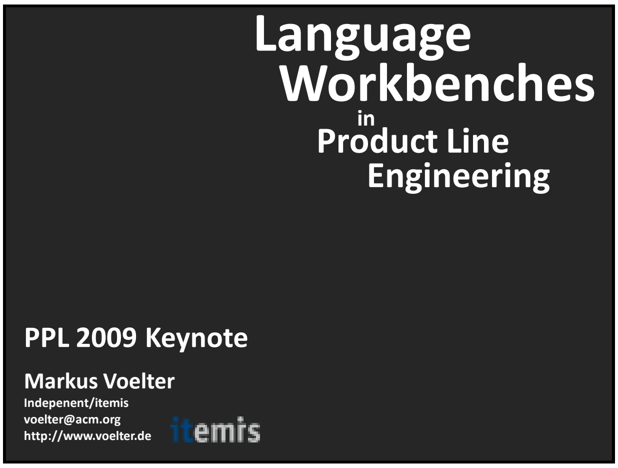
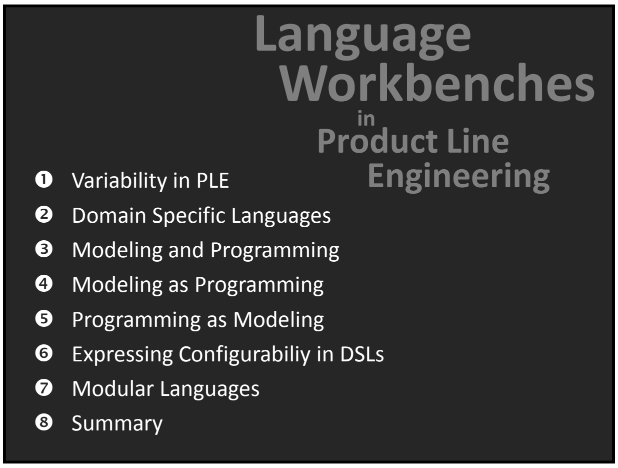


Language Workbenches in Product Line Engineering

PPL 2009 Keynote

Markus Voelter
Indepenent/itemis
voelter@acm.org
http://www.voelter.de

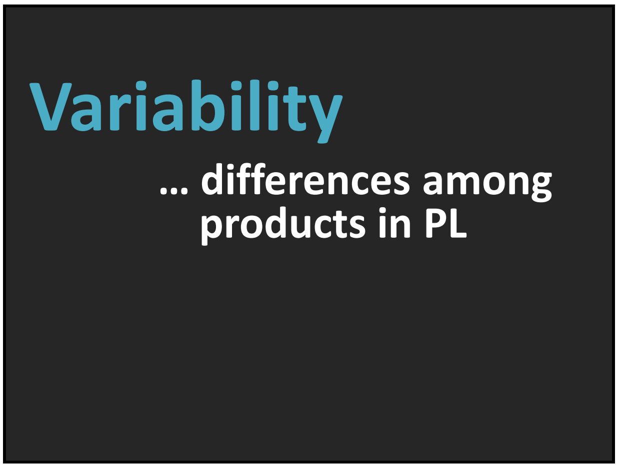
itemis



Language Workbenches in Product Line Engineering

❶ Variability in PLE
❷ Domain Specific Languages
❸ Modeling and Programming
❹ Modeling as Programming
❺ Programming as Modeling
❻ Expressing Configurabiliy in DSLs
❼ Modular Languages
❽ Summary

❶ Variability in PLE

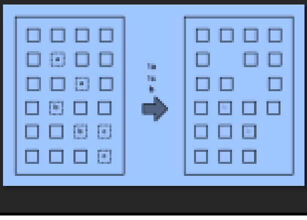


Variability

… differences among products in PL

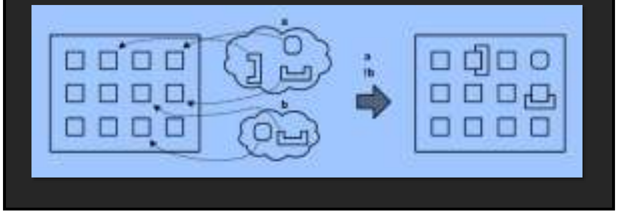
Variability Mechanisms
Removal

… optionally take away from overall whole



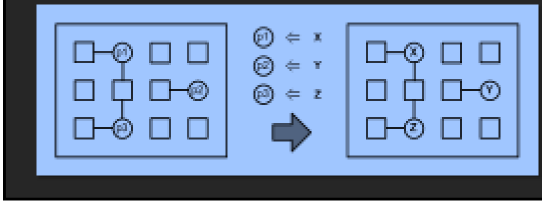
Variability Mechanisms
Injection

… optionally add to

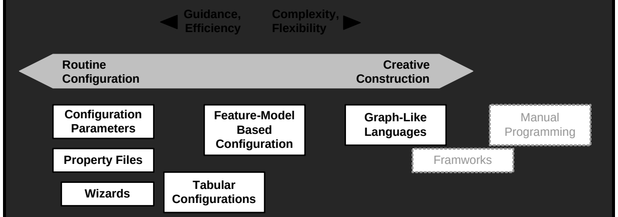


Variability Mechanisms
Parametrization

… define values for predefined params



Configuration vs. Customization
Variability



Guidance, Efficiency

Complexity, Flexibility

Routine Configuration

Creative Construction

Configuration Parameters

Property Files

Wizards

Feature-Model Based Configuration

Tabular Configurations

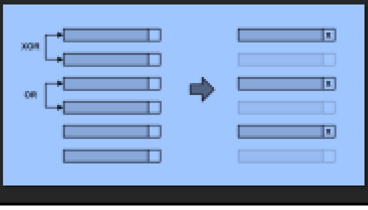
Graph-Like Languages

Manual Programming

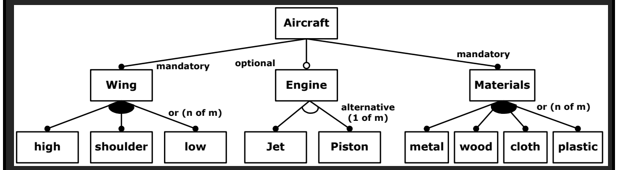
Framworks

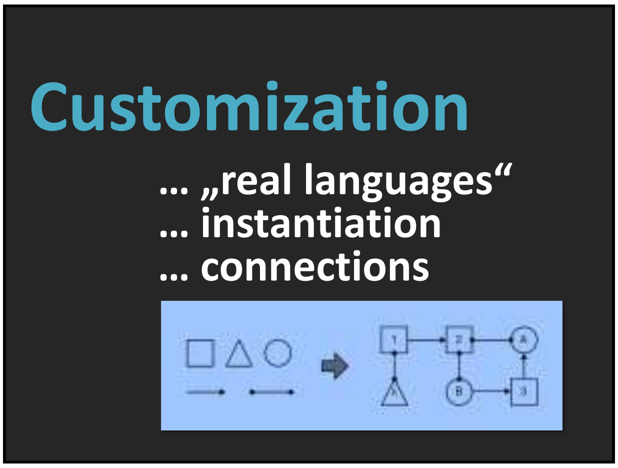
Configuration

… selecting options
… setting param values



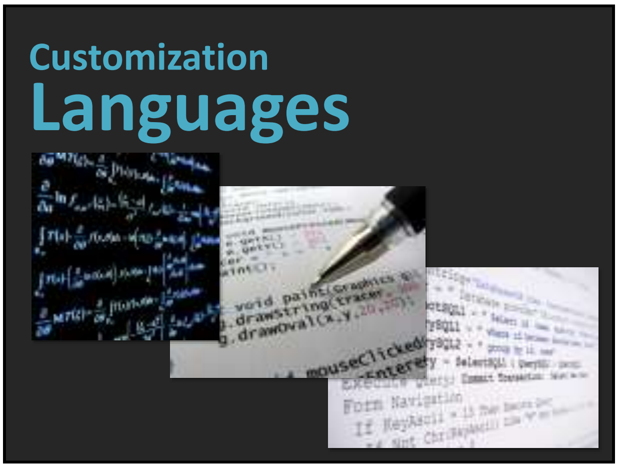
Configuration
Feature Models





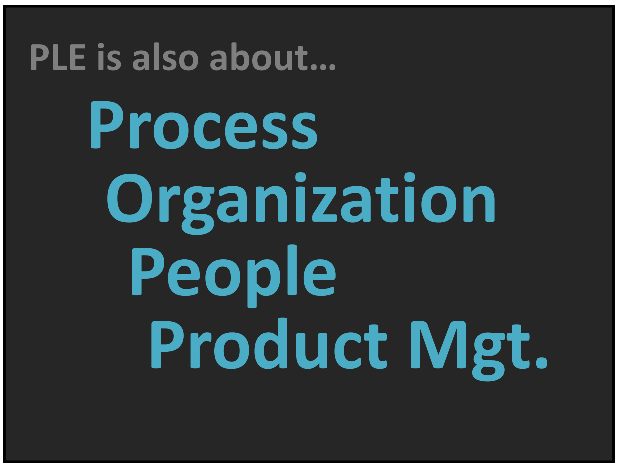
Customization

… „real languages“
… instantiation
… connections



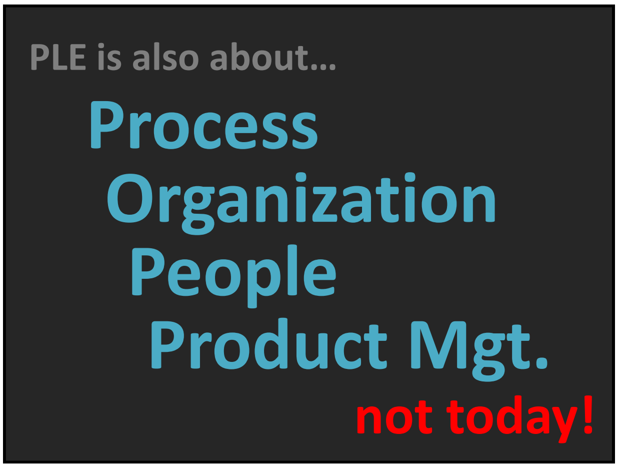
Customization

Languages



PLE is also about…

Process
Organization
People
Product Mgt.

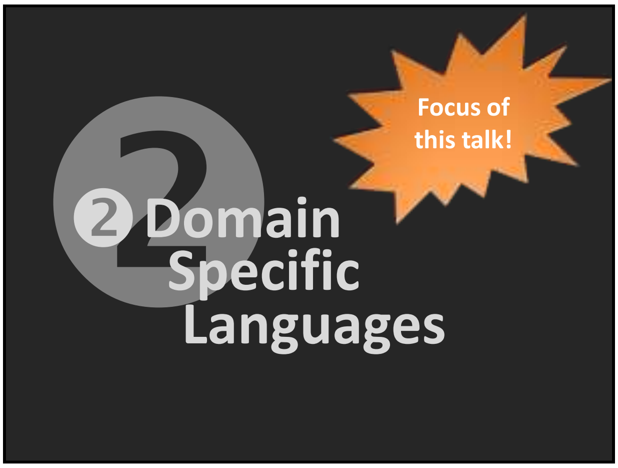


PLE is also about…

Process
Organization
People
Product Mgt.

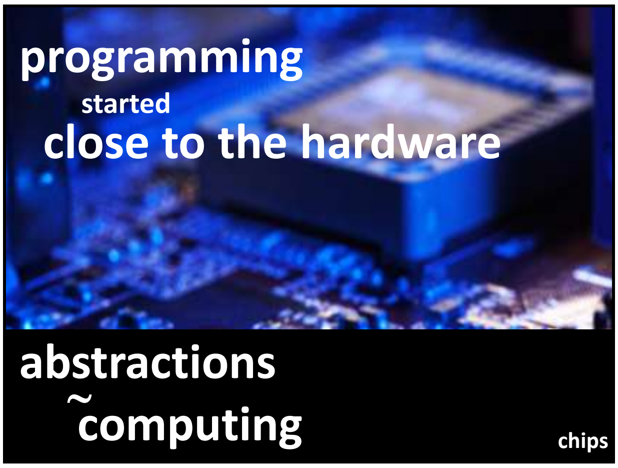
not today!

② Domain
Specific
Languages



② Domain
Specific
Languages

Focus of
this talk!



programming
started
close to the hardware

abstractions
~computing

chips

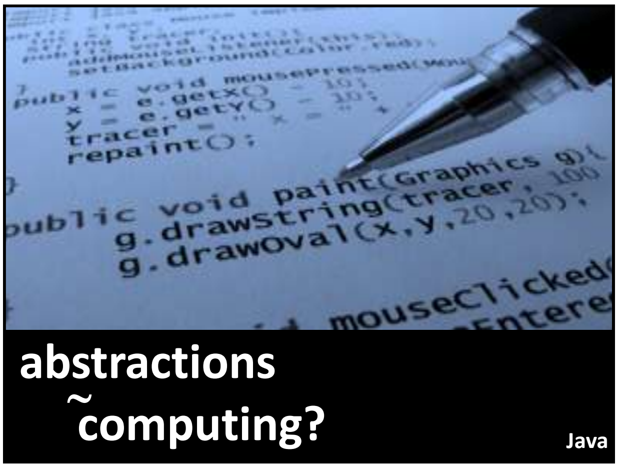
abstractions
~computing

bits

```
decodeMessage(
0; i < MAX_RES
i = 0;
s.length) {
1) buf[loc
ES_LEN)
rrCode
urn null;
extractMessage(res);
public int[] extractMessage(int[] res) {
for (int i = 0; i < MAX_RES_LEN; i++) buf[i] = 0;
int loc = 0, i = 0;
while (i < res.length) {
```

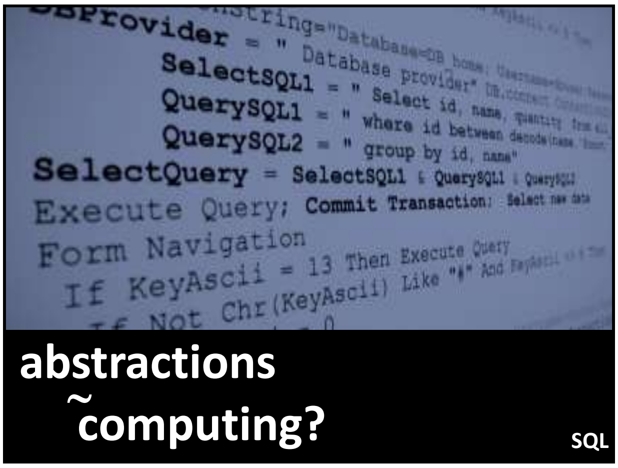
abstractions
~computing

C



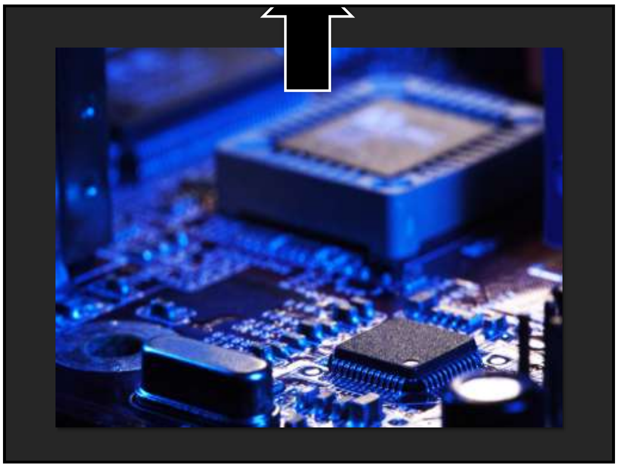
abstractions
~computing?

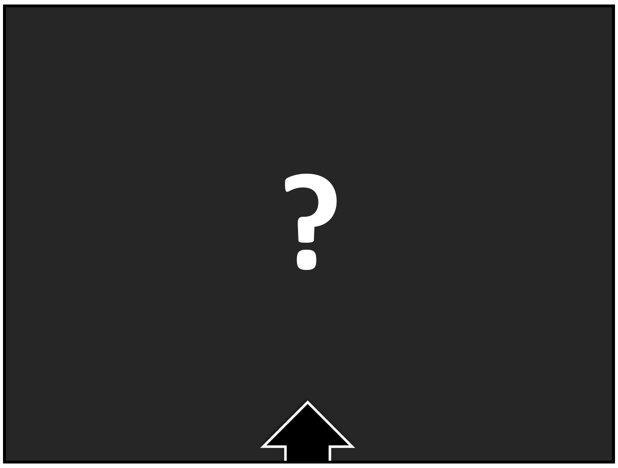
Java


DBProvider = " Database provider"
SelectSQL1 = " Select id, name, quantity
QuerySQL1 = " where id between
QuerySQL2 = " group by id, name"
SelectQuery = SelectSQL1 & QuerySQL1 & QuerySQL2
Execute Query; Commit Transaction; Select new data
Form Navigation
If KeyAscii = 13 Then Execute Query
If Not Chr(KeyAscii) Like "#" And KeyAscii

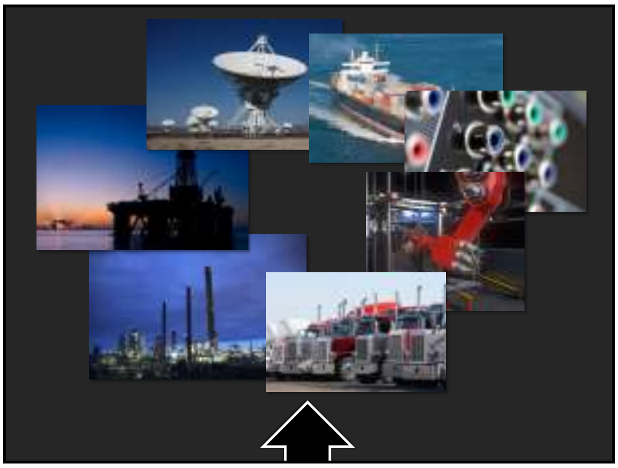
abstractions
~computing?

SQL

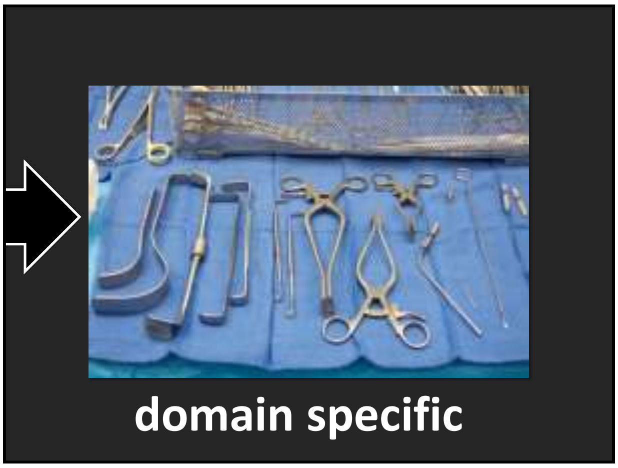




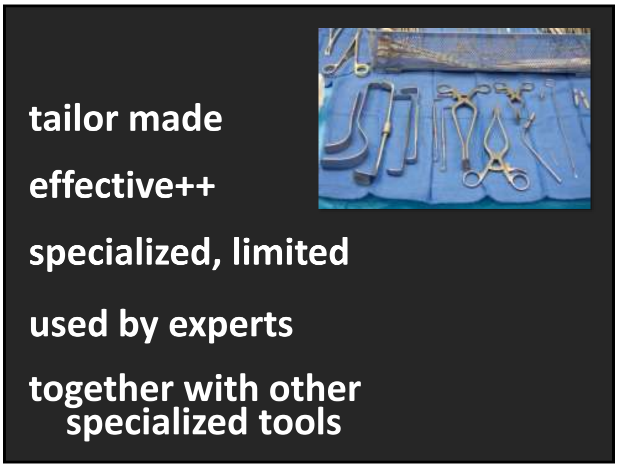
?



general purpose



domain specific



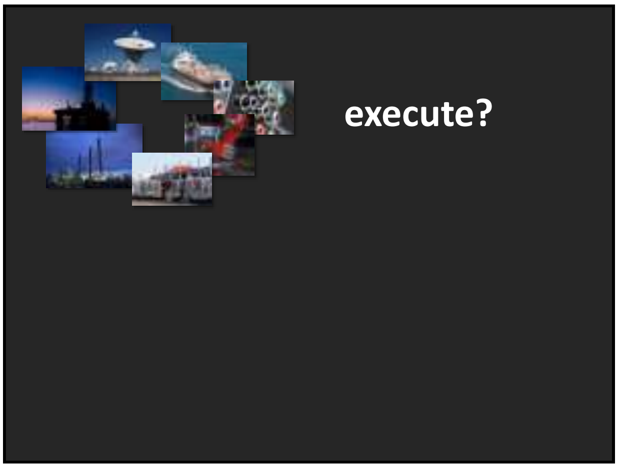
tailor made

effective++

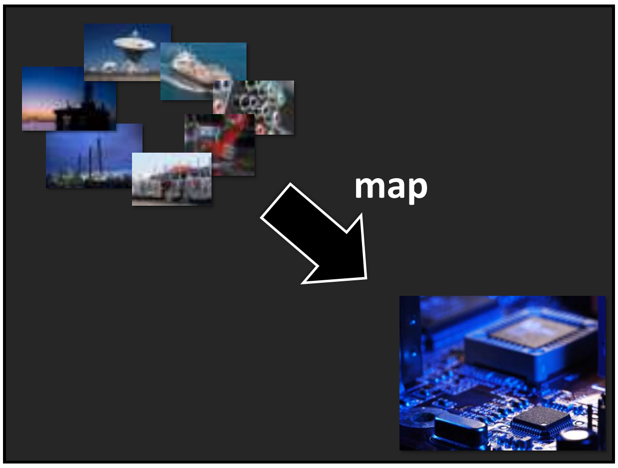
specialized, limited

used by experts

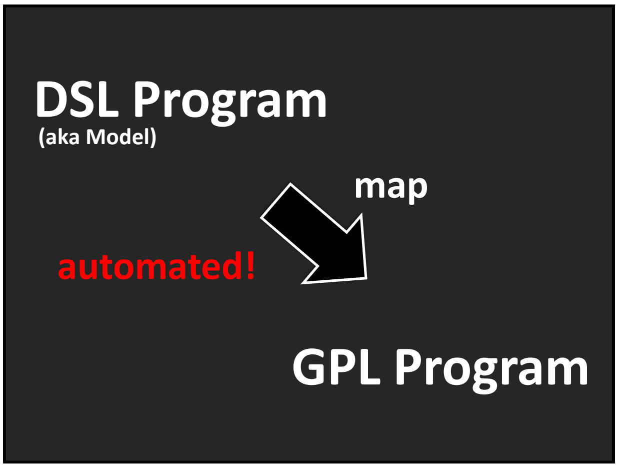
together with other specialized tools



execute?



map

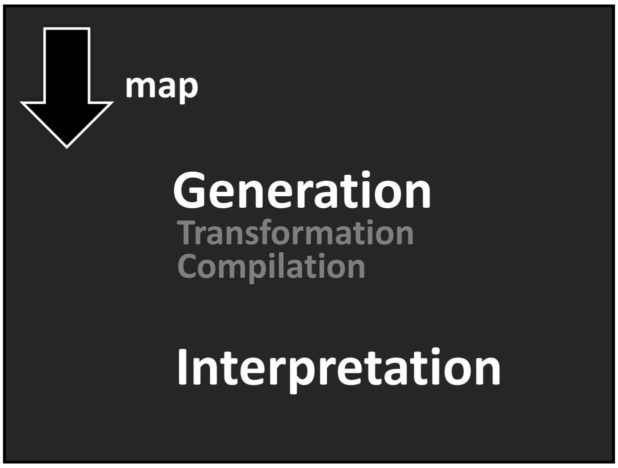


DSL Program
(aka Model)

map

automated!

GPL Program



map

Generation
Transformation
Compilation

Interpretation

Platform Architecture

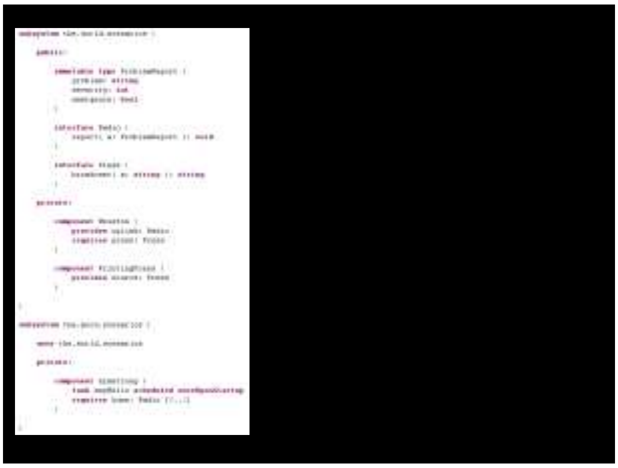
DSL

As you understand and develop your Platform Architecture…

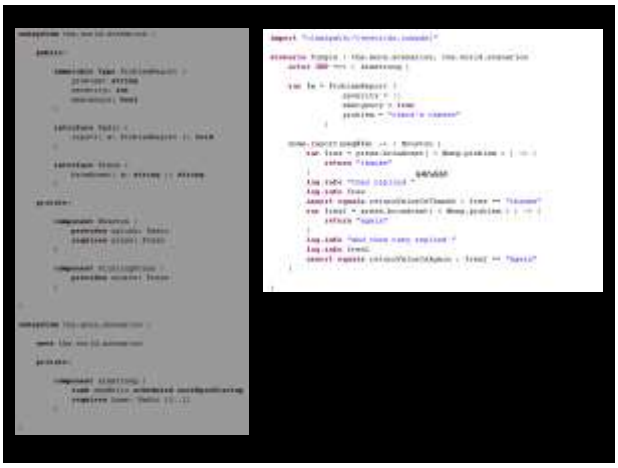
Develop a language to express it!

Language resembles architectural concepts…

Express the applications with the language.



[illegible]

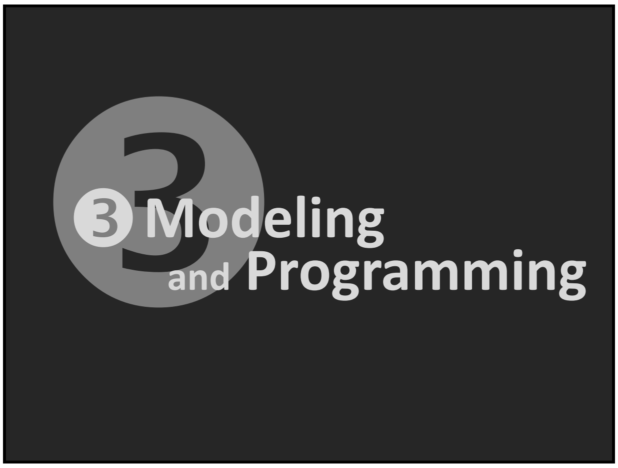


[illegible]

[illegible]

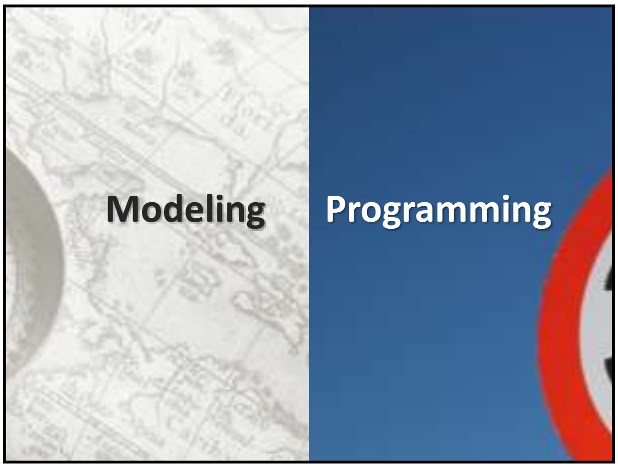
Another Story…

Architecture As Language



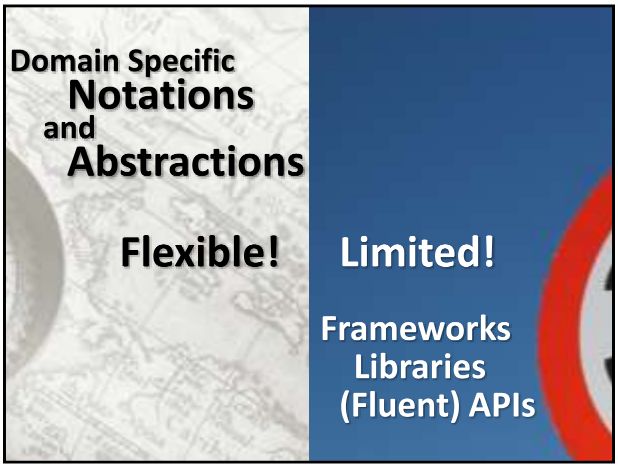
3 Modeling
and Programming

Compare



Modeling

Programming

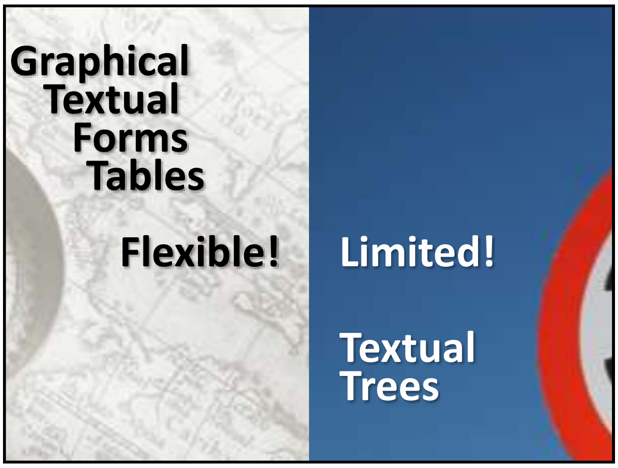


Domain Specific
Notations
and
Abstractions

Flexible!

Limited!

Frameworks
Libraries
(Fluent) APIs

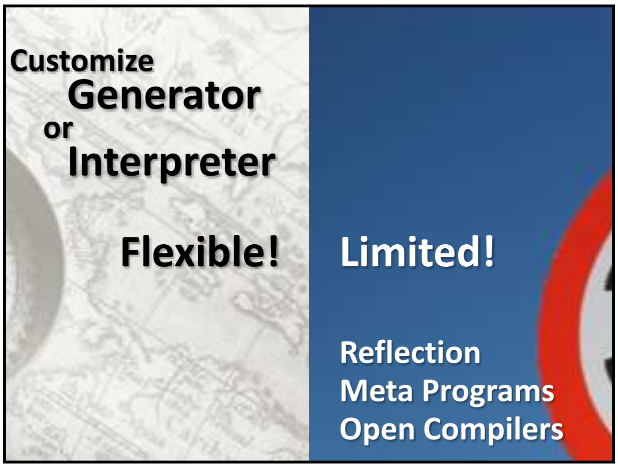


Graphical
Textual
Forms
Tables

Flexible!

Limited!

Textual
Trees

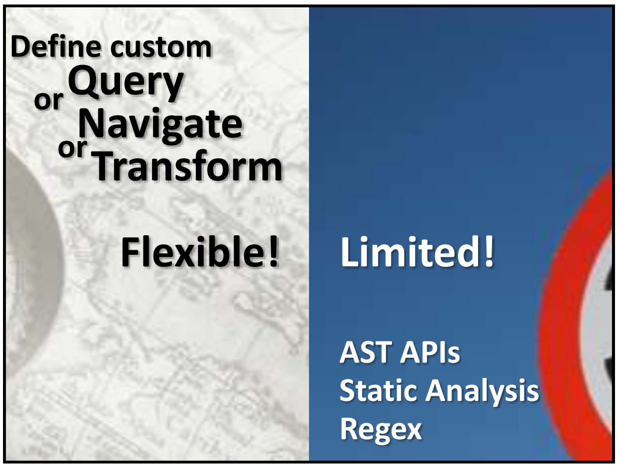


Customize
Generator
or
Interpreter

Flexible!

Limited!

Reflection
Meta Programs
Open Compilers

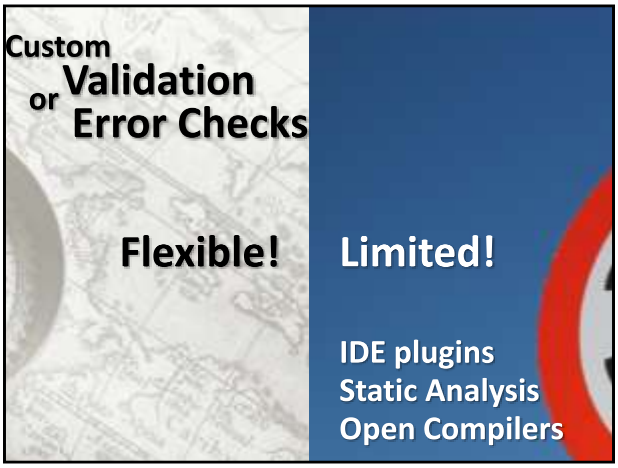


Define custom
Query
or Navigate
or Transform

Flexible!

Limited!

AST APIs
Static Analysis
Regex

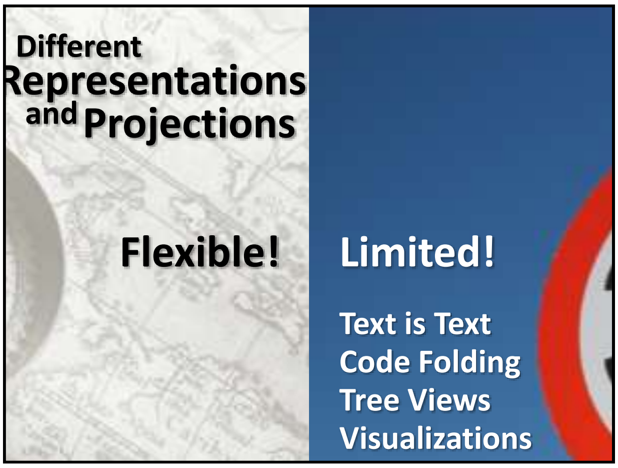


Custom
Validation
or Error Checks

Flexible!

Limited!

IDE plugins
Static Analysis
Open Compilers

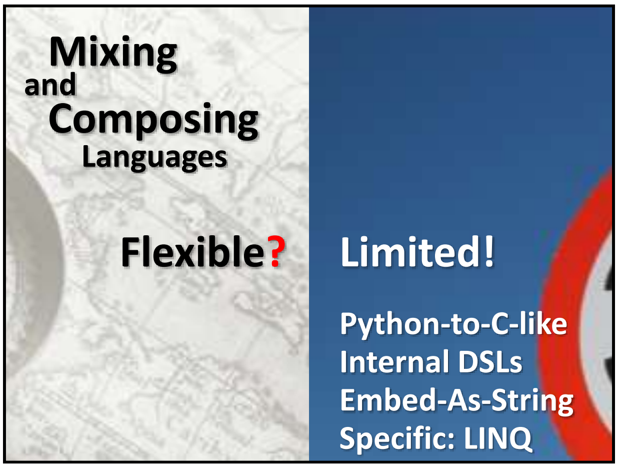


Different
Representations
and Projections

Flexible!

Limited!

Text is Text
Code Folding
Tree Views
Visualizations

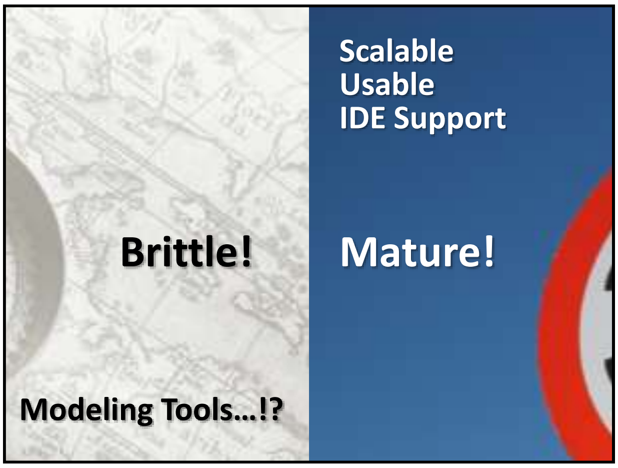


Mixing
and
Composing
Languages

Flexible?

Limited!

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

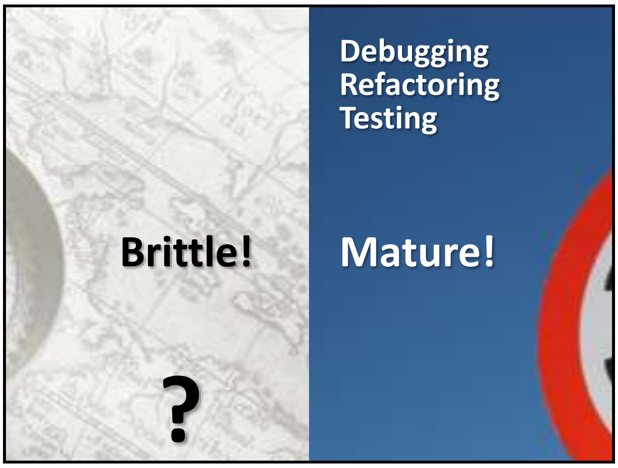


Scalable
Usable
IDE Support

Brittle!

Mature!

Modeling Tools...!?

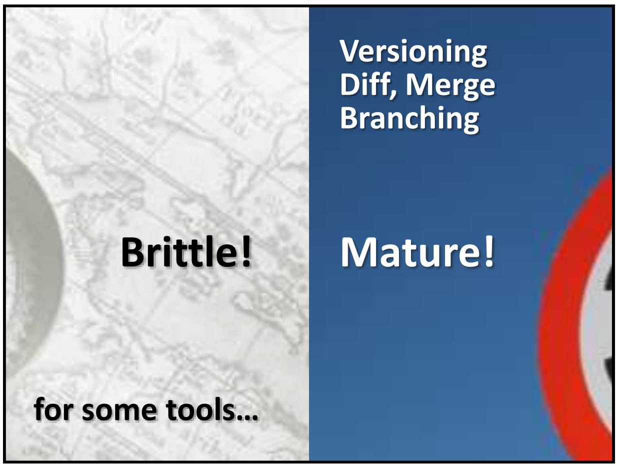


Debugging
Refactoring
Testing

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...

Different Worlds

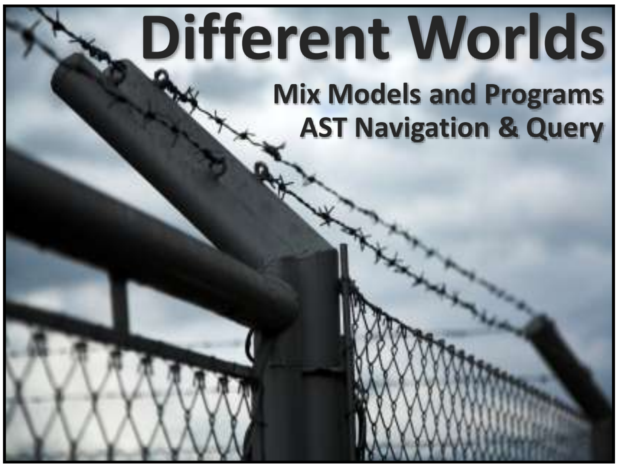
Programming Tools
!=
Modeling Tools

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

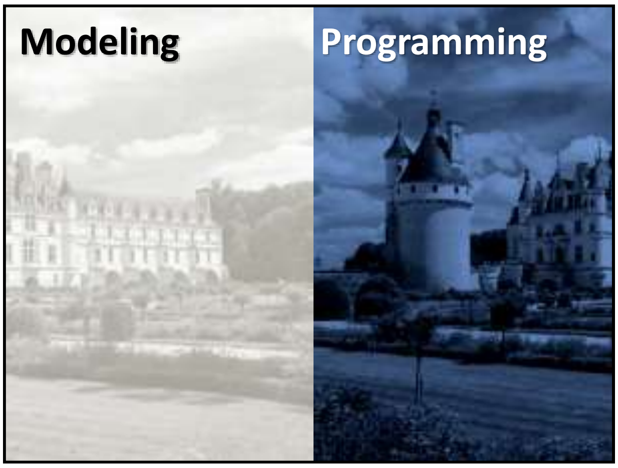
Different Worlds

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

Why

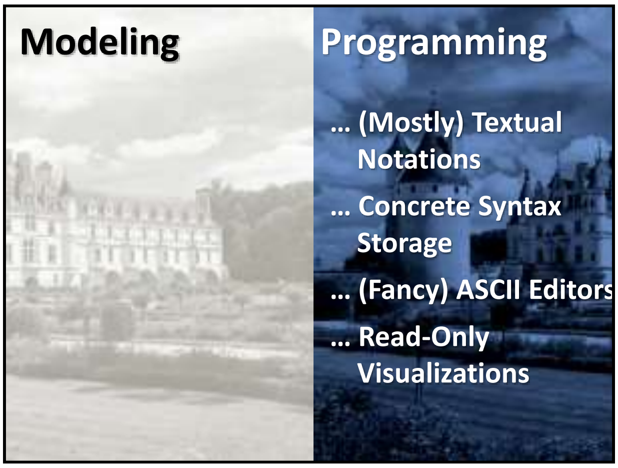
the difference?

History?



Modeling

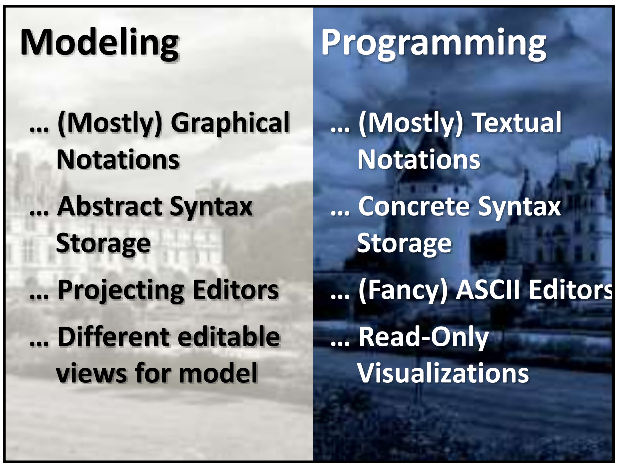
Programming



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

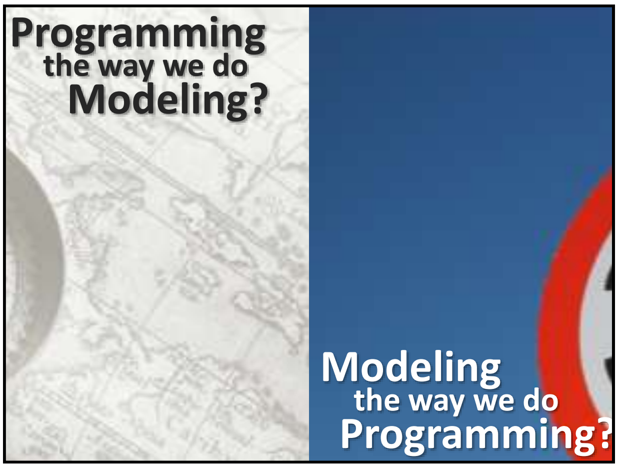
- … (Mostly) Textual Notations
- … Concrete Syntax Storage
- … (Fancy) ASCII Editors
- … Read-Only Visualizations

Why

the difference?

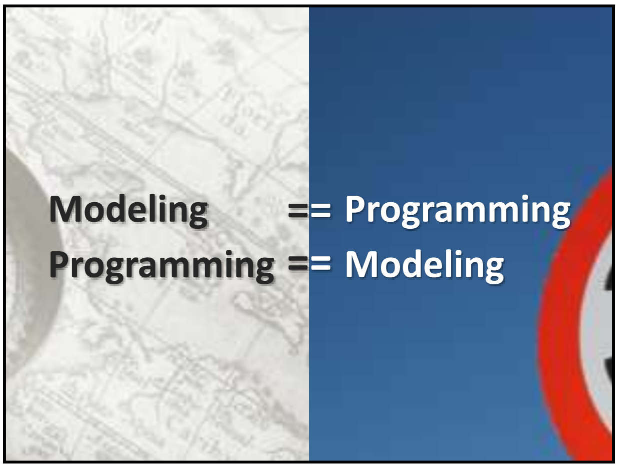
It is time for …

… a Different Perspective



Programming
the way we do
Modeling?

Modeling
the way we do
Programming?



Modeling == Programming

Programming == Modeling

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

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

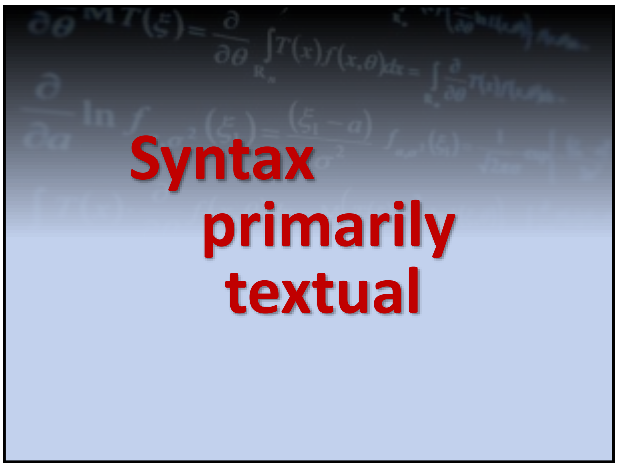
+

language definition implies IDE definition

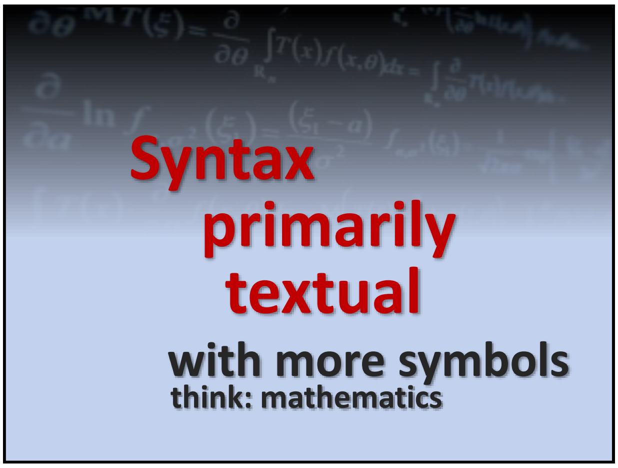
Language Workbench

(Martin Fowler)

support for „classical“ programming and „classical“ modeling



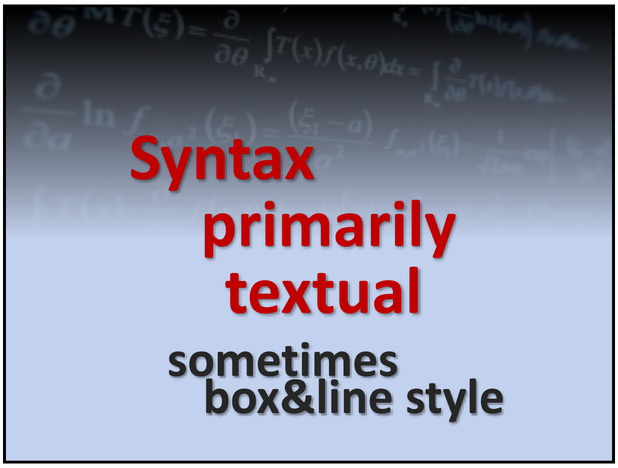
Syntax primarily textual



Syntax primarily textual

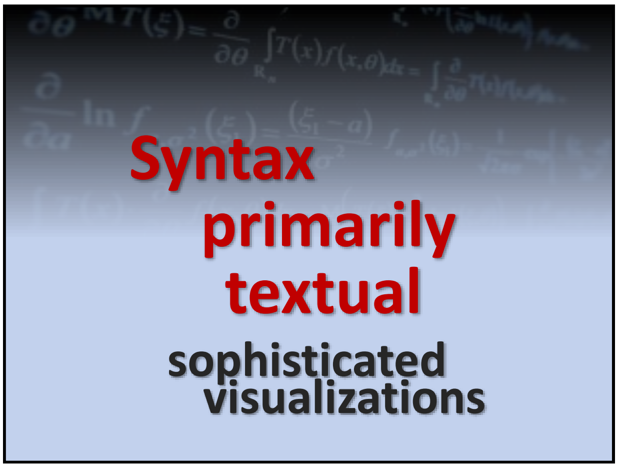
with more symbols

think: mathematics



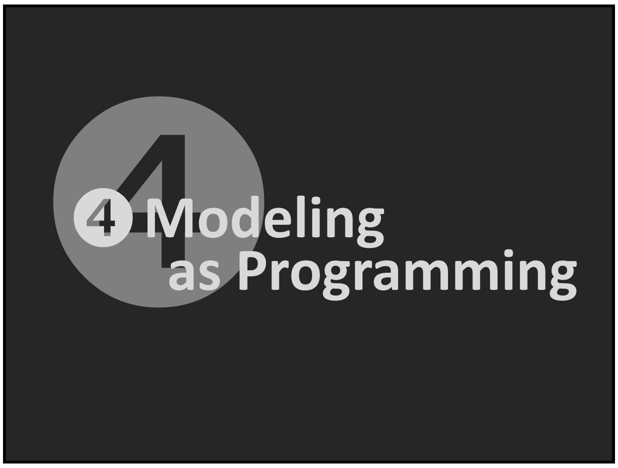
Syntax primarily textual

sometimes box&line style

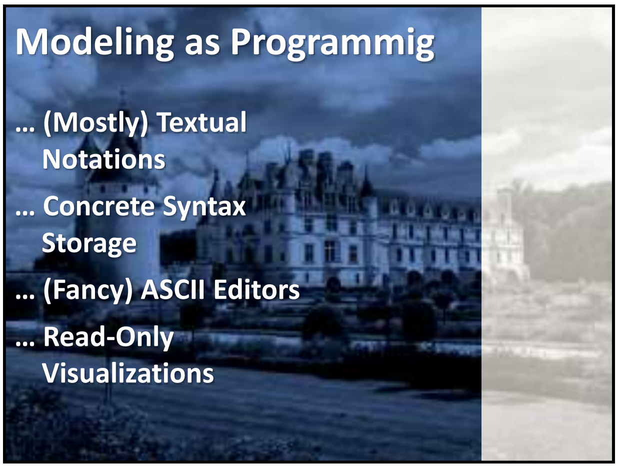


Syntax primarily textual

sophisticated visualizations

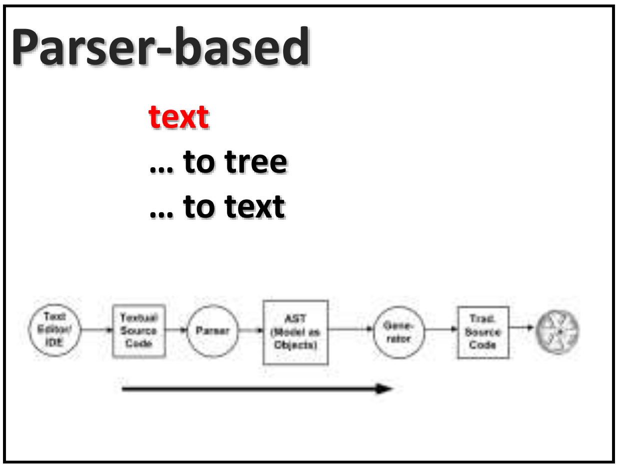


4 Modeling
as Programming



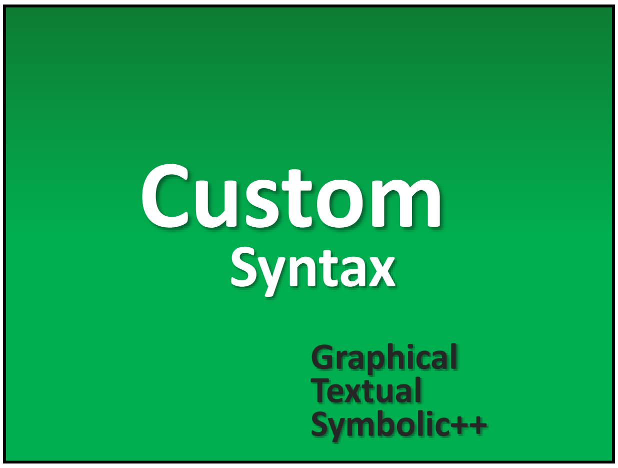
Modeling as Programmig

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



Parser-based

text
… to tree
… to text

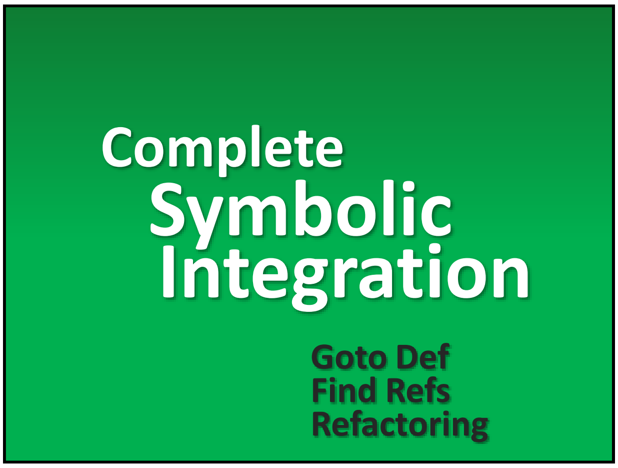


Custom
Syntax

Graphical
Textual
Symbolic++

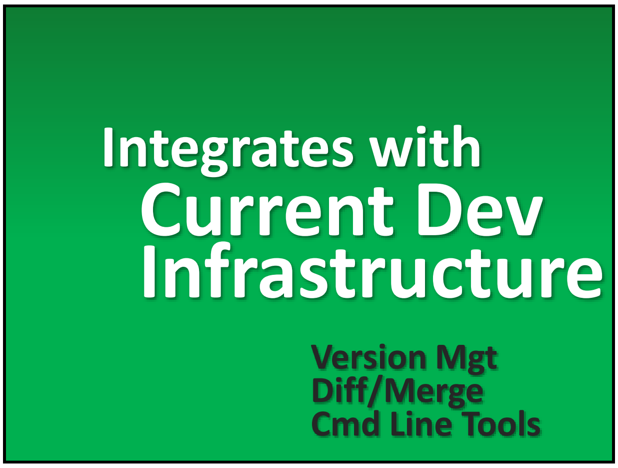
IDE
Support

Teamwork
Debugging
Custom Editors



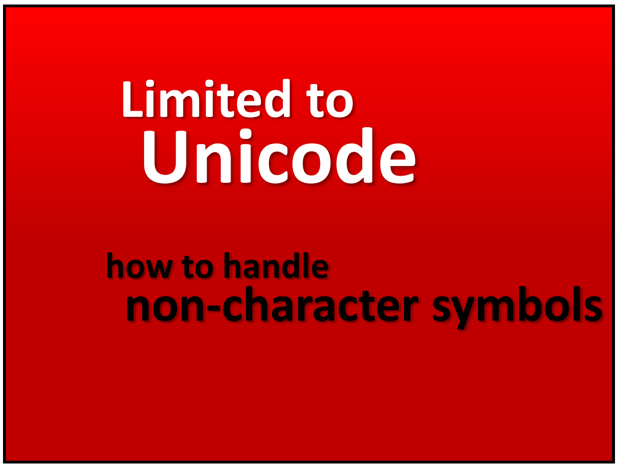
Complete
Symbolic
Integration

Goto Def
Find Refs
Refactoring



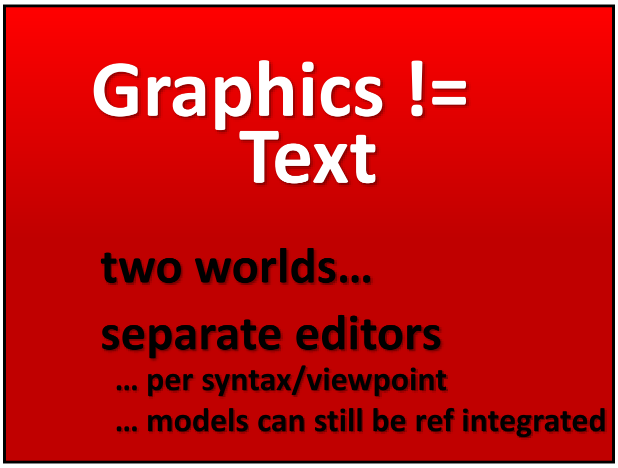
Integrates with
Current Dev
Infrastructure

Version Mgt
Diff/Merge
Cmd Line Tools



Limited to
Unicode

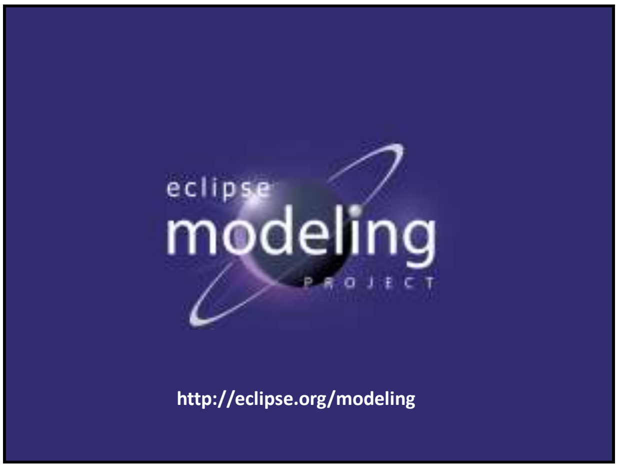
how to handle
non-character symbols



Graphics !=
Text

two worlds…

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



http://eclipse.org/modeling

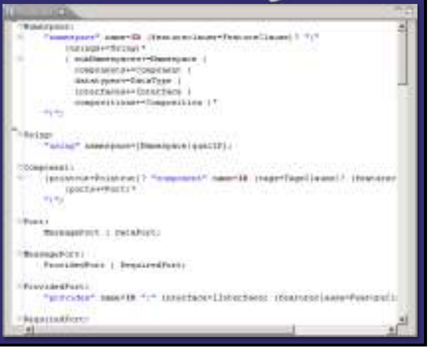
xtext

eclipse modeling PROJECT

http://eclipse.org/xtext

Eclipse Xtext

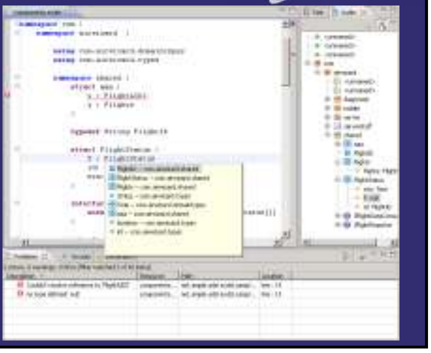
Building Textual Editors



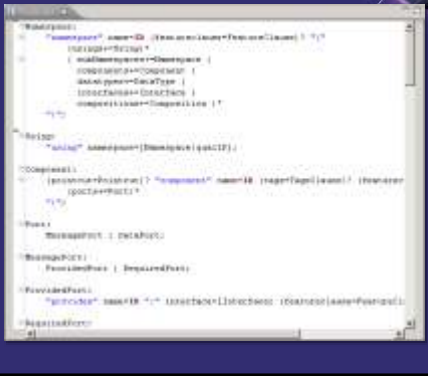
```
Namespace:
    "namespace" [illegible]
    [illegible]
Using:
    "using" [illegible]
Component:
    [illegible]
Port:
    MessagePort | DataPort;
MessagePort:
    ProvidedPort | RequiredPort;
ProvidedPort:
    "provides" [illegible]
RequiredPort:
```

Eclipse Xtext

Building Textual Editors

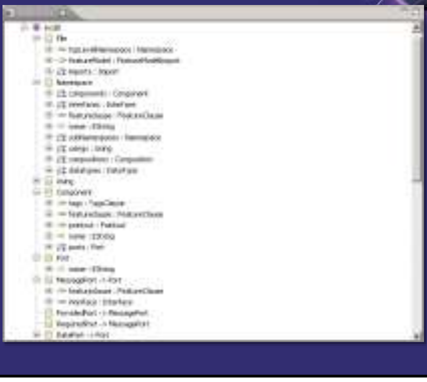


Xtext: Specify Grammar

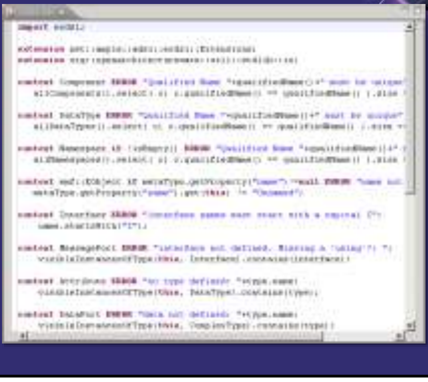


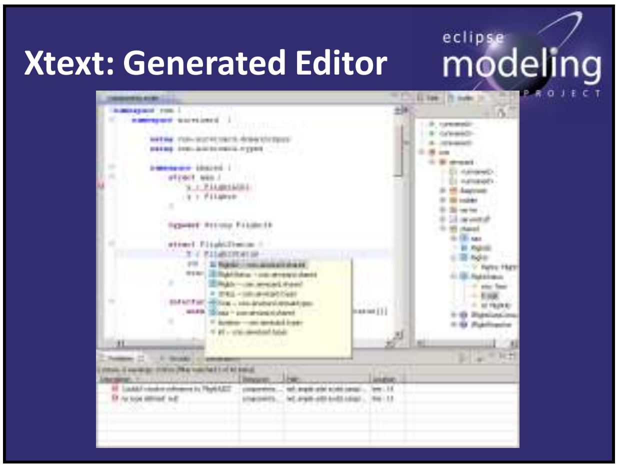
```
Namespace:
    "namespace" [illegible]
    [illegible]
Using:
    "using" [illegible]
Component:
    [illegible]
Port:
    MessagePort | DataPort;
MessagePort:
    ProvidedPort | RequiredPort;
ProvidedPort:
    "provides" [illegible]
RequiredPort:
```

Xtext: Gen. Meta Model

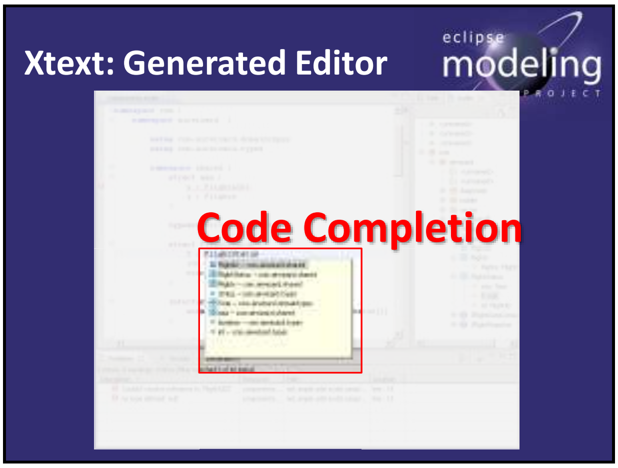


Xtext: Constraints



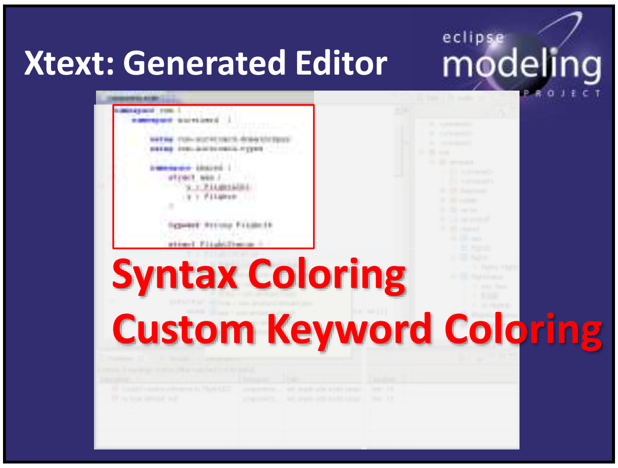


Xtext: Generated Editor



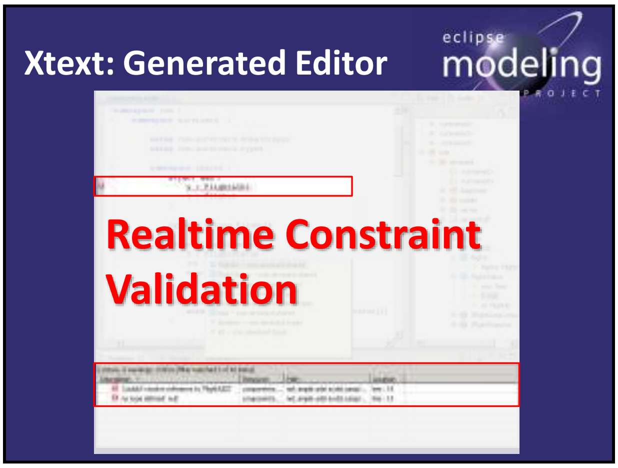
Xtext: Generated Editor

Code Completion



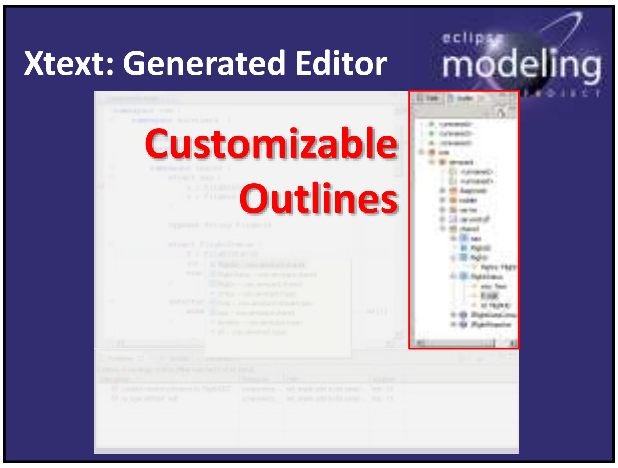
Xtext: Generated Editor

Syntax Coloring
Custom Keyword Coloring



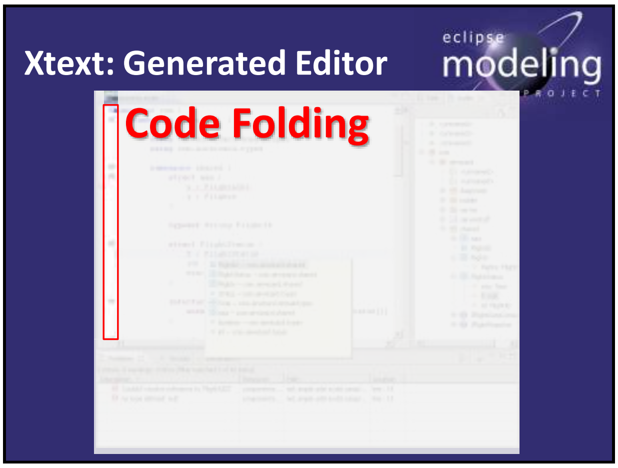
Xtext: Generated Editor

Realtime Constraint Validation



Xtext: Generated Editor

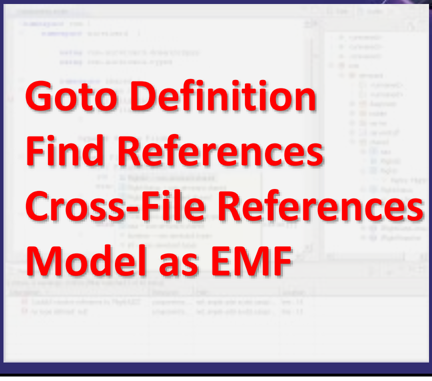
Customizable Outlines



Xtext: Generated Editor

Code Folding

Xtext: Generated Editor



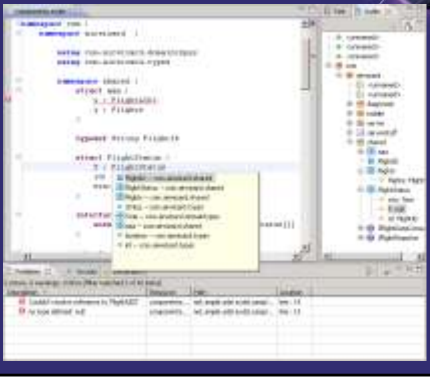
Goto Definition

Find References

Cross-File References

Model as EMF

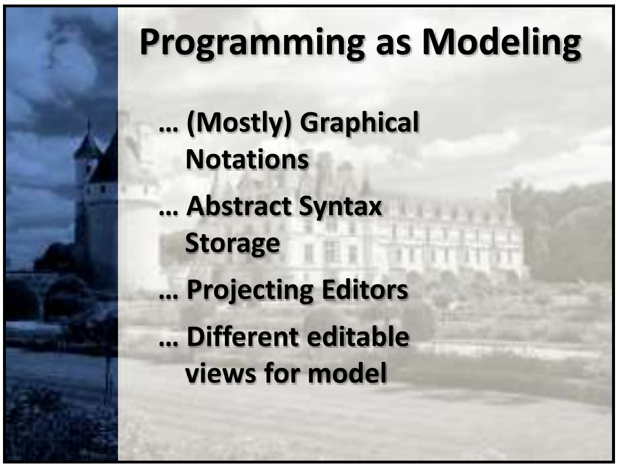
Xtext: Generated Editor



DEMO I

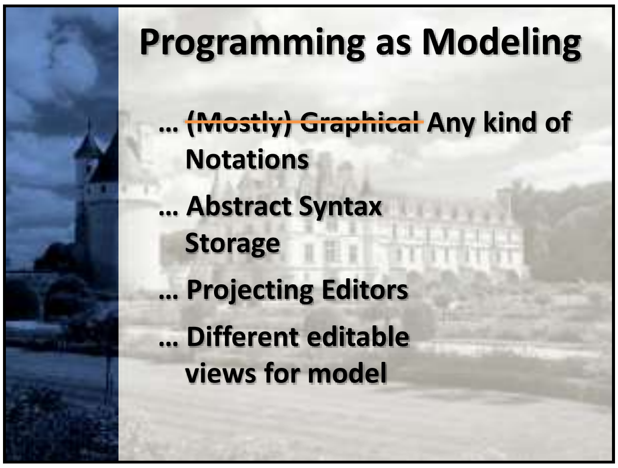
Building DSLs with
Eclipse Xtext

5 Programming as Modeling



Programming as Modeling

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

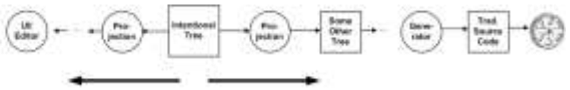


Programming as Modeling

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

Projectional

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



Language Composition

There's no parsing.

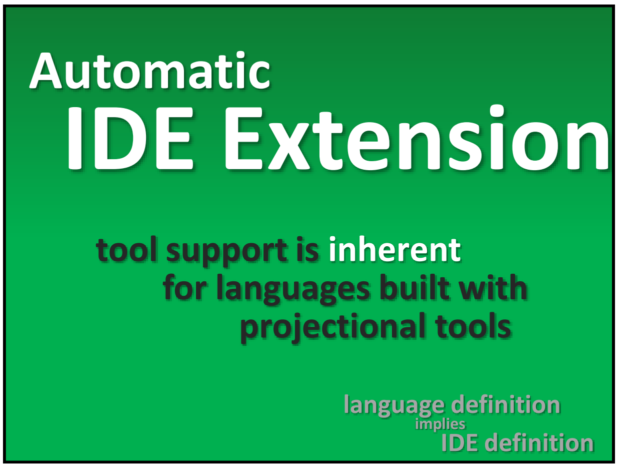
Unique Language Element Identity.

Unlimited language composition.

Flexible Notations

Textual
like ASCII
Graphical
box & line
Semi-Graphical
mathematical

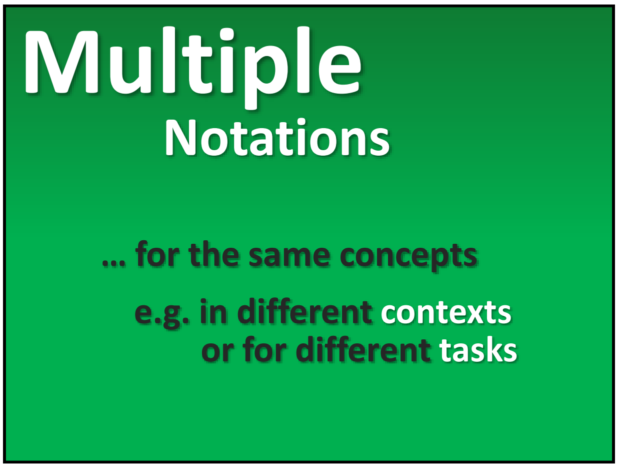
treated the same
can be mixed



Automatic IDE Extension

tool support is inherent
for languages built with
projectional tools

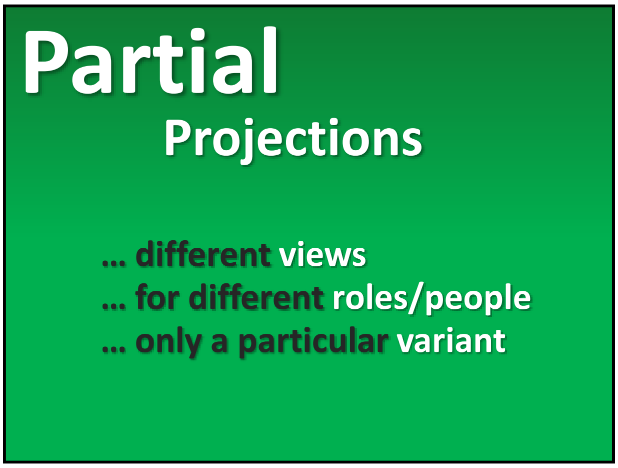
language definition
implies
IDE definition



Multiple Notations

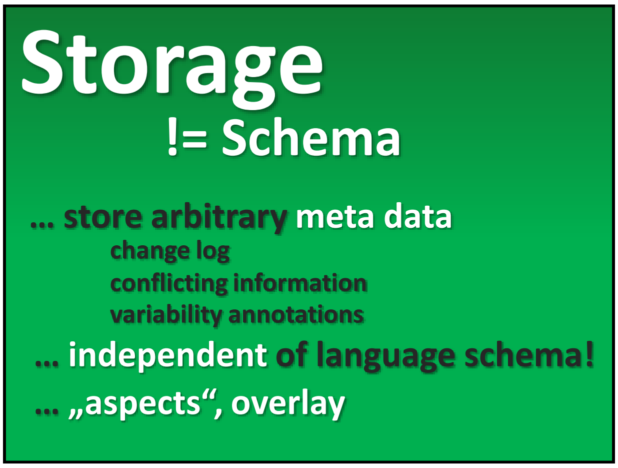
… for the same concepts

e.g. in different contexts
or for different tasks



Partial Projections

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

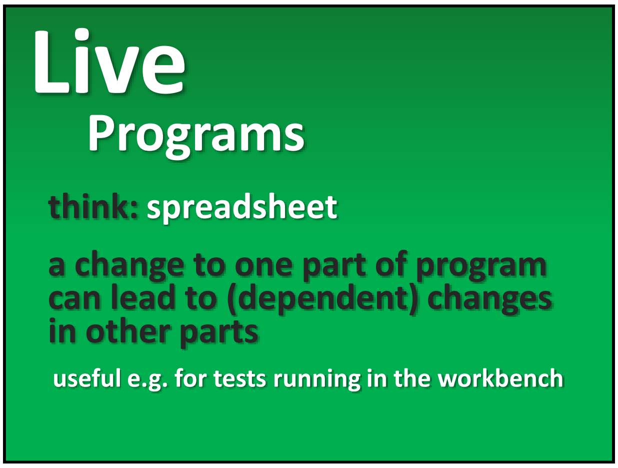


Storage != Schema

… store arbitrary 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

useful e.g. for tests running in the workbench

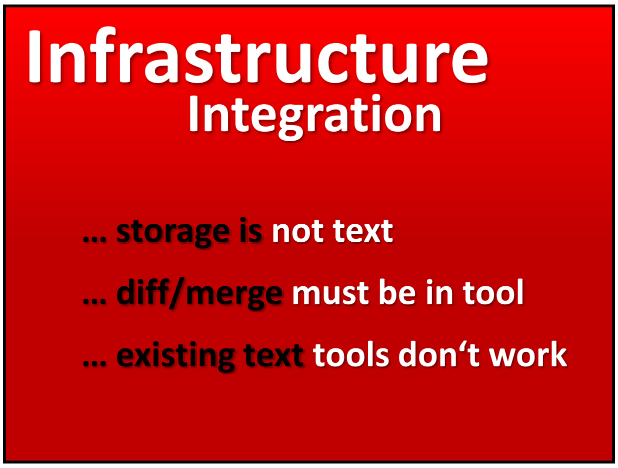
Tree Editing

… 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

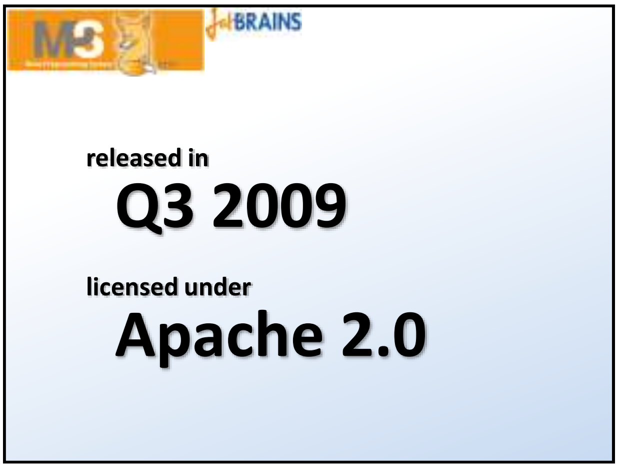
… existing text tools don‘t work

Proprietary
Tools

… no standards

… no out-of-the-box interop

Jetbrains‘
Meta
Programming
System



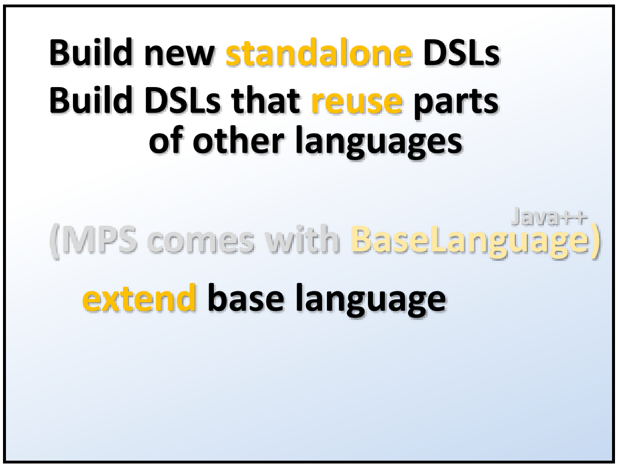
released in
Q3 2009

licensed under
Apache 2.0

Build new standalone DSLs

Build new standalone DSLs

Build DSLs that reuse parts of other languages



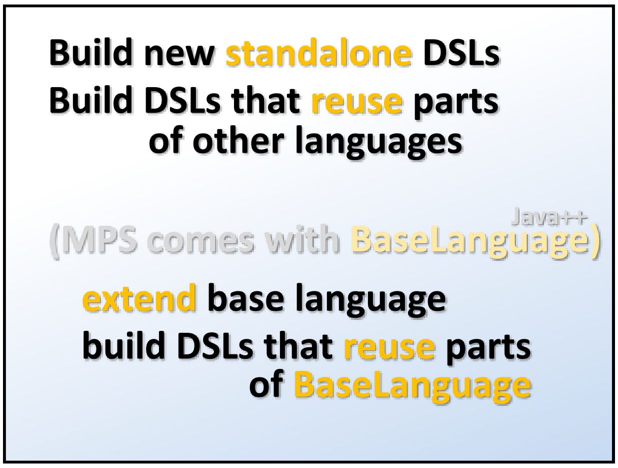
Build new standalone DSLs

Build DSLs that reuse parts of other languages

Java++

(MPS comes with BaseLanguage)

extend base language



Build new standalone DSLs

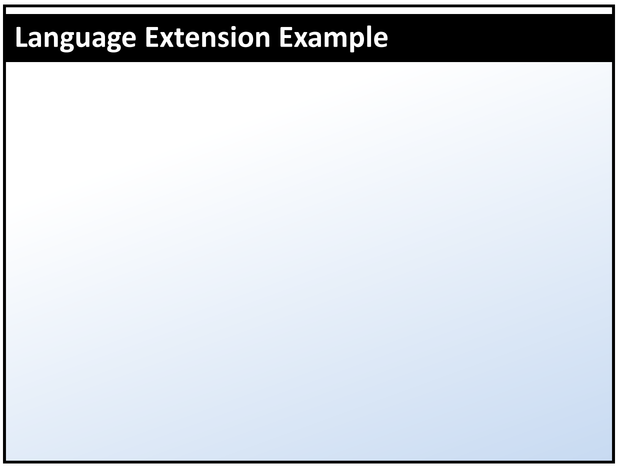
Build DSLs that reuse parts of other languages

Java++

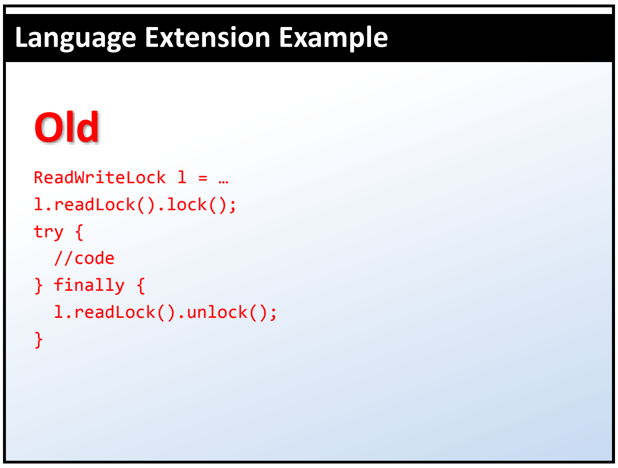
(MPS comes with BaseLanguage)

extend base language

build DSLs that reuse parts of BaseLanguage



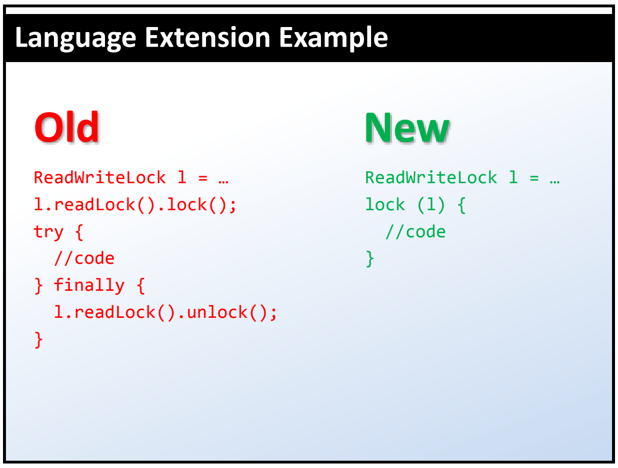
Language Extension Example



Language Extension Example

Old

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



Language Extension Example

Old

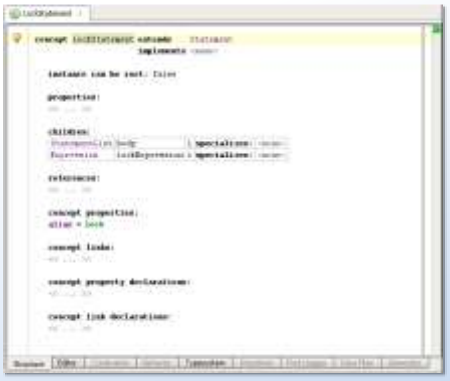
```
ReadWriteLock l = …
l.readLock().lock();
try {
  //code
} finally {
  l.readLock().unlock();
}
```

New

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

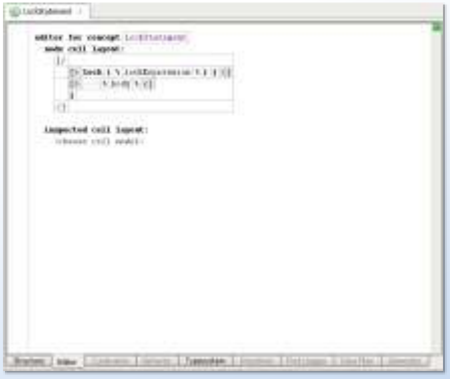
Structure ◆ Editor ◆ Typesystem ◆ Generator

Structure ◆ Editor ◆ Typesystem ◆ Generator



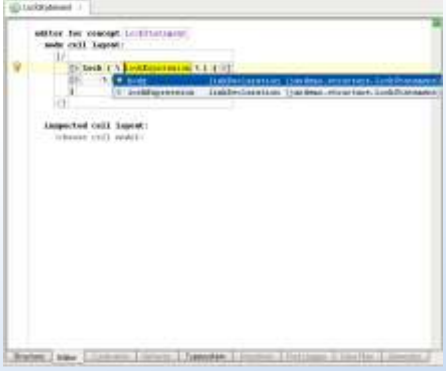
```
concept LockStatement extends Statement
                      implements <none>

  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 ◆ Typesystem ◆ Generator



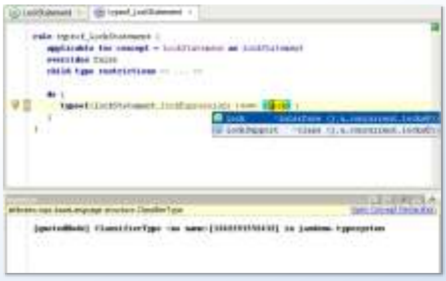
```
editor for concept LockStatement
  node cell layout:
    [/
      [> lock ( % lockExpression % ) { <]
      [>     % body % <]
      }
    /]

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

Structure ◆ Editor ◆ Typesystem ◆ Generator



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

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

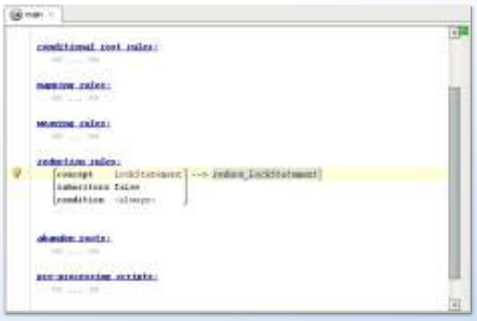
Structure ◆ Editor ◆ Typesystem ◆ Generator



```
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)
```

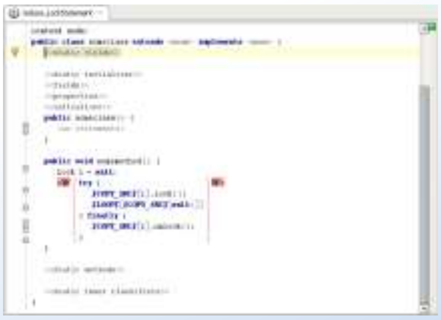
```
jetbrains.mps.baseLanguage.structure.ClassifierType

[quotedNode] ClassifierType <no name>[[illegible]] in jetbrains.typesystem
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

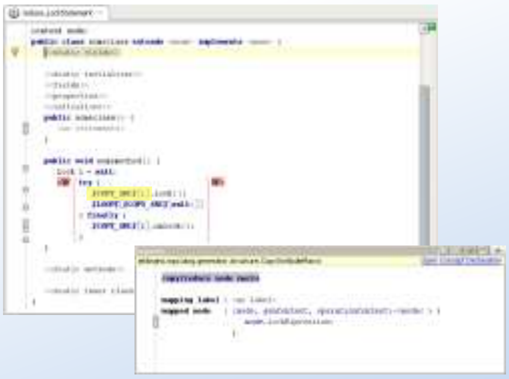


```
conditional root rules:
  << ... >>

mapping rules:
  << ... >>

weaving rules:
  << ... >>

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

abandon roots:
  << ... >>

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

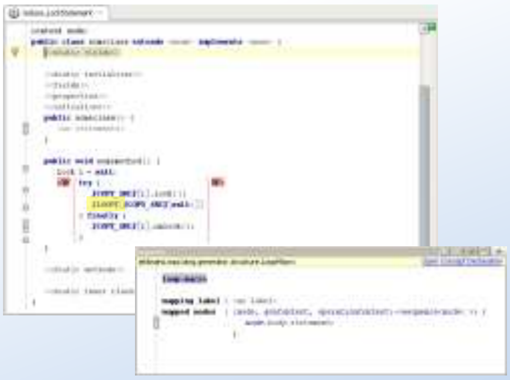
Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator



Language Extension Example

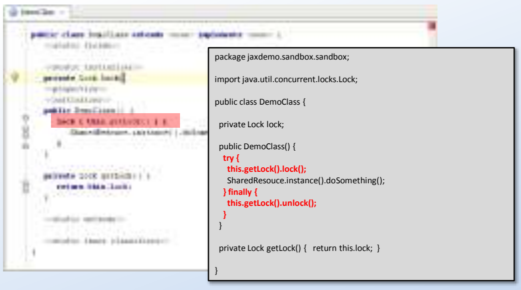
Result behaves like a native base language construct

Language Extension Example

Result behaves like a native base language construct

Language Extension Example

Translated to regular Java code based on the generator



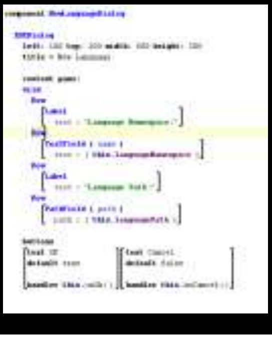
```
package jaxdemo.sandbox.sandbox;

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

Extending Java with
JetBrains MPS

Example Languages

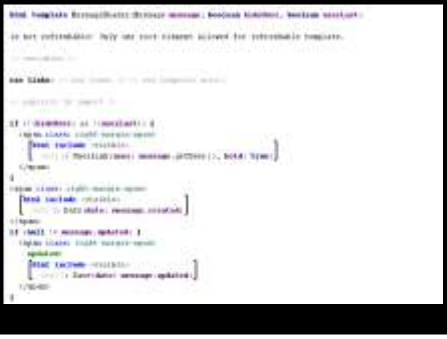
UI Language



```
component NewLanguageDialog

  [illegible]
    left: 100 top: 200 width: 500 height: 150
    title = New Language

    content pane:
    [illegible]
      Row
        [Label
          text = "Language Namespace:"]
      Row
        [TextField { user }
          text = { this.languageNamespace }]
      Row
        [Label
          text = "Language Path:"]
      Row
        [PathField { path }
          path = { this.languagePath }]

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

Example Languages

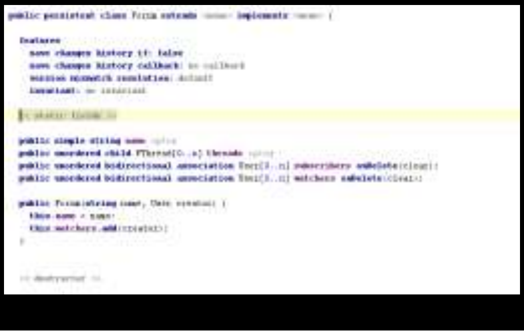
HTML Templates



```
html template [illegible](Message message, boolean [illegible], boolean [illegible])

is not refreshable: [illegible]

<< [illegible] >>

use links: [illegible]

<< [illegible] >>

if ( [illegible] ) {
  <span class: [illegible]>
    [html include [illegible]
      [illegible] message.getUser() [illegible] bold: true]
  </span>
}
<span class: [illegible]>
  [html include [illegible]
    [illegible] date: message.created]
</span>
if (null != message.updated) {
  <span class: [illegible]>
    updated:
    [html include [illegible]
      [illegible] 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: << callback
    version mismatch resolution: default
    invariant: << invariant

  << static fields >>

  public simple string name [illegible]
  public unordered child FThread[0..n] threads [illegible]
  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 >>
```

INTENTIONAL SOFTWARE

Version 1.0 released in

Dec 15, 2008

currently at 1.5

INTENTIONAL SOFTWARE

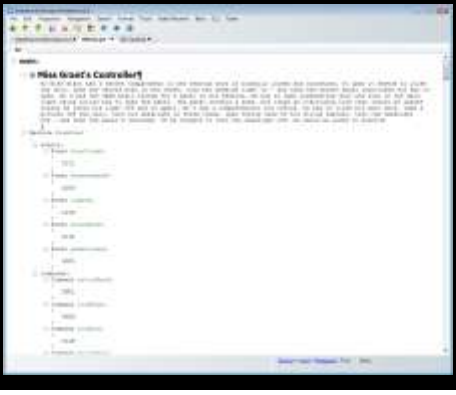
Commercial Product

Eval available

upon request

Statemachine Example

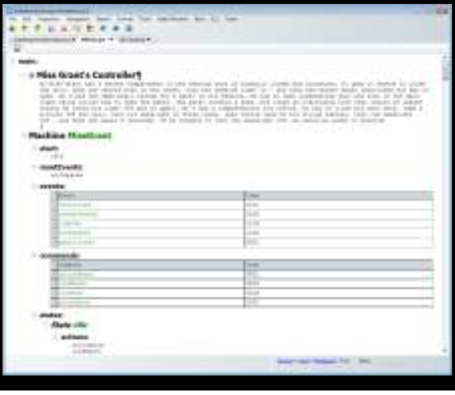
Native Projection



Miss Grant's Controller¶

Statemachine Example

Tabular Projection

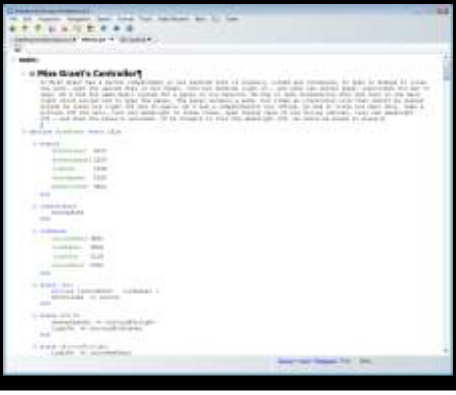


Miss Grant's Controller¶

Machine MissGrant

Statemachine Example

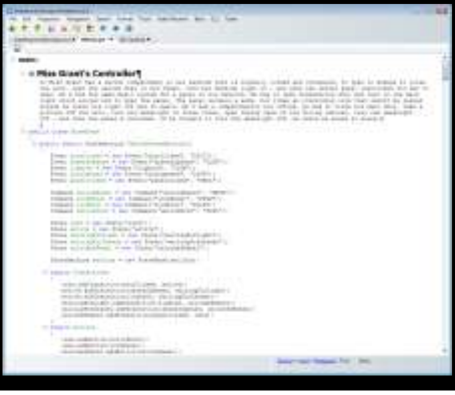
Textual DSL Projection



Miss Grant's Controller¶

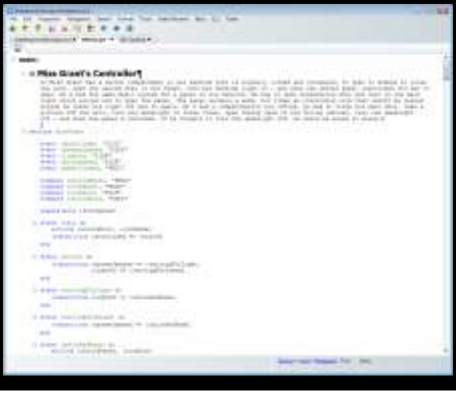
Statemachine Example

Java Projection


Miss Grant's Controller¶

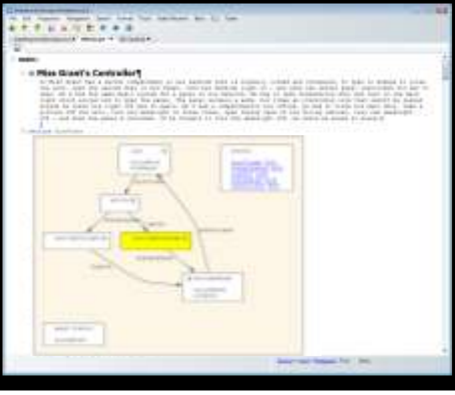
Statemachine Example

Ruby Projection


Miss Grant's Controller¶

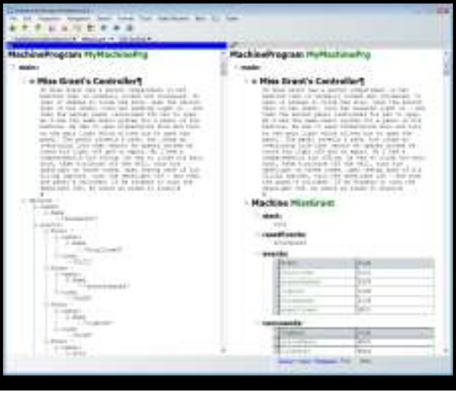
Statemachine Example

Ruby Projection


Miss Grant's Controller¶

Statemachine Example

Two projections side/side


MachineProgram MyMachinePrg

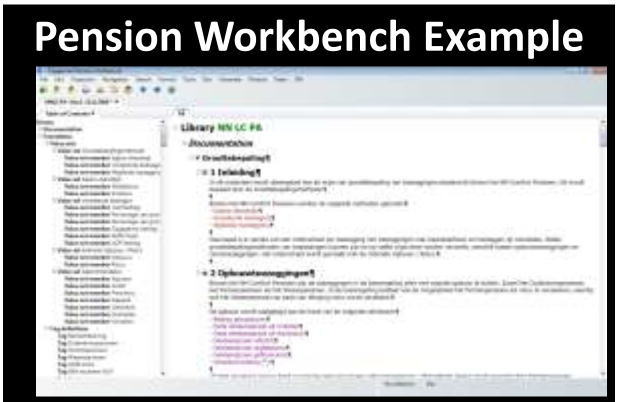
Miss Grant's Controller¶

MachineProgram MyMachinePrg

Miss Grant's Controller¶

Machine MissGrant

Pension Workbench Example


Library NN LC PA

Documentation

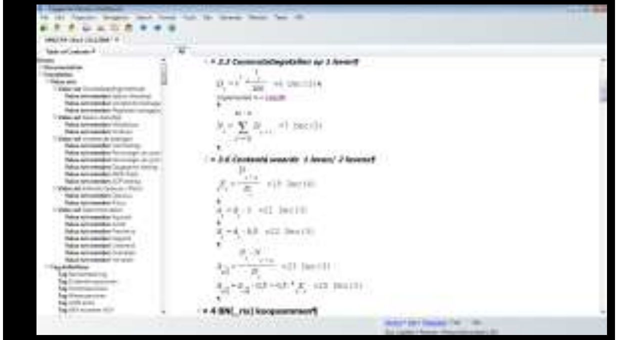
Grootheidsbepaling¶

1 Inleiding¶

2 Opbouwsteunsgegevens¶

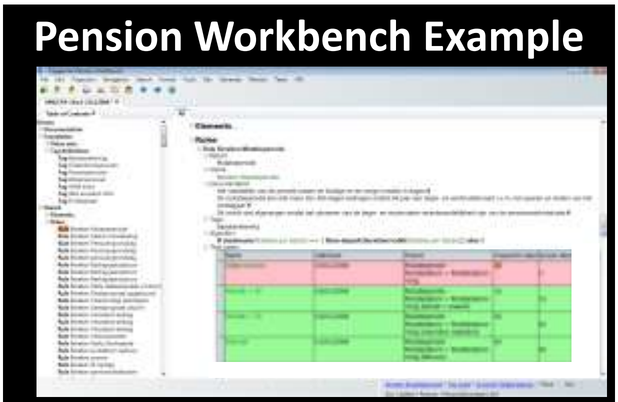
Text Editing Domain

Pension Workbench Example



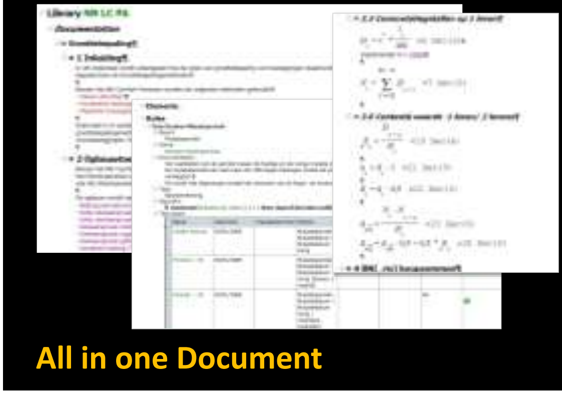
Insurance Mathematics Domain

Pension Workbench Example



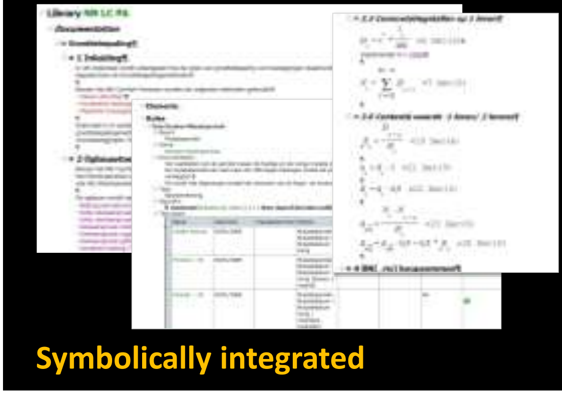
Pension Rules Domain w/ tests

Pension Workbench Example



All in one Document

Pension Workbench Example



Symbolically integrated

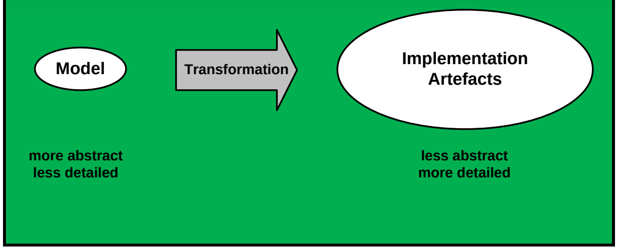
6 Expressing Configurability

Configuration on model level

fewer variation points

less complex

„expanded“ via generator

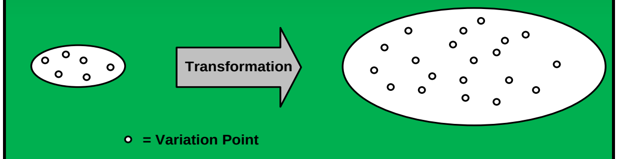


Configuration on model level

fewer variation points

less complex

„expanded“ via generator



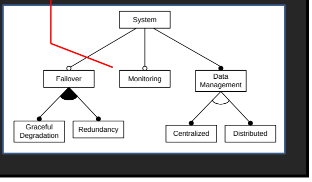
Negative Variability: Conditionally taking something away

Negative Variability: Conditionally taking something away

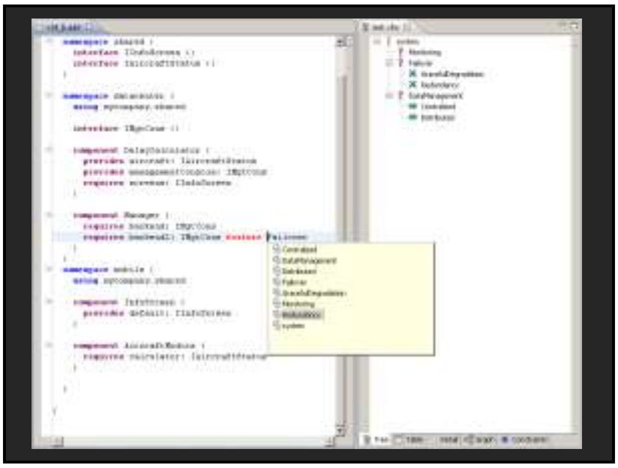
Feature Models

```
component DelayCalculator {
  provides default: IDelayCalculator
  requires screens[0..n]: IInfoScreen
  provides mon: IMonitoring feature monitoring
}
```

```
component DelayCalculator {
  provides default: IDelayCalculator
  requires screens[0..n]: IInfoScreen
  provides mon: IMonitoring feature monitoring
}
```



```
namespace monitoringStuff feature monitoring {

  component MonitoringConsole {
    requires devices:[*]: IMonitor
  }

  instance monitor: MonitoringConsole

  dynamic connect monitor.devices query {
    type = IMonitor
  }

}
```

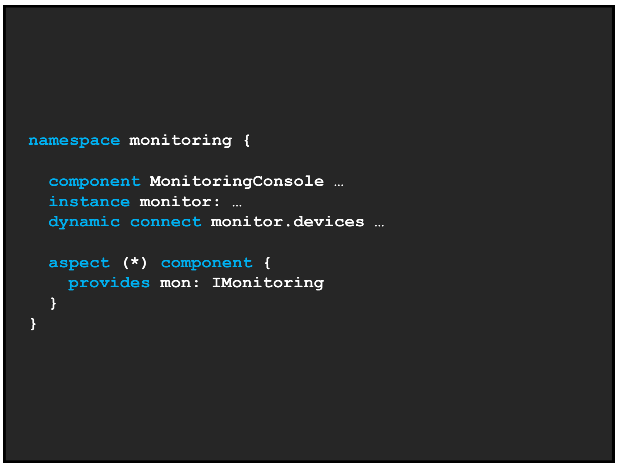


```
namespace shared {
  interface IInfoScreen {}
  interface IAircraftStatus {}
}

namespace datacenter {
  using mycompany.shared

  interface IMgrCons {}

  component DelayCalculator {
    provides aircraft: IAircraftStatus
    provides managementConsole: IMgrCons
    requires screens: IInfoScreen
  }

  component Manager {
    requires backend: IMgrCons
    requires backend2: IMgrCons feature Failover
  }
}
namespace mobile {
  using mycompany.shared

  component InfoScreen {
    provides default: IInfoScreen
  }

  component AircraftModule {
    requires calculator: IAircraftStatus
  }

}
```

Positive Variability: Conditionally adding something to a minimal core

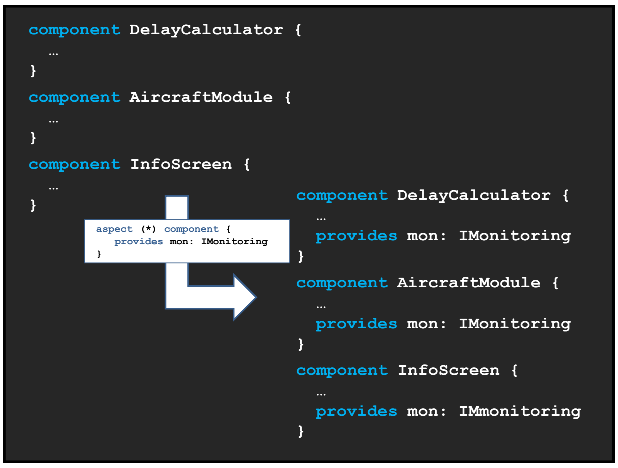
Positive Variability: Conditionally adding something to a minimal core

Aspects



```
namespace monitoring {

  component MonitoringConsole …
  instance monitor: …
  dynamic connect monitor.devices …

  aspect (*) component {
    provides mon: IMonitoring
  }
}
```

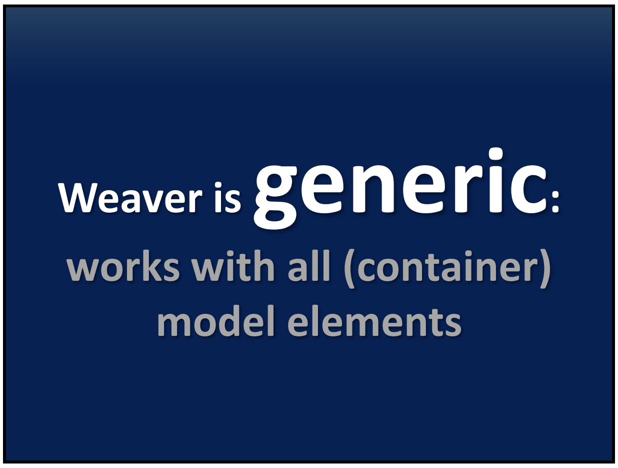
```
component DelayCalculator {
  …
}
component AircraftModule {
  …
}
component InfoScreen {
  …
}
```



```
component DelayCalculator {
  …
}
component AircraftModule {
  …
}
component InfoScreen {
  …
}
```

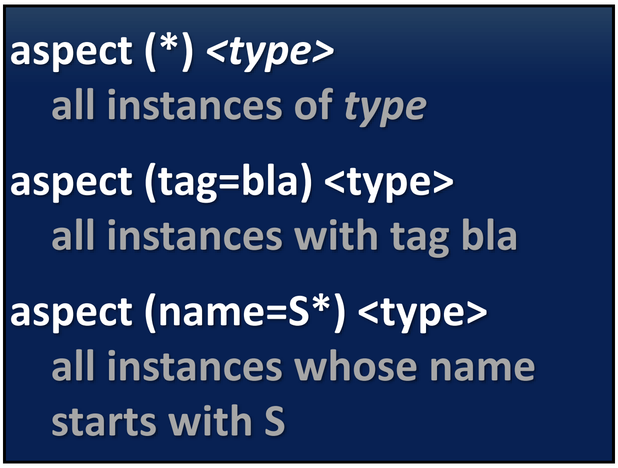
```
aspect (*) component {
  provides mon: IMonitoring
}
```

```
component DelayCalculator {
  …
  provides mon: IMonitoring
}
component AircraftModule {
  …
  provides mon: IMonitoring
}
component InfoScreen {
  …
  provides mon: IMmonitoring
}
```



Weaver is **generic**:

works with all (container) model elements



aspect (*) <*type*>

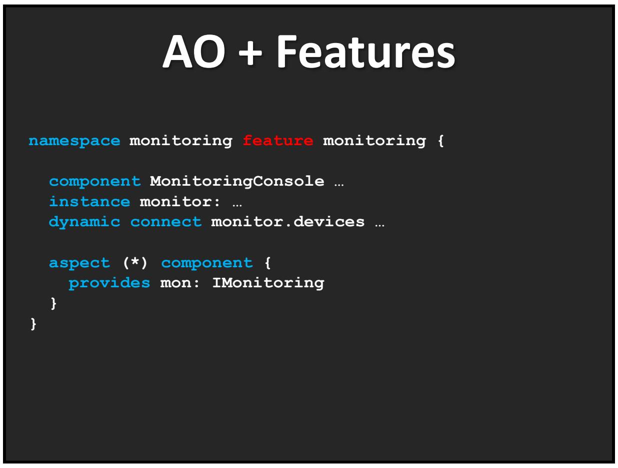
all instances of *type*

aspect (tag=bla) <type>

all instances with tag bla

aspect (name=S*) <type>

all instances whose name starts with S



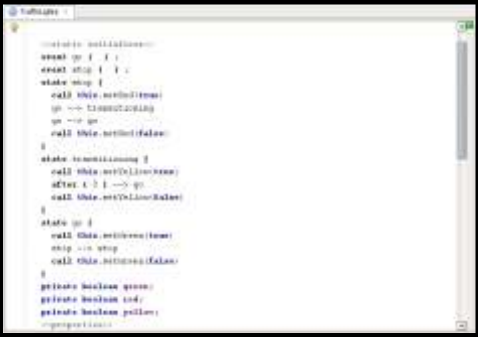
AO + Features

```
namespace monitoring feature monitoring {

  component MonitoringConsole …
  instance monitor: …
  dynamic connect monitor.devices …

  aspect (*) component {
    provides mon: IMonitoring
  }
}
```

DEMO III

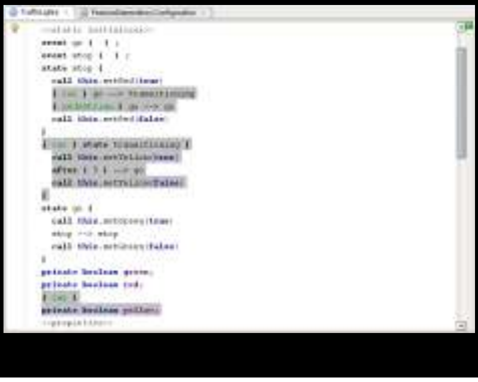
Adding Variability and connectivity to a feature model to an Xtext DSL

A statemachine



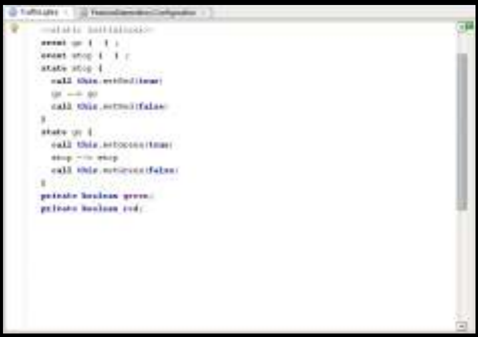
```
<<static initializer>>
event go { } ;
event stop { } ;
state stop {
  call this.setRed(true)
  go --> transitioning
  go --> go
  call this.setRed(false)
}
state transitioning {
  call this.setYellow(true)
  after ( 3 ) --> go
  call this.setYellow(false)
}
state go {
  call this.setGreen(true)
  stop --> stop
  call this.setGreen(false)
}
private boolean green;
private boolean red;
private boolean yellow;
<<properties>>
```

Actually, several…!



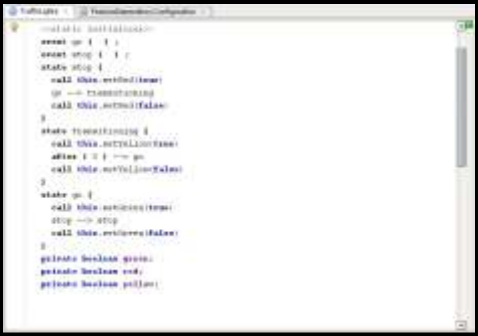
```
<<static initializer>>
event go { } ;
event stop { } ;
state stop {
  call this.setRed(true)
  { car } go --> transitioning
  { pedestrian } go --> go
  call this.setRed(false)
}
{ car } state transitioning {
  call this.setYellow(true)
  after ( 3 ) --> go
  call this.setYellow(false)
}
state go {
  call this.setGreen(true)
  stop --> stop
  call this.setGreen(false)
}
private boolean green;
private boolean red;
{ car }
private boolean yellow;
<<properties>>
```

Variant for Pedestrians



```
<<static initializer>>
event go { } ;
event stop { } ;
state stop {
  call this.setRed(true)
  go --> go
  call this.setRed(false)
}
state go {
  call this.setGreen(true)
  stop --> stop
  call this.setGreen(false)
}
private boolean green;
private boolean red;
```

Variant for Cars



```
<<static initializer>>
event go { } ;
event stop { } ;
state stop {
  call this.setRed(true)
  go --> transitioning
  call this.setRed(false)
}
state transitioning {
  call this.setYellow(true)
  after { 3 } --> go
  call this.setYellow(false)
}
state go {
  call this.setGreen(true)
  stop --> stop
  call this.setGreen(false)
}
private boolean green;
private boolean red;
private boolean yellow;
```

DEMO IV

Expressing Variability
with MPS

Facilities to express
Variability
are completely
independent
of the target language!

Facilities to express
Variability
are completely
independent
of the target language!

which leads to…

7 Modular Languages

Viewpoints

Business

custom
purpose-built

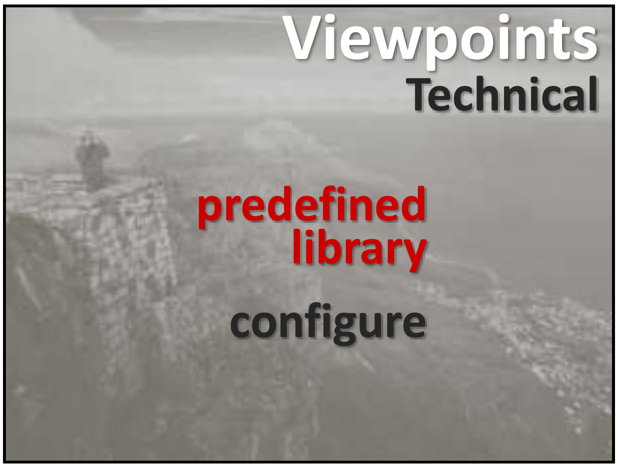
create/include

Viewpoints

Business

Custom
Notations

real
business
expert integration



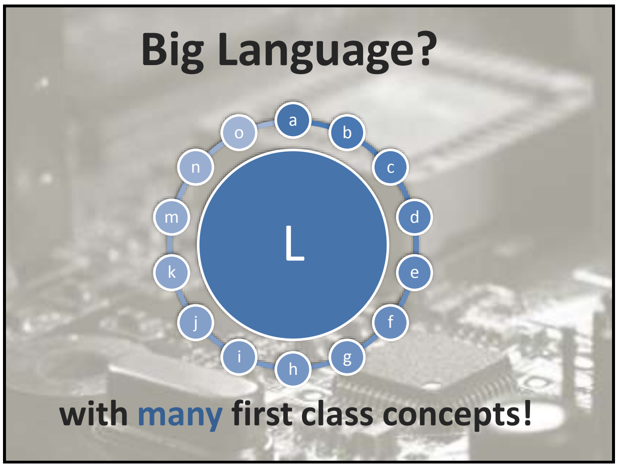
Viewpoints

Technical

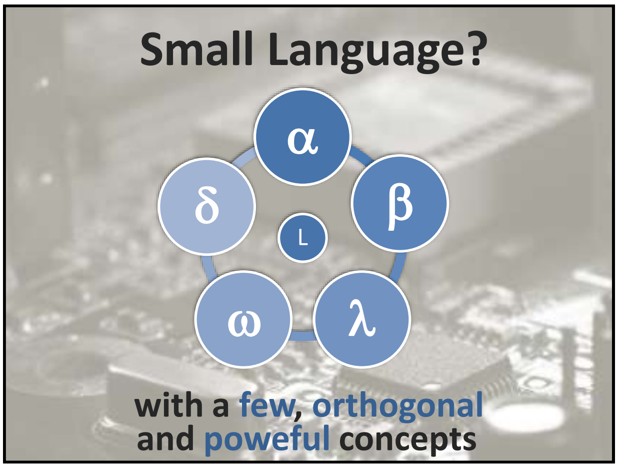
predefined
library

configure

Big Language?

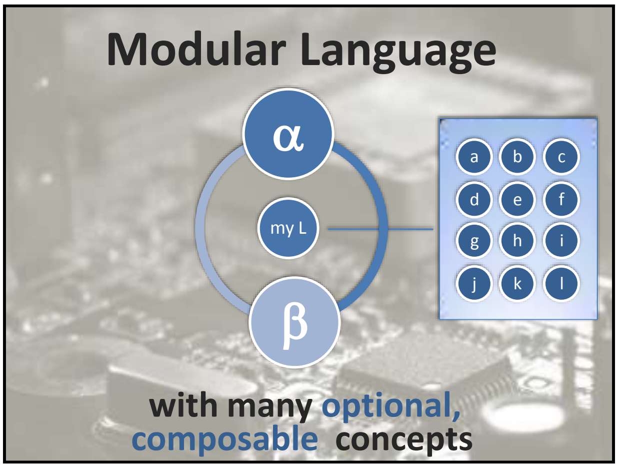


with many first class concepts!



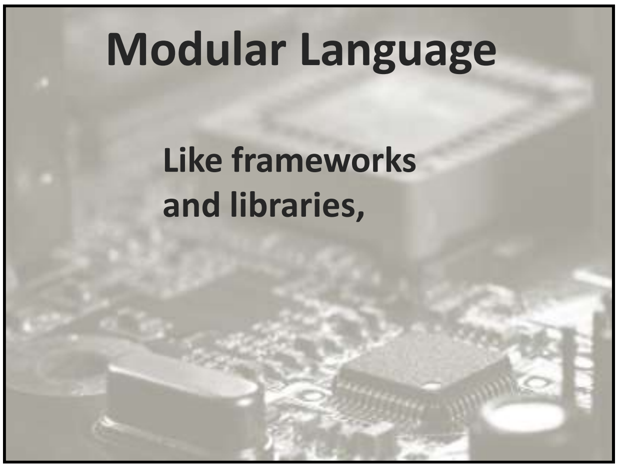
Small Language?

with a few, orthogonal
and poweful concepts



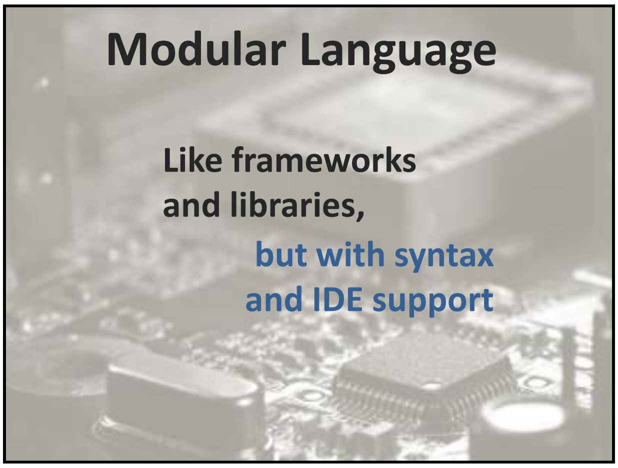
Modular Language

with many optional,
composable concepts



Modular Language

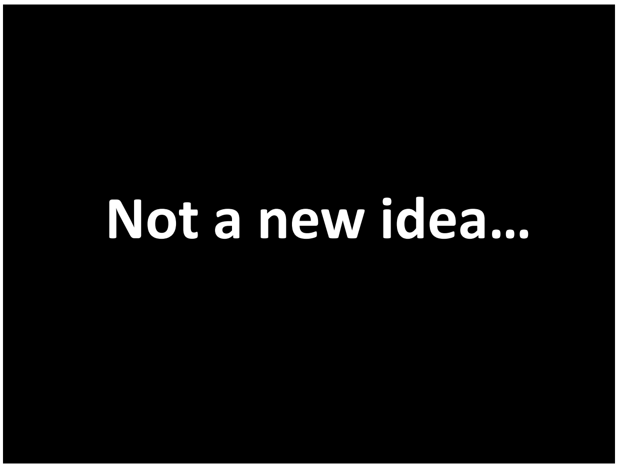
Like frameworks
and libraries,



Modular Language

Like frameworks
and libraries,

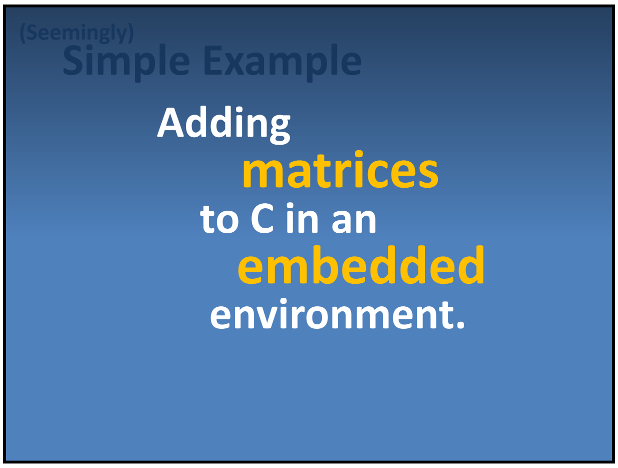
but with syntax
and IDE support



Not a new idea…

Growing A Language

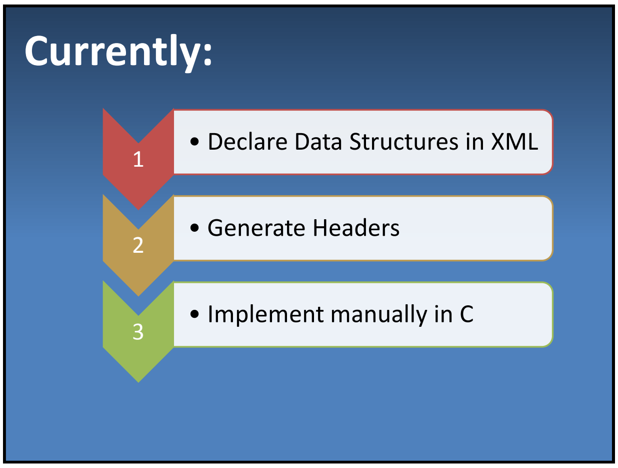
(Guy L Steele)



(Seemingly)

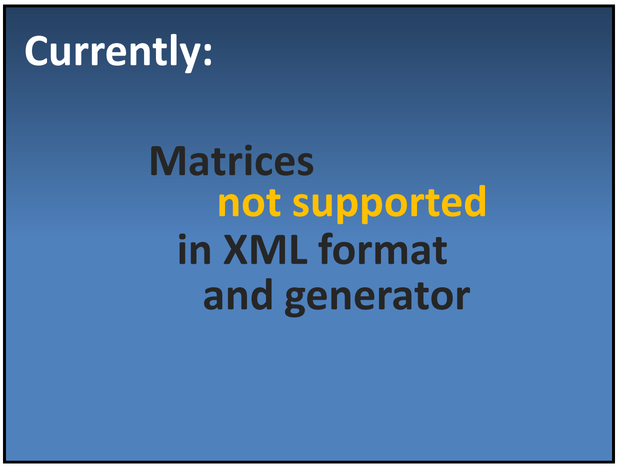
Simple Example

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

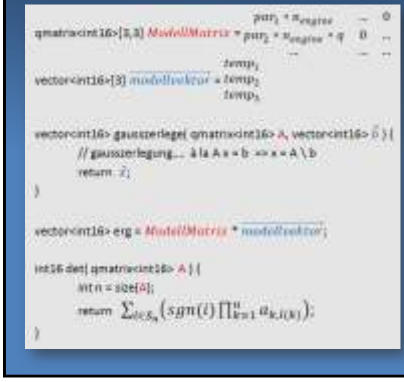
… a lot of work
… busy
… one central tool

Currently:

No real
compiler support
in the resulting C code

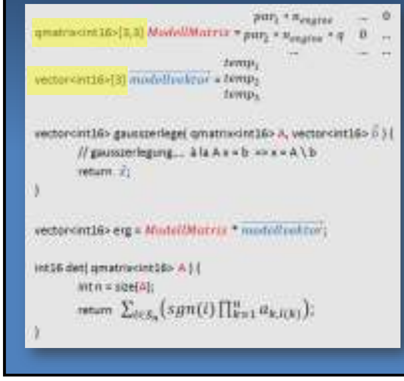
… type checks
… operator overloading
… generics (matrix<int>)
… matrix syntax?

Better Solution



```
                                         par1 * n_engine          –    0
qmatrix<int16>[3,3] ModellMatrix = par2 * n_engine * q   0    –
                                               –               –    –

                                   temp1
vector<int16>[3] modellvektor = temp2
                                   temp3

vector<int16> gausszerlegel( 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)});
}
```

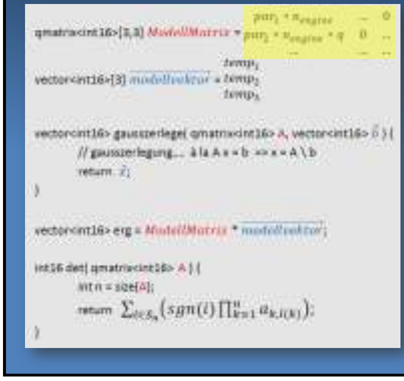
Better Solution



```
                                         par1 * n_engine          –    0
qmatrix<int16>[3,3] ModellMatrix = par2 * n_engine * q   0    –
                                               –               –    –

                                   temp1
vector<int16>[3] modellvektor = temp2
                                   temp3

vector<int16> gausszerlegel( 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)});
}
```

generic
matrix
and
vector
types

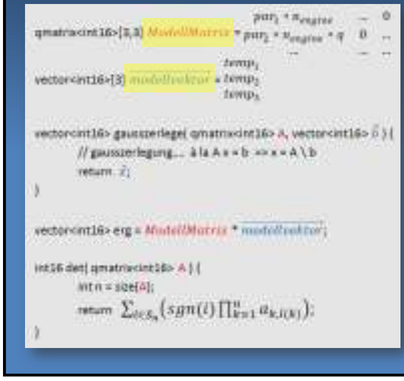
Better Solution



```
                                         par1 * n_engine          –    0
qmatrix<int16>[3,3] ModellMatrix = par2 * n_engine * q   0    –
                                               –               –    –

                                   temp1
vector<int16>[3] modellvektor = temp2
                                   temp3

vector<int16> gausszerlegel( 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)});
}
```

real
matrix
and
vector
literals

Better Solution



```
                                   pur1 * πengree        —   0
qmatrix<int16>[3,3] ModellMatrix = pur2 * πengree * 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 Σ_{i∈S_n}(sgn(i) Π_{k=1}^{n} a_{k,i(k)});
}
```

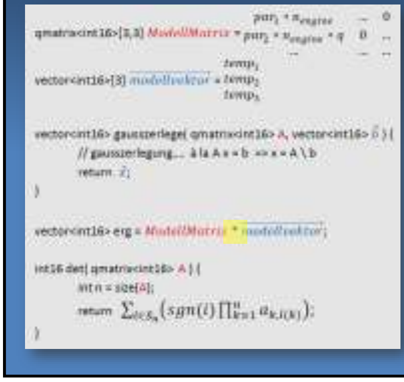
syntax
highlights
for
vectors
and
matrices

Better Solution

```
                                   pur1 * πengree        —   0
qmatrix<int16>[3,3] ModellMatrix = pur2 * πengree * 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 Σ_{i∈S_n}(sgn(i) Π_{k=1}^{n} a_{k,i(k)});
}
```

operator
overloading

Better Solution



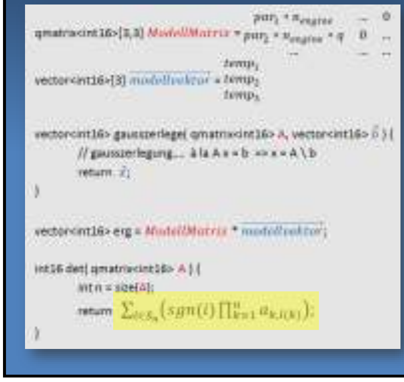
```
                                   pur1 * πengree        —   0
qmatrix<int16>[3,3] ModellMatrix = pur2 * πengree * q    0   ...
                                   ...                   —   ...
                                    temp1
vector<int16>[3] modellvektor =     temp2
                                    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 Σ_{i∈S_n}(sgn(i) Π_{k=1}^{n} a_{k,i(k)});
}
```

operator
overloading

static
optimization

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

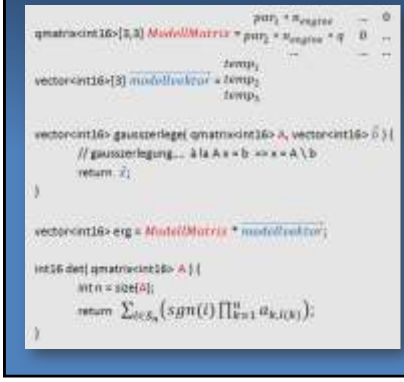
Better Solution



```
                                   pur1 * πengree        —   0
qmatrix<int16>[3,3] ModellMatrix = pur2 * πengree * 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 Σ_{i∈S_n}(sgn(i) Π_{k=1}^{n} a_{k,i(k)});
}
```

math
notation

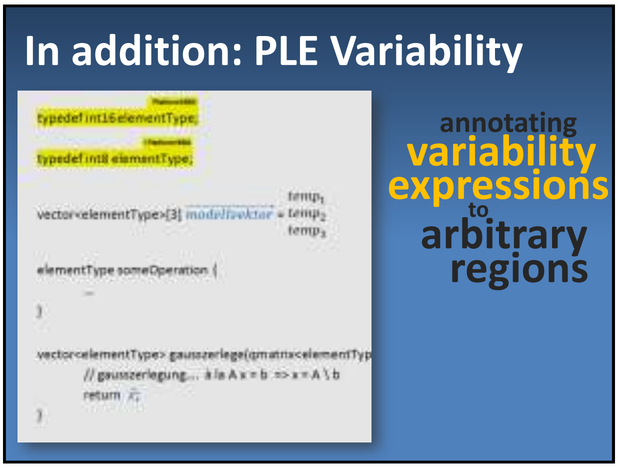
Better Solution



```
                                   pur1 * πengree        —   0
qmatrix<int16>[3,3] ModellMatrix = pur2 * πengree * 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 Σ_{i∈S_n}(sgn(i) Π_{k=1}^{n} a_{k,i(k)});
}
```

a
separate
language
module

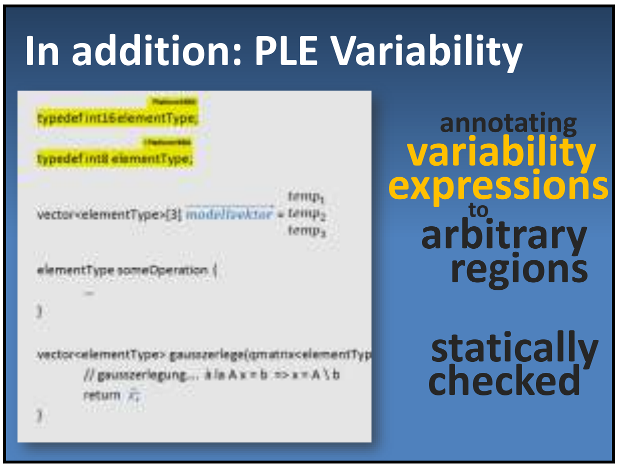
used only by
those who
really need it



In addition: PLE Variability

```
typedef int16 elementType;

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

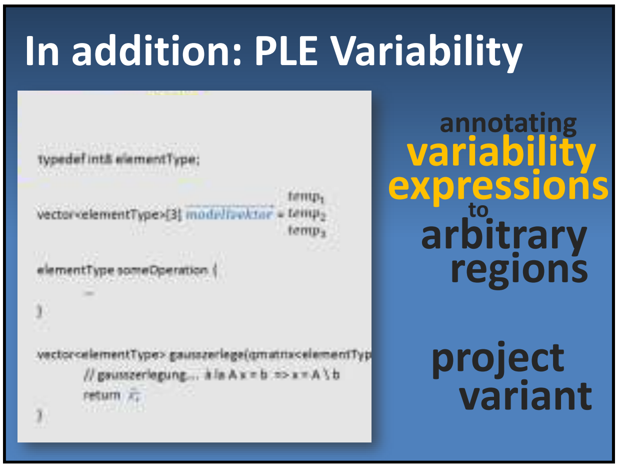


In addition: PLE Variability

```
typedef int16 elementType;

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

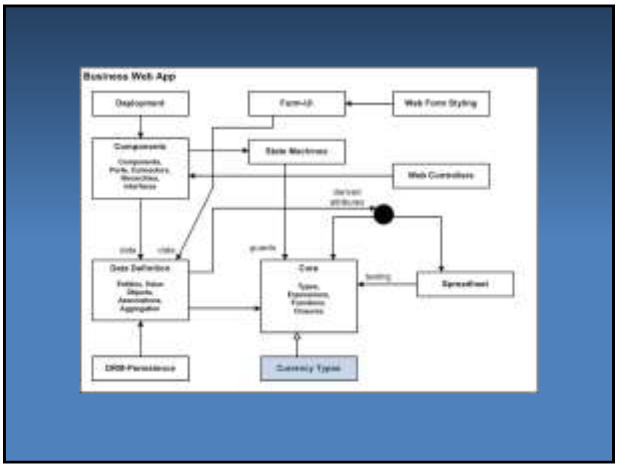


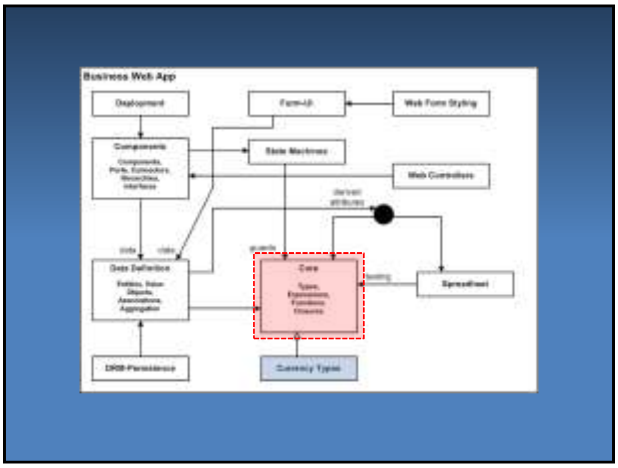
In addition: PLE Variability

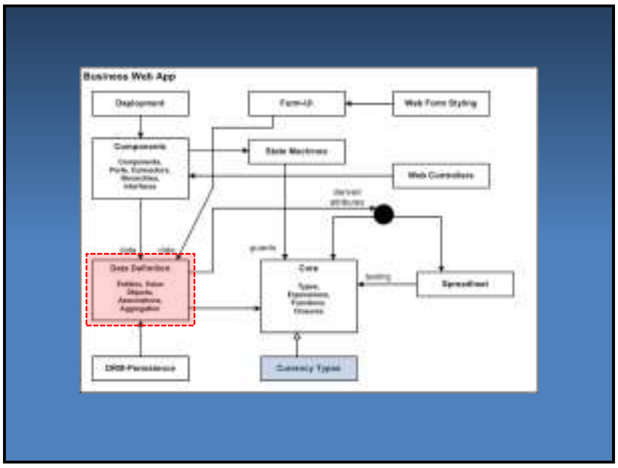
```
typedef int8 elementType;

                                           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







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

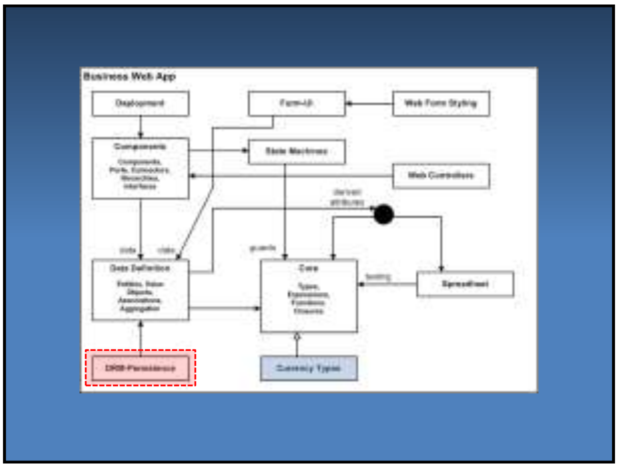
Types, Expressions, Functions, Closures

testing

Spreadsheet

DRB-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

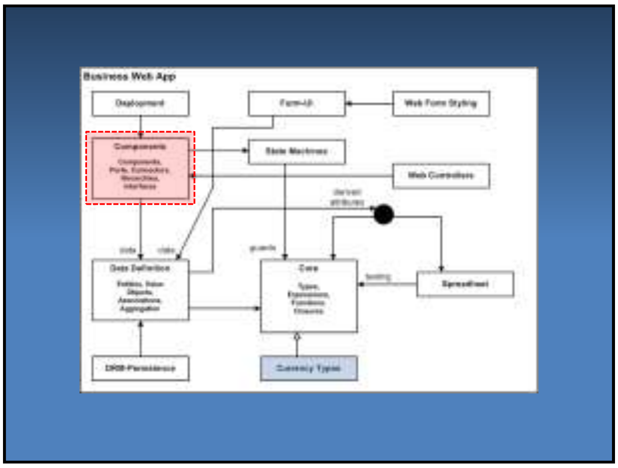
Types, Expressions, Functions, Closures

testing

Spreadsheet

DRB-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

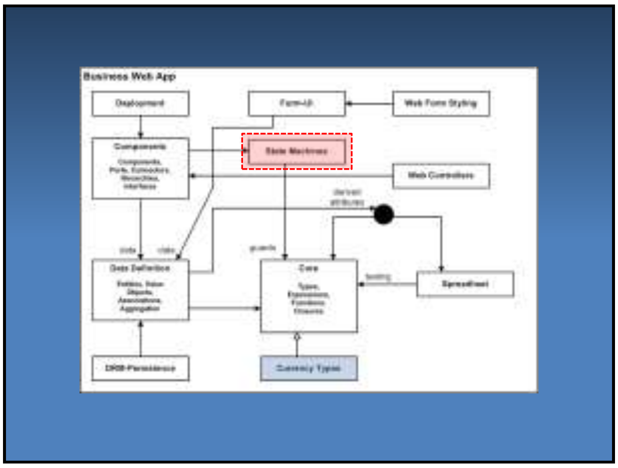
Types, Expressions, Functions, Closures

testing

Spreadsheet

DRB-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

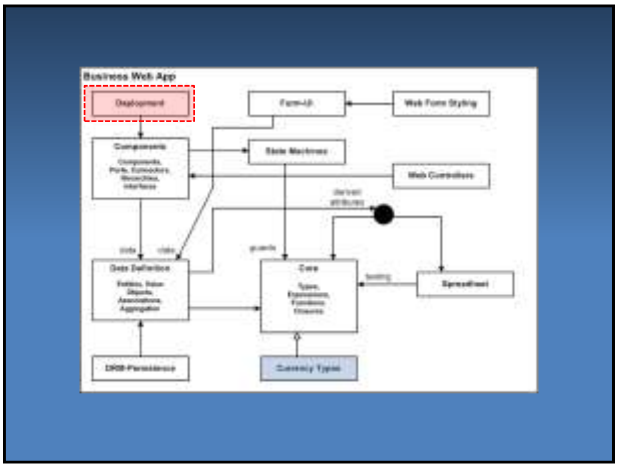
Types, Expressions, Functions, Closures

testing

Spreadsheet

DRB-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

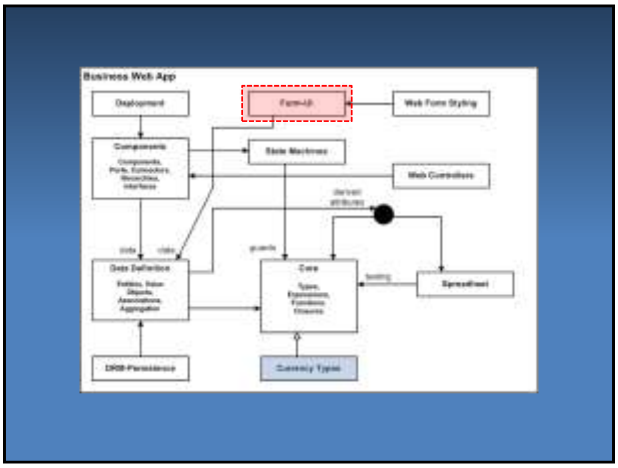
Types, Expressions, Functions, Closures

testing

Spreadsheet

DRB-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

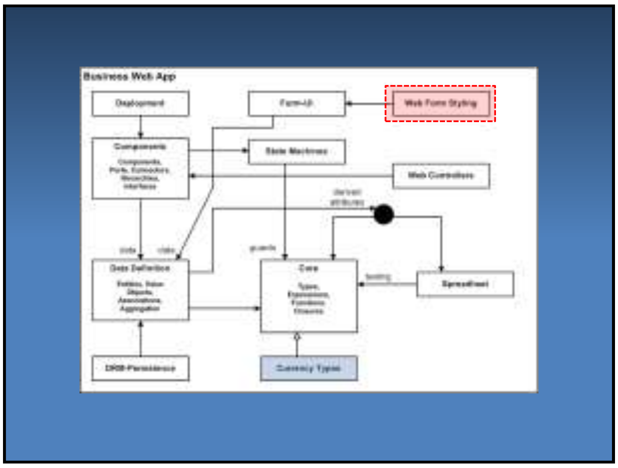
Types, Expressions, Functions, Closures

testing

Spreadsheet

DRB-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

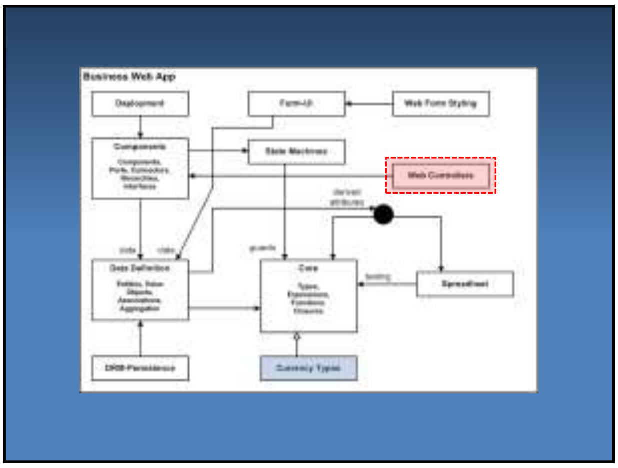
Types, Expressions, Functions, Closures

testing

Spreadsheet

ORM-Persistence

Currency Types



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

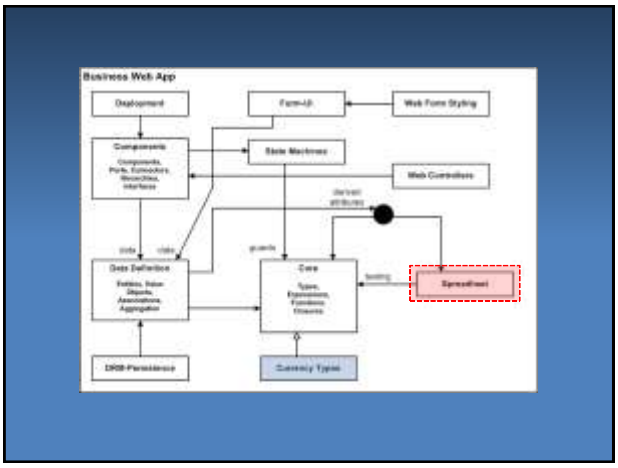
Types, Expressions, Functions, Closures

testing

Spreadsheet

ORM-Persistence

Currency Types



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

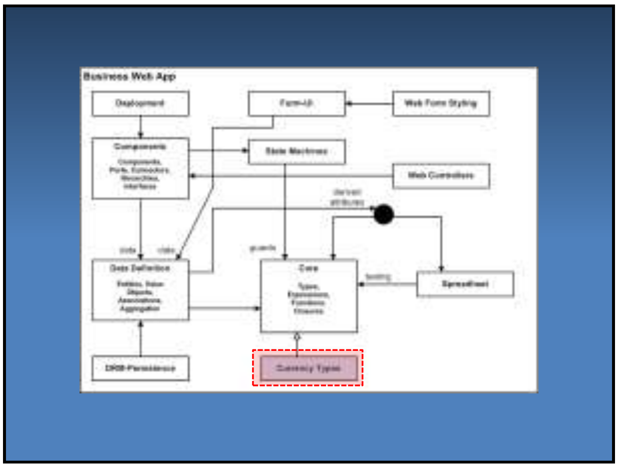
Types, Expressions, Functions, Closures

testing

Spreadsheet

ORM-Persistence

Currency Types



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

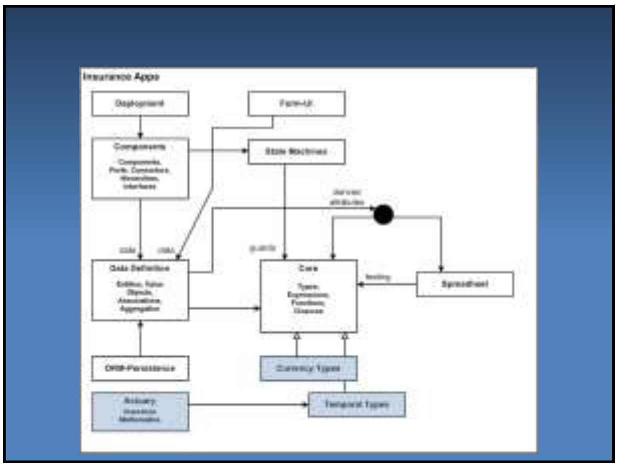
Types, Expressions, Functions, Closures

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

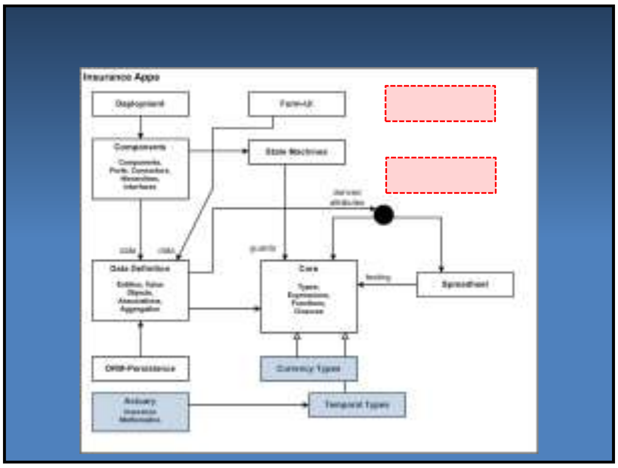
ORM-Persistence

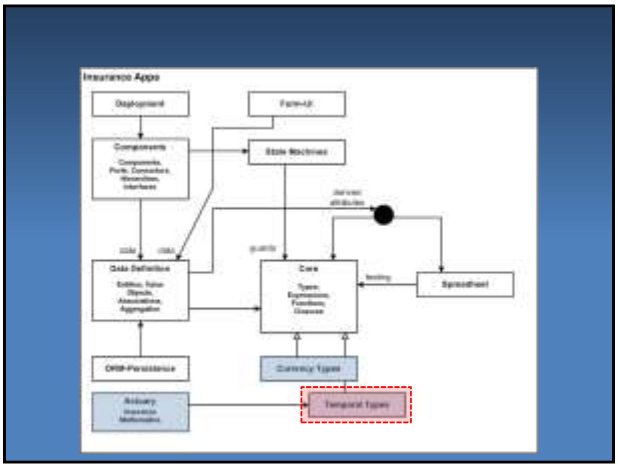
Currency Types

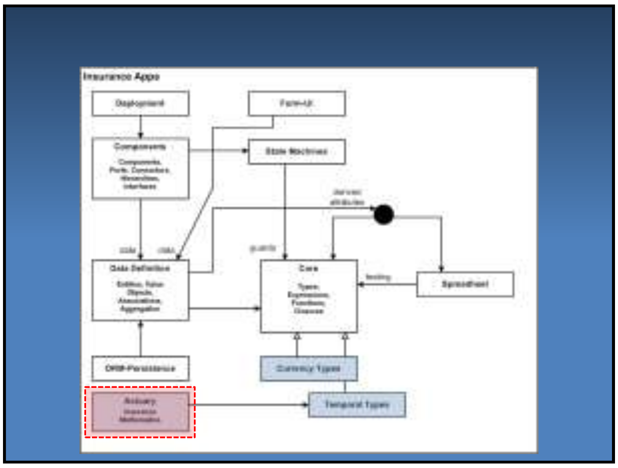
Actuary

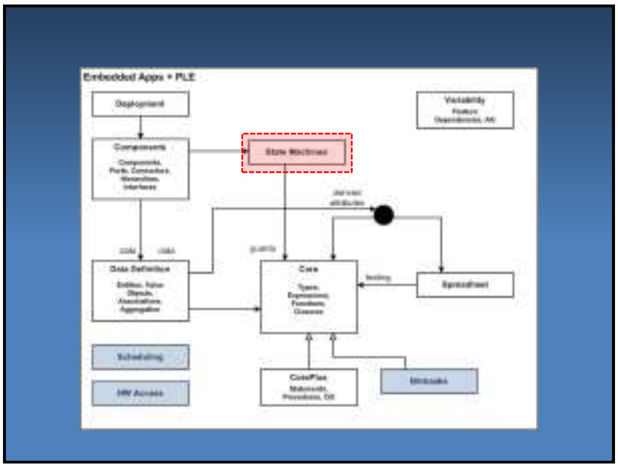
Insurance Mathematics

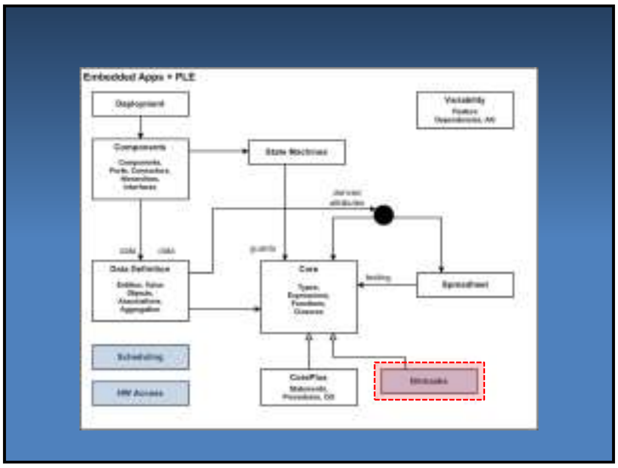
Temporal Types

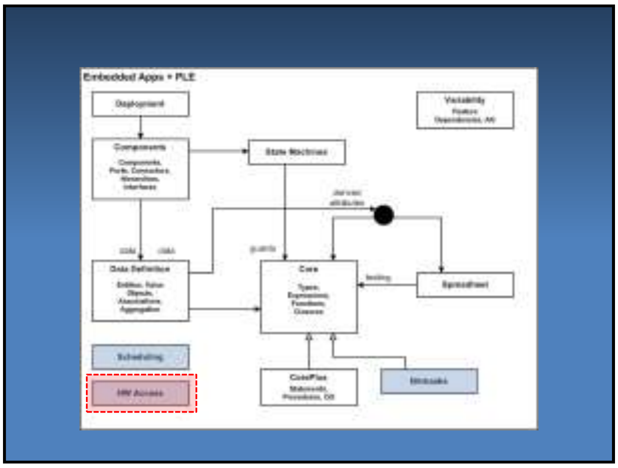

Insurance Apps
Deployment
Form-UI
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadsheet
ORM-Persistence
Currency Types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


Insurance Apps
Deployment
Form-UI
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
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
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
Variability
Feature Dependencies, AM
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadsheet
Scheduling
HW Access
CoreFlex
Statements, Procedures, OO
Intervals


Embedded Apps + PLE
Deployment
Variability
Feature Dependencies, AM
Components
Components, Ports, Connectors, Hierarchies, Interfaces
State Machines
derived attributes
guards
Data Definition
Entities, Value Objects, Associations, Aggregation
Core
Types, Expressions, Functions, Closures
testing
Spreadsheet
Scheduling
HW Access
CoreFlex
Statements, Procedures, OO
Intervals



Embedded Apps + PLE

Deployment

Variability

Components

State Machines

derived attributes

data

guards

Data Definition

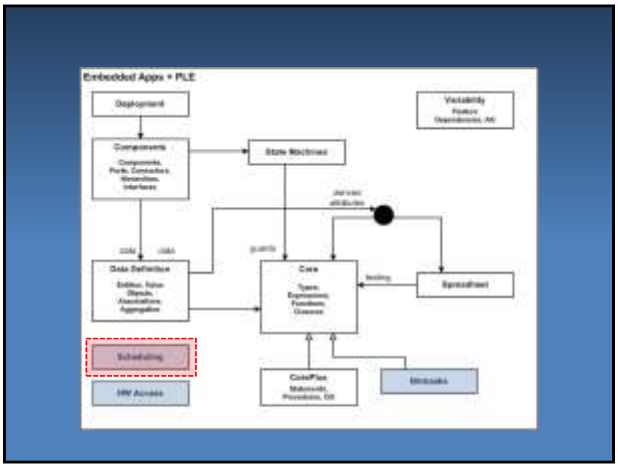
Core

testing

Spreadsheet

Scheduling

HW Access



Embedded Apps + PLE

Deployment

Variability

Components

State Machines

derived attributes

data

guards

Data Definition

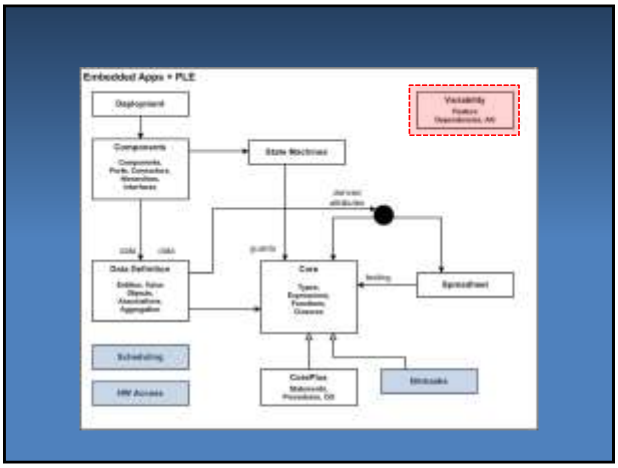
Core

testing

Spreadsheet

Scheduling

HW Access



Embedded Apps + PLE

Deployment

Variability

Components

State Machines

derived attributes

data

guards

Data Definition

Core

testing

Spreadsheet

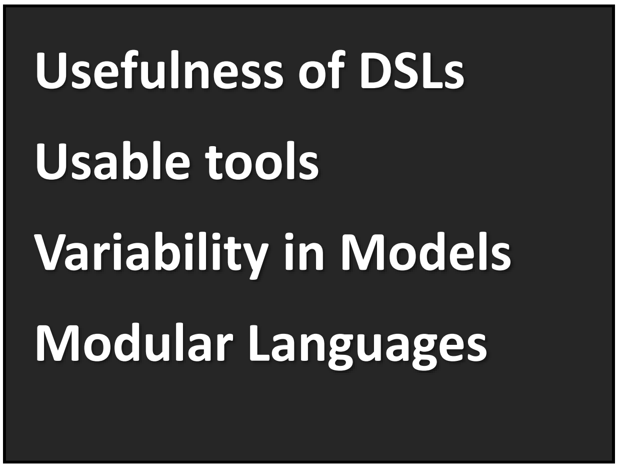
Scheduling

HW Access

DEMO V

Language Modularity
with MPS

8 Summary

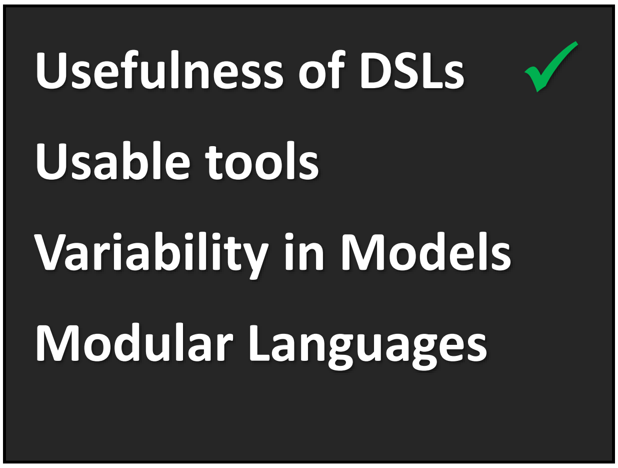


Usefulness of DSLs

Usable tools

Variability in Models

Modular Languages

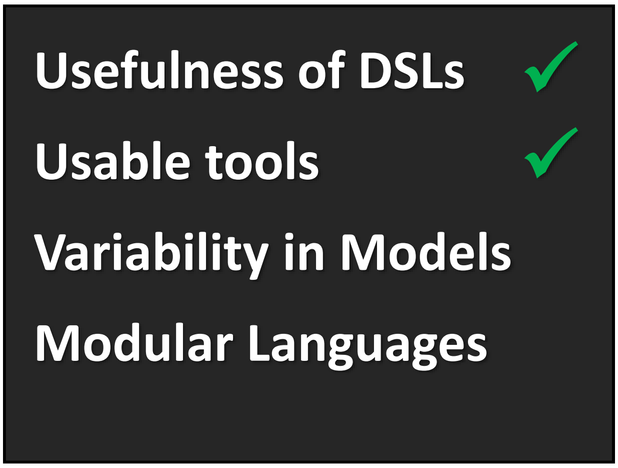


Usefulness of DSLs ✓

Usable tools

Variability in Models

Modular Languages

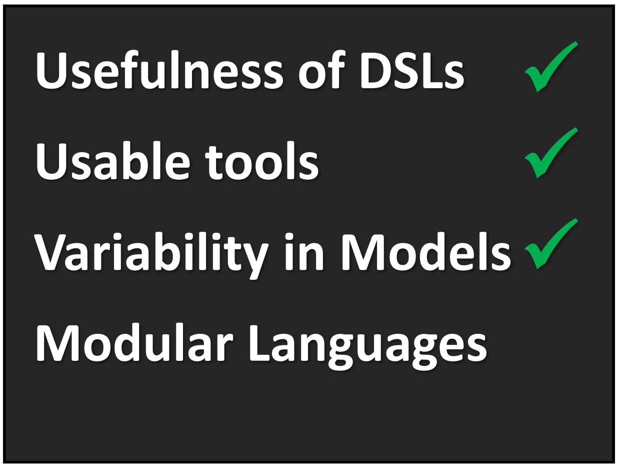


Usefulness of DSLs ✓

Usable tools ✓

Variability in Models

Modular Languages

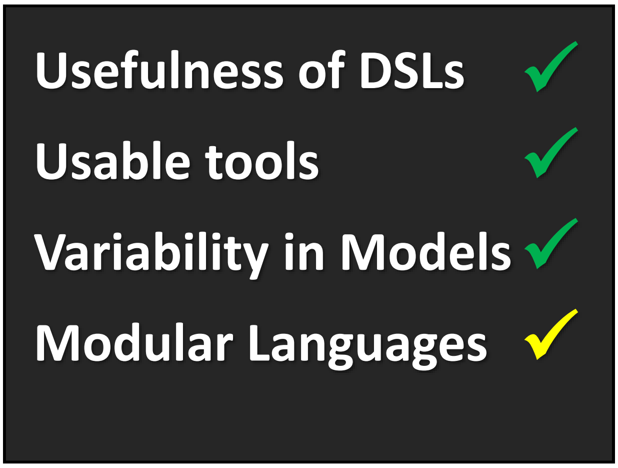


Usefulness of DSLs ✓

Usable tools ✓

Variability in Models ✓

Modular Languages



- Usefulness of DSLs ✓
- Usable tools ✓
- Variability in Models ✓
- Modular Languages ✓

THE END

Markus Voelter
Indepenent/itemis
voelter@acm.org
http://www.voelter.de