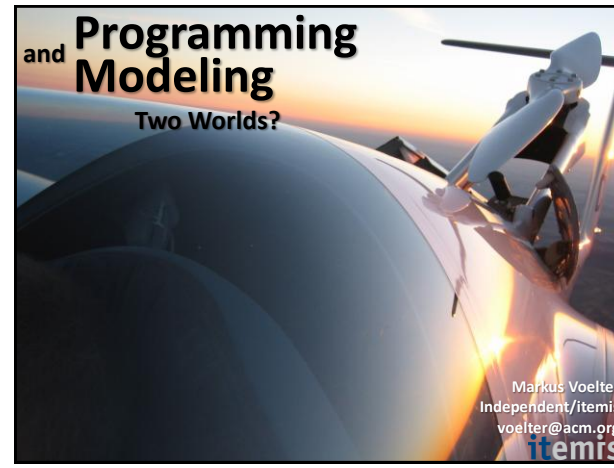
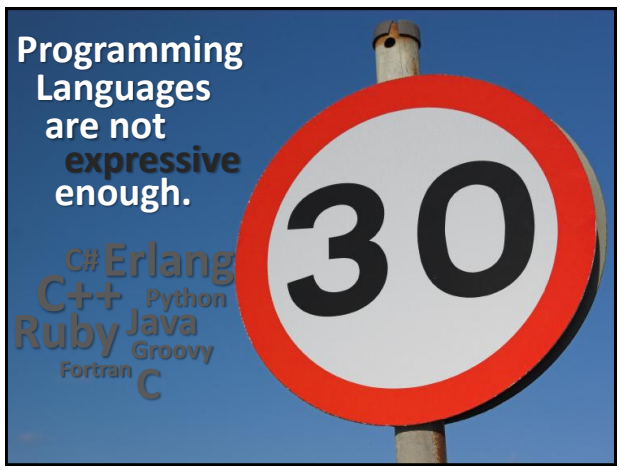
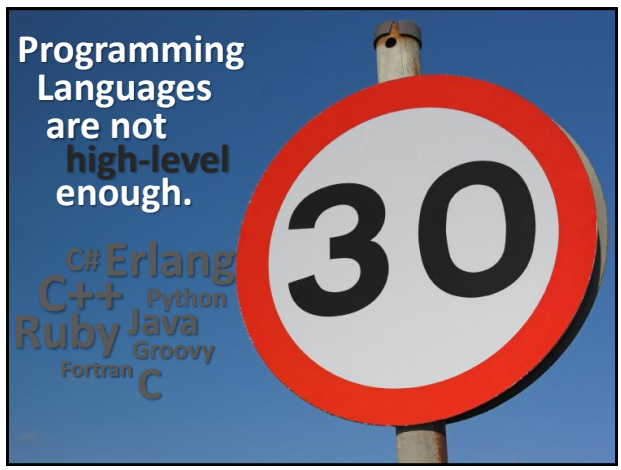
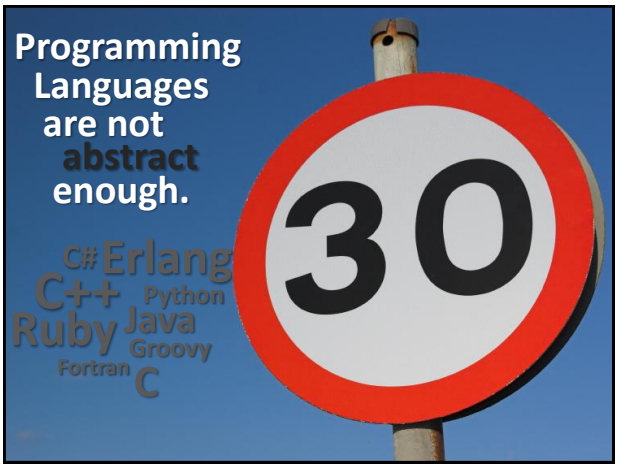

und Programmierung
Modellierung
Zwei Welten?
Markus Voelter
Independent/itemis
voelter@acm.org
itemis

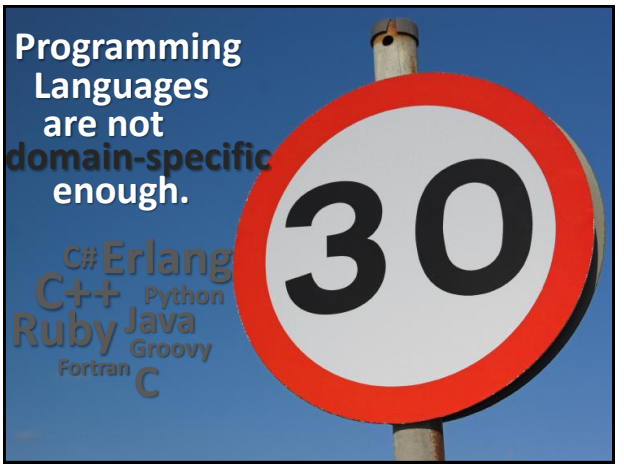

and Programming
Modeling
Two Worlds?
Markus Voelter
Independent/itemis
voelter@acm.org
itemis

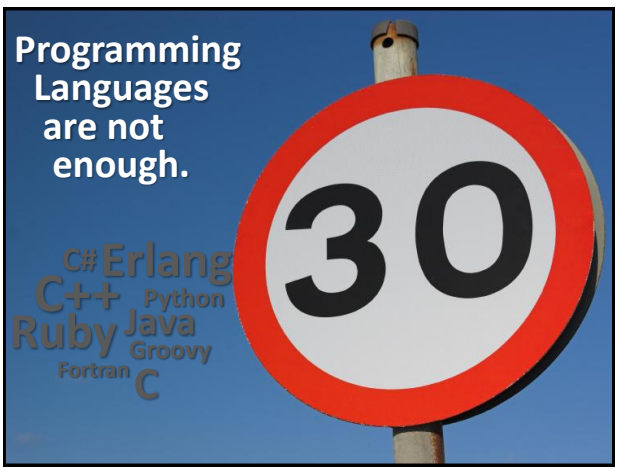

Programming
Languages
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C
30


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

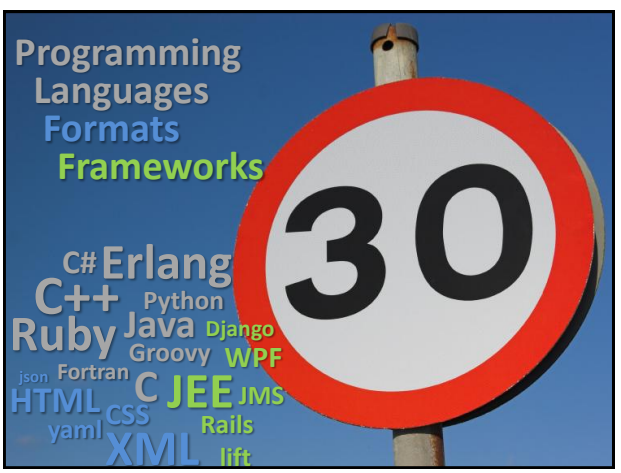

Programming
Languages
are not
high-level
enough.
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C
30

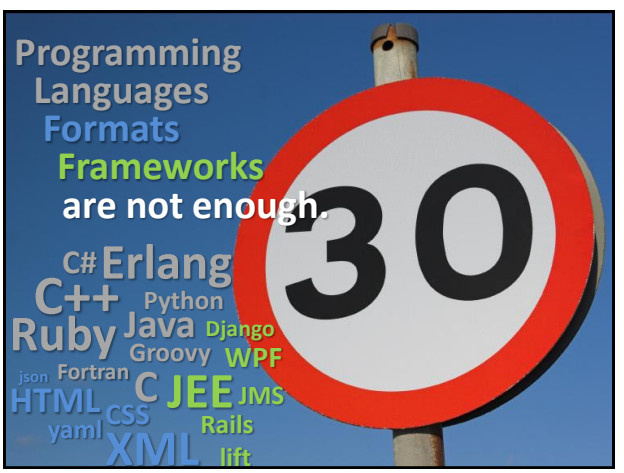

Programming
Languages
are not
abstract
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


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


Programming
Languages
Formats
Frameworks
C# Erlang
C++ Python
Ruby Java Django
Groovy WPF
json Fortran C JEE JMS
HTML CSS
yaml Rails
XML lift
30


Programming
Languages
Formats
Frameworks
are not enough.
C# Erlang
C++ Python
Ruby Java Django
Groovy WPF
json Fortran C JEE JMS
HTML CSS
yaml Rails
XML lift
30

So?

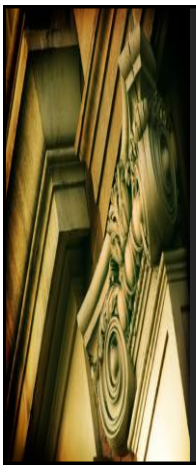
Modeling

... Higher Level

... Domain Specific
Concepts & Notations

... Code Generation
Interpretation

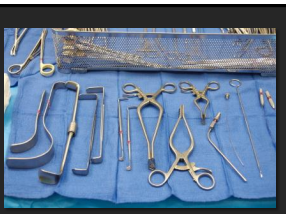
Solves the Problem!



DSL

general purpose

domain specific



tailor made

effective++

specialized, limited

used by experts

together with other specialized tools

But:

Different Worlds

Programming Tools
!=
Modeling Tools

Modeling	Programming

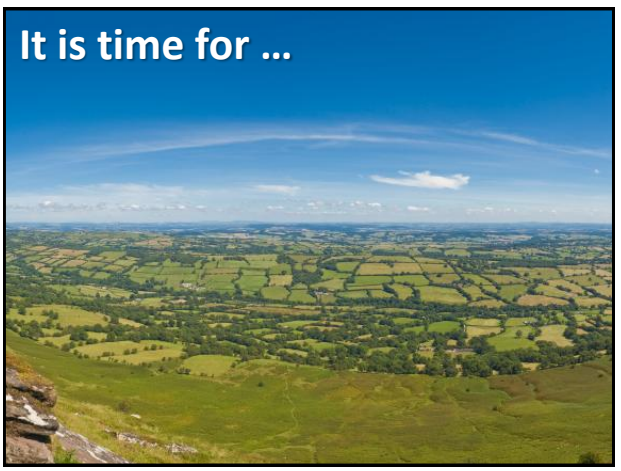
Modeling	Programming
	… (Mostly) Textual Notations … Concrete Syntax Storage … (Fancy) ASCII Editors … Read-Only Visualizations

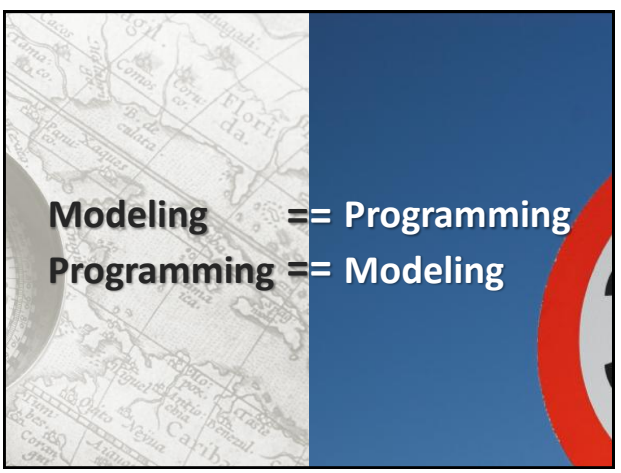
Modeling	Programming
… (Mostly) Graphical Notations … Abstract Syntax Storage … Projecting Editors … Different editable views for model	… (Mostly) Textual Notations … Concrete Syntax Storage … (Fancy) ASCII Editors … Read-Only Visualizations

Why

the difference?

Why
the difference?





modeling == programming

modeling == programming

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

modeling == programming

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

modeling == programming

And always:
precise and tool processable

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

When a graphical
notation
is better, you can
visualize.

1 Available Tooling

Eclipse
Xtext

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

Infrastructure Integration

... **storage** not text

... **diff/merge** with existing tools

... **existing text** work well!

Language Composition

Grammar composition with traditional parsers is tough!

More advanced parsers currently resarch

Limited to Unicode

how to handle non-character symbols

Graphics != Text

two worlds…

separate editors

… per syntax/viewpoint

… models can still be ref integrated

eclipse modeling PROJECT

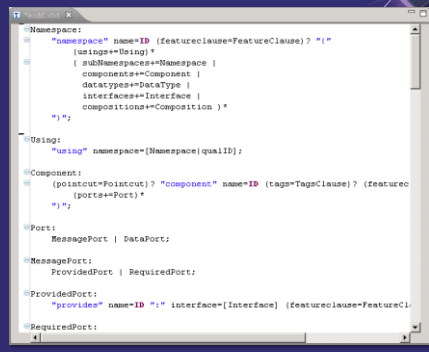
http://eclipse.org/modeling

xtext

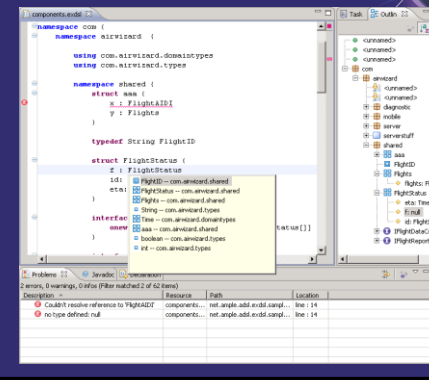
eclipse modeling PROJECT

http://eclipse.org/xtext

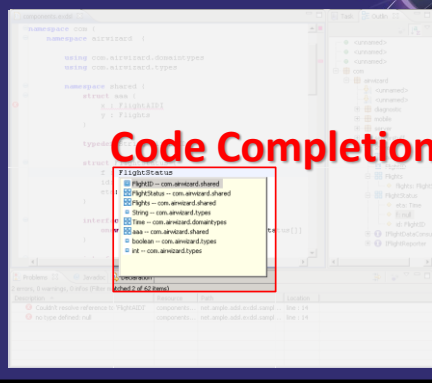
Xtext: Specify Grammar



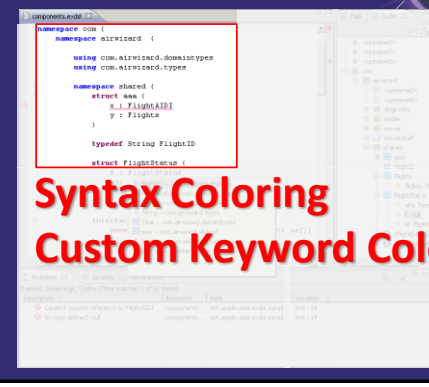
Xtext: Generated Editor



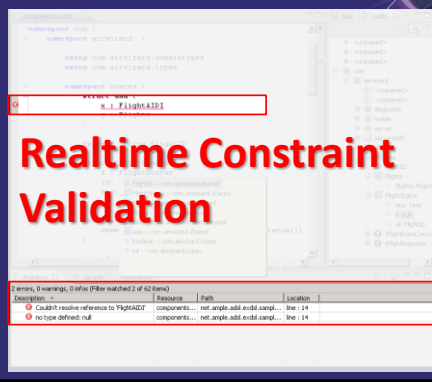
Xtext: Generated Editor



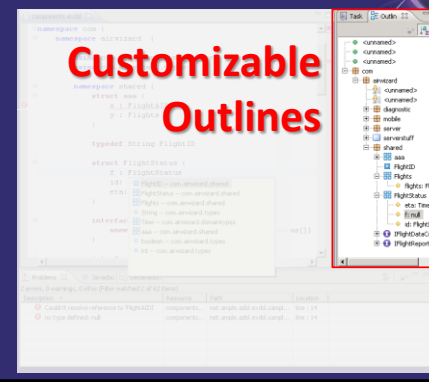
Xtext: Generated Editor



Xtext: Generated Editor

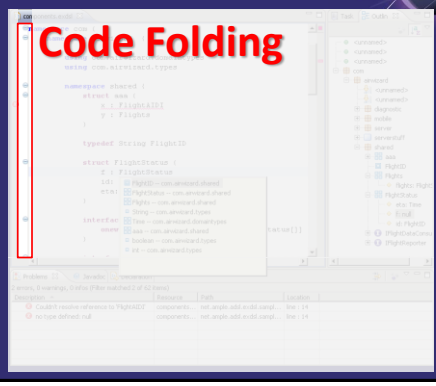


Xtext: Generated Editor



Xtext: Generated Editor

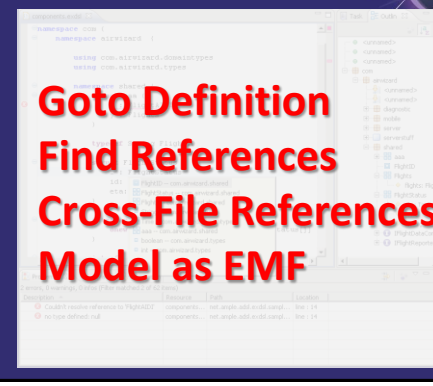
eclipse modeling PROJECT



Code Folding

Xtext: Generated Editor

eclipse modeling PROJECT



Goto Definition
Find References
Cross-File References
Model as EMF

DEMO I

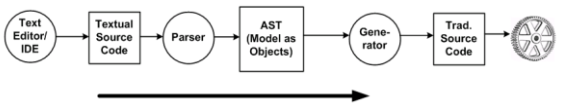
Building DSLs with
Eclipse Xtext

2 Available Tooling

Jetbrains'
Meta Programming System

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

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

licensed under

Apache 2.0

released in

Q2 2009

currently

1.1 RC1

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

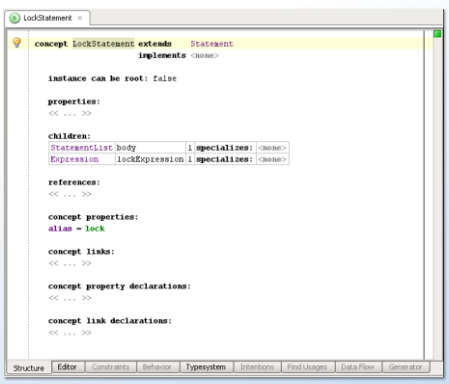
```
ReadWriteLock l = …
l.readLock().lock();
try {
  //code
} finally {
  l.readLock().unlock();
}
```

New

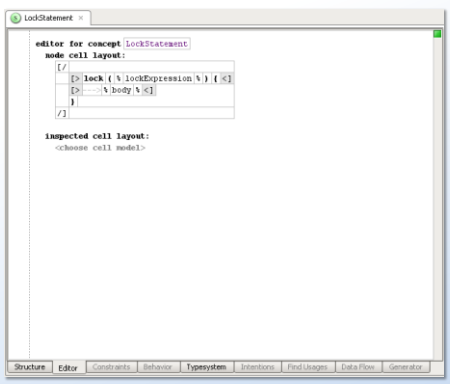
```
ReadWriteLock l = …
lock (l) {
  //code
}
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator

```
LockStatement
editor for concept LockStatement
  node cell layout:
    [/
      lock ( %lockExpression% ) { <> }
      ...%body%
    }
    /]
    body                 linkDeclaration (jaxdemo.structure.LockStatement)
    lockExpression       linkDeclaration (jaxdemo.structure.LockStatement)

  inspected cell layout:
    <choose cell model>

Structure | Editor | Constraints | Behavior | Typesystem | Intentions | Find Usages | Data Flow | 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;  }

}
```

DEMO II

Building DSLs with JetBrains MPS

3 A vision for Programming

Programming Languages are not **MODULAR** enough.

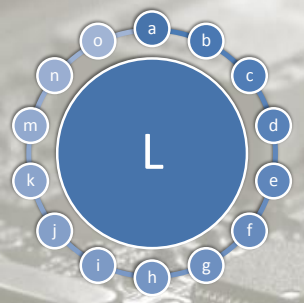
Programming Languages are not **COMPOSABLE** enough.

Programming Languages are not **CONFIGURABLE** enough.

Programming Languages are not **ADAPTABLE** enough.

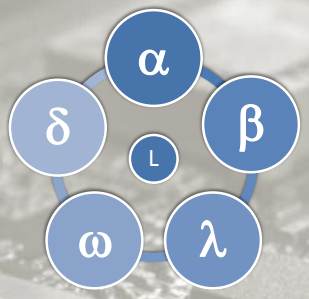
Programming Language Syntax is not **FLEXIBLE** enough.

Big Language?



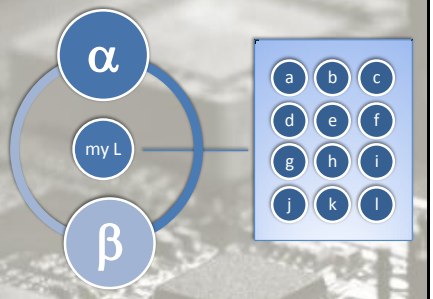
with many first class concepts!

Small Language?



with a few, orthogonal and poweful concepts

Modular Language



with many optional, composable concepts

Modular Language

Like frameworks and libraries,

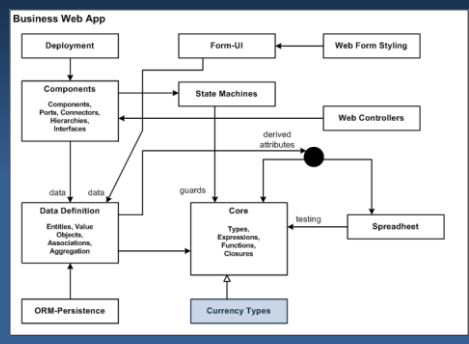
Modular Language

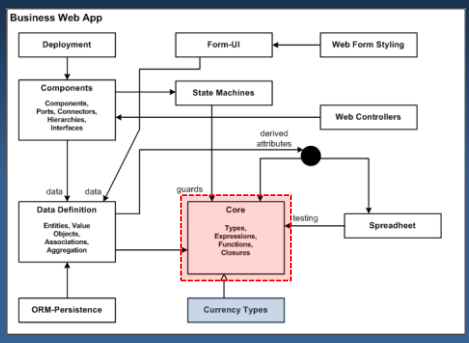
Like frameworks and libraries,

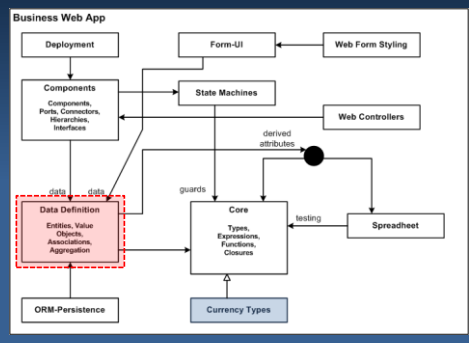
but with syntax and IDE support

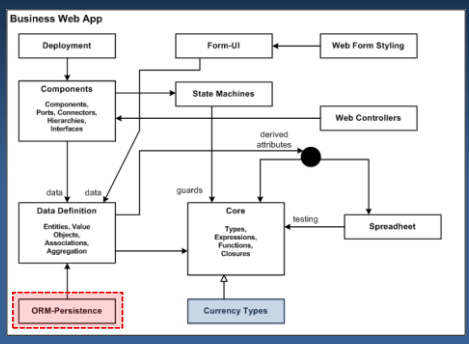
Not a new idea…

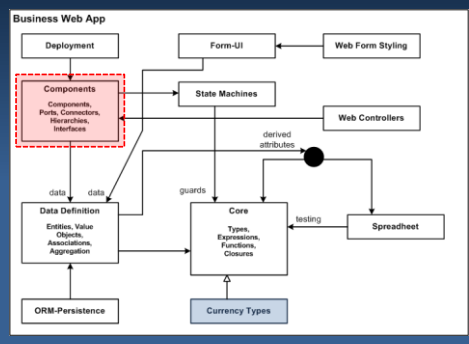
Example Languages

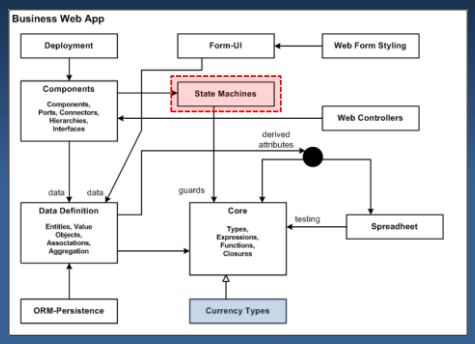












Business Web App

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

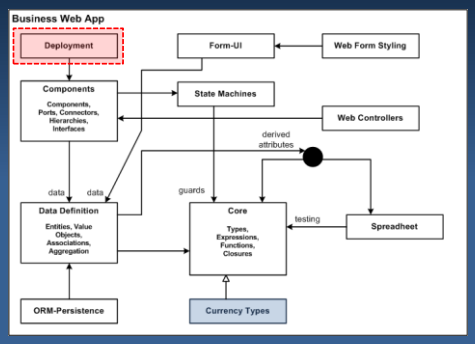
Types, Expressions, Functions, Closures

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

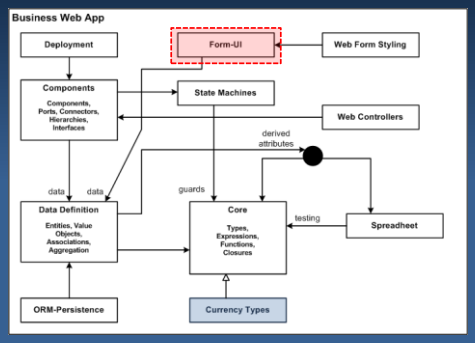
Types, Expressions, Functions, Closures

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

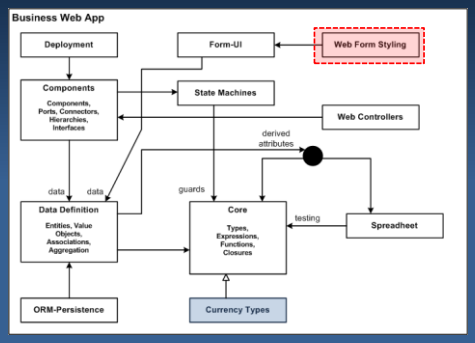
Types, Expressions, Functions, Closures

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

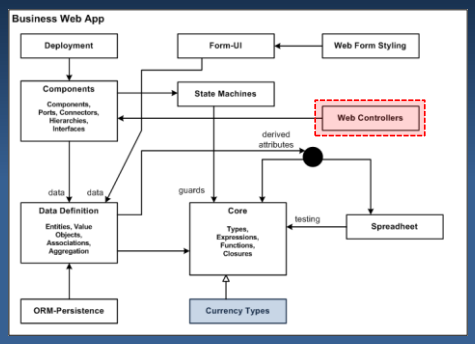
Types, Expressions, Functions, Closures

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

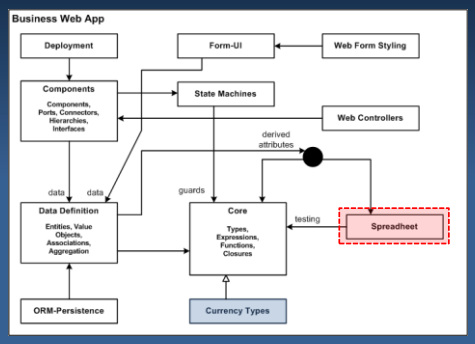
Types, Expressions, Functions, Closures

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

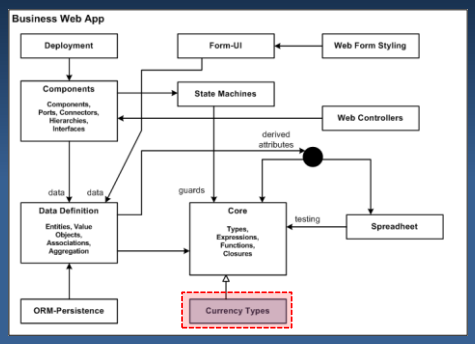
Types, Expressions, Functions, Closures

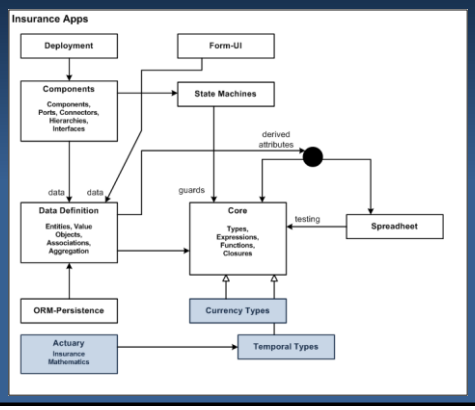
testing

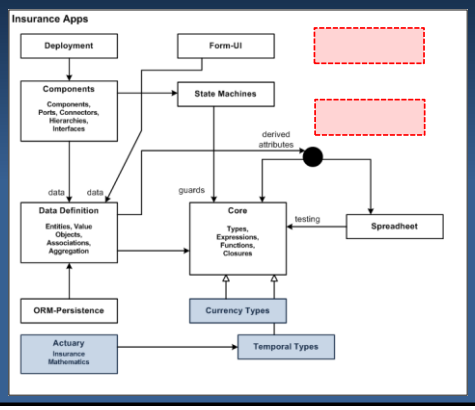
Spreadheet

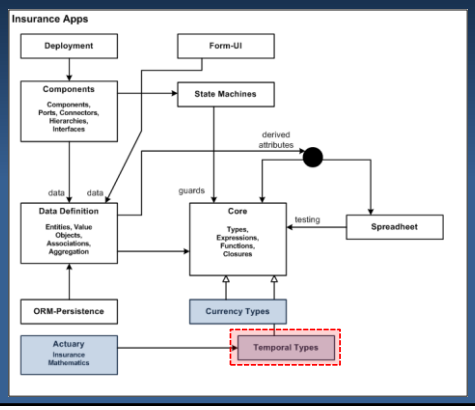
ORM-Persistence

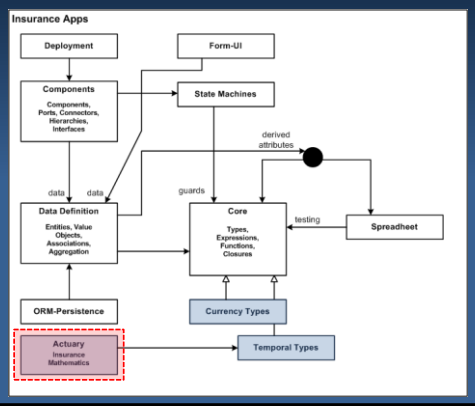
Currency Types

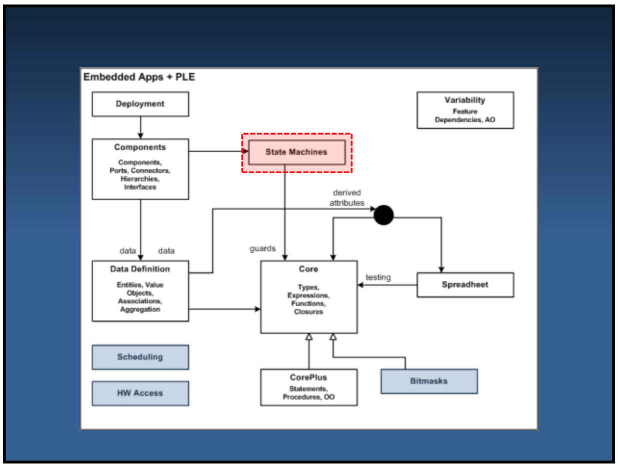

Business Web App
Deployment
Form-UI
Web Form Styling
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
Web Controllers
derived attributes
data
data
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing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



Embedded Apps + PLE

Deployment

Components
Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Variability
Feature Dependencies, AO

derived attributes

data

data

guards

Data Definition
Entities, Value Objects, Associations, Aggregation

Core
Types, Expressions, Functions, Closures

testing

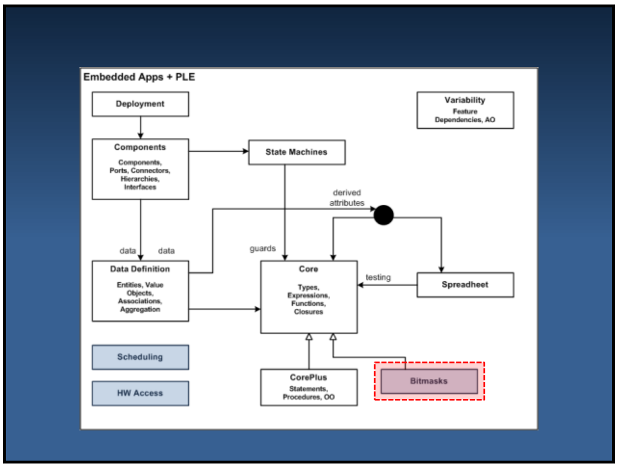
Spreadheet

Scheduling

HW Access

CorePlus
Statements, Procedures, OO

Bitmasks



Embedded Apps + PLE

Deployment

Components
Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Variability
Feature Dependencies, AO

derived attributes

data

data

guards

Data Definition
Entities, Value Objects, Associations, Aggregation

Core
Types, Expressions, Functions, Closures

testing

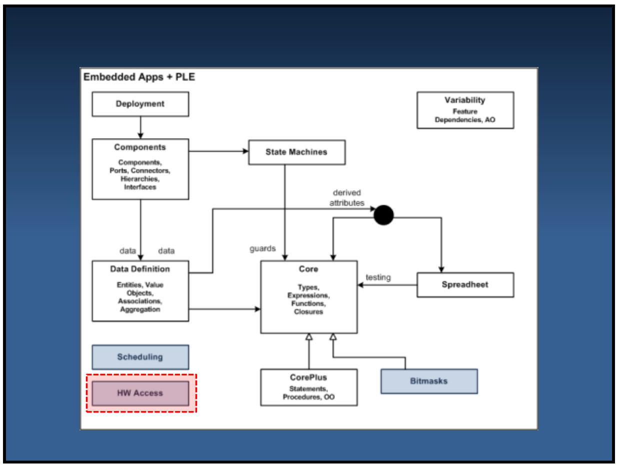
Spreadheet

Scheduling

HW Access

CorePlus
Statements, Procedures, OO

Bitmasks



Embedded Apps + PLE

Deployment

Components
Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Variability
Feature Dependencies, AO

derived attributes

data

data

guards

Data Definition
Entities, Value Objects, Associations, Aggregation

Core
Types, Expressions, Functions, Closures

testing

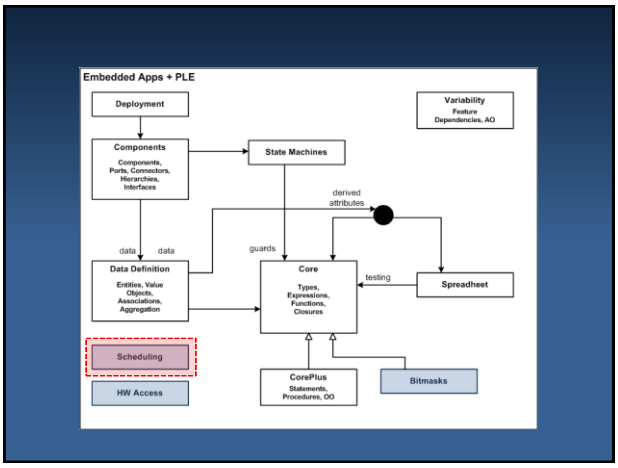
Spreadheet

Scheduling

HW Access

CorePlus
Statements, Procedures, OO

Bitmasks



Embedded Apps + PLE

Deployment

Components
Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Variability
Feature Dependencies, AO

derived attributes

data

data

guards

Data Definition
Entities, Value Objects, Associations, Aggregation

Core
Types, Expressions, Functions, Closures

testing

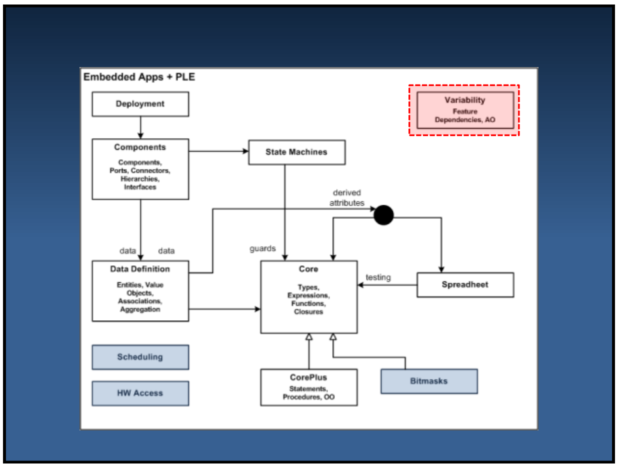
Spreadheet

Scheduling

HW Access

CorePlus
Statements, Procedures, OO

Bitmasks



Embedded Apps + PLE

Deployment

Components
Components, Ports, Connectors, Hierarchies, Interfaces

State Machines

Variability
Feature Dependencies, AO

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 = | par1 * n_engine   ...   0 |
                                   | par2 * n_engine * q   0   ... |
                                   |        ...        ...   ... |
vector<int16>[3] modellvektor = | temp1 |
                                | 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  Σ_{i∈S_n} ( sgn(i) Π_{k=1}^{n} a_{k,i(k)} );
}
```

Better Solution

```
qmatrix<int16>[3,3] ModellMatrix = | par1 * n_engine   ...   0 |
                                   | par2 * n_engine * q   0   ... |
                                   |        ...        ...   ... |
vector<int16>[3] modellvektor = | temp1 |
                                | 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  Σ_{i∈S_n} ( sgn(i) Π_{k=1}^{n} a_{k,i(k)} );
}
```

generic matrix and vector types

Better Solution

```
qmatrix<int16>[3,3] ModellMatrix = | par1 * n_engine   ...   0 |
                                   | par2 * n_engine * q   0   ... |
                                   |        ...        ...   ... |
vector<int16>[3] modellvektor = | temp1 |
                                | 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  Σ_{i∈S_n} ( sgn(i) Π_{k=1}^{n} a_{k,i(k)} );
}
```

real matrix and vector literals

Better Solution

```
qmatrix<int16>[3,3] ModellMatrix = | par1 * n_engine   ...   0 |
                                   | par2 * n_engine * q   0   ... |
                                   |        ...        ...   ... |
vector<int16>[3] modellvektor = | temp1 |
                                | 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  Σ_{i∈S_n} ( sgn(i) Π_{k=1}^{n} a_{k,i(k)} );
}
```

syntax highlights for vectors and matrices

Better Solution

```
qmatrix<int16>[3,3] ModellMatrix = | par1 * n_engine   ...   0 |
                                   | par2 * n_engine * q   0   ... |
                                   |        ...        ...   ... |
vector<int16>[3] modellvektor = | temp1 |
                                | 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  Σ_{i∈S_n} ( sgn(i) Π_{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

4 Available Tooling

Intentional Software

Domain Workbench

INTENTIONAL SOFTWARE

Commercial Product

Eval available

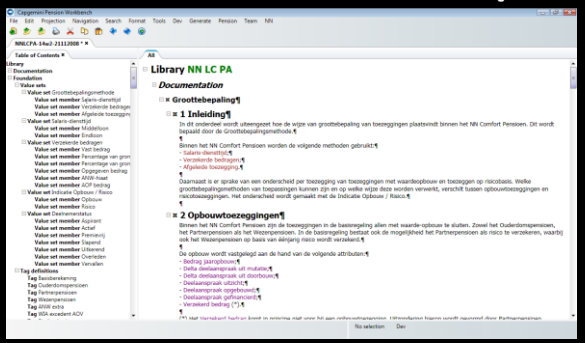
upon request

INTENTIONAL SOFTWARE

Eval available

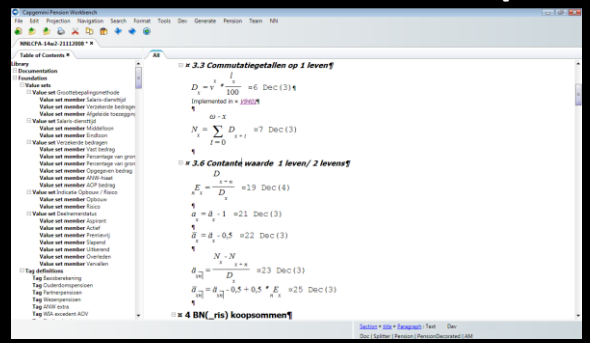
upon request

Pension Workbench Example



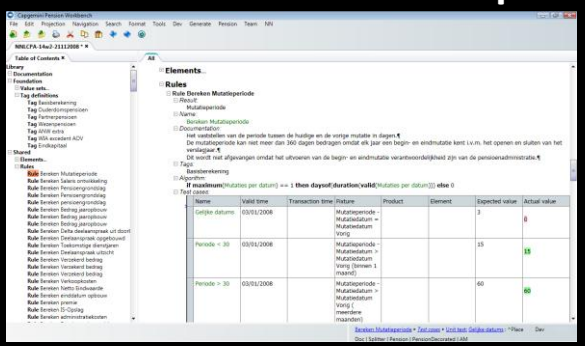
Text Editing Domain

Pension Workbench Example



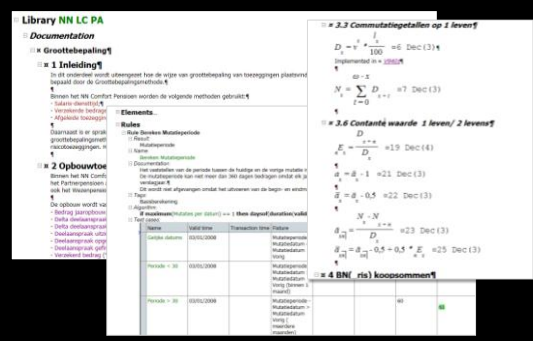
Insurance Mathematics Domain

Pension Workbench Example



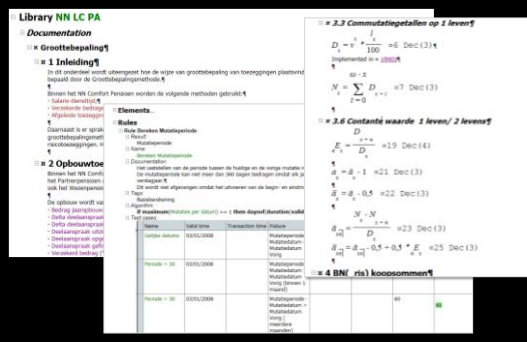
Pension Contract Rules Domain

Pension Workbench Example

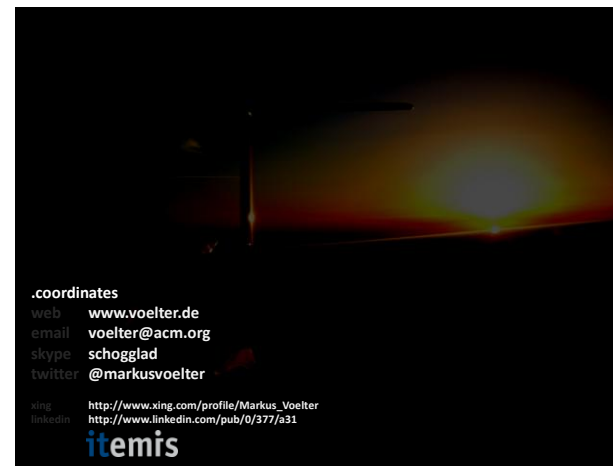


All in one Document

Pension Workbench Example



Symbolically integrated



.coordinates

web www.voelter.de
email voelter@acm.org
skype schogglad
twitter @markusvoelter

xing http://www.xing.com/profile/Markus_Voelter
linkedin http://www.linkedin.com/pub/0/377/a31

itemis