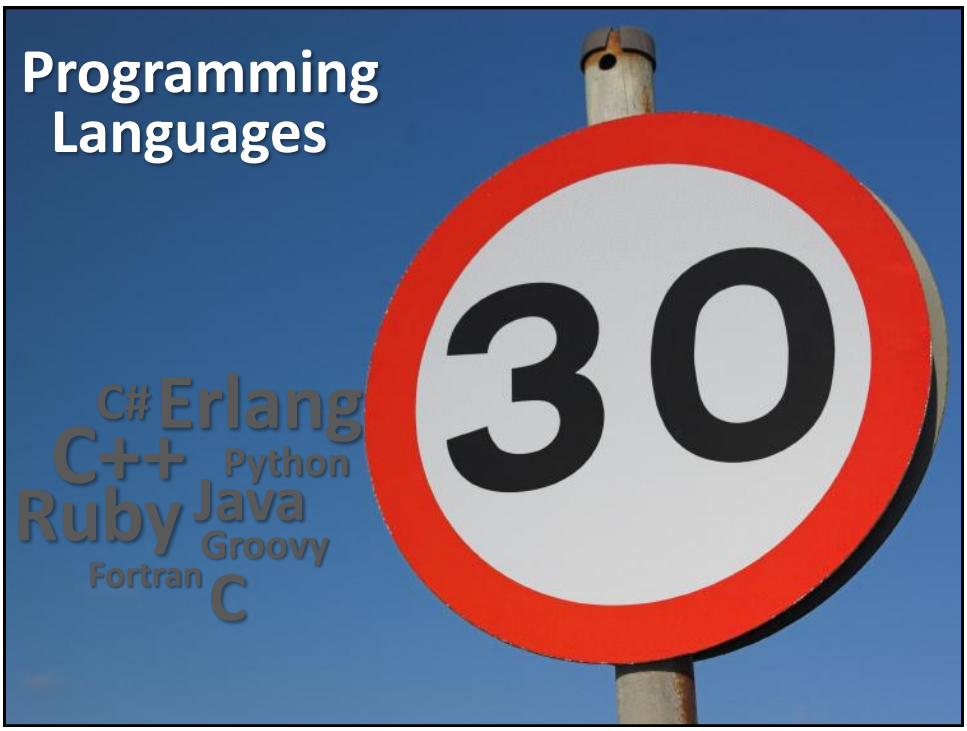
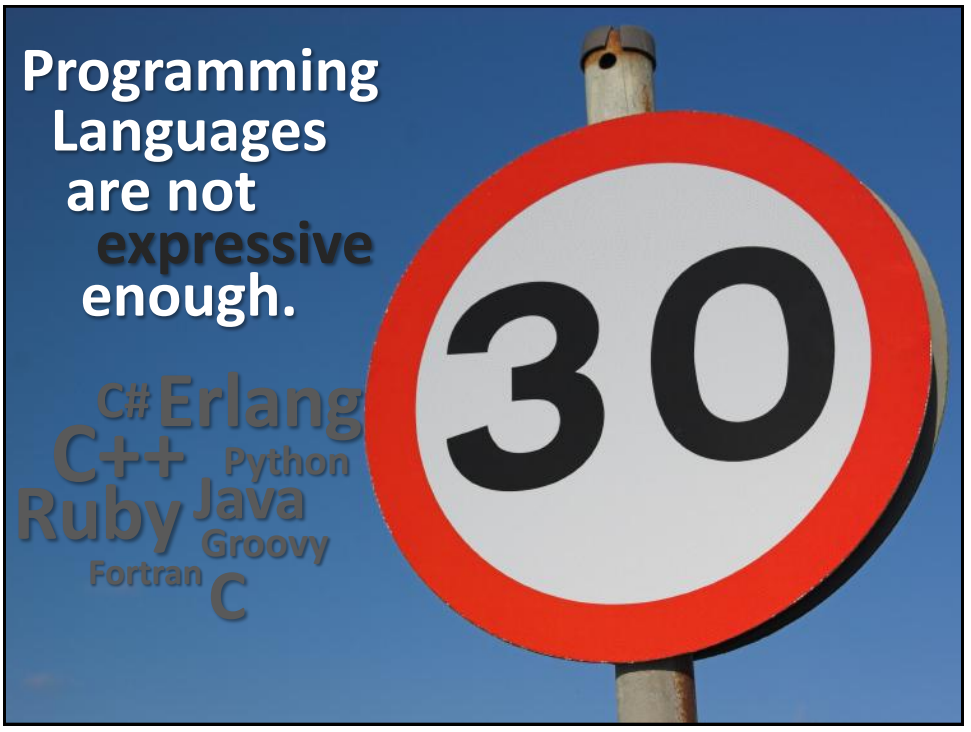
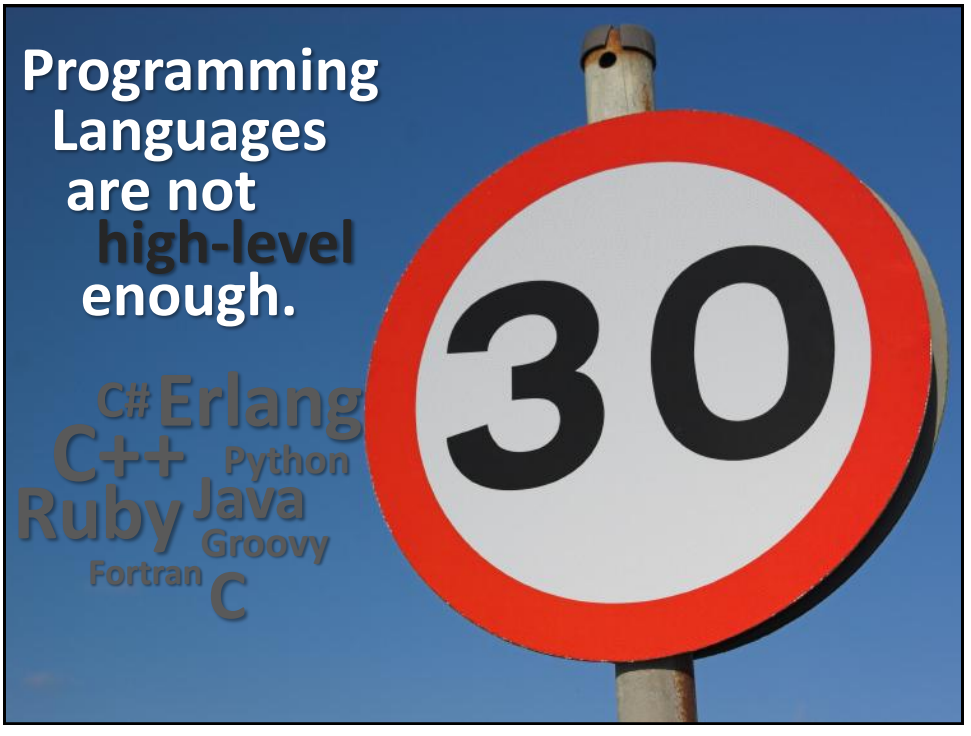
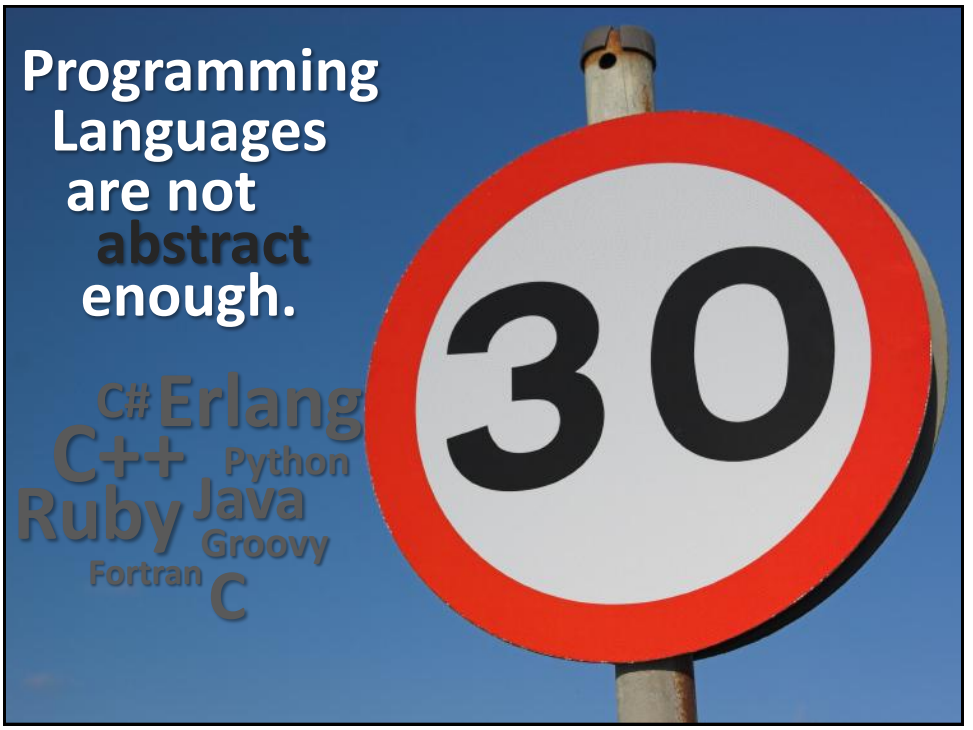
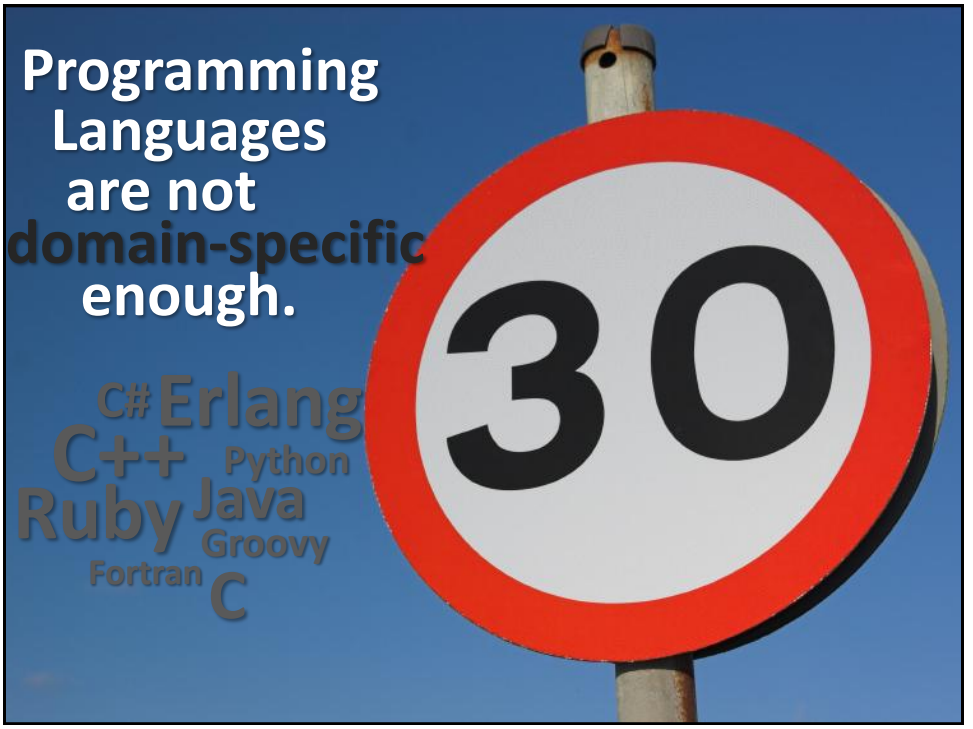

from
to
Programming
Modeling
... and back again.
Markus Voelter
Independent/itemis
voelter@acm.org
itemis

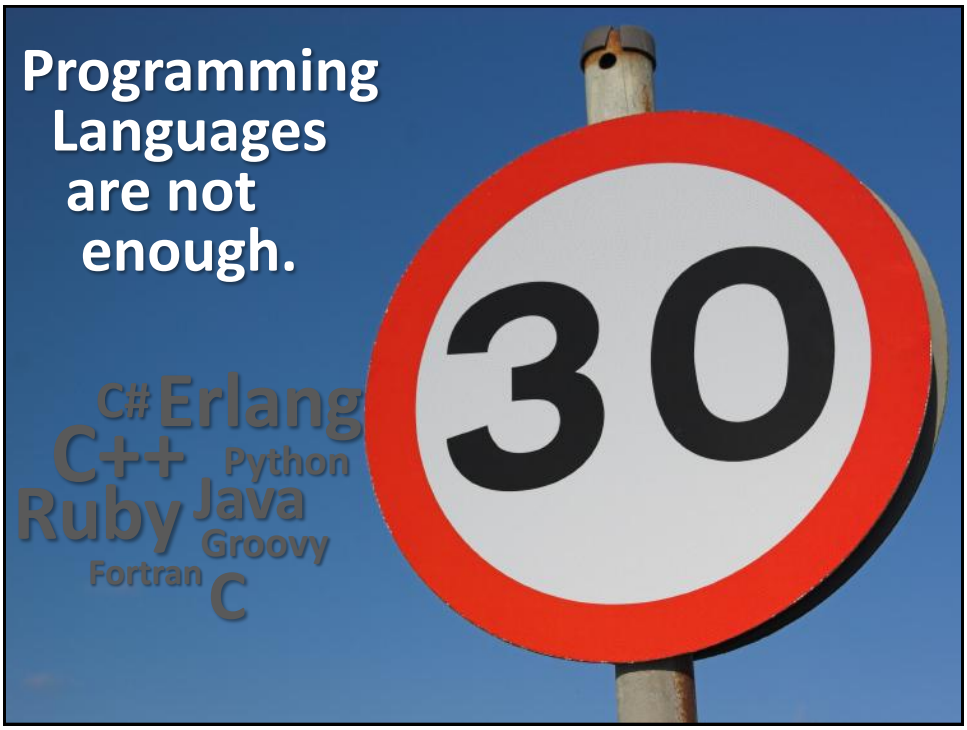

Programming
Languages
30
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C


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

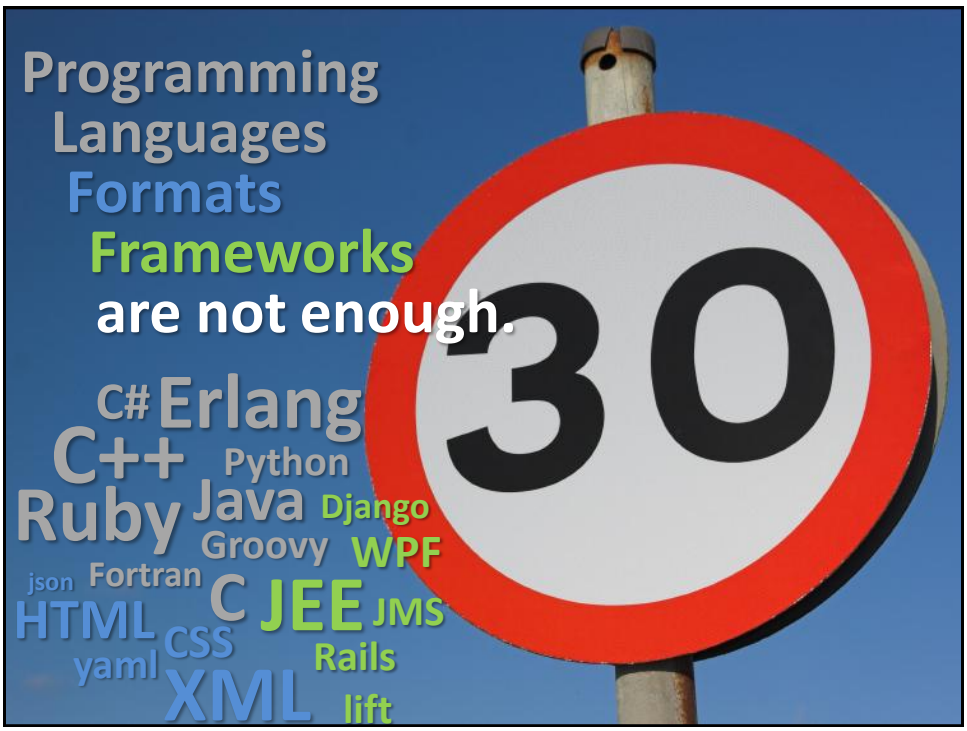

Programming
Languages
are not
abstract
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


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

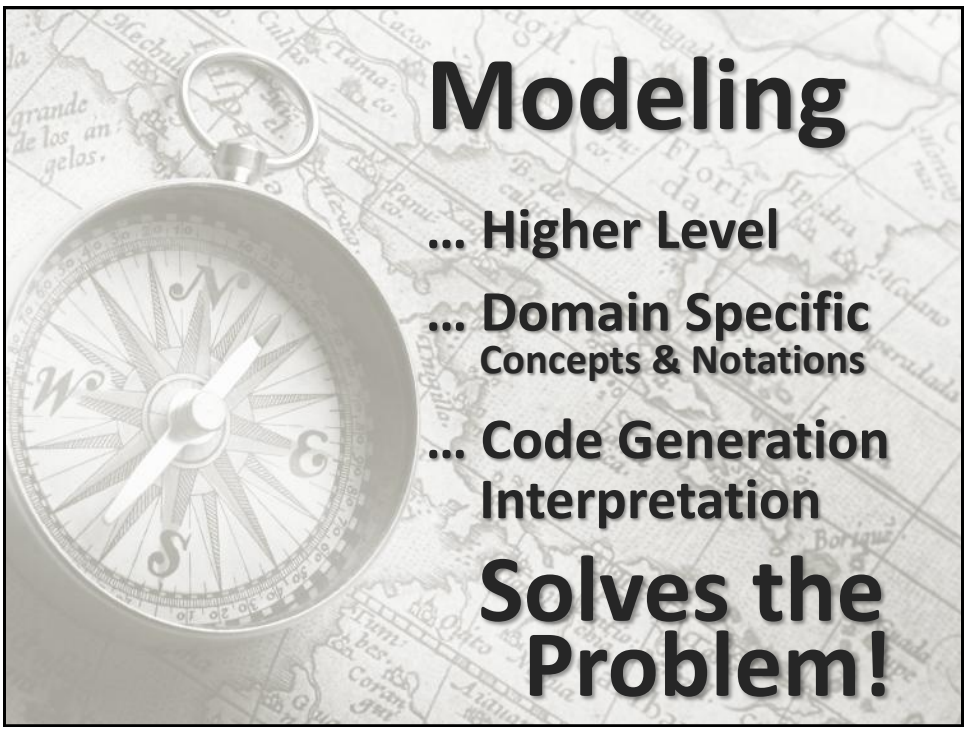

Programming
Languages
are not
enough.
C# Erlang
C++ Python
Ruby Java
Groovy
Fortran C
30


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


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


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

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!

?

Versioning
Diff, Merge
Branching

Brittle!

Mature!

for some tools…

Gets some Jobs done.

some people doubt that…

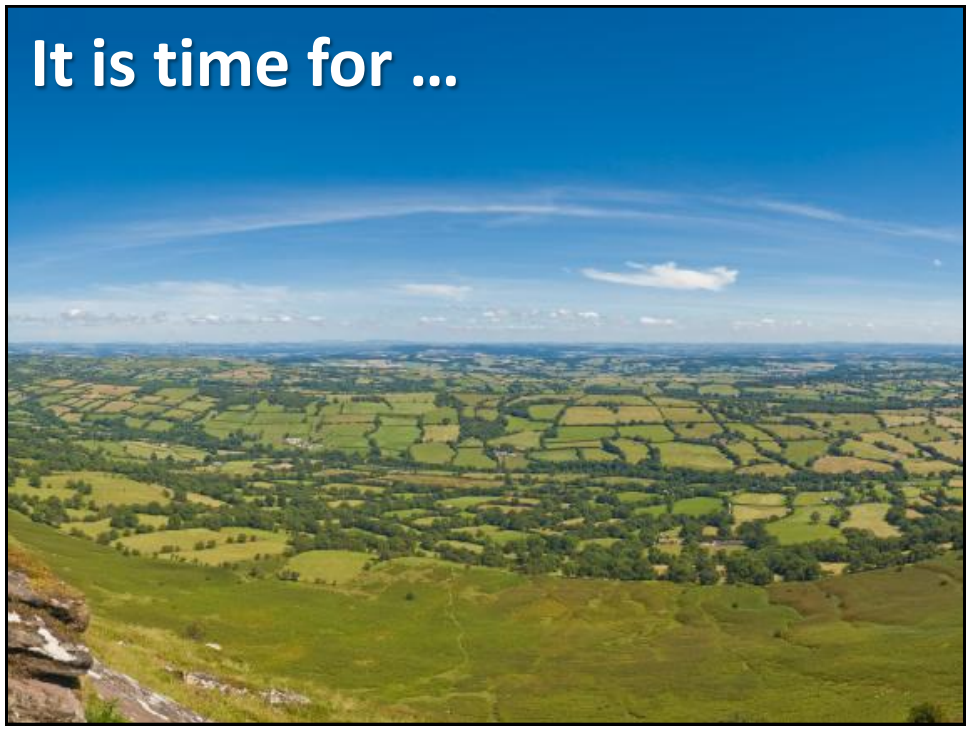
Gets the Job Done!

everybody agrees…

Why

the difference?

Why
the difference?


It is time for …


… a Different Perspective

Modeling == Programming
Programming == Modeling

modeling == programming

modeling == programming

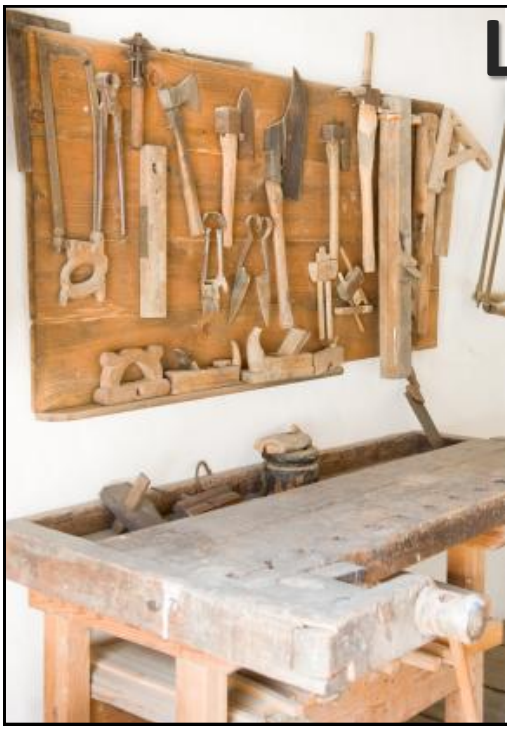
… at different levels of abstraction
… from different viewpoints
… integrated!

modeling == programming

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

modeling == programming

And always:
precise and tool processable



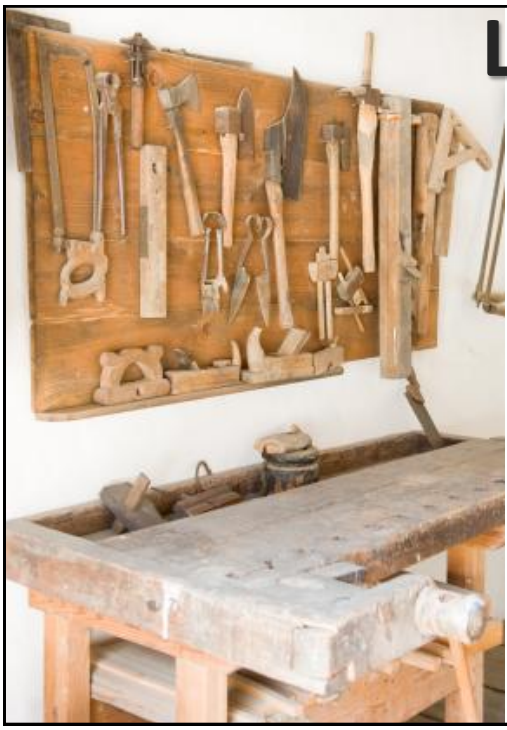
Language Workbench

(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

[redacted]

Syntax primarily textual

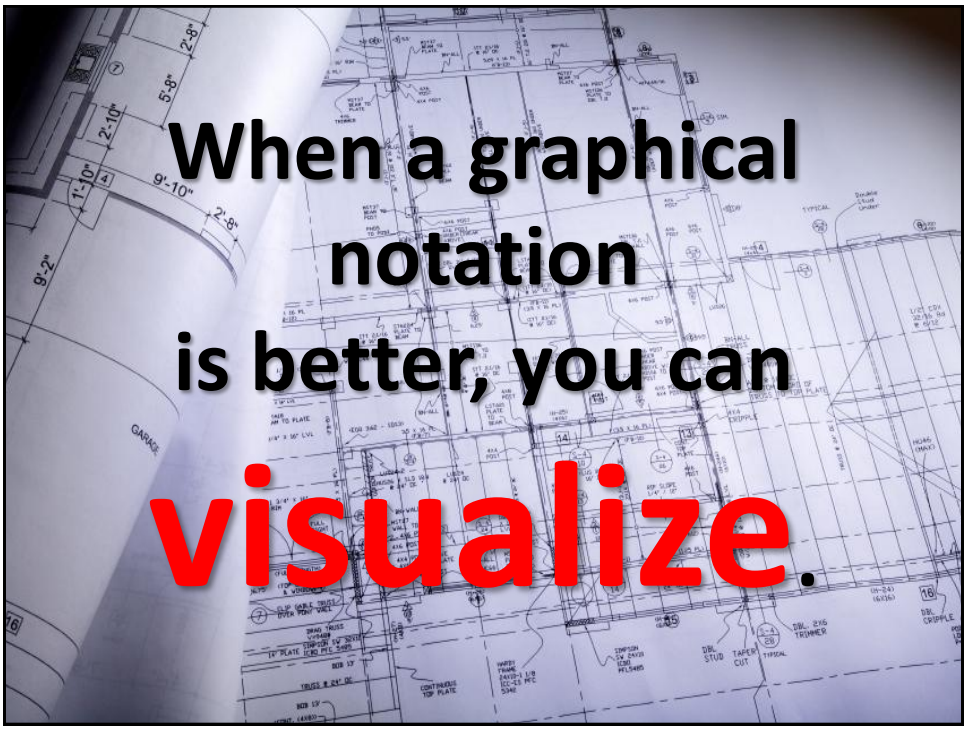
Syntax primarily textual

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

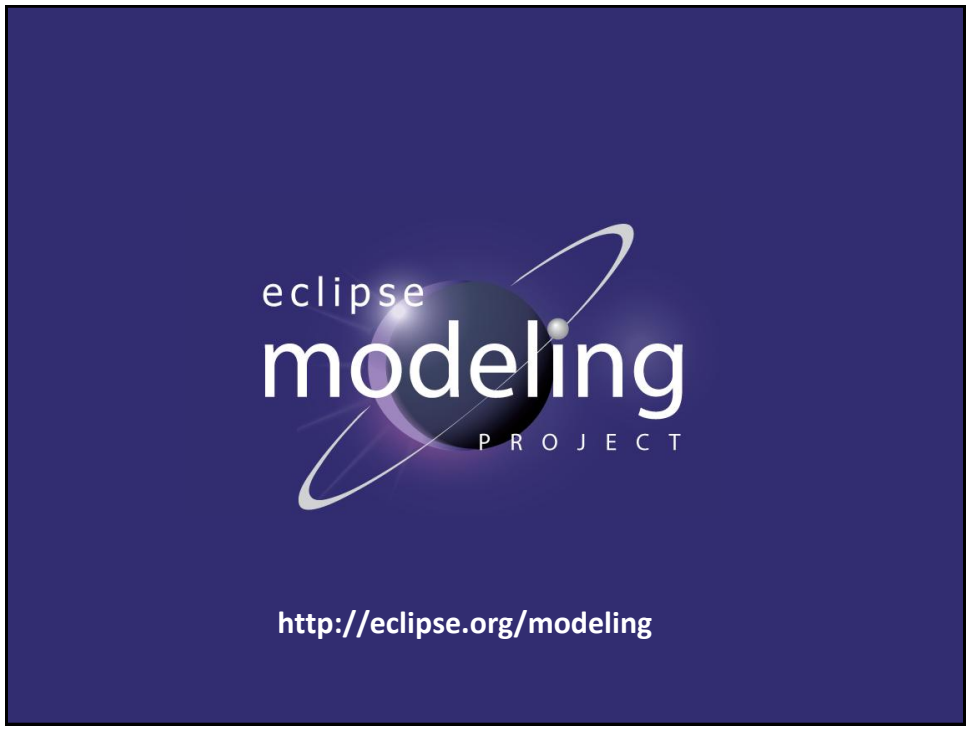
how to handle non-character symbols

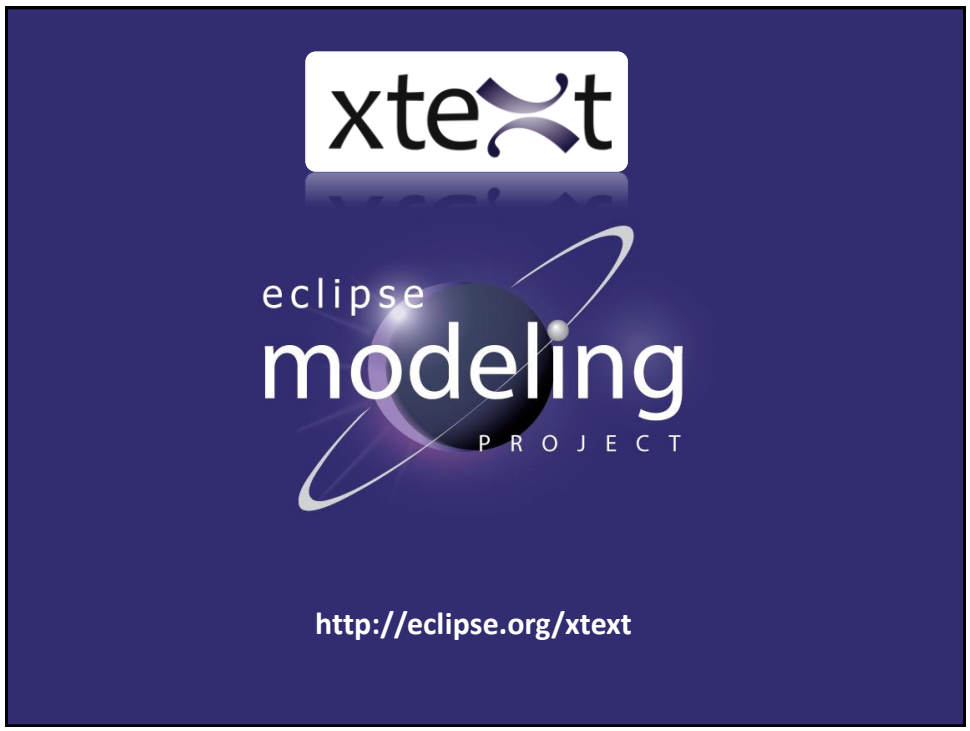
Graphics != Text

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

eclipse modeling PROJECT

```
*exdsl.xtxt
Namespace:
    "namespace" name=ID (featureclause=FeatureClause)? "{"
        (usings+=Using)*
        ( subNamespaces+=Namespace |
          components+=Component |
          datatypes+=DataType |
          interfaces+=Interface |
          compositions+=Composition )*
    "}";

Using:
    "using" namespace=[Namespace|qualID];

Component:
    (pointcut=Pointcut)? "component" name=ID (tags=TagsClause)? (featurec
        (ports+=Port)*
    "}";

Port:
    MessagePort | DataPort;

MessagePort:
    ProvidedPort | RequiredPort;

ProvidedPort:
    "provides" name=ID ":" interface=[Interface] (featureclause=FeatureCl

RequiredPort:
```

Xtext: Generated Editor

eclipse modeling PROJECT

```
components.exdsl
namespace com {
    namespace airwizard  {

        using com.airwizard.domaintypes
        using com.airwizard.types

        namespace shared {
            struct aaa {
                x : FlightAIDI
                y : Flights
            }

            typedef String FlightID

            struct FlightStatus {
                f : FlightStatus
                id:
                eta:
            }

            interfac
                onew                                    tatus[]]
            }
```

FlightID -- com.airwizard.shared
FlightStatus -- com.airwizard.shared
Flights -- com.airwizard.shared
String -- com.airwizard.types
Time -- com.airwizard.domaintypes
aaa -- com.airwizard.shared
boolean -- com.airwizard.types
int -- com.airwizard.types

Task | Outlin

<unnamed>
<unnamed>
<unnamed>
com
airwizard
<unnamed>
<unnamed>
diagnostic
mobile
server
serverstuff
shared
aaa
FlightID
Flights
flights: Flights
FlightStatus
eta: Time
f: null
id: FlightID
IFlightDataConsu
IFlightReporter

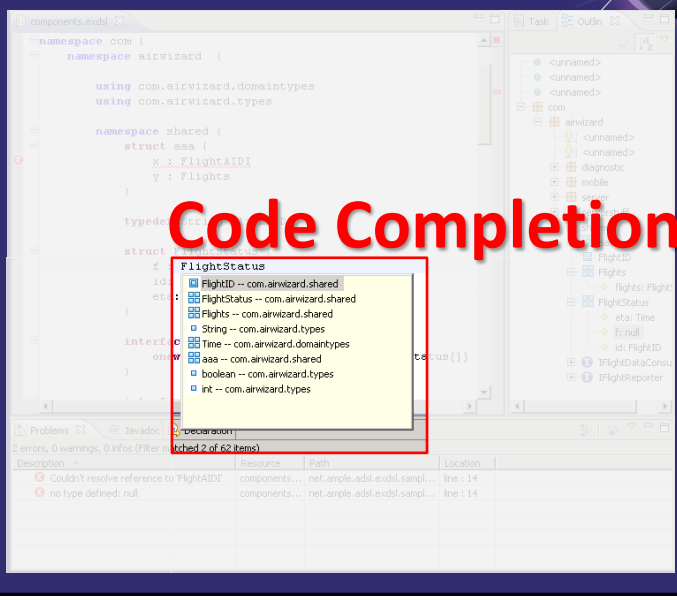
Problems | Javadoc | Declaration

2 errors, 0 warnings, 0 infos (Filter matched 2 of 62 items)

Description	Resource	Path	Location
Couldn't resolve reference to 'FlightAIDI'	components...	net.ample.adsl.exdsl.sampl...	line : 14
no type defined: null	components...	net.ample.adsl.exdsl.sampl...	line : 14

Xtext: Generated Editor

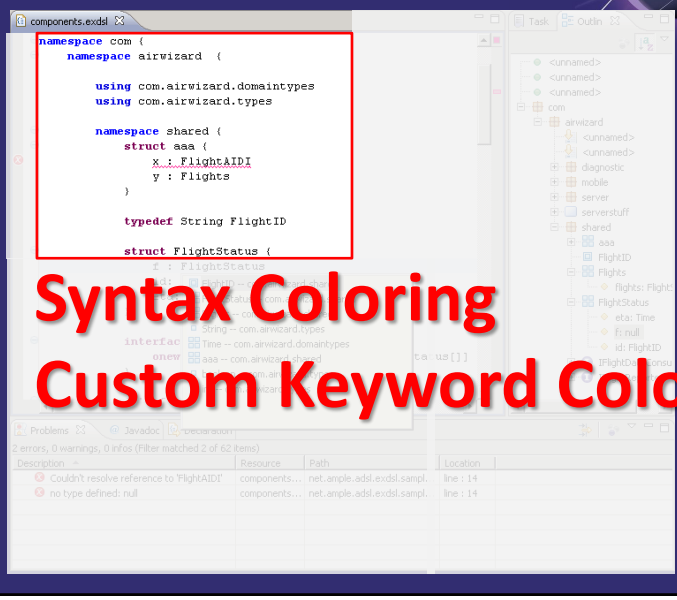
eclipse modeling PROJECT



Code Completion

Xtext: Generated Editor

eclipse modeling PROJECT

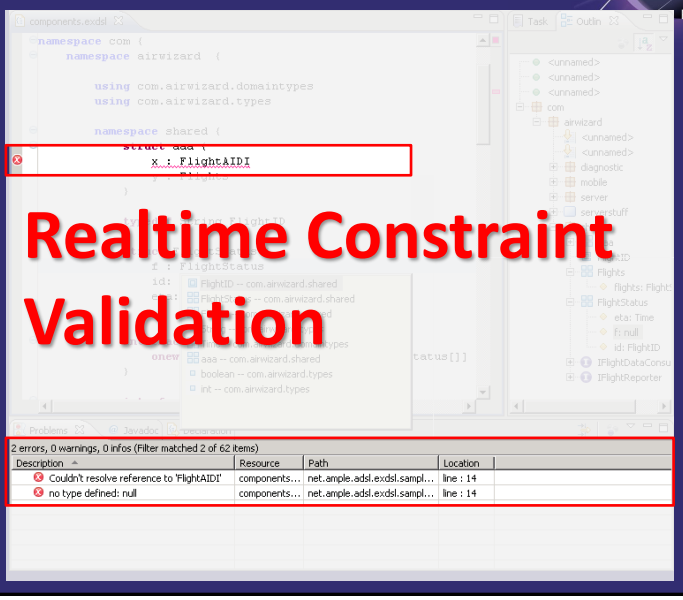


Syntax Coloring

Custom Keyword Coloring

Xtext: Generated Editor

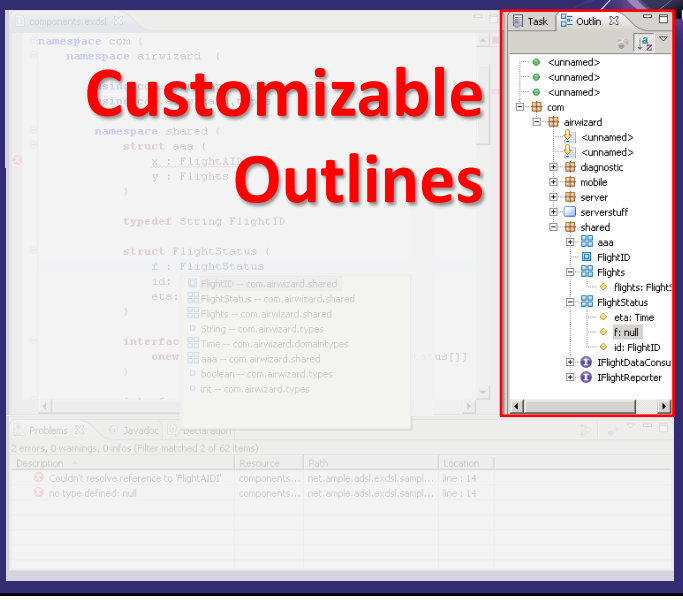
eclipse modeling PROJECT



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

2

Available Tooling

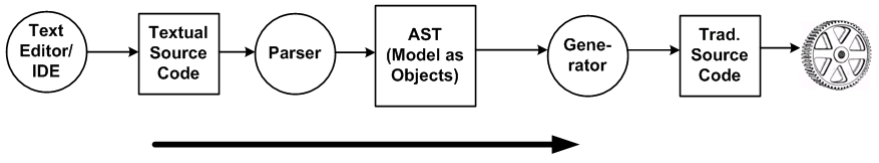
Jetbrains‘ Meta Programming System

Parser-based

text

… to tree

… to text



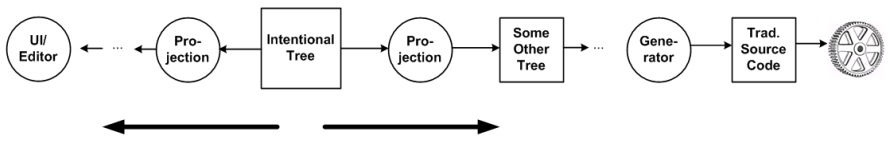
Projectional

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 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

LockStatement

```
concept LockStatement extends    Statement
                      implements <none>

  instance can be root: false

  properties:
  << ... >>

  children:
  StatementList | body          | 1 | specializes: | <none>
  Expression    | lockExpression | 1 | specializes: | <none>

  references:
  << ... >>

  concept properties:
  alias = lock

  concept links:
  << ... >>

  concept property declarations:
  << ... >>

  concept link declarations:
  << ... >>
```

Structure | Editor | Constraints | Behavior | Typesystem | Intentions | Find Usages | Data Flow | Generator

Structure ◆ Editor ◆ Typesystem ◆ Generator

LockStatement

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

  inspected cell layout:
    <choose cell model>
```

Structure | Editor | Constraints | Behavior | Typesystem | Intentions | Find Usages | Data Flow | Generator

Structure ◆ Editor ◆ Typesystem ◆ Generator

```
LockStatement

editor for concept LockStatement
  node cell layout:
    [/
      [> lock ( % lockExpression % ) { <]
      [> ---> % ]
      }
    /]

  inspected cell layout:
    <choose cell model>

body                 linkDeclaration (jaxdemo.structure.LockStatement)
lockExpression       linkDeclaration (jaxdemo.structure.LockStatement)

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@j)
LockSupport   ^class (j.u.concurrent.locks@j)

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 {
            $COPY_SRC$[l].lock();
            $LOOP$[$COPY_SRC$[null:]]
         } finally {
            $COPY_SRC$[l].unlock();
         } ] TF>
   }

   <<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>
            $COPY_SRC$[l].lock();
            $LOOP$[$COPY_SRC$[null:]]
        } finally {
            $COPY_SRC$[l].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>
            $COPY_SRC$[l].lock();
            $LOOP$[$COPY_SRC$[null:]]
        } finally {
            $COPY_SRC$[l].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 ( 1 ) {
         SharedResouce.instance().doSomething();
      }
   }

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

   <<static methods>>

   <<static inner classifiers>>
}
```

Typesystem error: int is not a subtype of Lock

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;  }

}
```

3 A vision for Programming

Programming Languages are not **MODULAR** enough.

Programming Languages are not **COMPOSABLE** enough.

Programming Languages are not **CONFIGURABLE** enough.

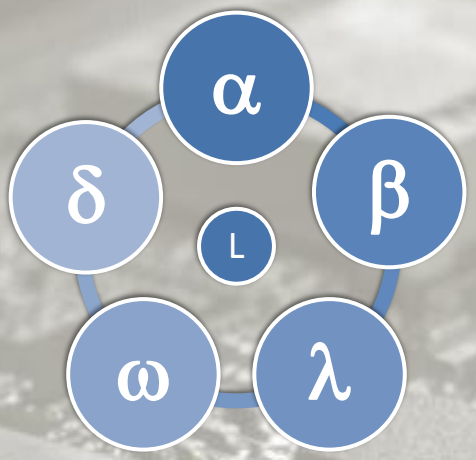
Programming Languages are not **ADAPTABLE** enough.

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

Big Language?

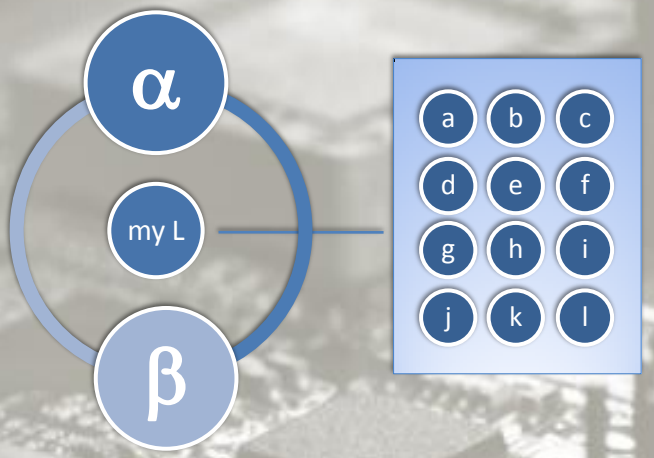
with many first class concepts!

Small Language?



with a few, orthogonal
and poweful concepts

Modular Language



with many optional,
composable concepts

Modular Language

Like frameworks
and libraries,

Modular Language

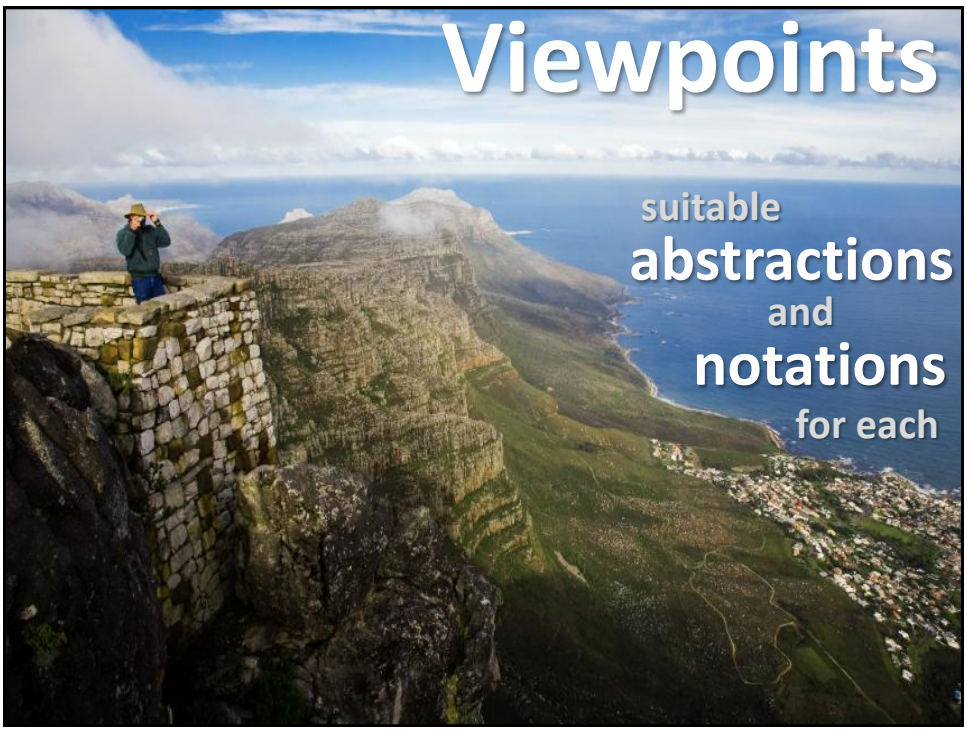
Like frameworks
and libraries,

but with syntax
and IDE support

Not a new idea…


Growing A Language
(Guy L Steele)


Viewpoints


Viewpoints
suitable
abstractions
and
notations
for each


Viewpoints
Integrated
via
symbolic
references
and
seamless
transitions


Viewpoints
Business
custom
purpose-built
create/include

Viewpoints
Business

Custom
Notations

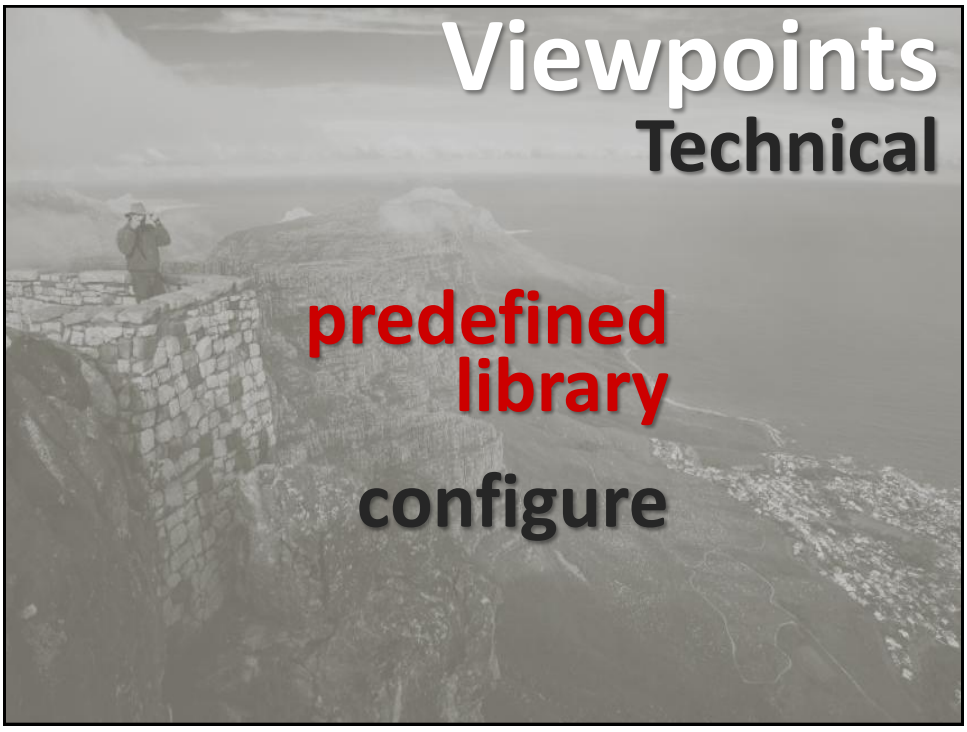
real
business
expert integration

Viewpoints
Business

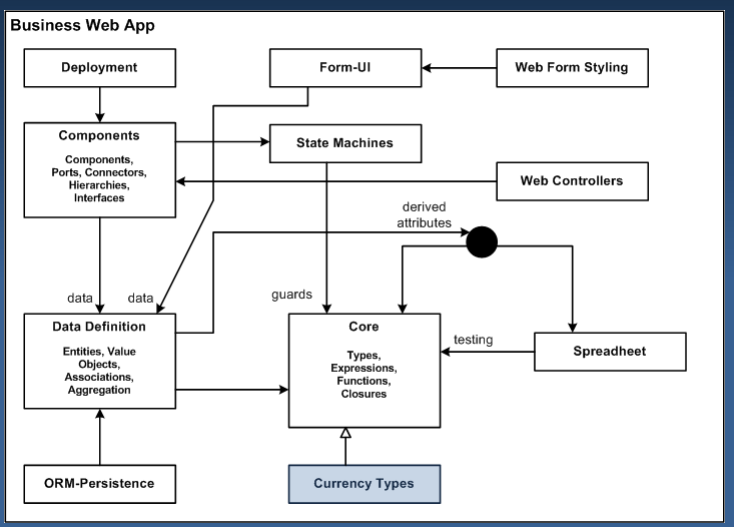
Custom
Notations

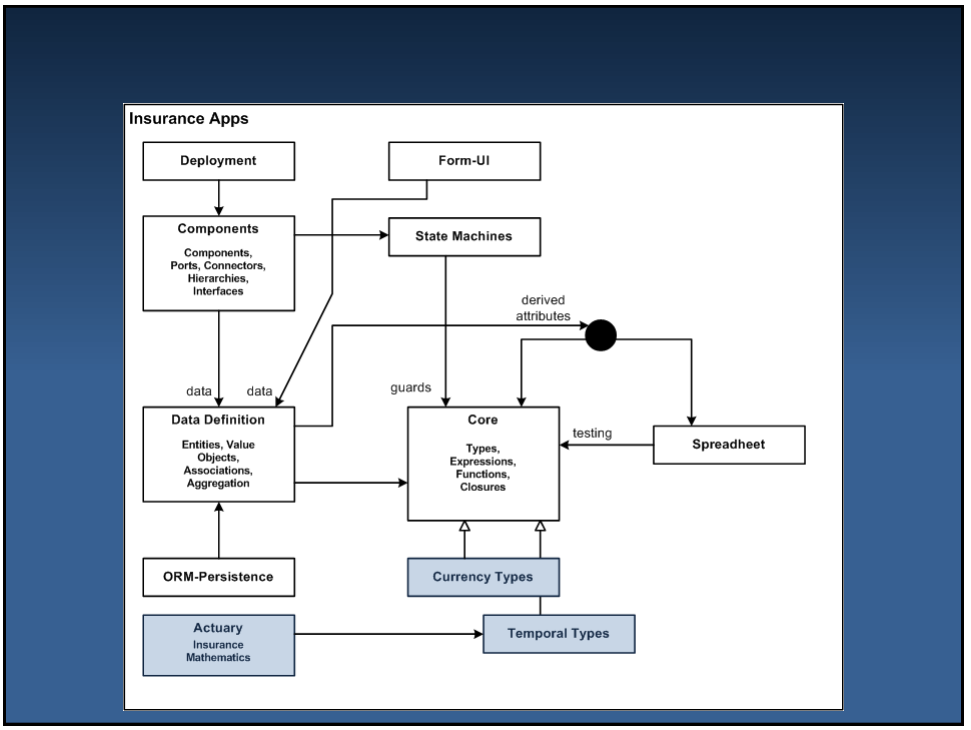
real
business
expert integration

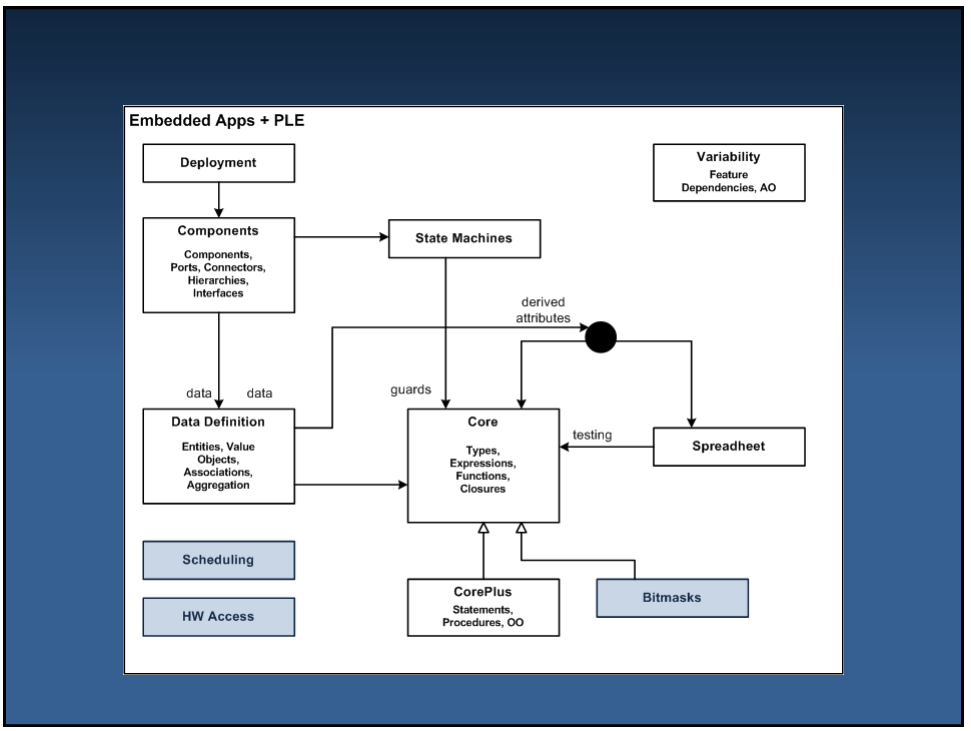
But that‘s another talk…


Viewpoints
Technical
predefined
library
configure

Example Languages




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

(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 = $ModellMatrix * \overrightarrow{modellvektor}$;

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

4 Available Tooling

Intentional Software
Domain Workbench

[redacted]

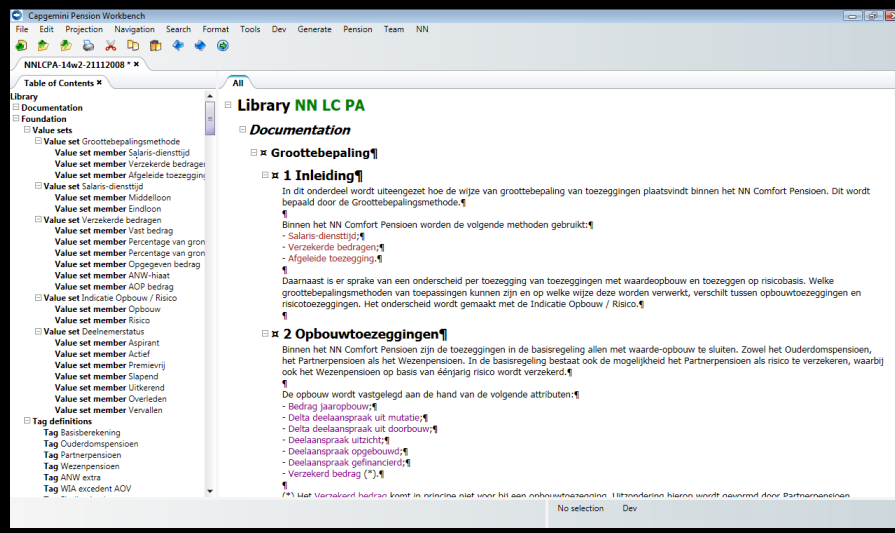
INTENTIONAL®
SOFTWARE

Commercial Product

Eval available

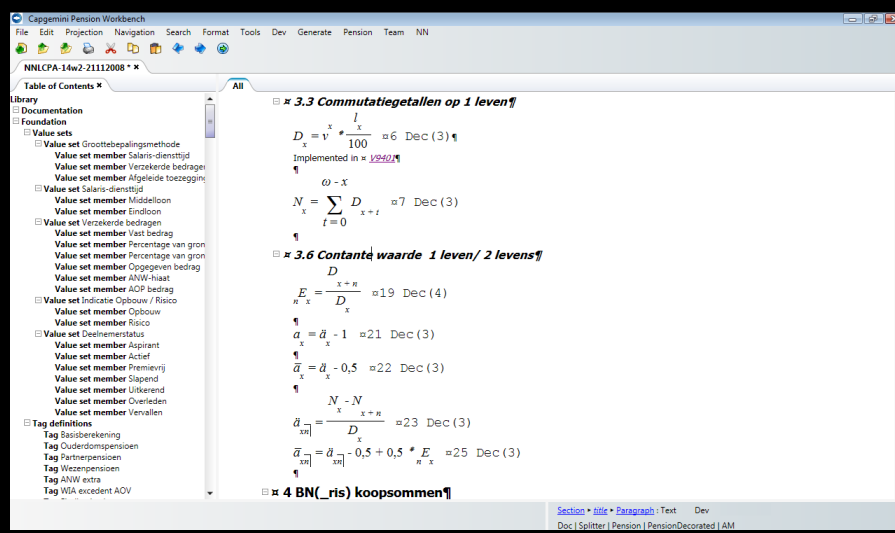
upon request

Pension Workbench Example



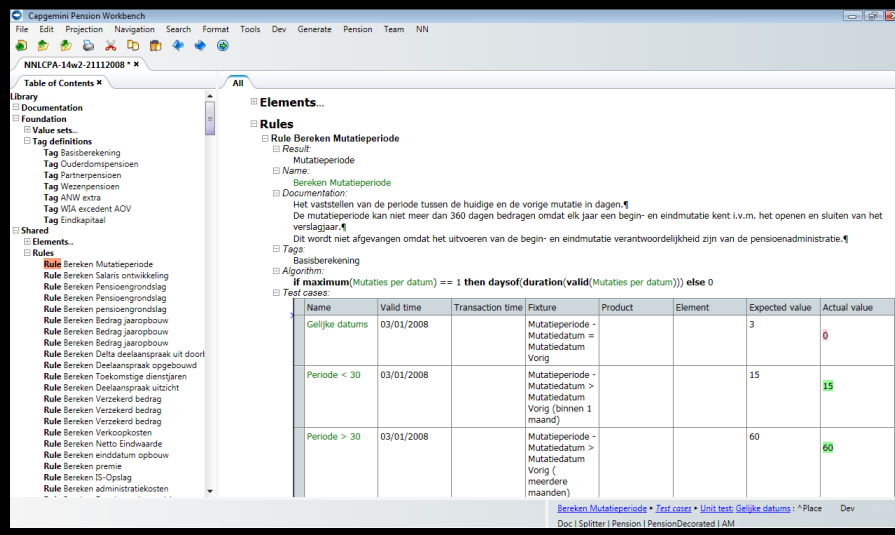
Text Editing Domain

Pension Workbench Example



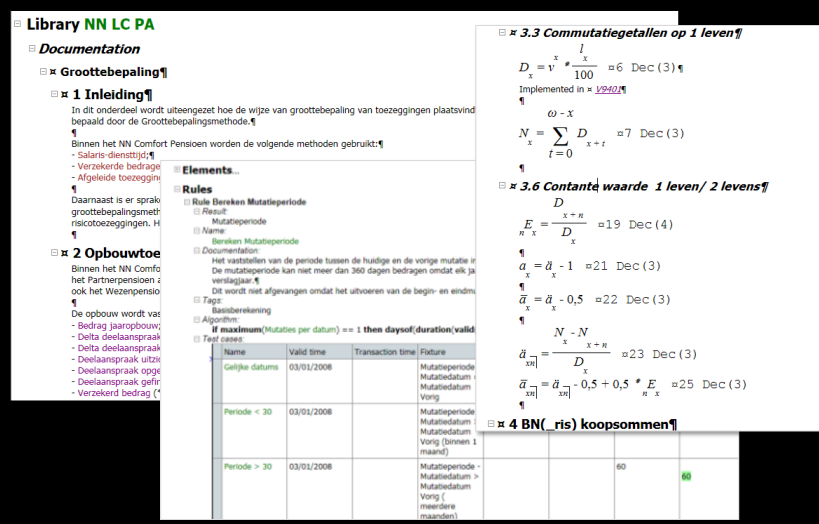
Insurance Mathematics Domain

Pension Workbench Example



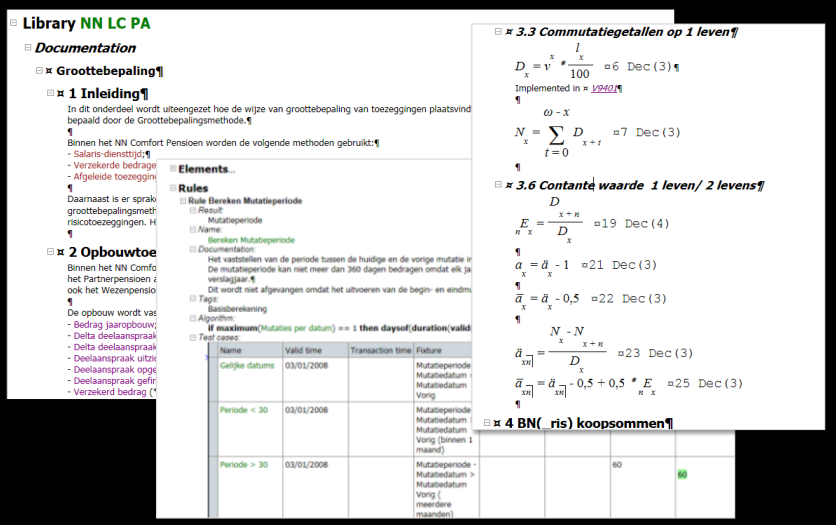
Pension Contract Rules Domain

Pension Workbench Example



All in one Document

Pension Workbench Example

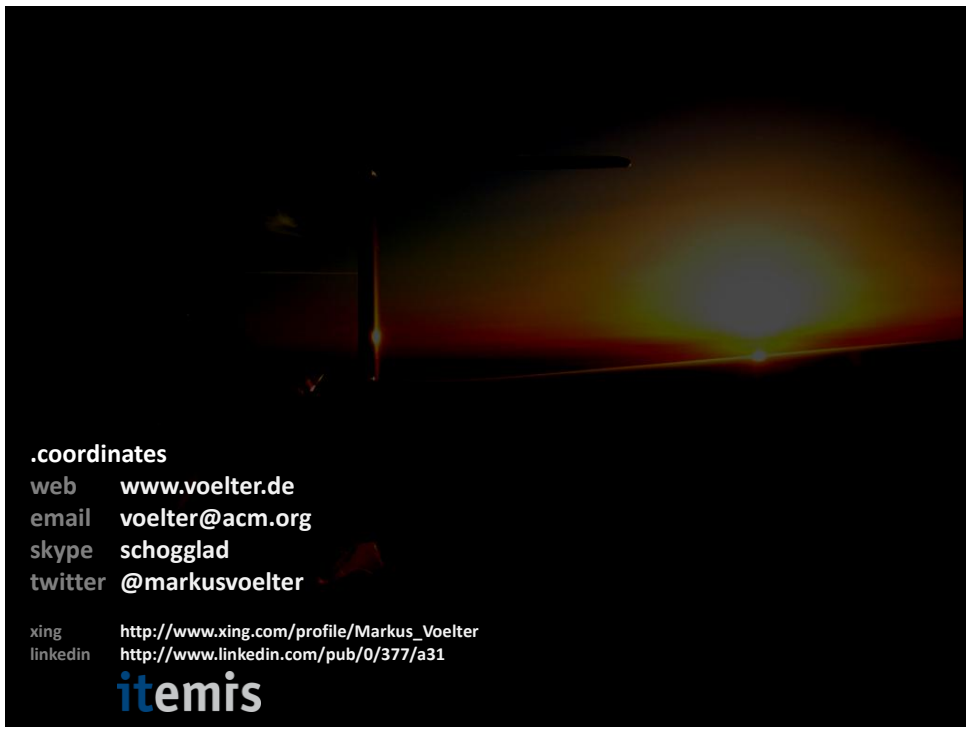


Symbolically integrated


THE END.


THE END.
.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


.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


.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

.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