurrent.locks@)
```

```
Inspector
jetbrains.mps.baseLanguage.structure.ClassifierType        Open Concept Declaration
[quotedNode] ClassifierType <no name>[1242291552632] in jaxdemo.typesystem
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

```
main
conditional root rules:
  << ... >>

mapping rules:
  << ... >>

weaving rules:
  << ... >>

reduction rules:
  concept    LockStatement --> reduce_LockStatement
  inheritors false
  condition  <always>

abandon roots:
  << ... >>

pre-processing scripts:
  << ... >>
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

```
reduce_LockStatement
content node:
public class someclass extends <none> implements <none> {
  <static fields>>

  <static initializer>>
  <fields>>
  <properties>>
  <initializer>>
  public someclass() {
    <no statements>
  }

  public void somemethod() {
    Lock l = null;
    <TF try {                                  TF>
        $COPY_SRC$[1].lock();
        $LOOP$[$COPY_SRC$[null:]]
      } finally {
        $COPY_SRC$[1].unlock();
      }
  }

  <static methods>>

  <static inner classifiers>>
}
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

```
reduce_LockStatement
content node:
public class someclass extends <none> implements <none> {
  <static fields>>

  <static initializer>>
  <fields>>
  <properties>>
  <initializer>>
  public someclass() {
    <no statements>
  }

  public void somemethod() {
    Lock l = null;
    <TF try {                                  TF>
        $COPY_SRC$[1].lock();
        $LOOP$[$COPY_SRC$[null:]]
      } finally {
        $COPY_SRC$[1].unlock();
      }
  }

  <static methods>>

  <static inner class
}
```

```
Inspector
jetbrains.mps.lang.generator.structure.CopySrcNodeMacro        Open Concept Declaration
copy/reduce node macro

mapping label : <no label>
mapped node   : (node, genContext, operationContext)->node< > {
                  node.lockExpression;
                }
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

```
reduce_LockStatement
content node:
public class someclass extends <none> implements <none> {
  <static fields>>

  <static initializer>>
  <fields>>
  <properties>>
  <initializer>>
  public someclass() {
    <no statements>
  }

  public void somemethod() {
    Lock l = null;
    <TF try {                                  TF>
        $COPY_SRC$[1].lock();
        $LOOP$[$COPY_SRC$[null:]]
      } finally {
        $COPY_SRC$[1].unlock();
      }
  }

  <static methods>>

  <static inner class
}
```

```
Inspector
jetbrains.mps.lang.generator.structure.LoopMacro        Open Concept Declaration
loop-macro

mapping label : <no label>
mapped nodes  : (node, genContext, operationContext)->sequence<node< >> {
                  node.body.statement;
                }
```

Language Extension Example

Result behaves like a native base language construct

```
DemoClass
public class DemoClass extends <none> implements <none> {
  <static fields>>

  <static initializer>>
  private Lock lock;
  <properties>>
  <initializer>>
  public DemoClass() {
    lock ( this.getLock() ) {
      SharedResouce.instance().doSomething();
    }
  }

  private Lock getLock() {
    return this.lock;
  }

  <static methods>>

  <static inner classifiers>>
}
```

Language Extension Example

Result behaves like a native base language construct

```
DemoClass
public class DemoClass extends <none> implements <none> {
    <<static fields>>

    <<static initializer>>
    private Lock lock;
    <<properties>>
    <<initializer>>
    public DemoClass() {
      lock ( i ) {
        SharedResouce.instance().doSomething();
      }                    Typesystem error: int is not a subtype of Lock
    }

    private Lock getLock() {
      return this.lock;
    }

    <<static methods>>

    <<static inner classifiers>>
}
```

Language Extension Example

Translated to regular Java code based on the generator

```
DemoClass
public class DemoClass extends <none> implements <none> {
    <<static fields>>

    <<static initializer>>
    private Lock lock;
    <<properties>>
    <<initializer>>
    public DemoClass() {
      lock ( this.getLock() ) {
        SharedResouce.instance().doSome
      }
    }

    private Lock getLock() {
      return this.lock;
    }

    <<static methods>>

    <<static inner classifiers>>
}
```

```
package jaxdemo.sandbox.sandbox;

import java.util.concurrent.locks.Lock;

public class DemoClass {

  private Lock lock;

  public DemoClass() {
    try {
      this.getLock().lock();
      SharedResouce.instance().doSomething();
    } finally {
      this.getLock().unlock();
    }
  }

  private Lock getLock() {  return this.lock;  }

}
```

Example Languages

UI Language

```
component NewLanguageDialog

  IDEDialog
    left: 100 top: 100 width: 600 height: 200
    title = New Language

    content pane:
    Grid
      Row
        [Label
          text : "Language Namespace:"]
      Row
        [TextField ( name )
          text : ( this.languageNamespace )]
      Row
        [Label
          text : "Language Path:"]
      Row
        [PathField ( path )
          path : ( this.languagePath )]

    buttons
      [text OK
       default true
       handler this.onOk()]
      [text Cancel
       default false
       handler this.onCancel()]
```

Example Languages

HTML Templates

```
html template MessageHeader(Message message, boolean hideUser, boolean userLast)

is not refreshable: Only one root element allowed for refreshable template.

<< variables >>

rss links: << rss links >> << rss template uris>>

<< explicit js import >>

if (!(hideUser) && !(userLast)) {
  <span class: right-margin-span>
    [html include <visible>
      .def[]: UserLink(user: message.getUser(), bold: true)]
  </span>
}
<span class: right-margin-span>
  [html include <visible>
    .def[]: Date(date: message.created)]
</span>
if (null != message.updated) {
  <span class: right-margin-span>
    updated:
    [html include <visible>
      .def[]: Date(date: message.updated)]
  </span>
}
```

Example Languages

Persistent Classes

```
public persistent class Forum extends <none> implements <none> {

  features
    save changes history if: false
    save changes history callback: no callback
    version mismatch resolution: default
    invariant: no invariant

  << static fields >>

  public simple string name opts;
  public unordered child FThread[0..n] threads opts;
  public unordered bidirectional association User[0..n] subscribers onDelete(clear);
  public unordered bidirectional association User[0..n] watchers onDelete(clear);

  public Forum(string name, User creator) {
    this.name = name;
    this.watchers.add(creator);
  }


  << destructor >>
```

DEMO II

Building DSLs with
JetBrains MPS

3 A vision for Programming

Programming Languages are not **MODULAR** enough.

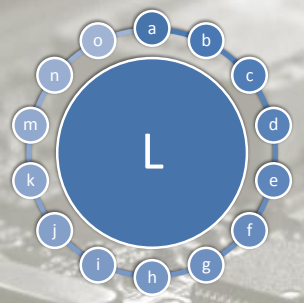
Programming Languages are not **COMPOSABLE** enough.

Programming Languages are not **CONFIGURABLE** enough.

Programming Languages are not **ADAPTABLE** enough.

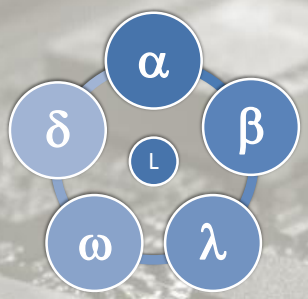
Programming Language Syntax is not **FLEXIBLE** enough.

Big Language?



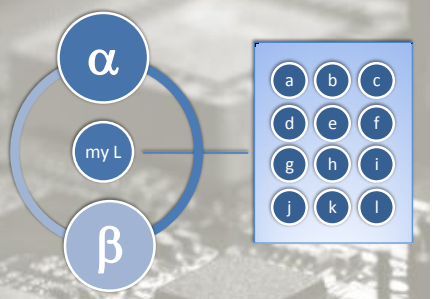
with many first class concepts!

Small Language?



with a few, orthogonal and poweful concepts

Modular Language



with many optional, composable concepts

Modular Language

Like frameworks
and libraries,

Modular Language

Like frameworks
and libraries,

but with syntax
and IDE support

Not a new idea…

Growing A Language
(Guy L Steele)

Language Workbench
(Martin Fowler)

Language Workbench
(Martin Fowler)

Freely
define
languages and
integrate
them

Language Workbench
(Martin Fowler)

use
persistent
abstract
representation
?

Language Workbench
(Martin Fowler)

language ::=
schema
+ **editors**
+ **generators**

Language Workbench
(Martin Fowler)

projectional
editing
?

Language Workbench
(Martin Fowler)

persist
incomplete
or
contradictory
information

Language Workbench
(Martin Fowler)

powerful
editing
testing
refactoring
debugging
groupware

language definition
implies
IDE definition

Language Workbench
(Martin Fowler)

support for
„classical“
programming
„classical“ and
modeling

Syntax
primarily
textual

Syntax
primarily
textual

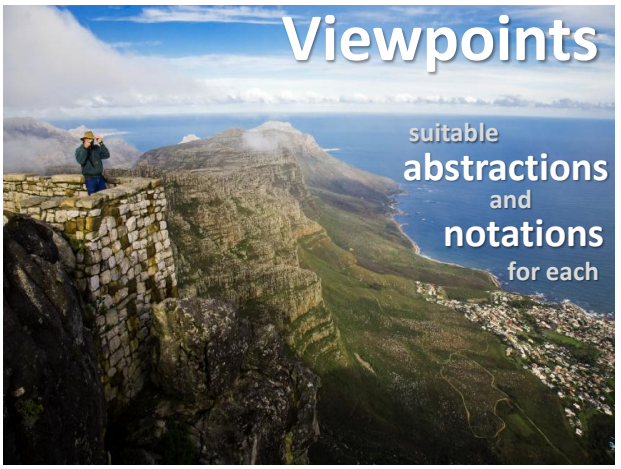
with more symbols
think: mathematics

Syntax
primarily
textual

sometimes
box&line style

Syntax
primarily
textual

sophisticated
visualizations



Viewpoints
Business

custom
purpose-built

create/include

Viewpoints
Business

Custom
Notations

real
business
expert integration

Viewpoints

Business

Custom
Notations

real
business
expert integration

But that's another talk...

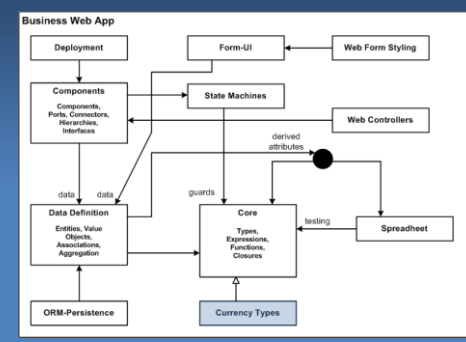
Viewpoints

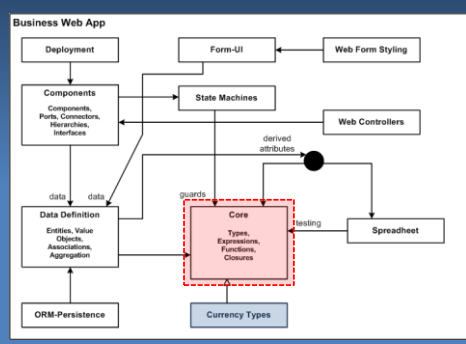
Technical

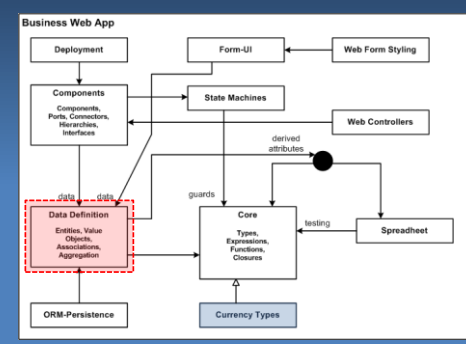
predefined
library

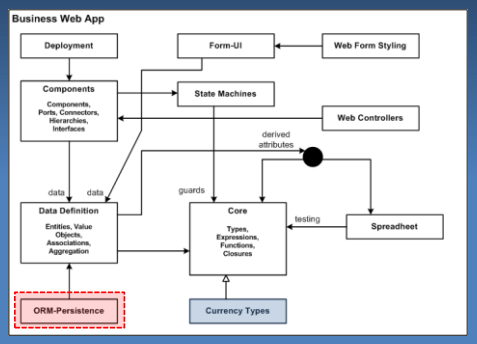
configure

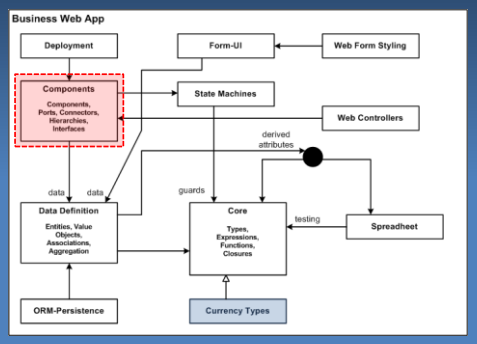
Example Languages

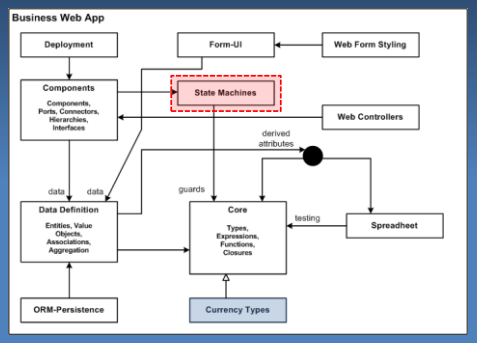


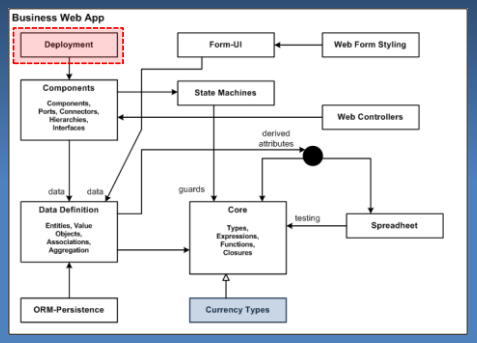


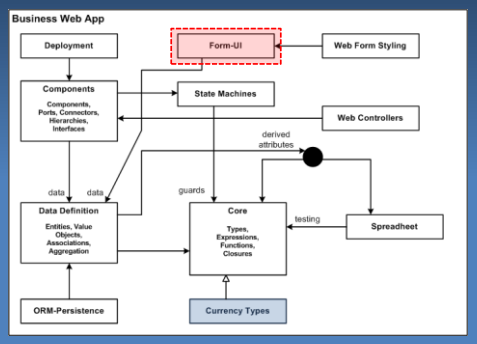


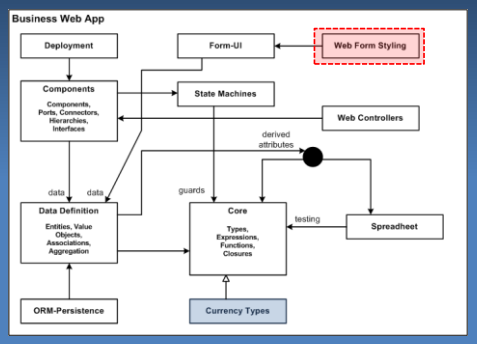

Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types

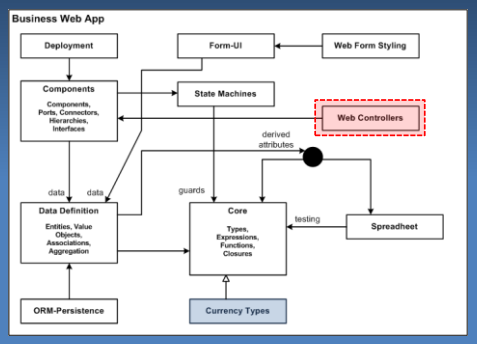

Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types


Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types


Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types


Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types


Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types



Business Web App

Deployment

Form-UI

Web Form Styling

Components

Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Web Controllers

derived attributes

data

data

guards

Data Definition

Entities, Value Objects, Associations, Aggregation

Core

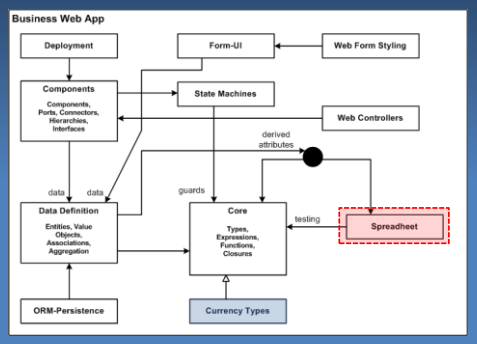
Types, Expressions, Functions, Closures

testing

Spreadheet

ORM-Persistence

Currency Types



Business Web App

Deployment

Form-UI

Web Form Styling

Components

Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Web Controllers

derived attributes

data

data

guards

Data Definition

Entities, Value Objects, Associations, Aggregation

Core

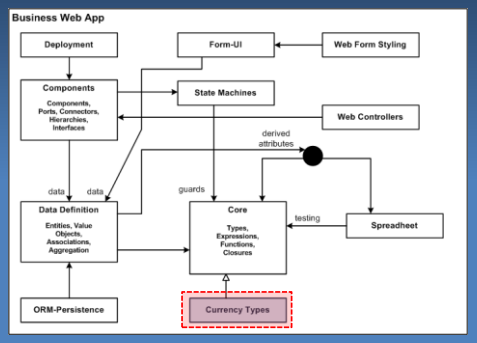
Types, Expressions, Functions, Closures

testing

Spreadheet

ORM-Persistence

Currency Types



Business Web App

Deployment

Form-UI

Web Form Styling

Components

Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Web Controllers

derived attributes

data

data

guards

Data Definition

Entities, Value Objects, Associations, Aggregation

Core

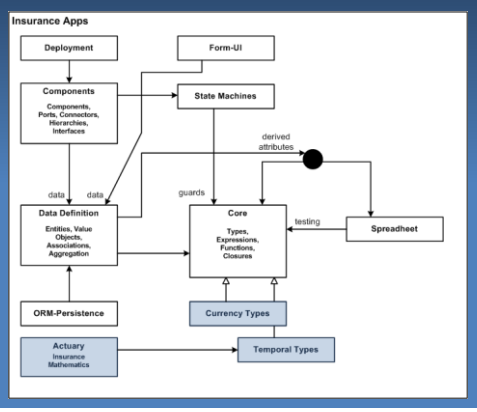
Types, Expressions, Functions, Closures

testing

Spreadheet

ORM-Persistence

Currency Types



Insurance Apps

Deployment

Form-UI

Components

Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

derived attributes

data

data

guards

Data Definition

Entities, Value Objects, Associations, Aggregation

Core

Types, Expressions, Functions, Closures

testing

Spreadheet

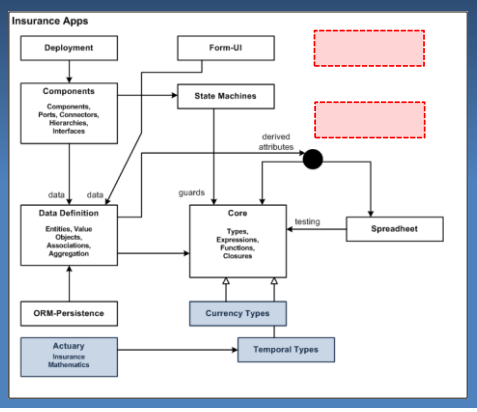
ORM-Persistence

Currency Types

Actuary

Insurance Mathematics

Temporal Types



Insurance Apps

Deployment

Form-UI

Components

Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

derived attributes

data

data

guards

Data Definition

Entities, Value Objects, Associations, Aggregation

Core

Types, Expressions, Functions, Closures

testing

Spreadheet

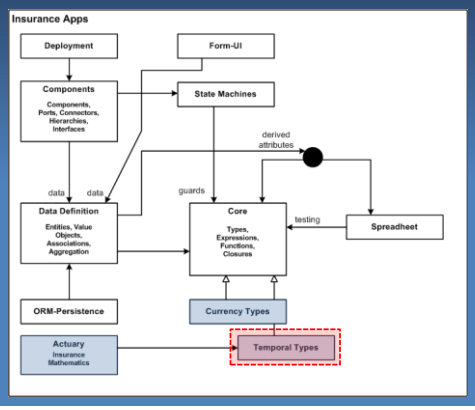
ORM-Persistence

Currency Types

Actuary

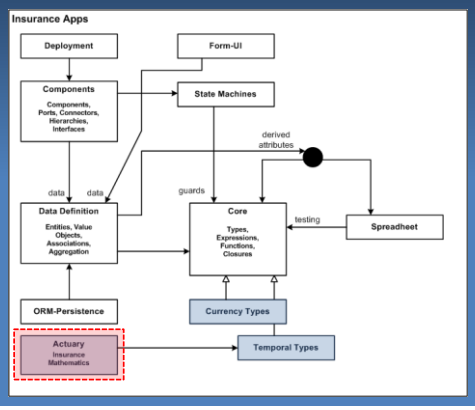
Insurance Mathematics

Temporal Types



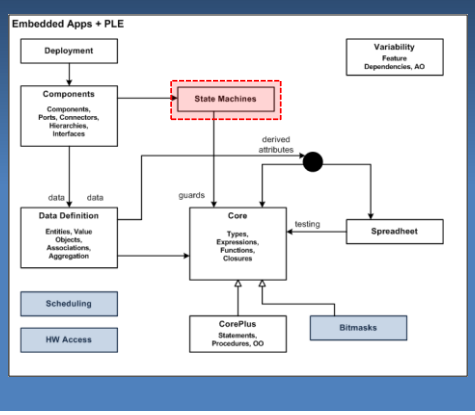
Insurance Apps

Deployment
Form-UI
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types
Actuary
Insurance Mathematics
Temporal Types



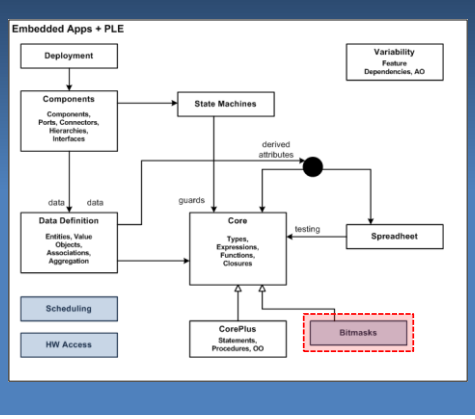
Insurance Apps

Deployment
Form-UI
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
ORM-Persistence
Currency Types
Actuary
Insurance Mathematics
Temporal Types



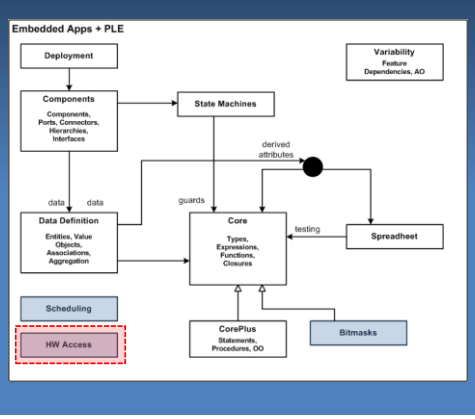
Embedded Apps + PLE

Deployment
Variability
Feature Dependencies, AO
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
Scheduling
HW Access
CorePlus
Statements, Procedures, OO
Bitmasks



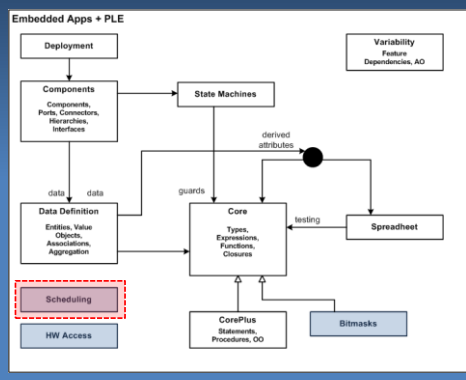
Embedded Apps + PLE

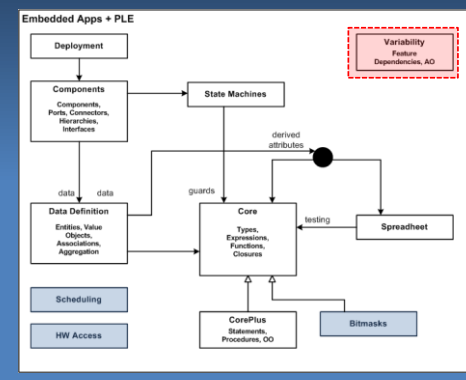
Deployment
Variability
Feature Dependencies, AO
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
Scheduling
HW Access
CorePlus
Statements, Procedures, OO
Bitmasks



Embedded Apps + PLE

Deployment
Variability
Feature Dependencies, AO
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadheet
Scheduling
HW Access
CorePlus
Statements, Procedures, OO
Bitmasks





(Seemingly)
Simple Example

Adding
matrices
to C in an
embedded
environment.

Currently:

1. Declare Data Structures in XML
2. Generate Headers
3. Implement manually in C

Currently:

Matrices
not supported
in XML format
and generator

Currently:

Tool team
would have to do the extension

- … a lot of work
- … busy
- … one central tool

Currently:

No real
compiler support
in the resulting C code

- … type checks
- … operator overloading
- … generics (matrix<int>)
- … matrix syntax?

Better Solution

qmatrix<int16>[3,3] $ModellMatrix = \begin{matrix} par_1 * n_{engine} & \dots & 0 \\ par_2 * n_{engine} * q & 0 & \dots \\ \dots & \dots & \dots \end{matrix}$

vector<int16>[3] $\overrightarrow{modellvektor} = \begin{matrix} temp_1 \\ temp_2 \\ temp_3 \end{matrix}$

```
vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b⃗ ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x⃗;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} (sgn(i) ∏_{k=1}^{n} a_{k,i(k)});
}
```

Better Solution

qmatrix<int16>[3,3] $ModellMatrix = \begin{matrix} par_1 * n_{engine} & \dots & 0 \\ par_2 * n_{engine} * q & 0 & \dots \\ \dots & \dots & \dots \end{matrix}$

vector<int16>[3] $\overrightarrow{modellvektor} = \begin{matrix} temp_1 \\ temp_2 \\ temp_3 \end{matrix}$

```
vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b⃗ ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x⃗;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} (sgn(i) ∏_{k=1}^{n} a_{k,i(k)});
}
```

generic matrix and vector types

Better Solution

qmatrix<int16>[3,3] $ModellMatrix = \begin{matrix} par_1 * n_{engine} & \dots & 0 \\ par_2 * n_{engine} * q & 0 & \dots \\ \dots & \dots & \dots \end{matrix}$

vector<int16>[3] $\overrightarrow{modellvektor} = \begin{matrix} temp_1 \\ temp_2 \\ temp_3 \end{matrix}$

```
vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b⃗ ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x⃗;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} (sgn(i) ∏_{k=1}^{n} a_{k,i(k)});
}
```

real matrix and vector literals

Better Solution

```
                                  par1 * nengine   ...   0
qmatrix<int16>[3,3] ModellMatrix = par2 * nengine * q   0   ...
                                  ...              ...   ...
                                  temp1
vector<int16>[3] modellvektor = temp2
                                  temp3

vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} ( sgn(i) ∏_{k=1}^{n} a_{k,i(k)} );
}
```

syntax **highlights** for **vectors** and **matrices**

Better Solution

```
                                  par1 * nengine   ...   0
qmatrix<int16>[3,3] ModellMatrix = par2 * nengine * q   0   ...
                                  ...              ...   ...
                                  temp1
vector<int16>[3] modellvektor = temp2
                                  temp3

vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} ( sgn(i) ∏_{k=1}^{n} a_{k,i(k)} );
}
```

operator overloading

Better Solution

```
                                  par1 * nengine   ...   0
qmatrix<int16>[3,3] ModellMatrix = par2 * nengine * q   0   ...
                                  ...              ...   ...
                                  temp1
vector<int16>[3] modellvektor = temp2
                                  temp3

vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} ( sgn(i) ∏_{k=1}^{n} a_{k,i(k)} );
}
```

operator overloading

static optimization

... symmetrical matrices
... identity matrix
... diagonal matrices

Better Solution

```
                                  par1 * nengine   ...   0
qmatrix<int16>[3,3] ModellMatrix = par2 * nengine * q   0   ...
                                  ...              ...   ...
                                  temp1
vector<int16>[3] modellvektor = temp2
                                  temp3

vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} ( sgn(i) ∏_{k=1}^{n} a_{k,i(k)} );
}
```

math notation

Better Solution

```
                                  par1 * nengine   ...   0
qmatrix<int16>[3,3] ModellMatrix = par2 * nengine * q   0   ...
                                  ...              ...   ...
                                  temp1
vector<int16>[3] modellvektor = temp2
                                  temp3

vector<int16> gausszerlege( qmatrix<int16> A, vector<int16> b ) {
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}

vector<int16> erg = ModellMatrix * modellvektor;

int16 det( qmatrix<int16> A ) {
        int n = size(A);
        return ∑_{i∈S_n} ( sgn(i) ∏_{k=1}^{n} a_{k,i(k)} );
}
```

a **separate** language **module**

used only by those who really need it

In addition: PLE Variability

```
Platform16Bit
typedef int16 elementType;
! Platform16Bit
typedef int8 elementType;

                                temp1
vector<elementType>[3] modellvektor = temp2
                                temp3

elementType someOperation {
        ...
}

vector<elementType> gausszerlege(qmatrix<elementTyp
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}
```

annotating **variability expressions** to **arbitrary regions**

In addition: PLE Variability

```
Platform16Bit
typedef int16 elementType;
! Platform16Bit
typedef int8 elementType;

                                temp1
vector<elementType>[3] modellvektor = temp2
                                temp3

elementType someOperation {
        ...
}

vector<elementType> gausszerlege(qmatrix<elementTyp
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}
```

annotating **variability expressions** to **arbitrary regions**

statically checked

In addition: PLE Variability

```
typedef int8 elementType;

                                temp1
vector<elementType>[3] modellvektor = temp2
                                temp3

elementType someOperation {
        ...
}

vector<elementType> gausszerlege(qmatrix<elementTyp
        // gausszerlegung.... à la A x = b  => x = A \ b
        return x;
}
```

annotating **variability expressions** to **arbitrary regions**

project variant

2 Available Tooling

2 Available Tooling

Intentional Software

Domain Workbench

INTENTIONAL SOFTWARE

Version 1.0 released in

Dec 15, 2008

currently at 1.4

INTENTIONAL SOFTWARE

Commercial Product

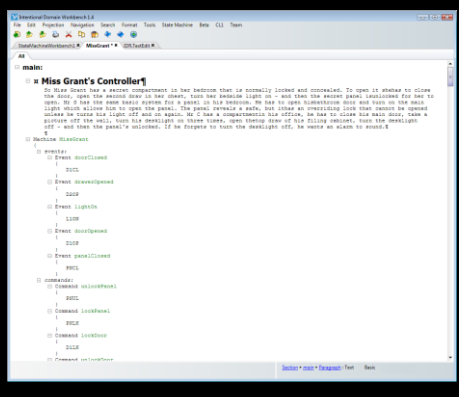
INTENTIONAL SOFTWARE

Eval available

upon request

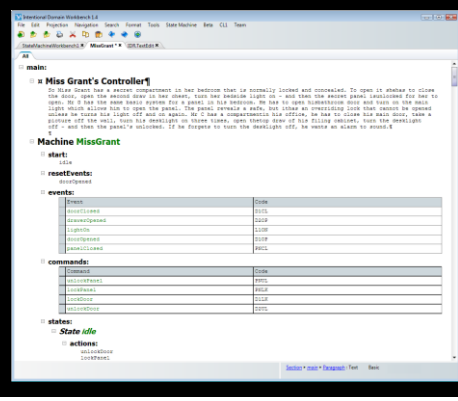
Statemachine Example

Native Projection



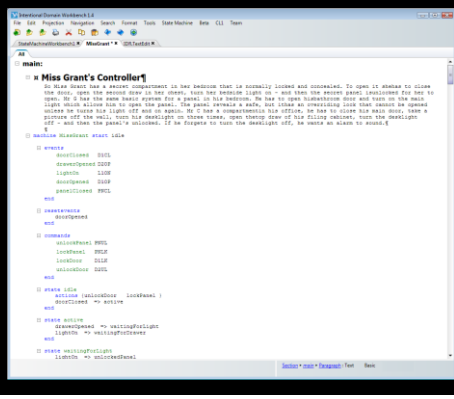
Statemachine Example

Tabular Projection



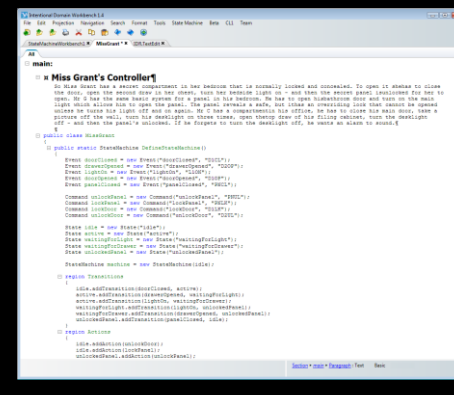
Statemachine Example

Textual DSL Projection



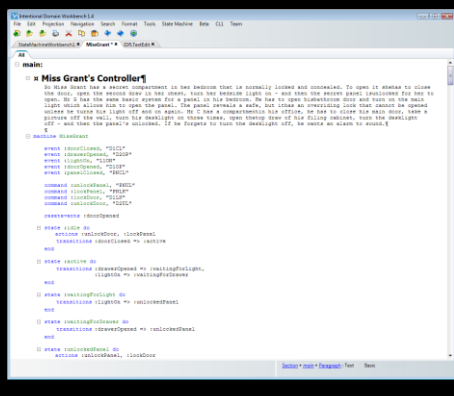
Statemachine Example

Java Projection



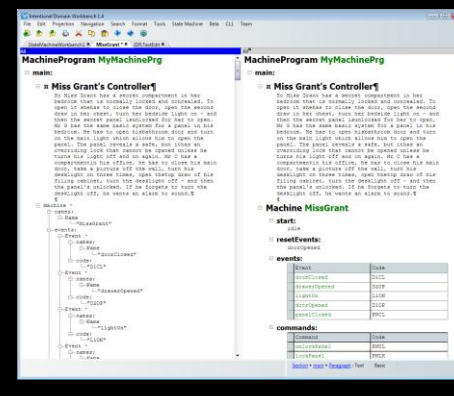
Statemachine Example

Ruby Projection



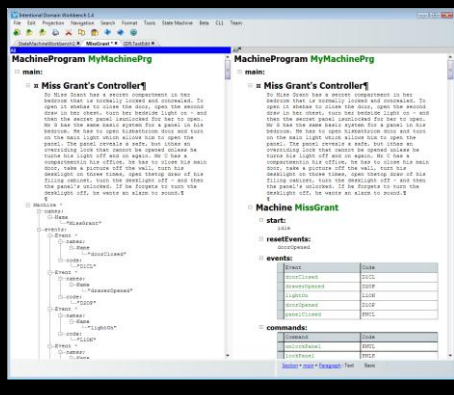
Statemachine Example

Two projections side/side

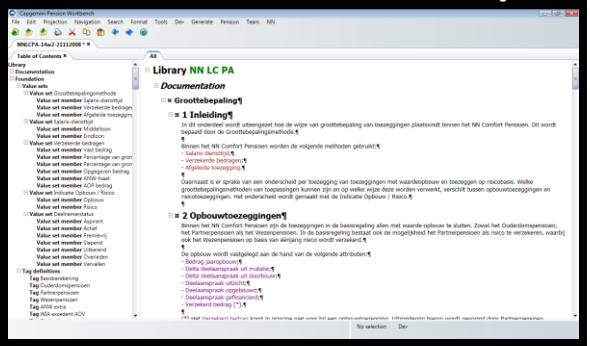


Statemachine Example

Two projections side/side

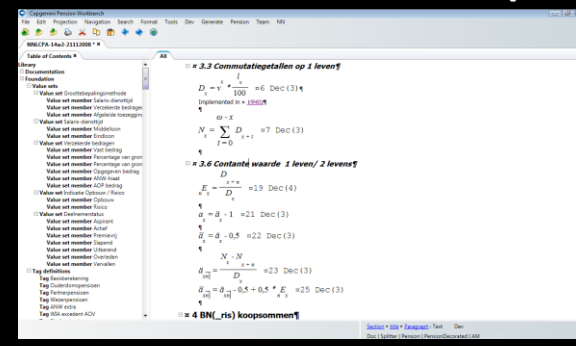


Pension Workbench Example



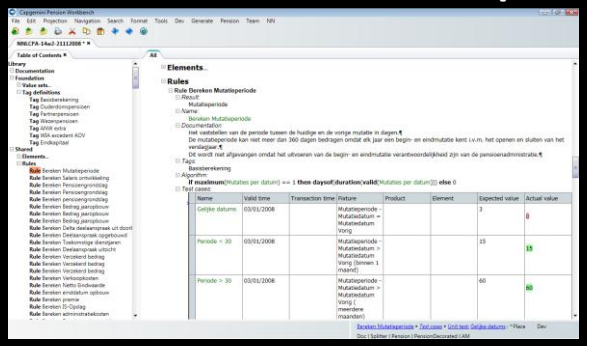
Text Editing Domain

Pension Workbench Example



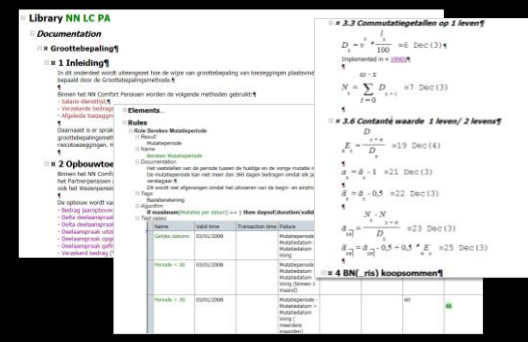
Insurance Mathematics Domain

Pension Workbench Example



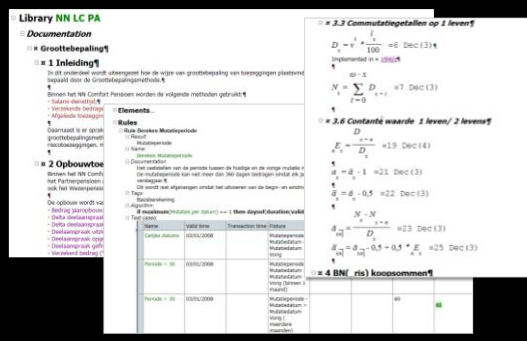
Pension Contract Rules Domain

Pension Workbench Example



All in one Document

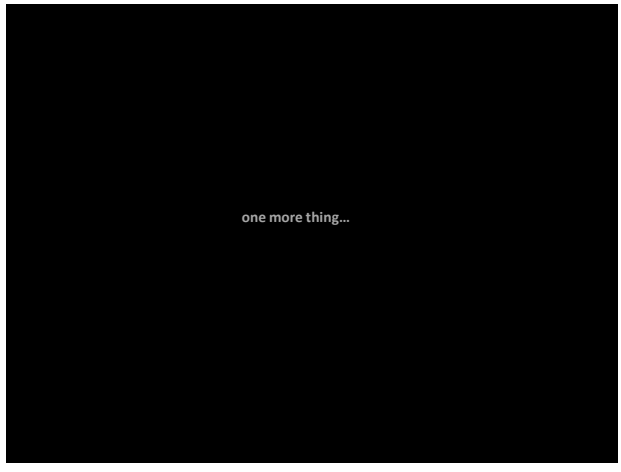
Pension Workbench Example



Symbolically integrated


A Journey…
Are we there?


A Journey…
.coordinates
web www.voelter.de
email voelter@acm.org
skype schogglad
xing http://www.xing.com/profile/Markus_Voelter
linkedin http://www.linkedin.com/pub/0/377/a31
itemis


one more thing…

Sioux Embedded Systems
Domain Specific Languages (for Product Lines)
http://sioux.nl

Feb 3 – 4, 2010
Eindhoven, NL

premium courses

itemis
Xtext
http://itemis.de

12./13.11.2009: Hamburg
18./19.02.2010: Hamburg
25./26.02.2010: Stuttgart
15./16.04.2010: München **http://bit.ly/3gRt8l**

itemis
Eclipse Modeling (GEF, EMF, GMF)
http://itemis.de

23./27.11.2009: Bonn
30./04.12.2009: München **http://bit.ly/2n9H9j**

THE END. Really.