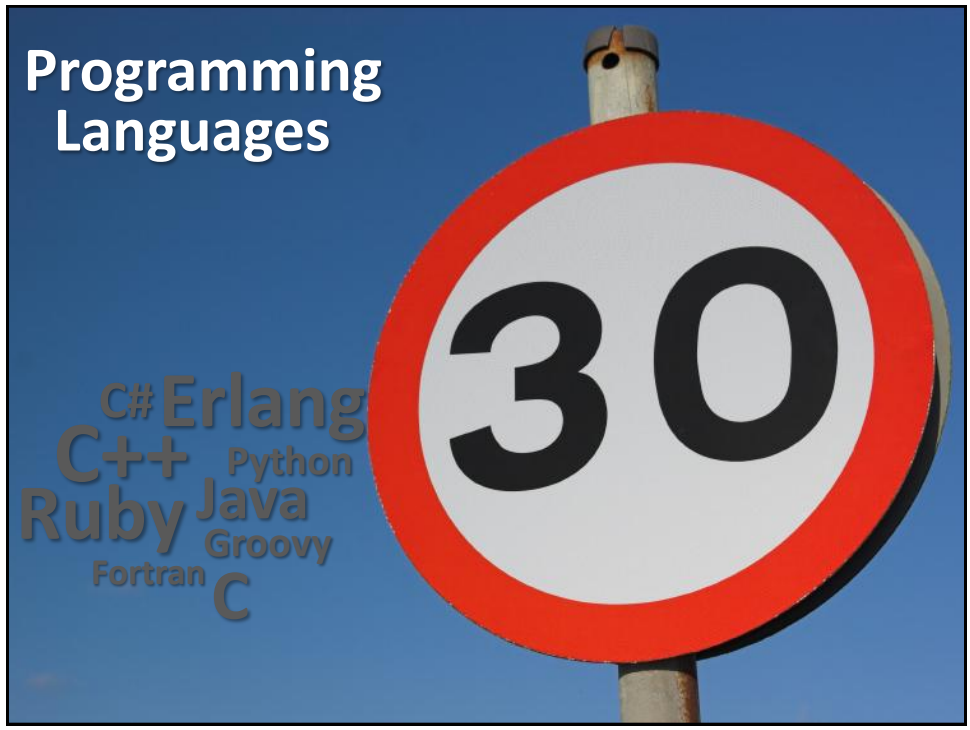
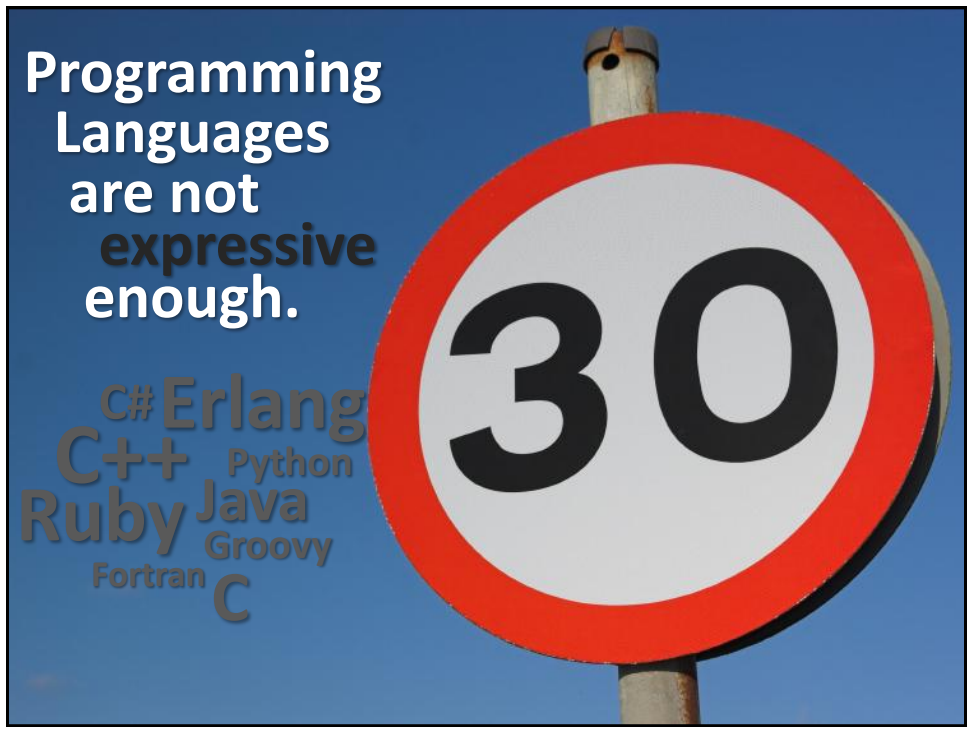
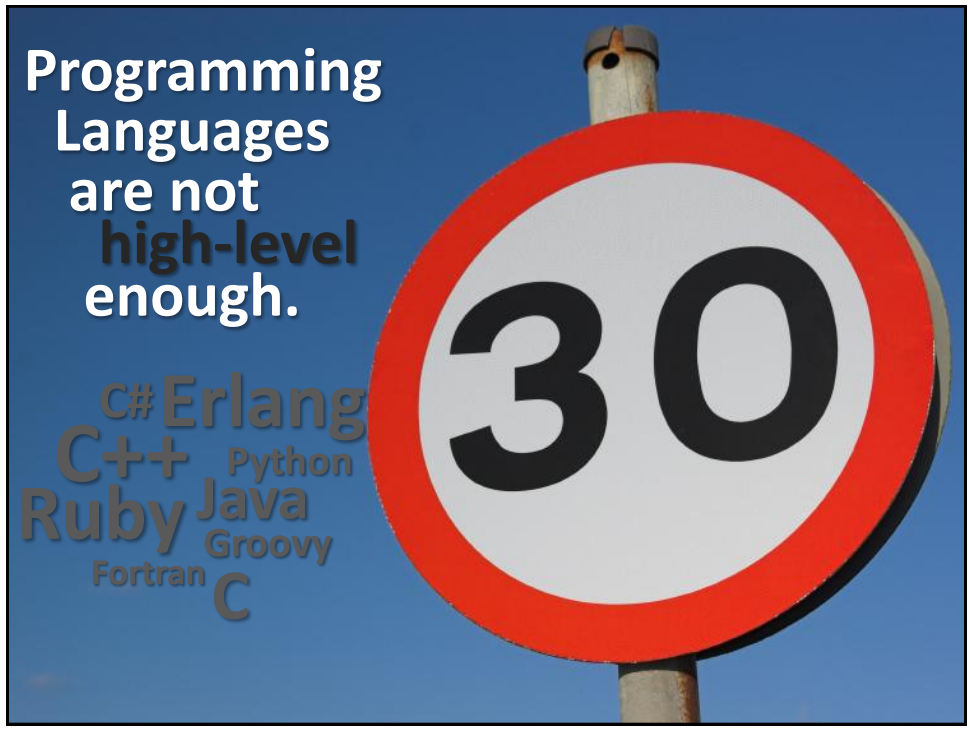
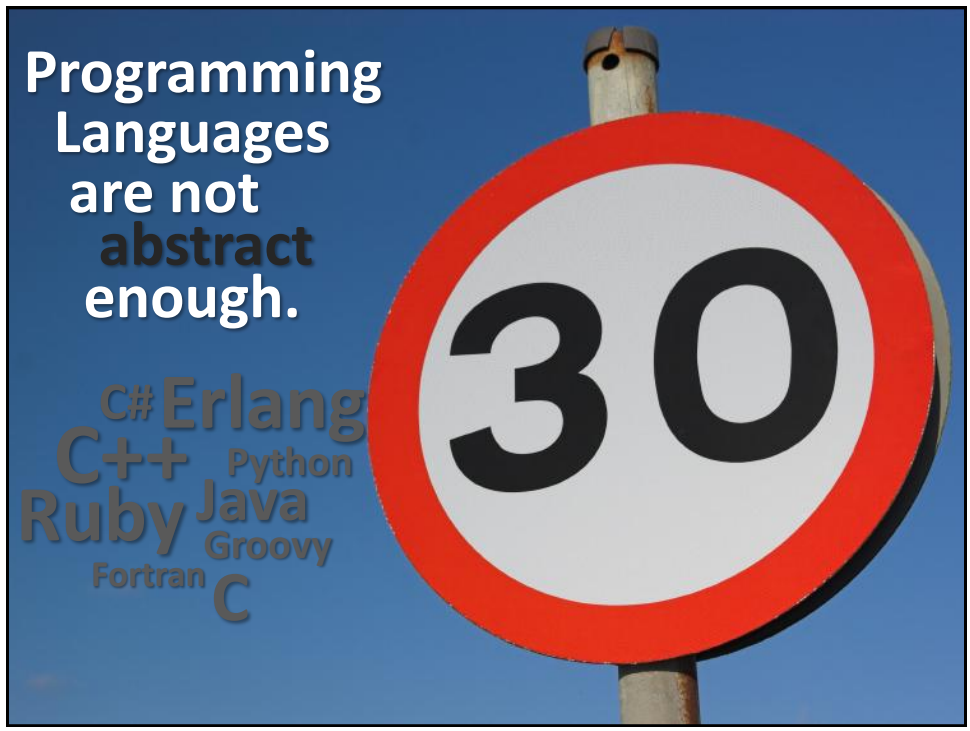
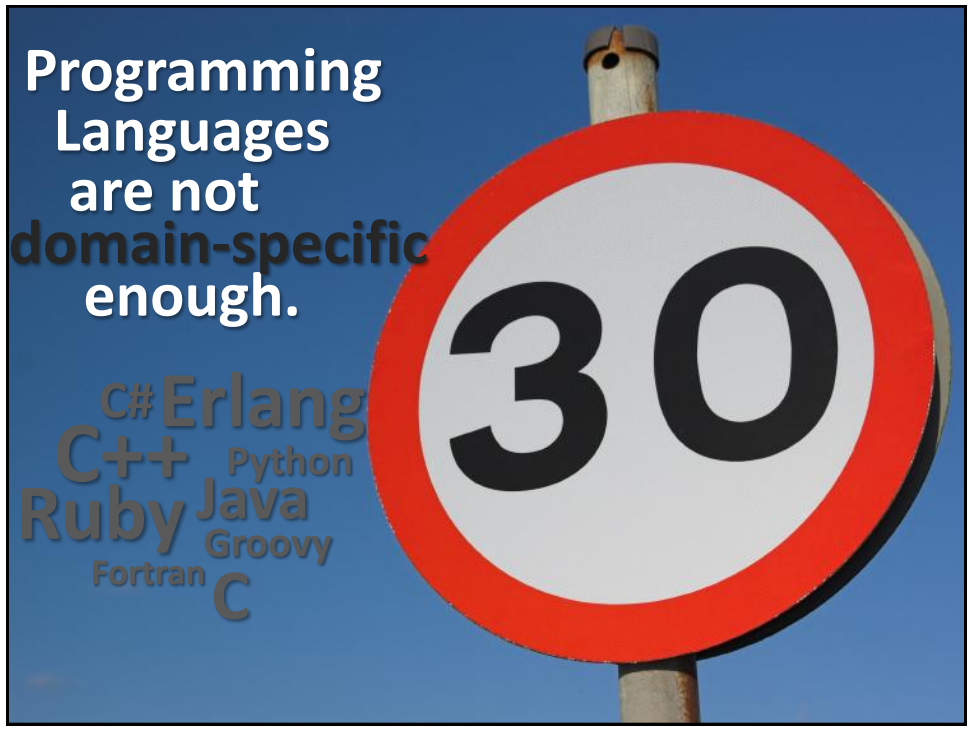
Software Engineering with
Domain-Specific Languages
and
Language Workbenches
Peter Friese
Itemis
peter.friese@itemis.de
Markus Voelter
Independent/itemis
voelter@acm.org
itemis

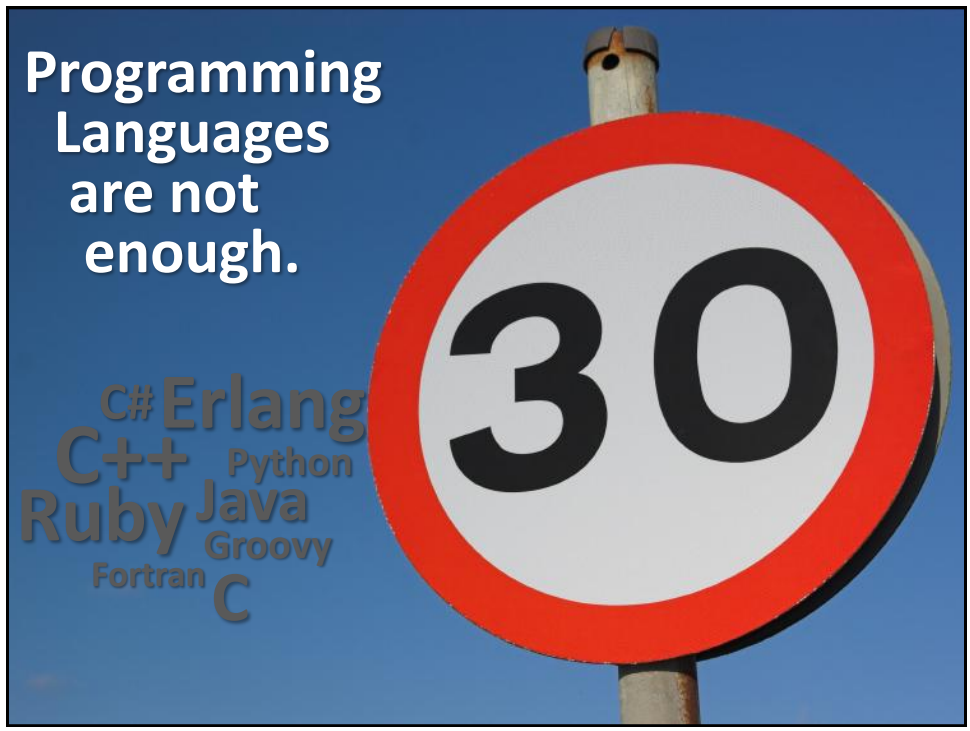

Programming
Languages
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C
30


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

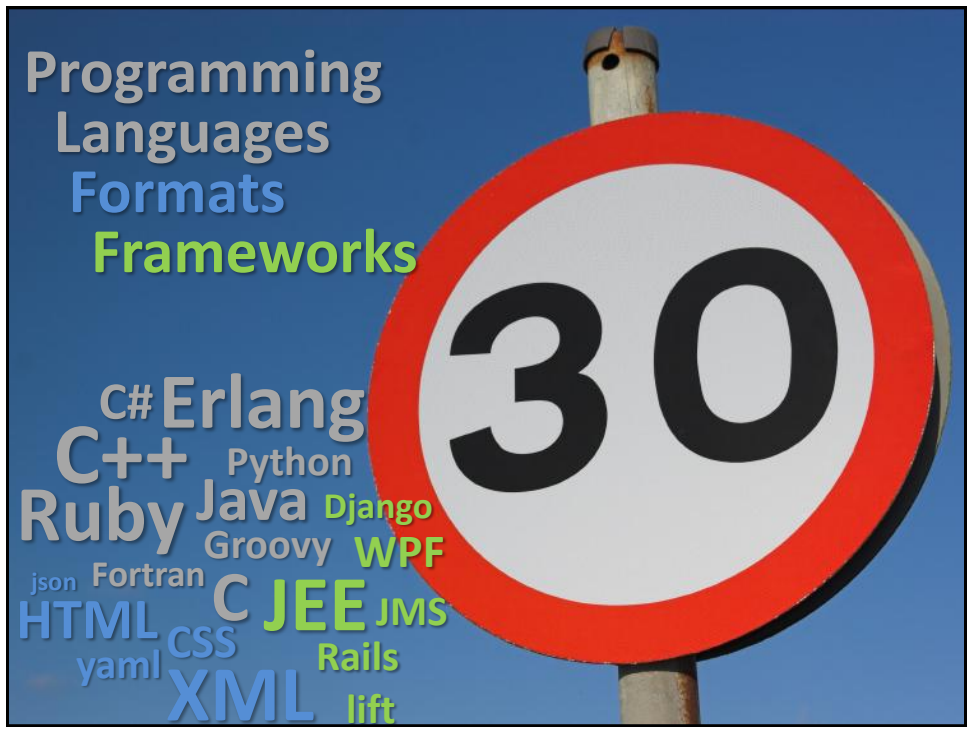

Programming
Languages
are not
high-level
enough.
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C
30

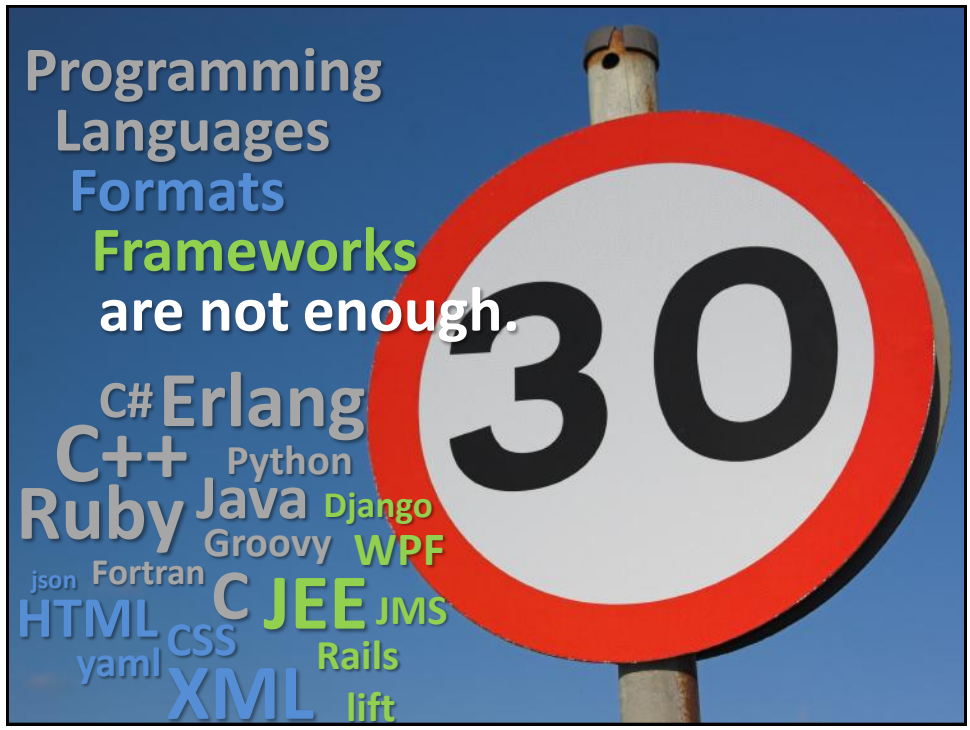

Programming
Languages
are not
abstract
enough.
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C
30


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

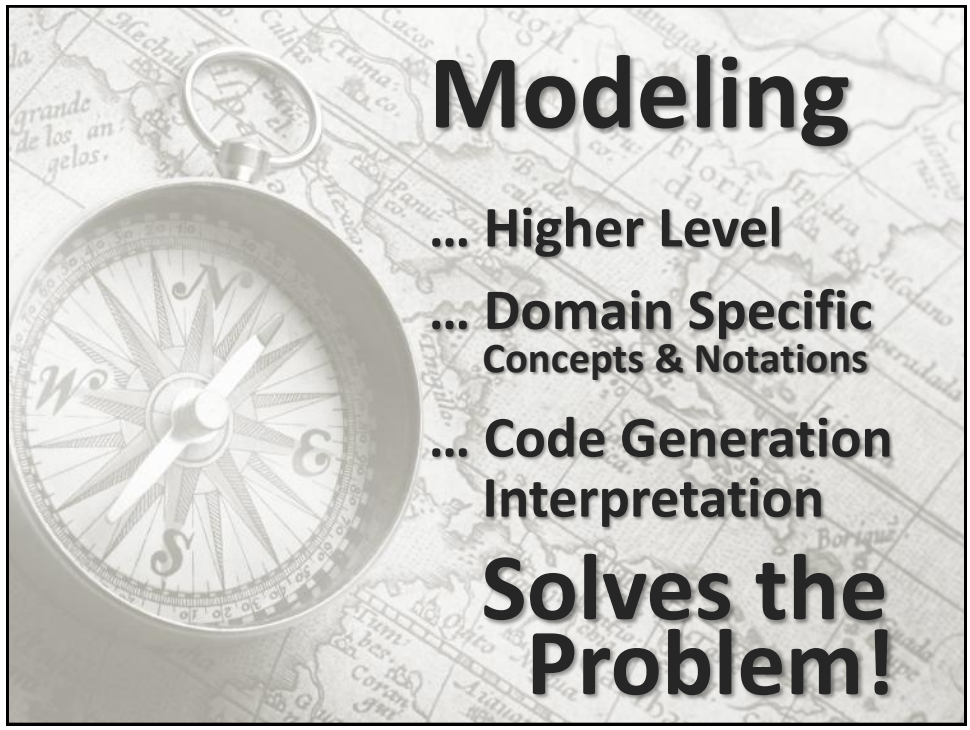

Programming
Languages
are not
enough.
C#
Erlang
C++
Python
Ruby
Java
Groovy
Fortran
C
30

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


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


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

So?


Modeling
… Higher Level
… Domain Specific
Concepts & Notations
… Code Generation
Interpretation
Solves the
Problem!

DSL

general purpose

domain specific

tailor made

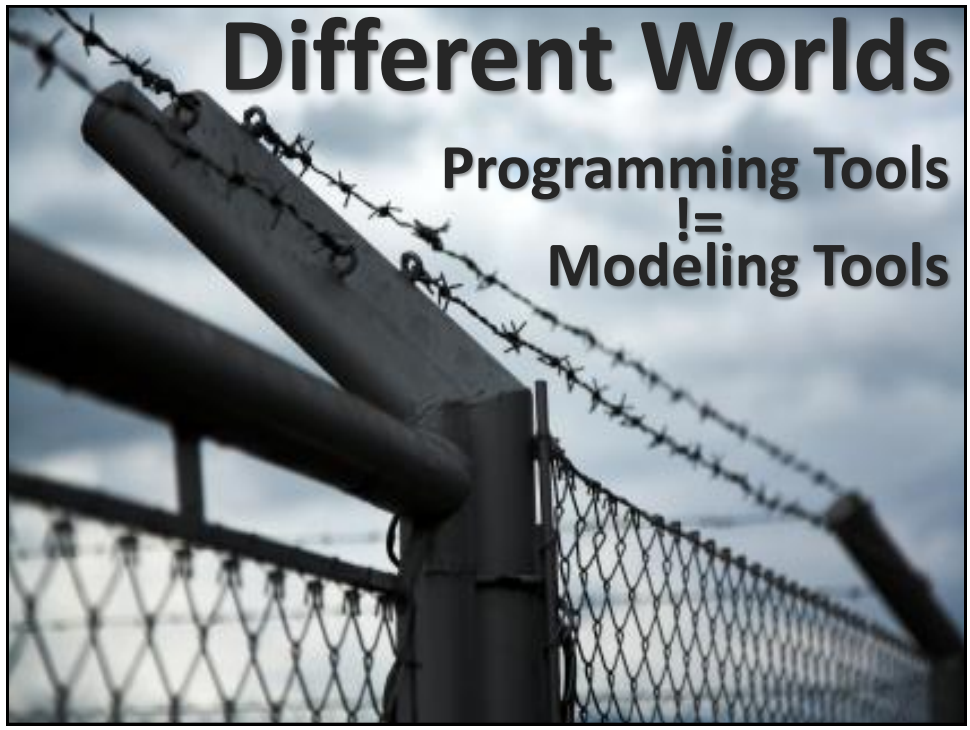
effective++

specialized, limited

used by experts

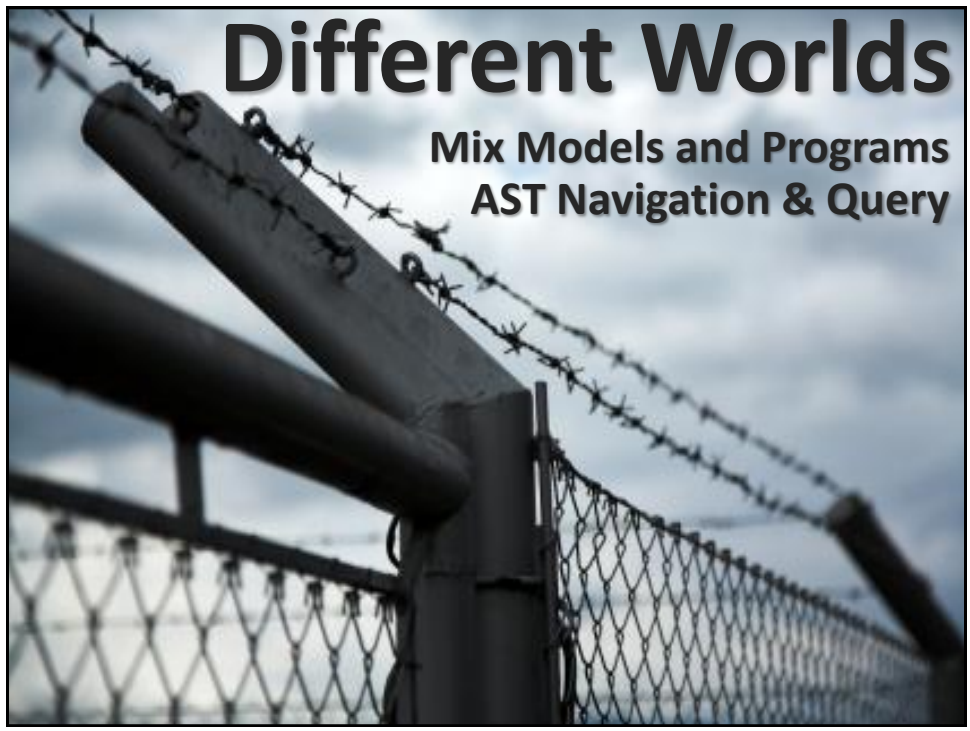
together with other specialized tools

But:


Different Worlds
Programming Tools
!=
Modeling Tools

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

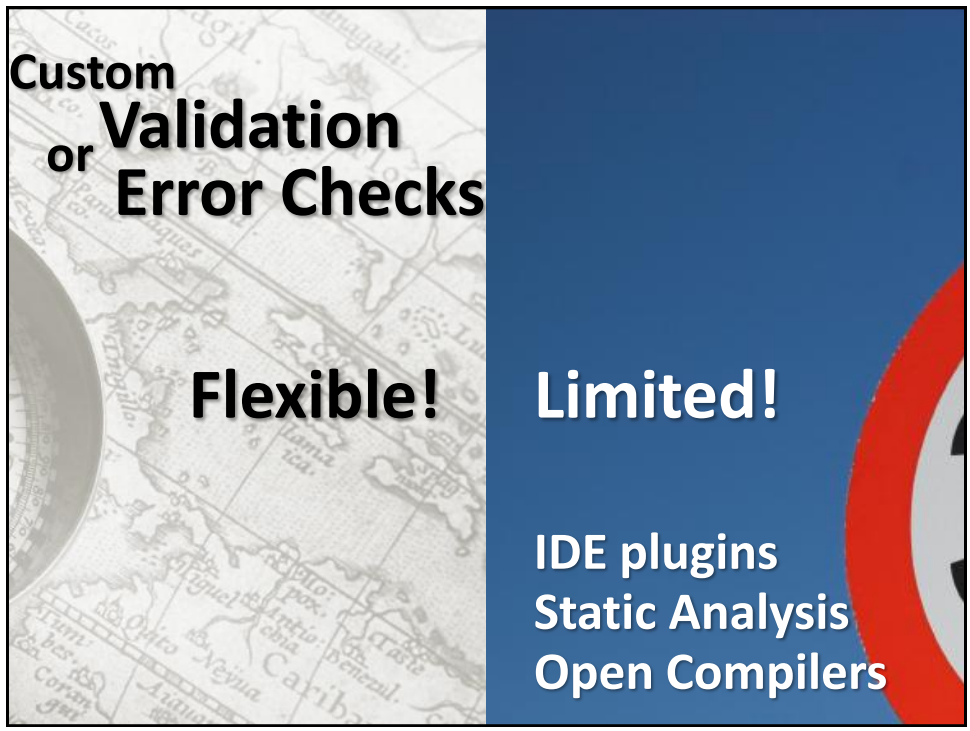
Compare

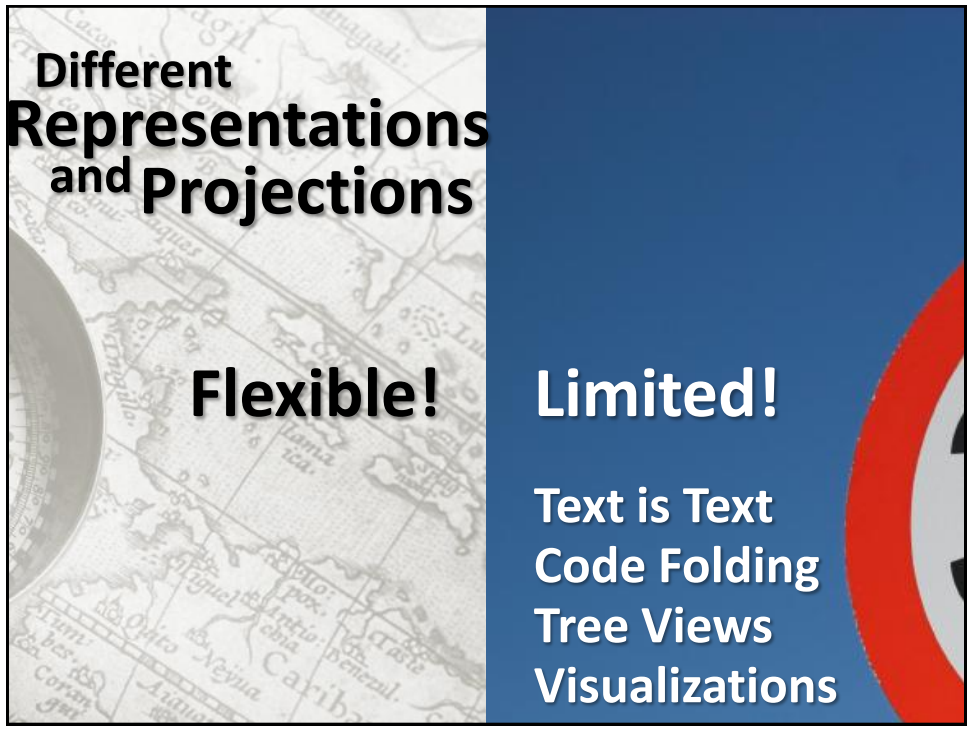
Domain Specific
Notations
and
Abstractions
Flexible!
Limited!
Frameworks
Libraries
(Fluent) APIs

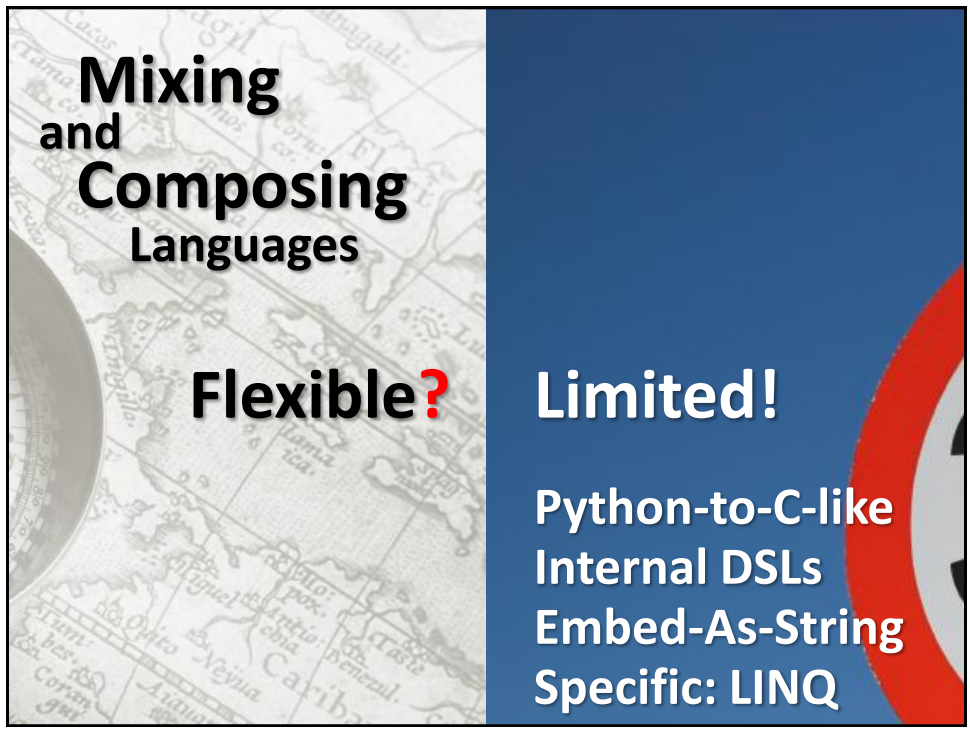
Graphical
Textual
Forms
Tables
Flexible!
Limited!
Textual
Trees

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

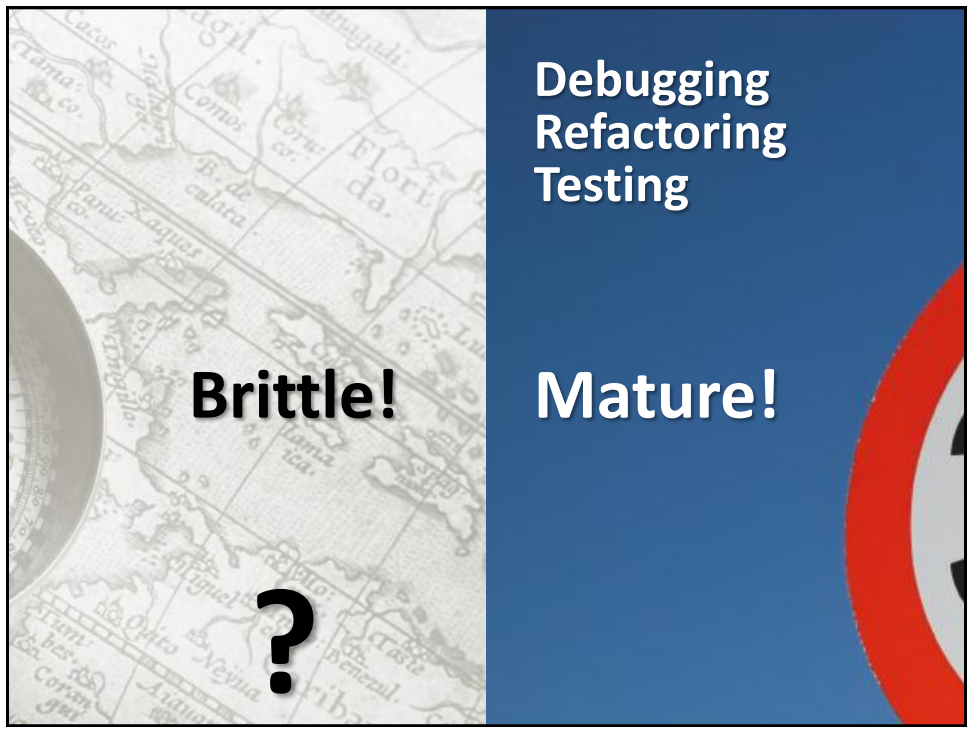
Define custom
Query
or Navigate
or Transform
Flexible!
Limited!
AST APIs
Static Analysis
Regex


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!
?

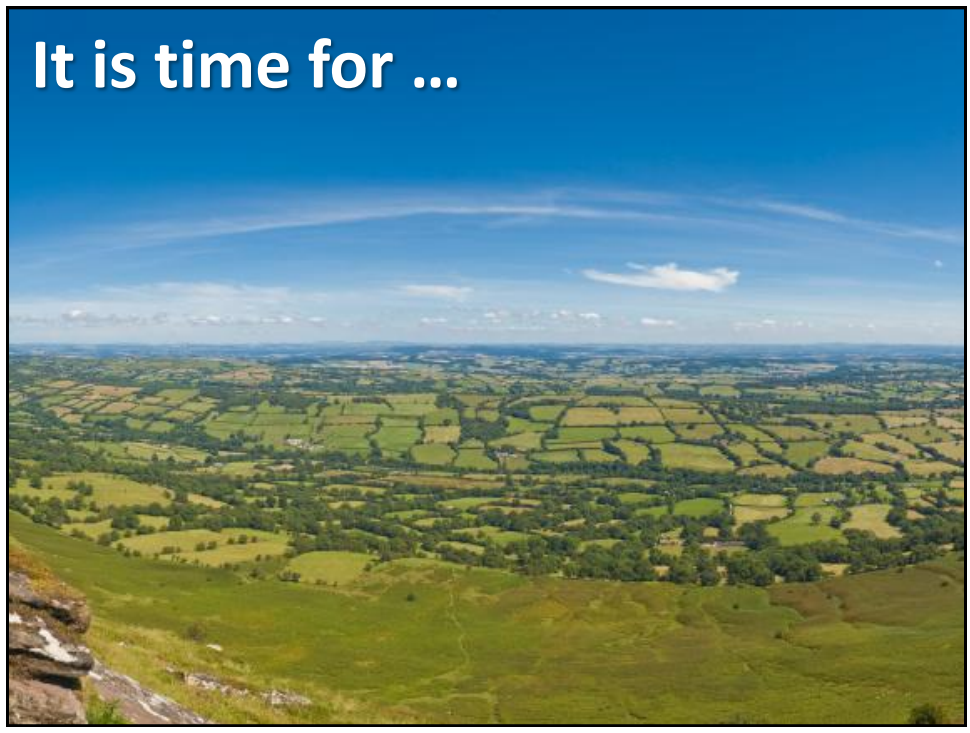
Versioning
Diff, Merge
Branching
Brittle!
Mature!
for some tools…

Gets some
Jobs done.
some people doubt that…
Gets the
Job Done!
everybody agrees…

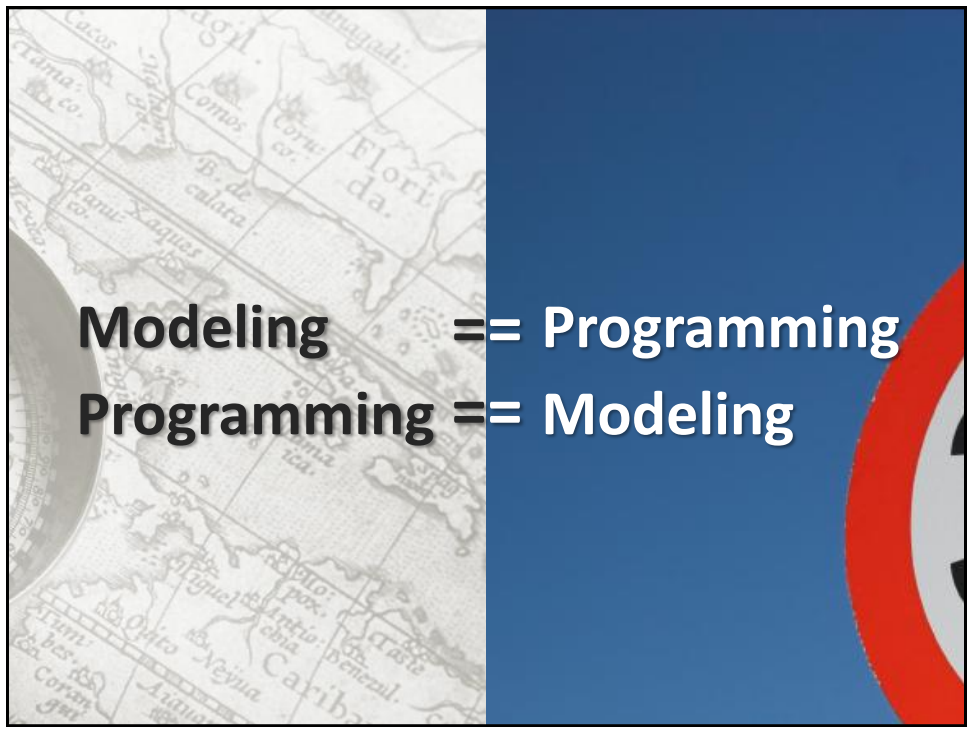
Why
the difference?

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**

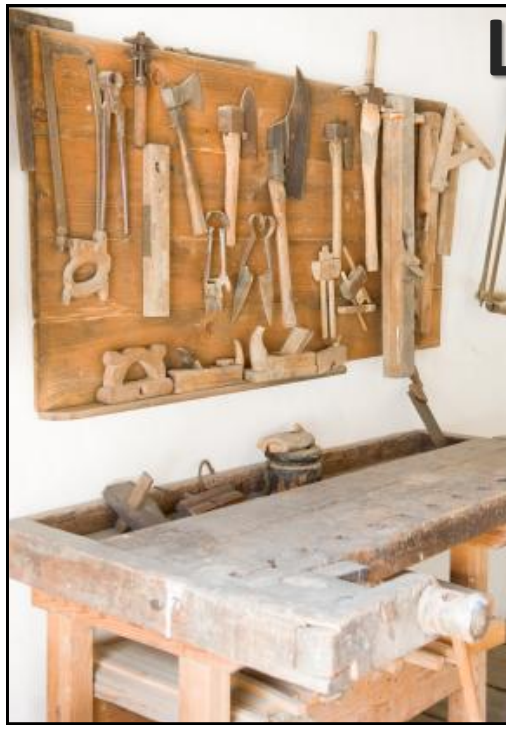
Language Workbench

(Martin Fowler)

Language Workbench

(Martin Fowler)

Freely
define
languages and
integrate
them



Language Workbench

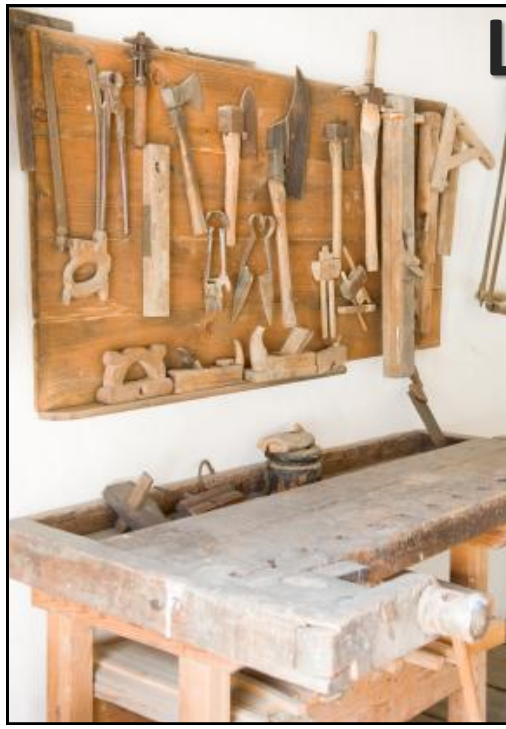
(Martin Fowler)

use
persistent?
abstract
representation

Language Workbench

(Martin Fowler)

language ::=
schema
+ editors
+ generators



Language Workbench

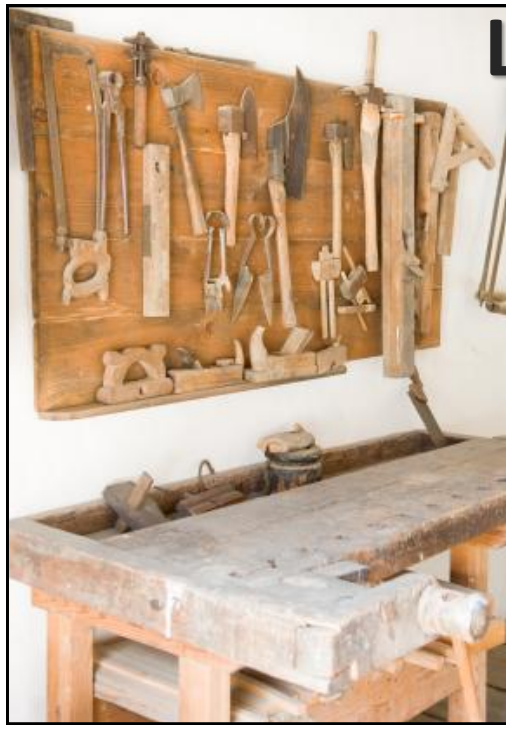
(Martin Fowler)

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

Syntax primarily textual

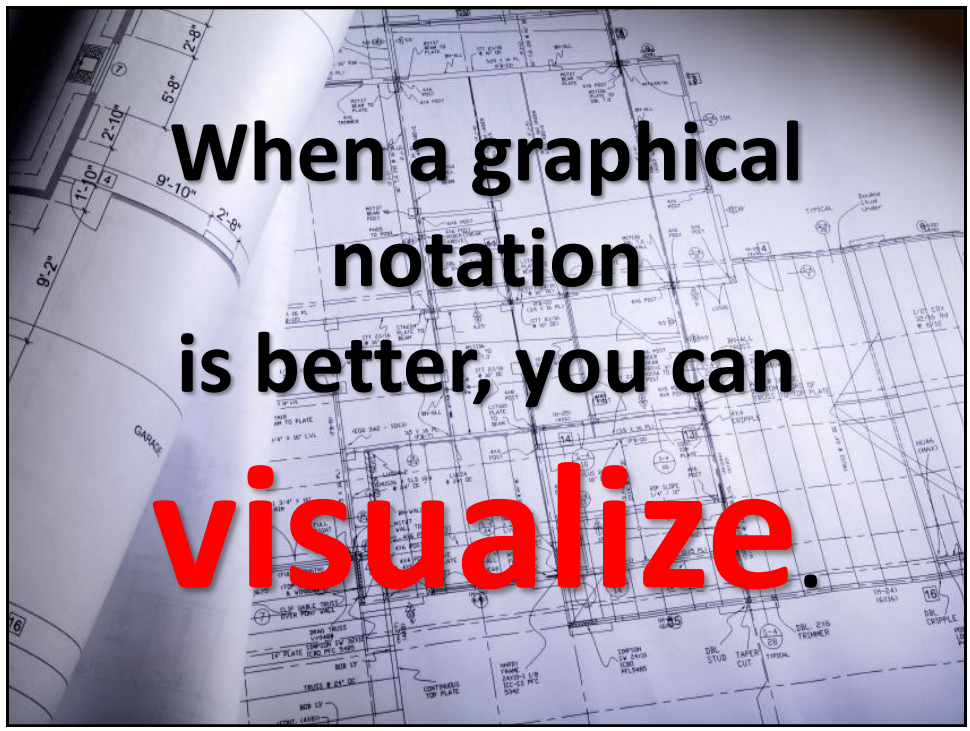
Syntax primarily textual

with more symbols

think: mathematics

Syntax
primarily
textual

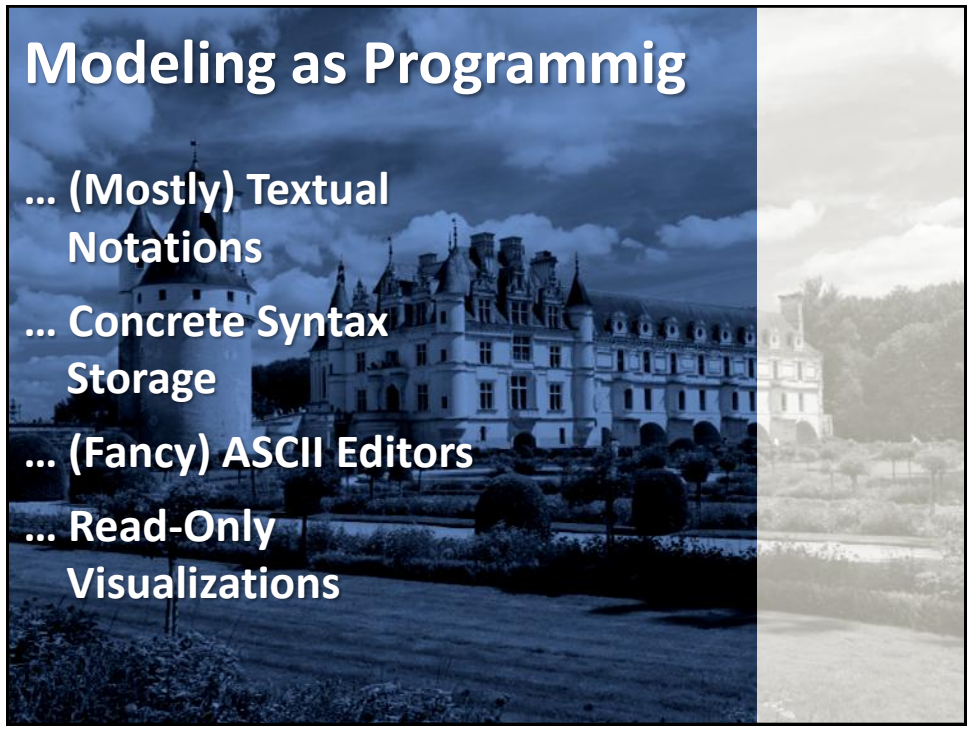
sometimes
box&line style



1 Available Tooling

Eclipse

Xtext



Custom
Syntax

Graphical
Textual
Symbolic++

IDE
Support

Teamwork
Debugging
Custom Editors

Complete
Symbolic Integration

Goto Def
Find Refs
Refactoring

Infrastructure Integration

… storage is text

… diff/merge with existing tools

… existing text tools work well!

Language Composition

Grammar composition with traditional parsers is tough!

More advanced parsers currently research

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

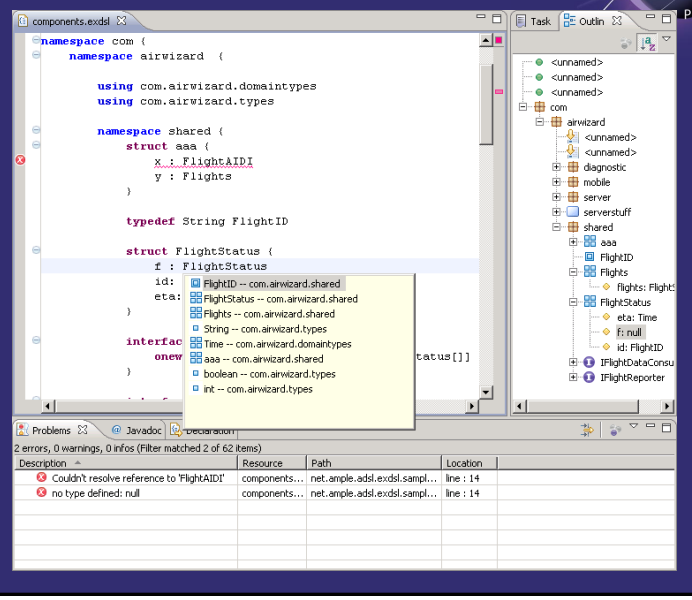
Xtext: Specify Grammar

eclipse modeling PROJECT

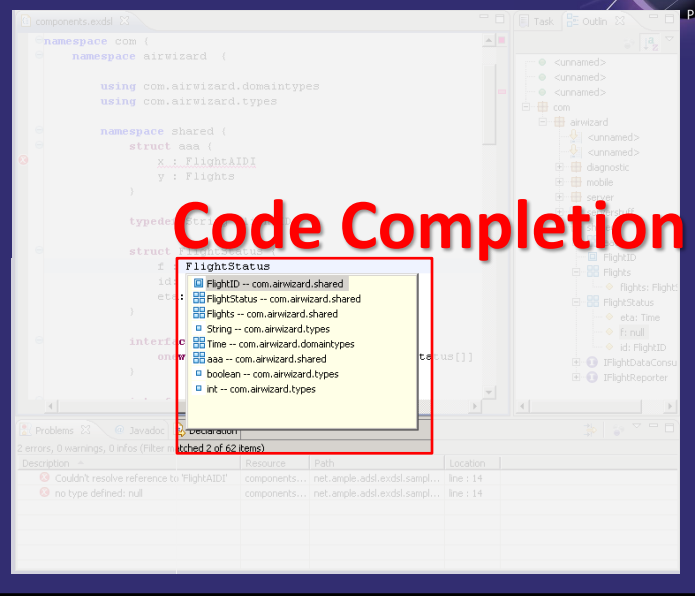
*exdsl.xtxt

```
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:
```

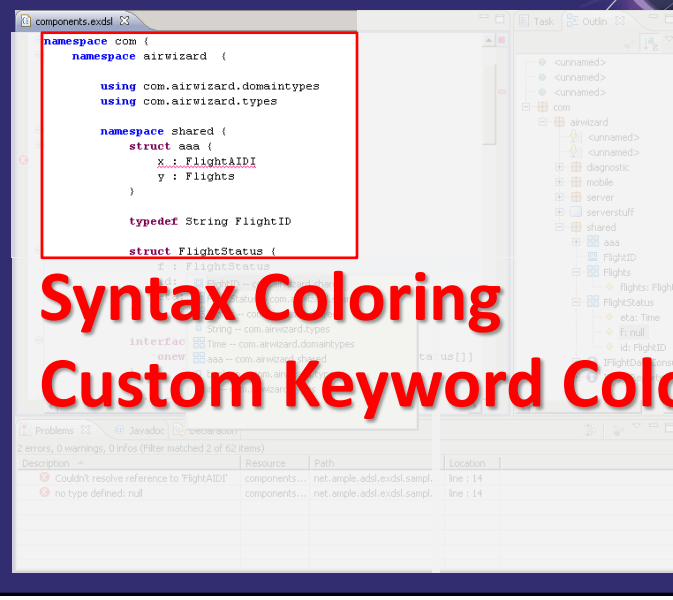
Xtext: Generated Editor



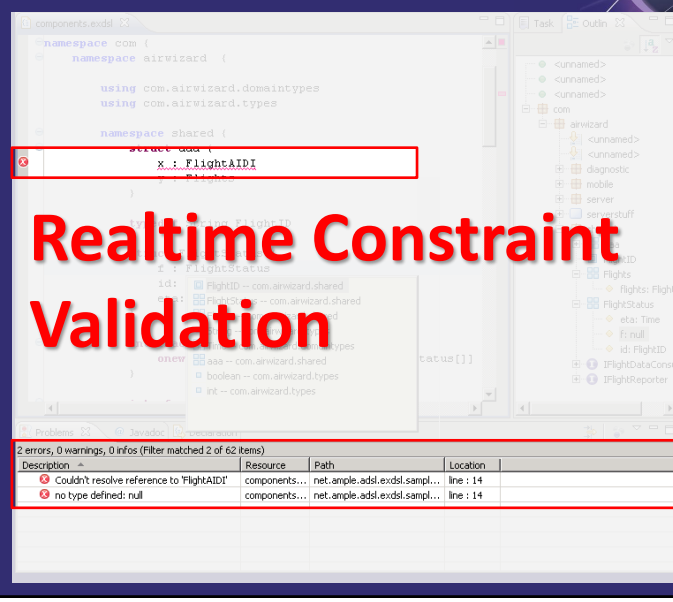
Xtext: Generated Editor



Xtext: Generated Editor

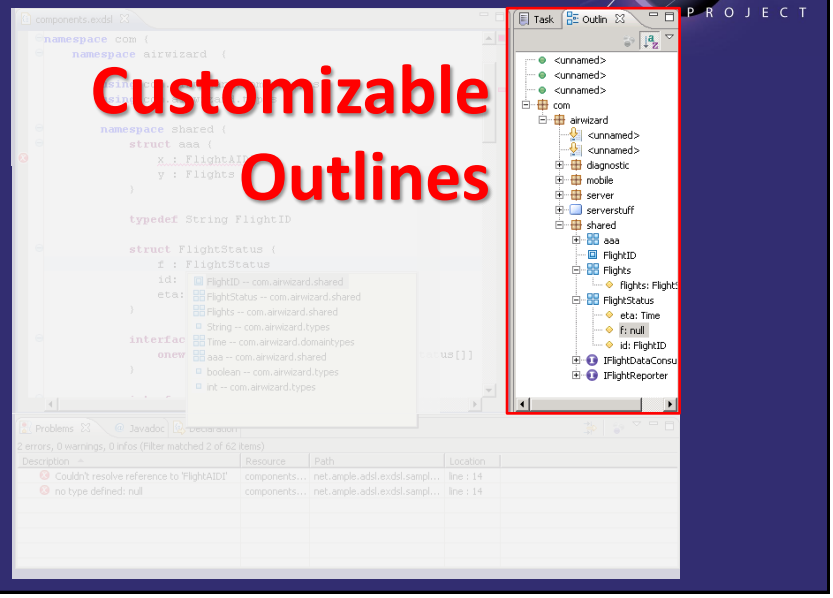


Xtext: Generated Editor



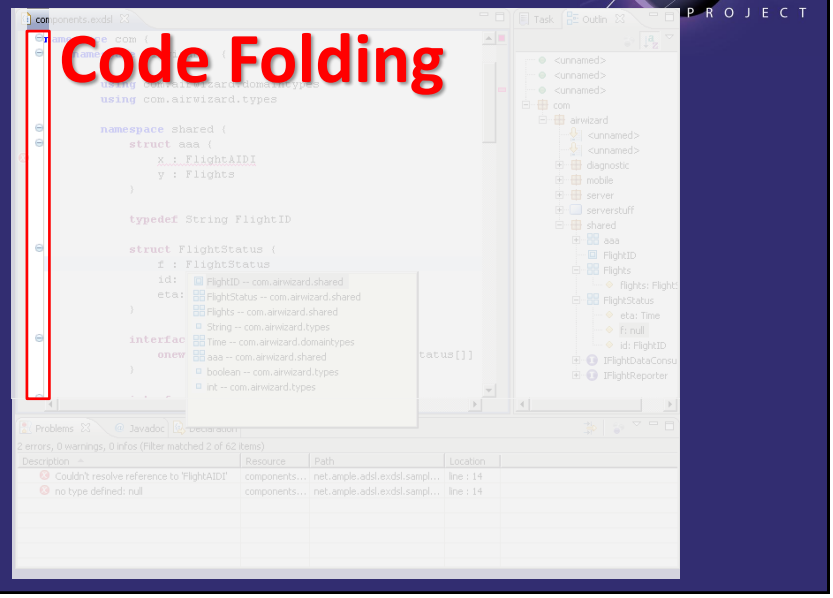
Xtext: Generated Editor

eclipse modeling PROJECT

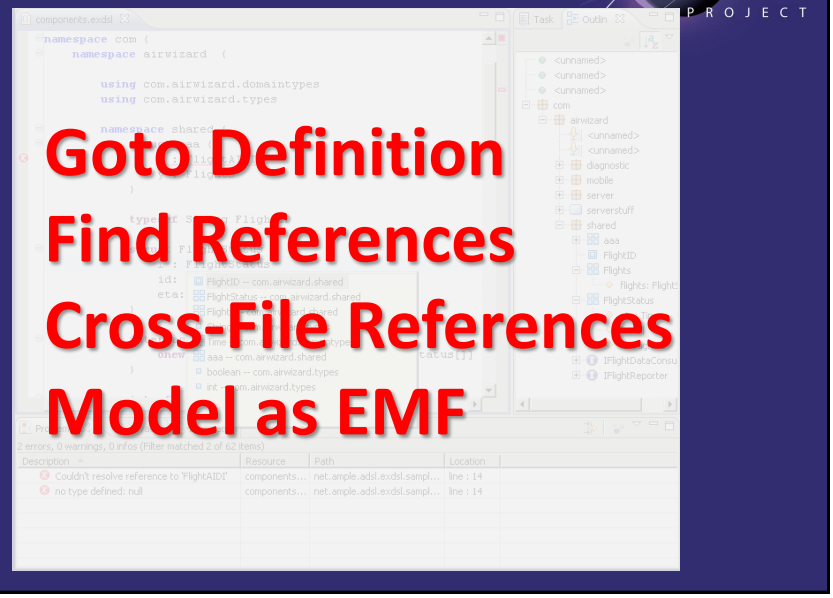


Xtext: Generated Editor

eclipse modeling PROJECT



Xtext: Generated Editor



DEMO I

Building DSLs with
Eclipse Xtext

2 Available Tooling

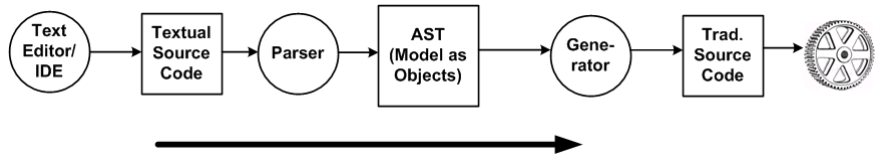
Jetbrains' Meta Programming System

Parser-based

text

… to tree

… to text



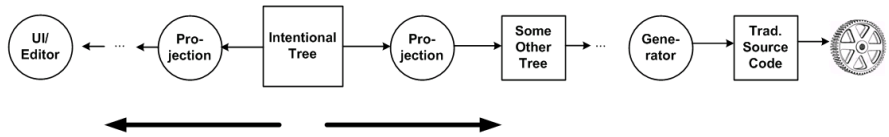
Projectional

tree

… to text-lookalike (editor)

… to other trees … [*]

… to text



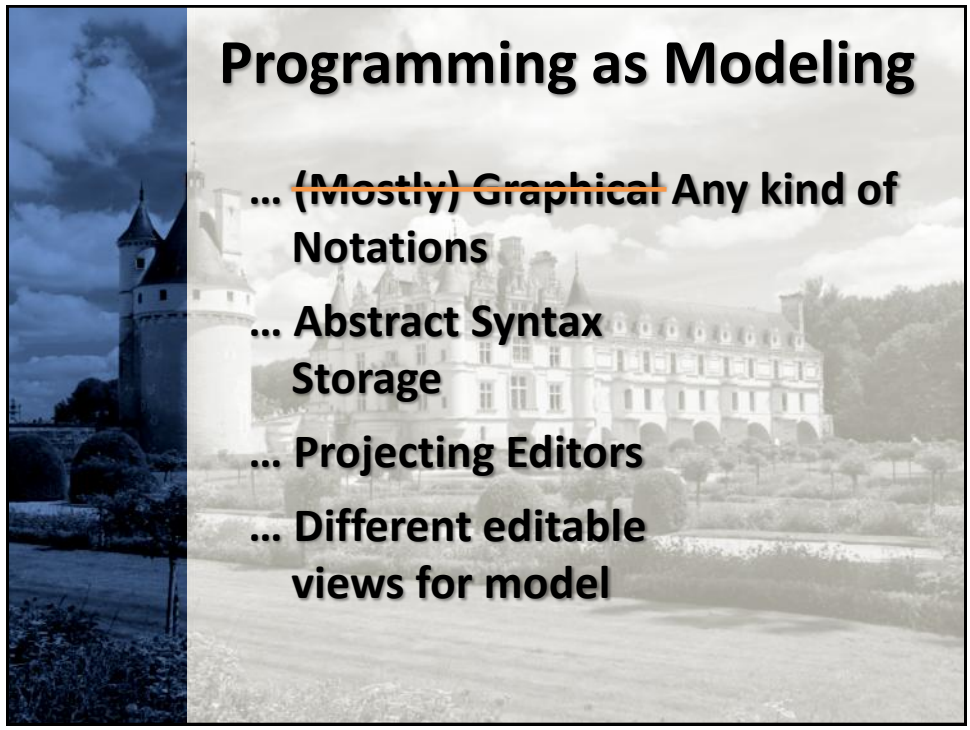
Programming as Modeling

… (Mostly) Graphical Notations

… Abstract Syntax Storage

… Projecting Editors

… Different editable views for model



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

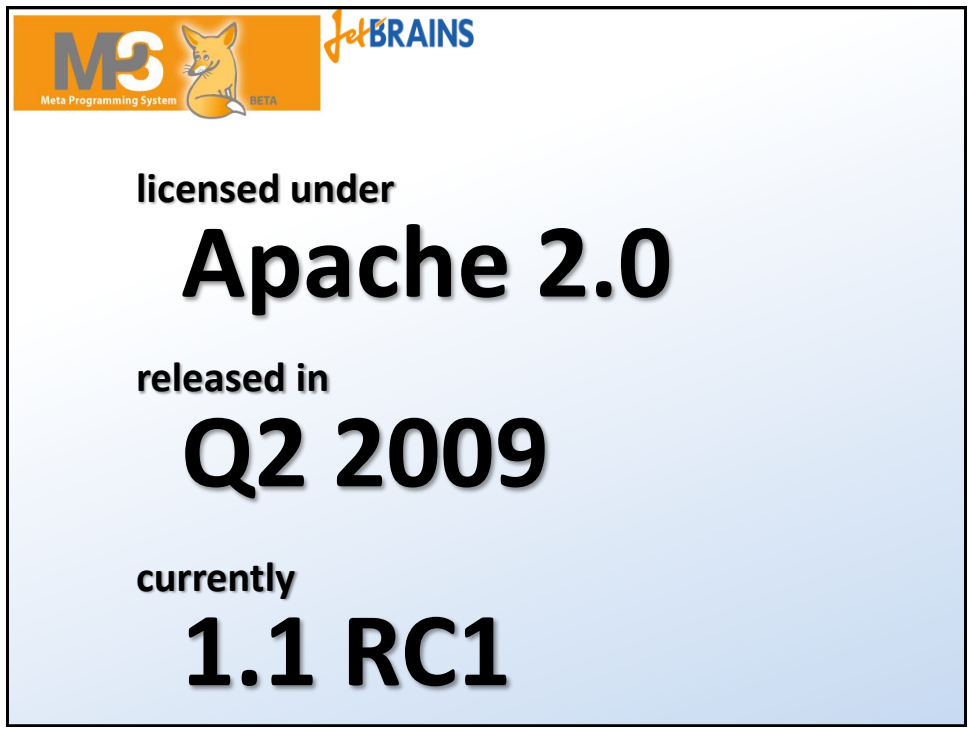
… diff/merge must be in tool

… existing text tools don‘t work

Proprietary
Tools

… no standards

… no interop


MPS
Meta Programming System
BETA
JetBRAINS
licensed under
Apache 2.0
released in
Q2 2009
currently
1.1 RC1

Build new standalone DSLs

Build new standalone DSLs

Build DSLs that reuse parts of other languages

Build new standalone DSLs

Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage / Java++)

extend base language

Build new standalone DSLs

Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage / Java++)

extend base language

build DSLs that reuse parts of BaseLanguage

Language Extension Example

Language Extension Example

Old

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

Language Extension Example

Old

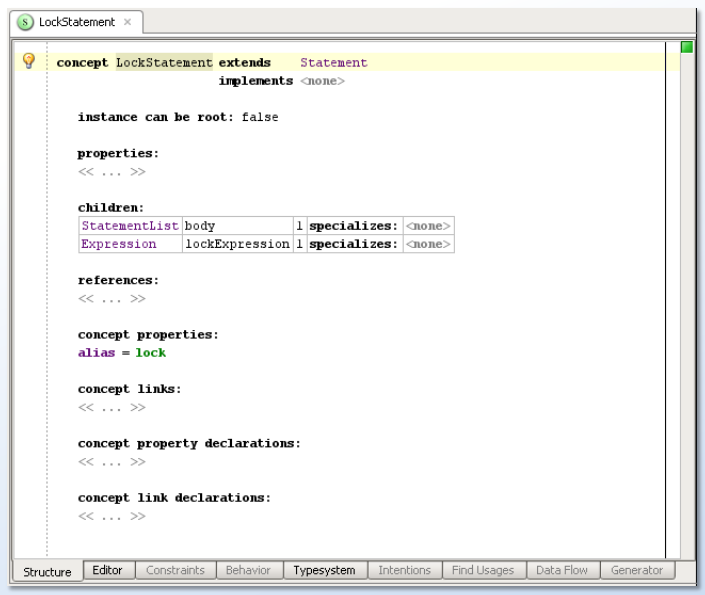
```
ReadWriteLock l = …
l.readLock().lock();
try {
  //code
} finally {
  l.readLock().unlock();
}
```

New

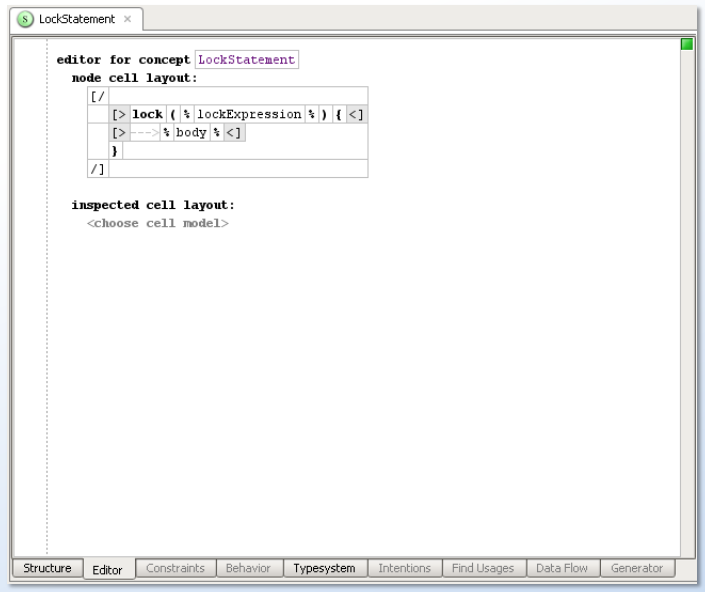
```
ReadWriteLock l = …
lock (l) {
  //code
}
```

Structure ◆ Editor ◆ Typesystem ◆ Generator

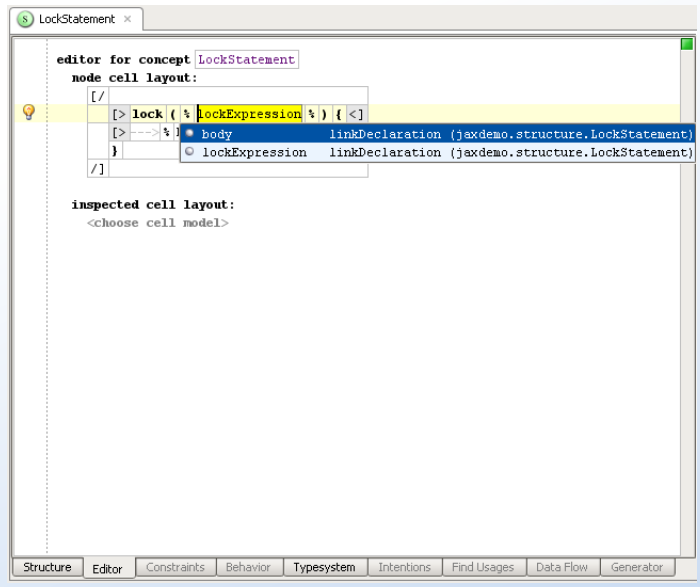
Structure ◆ Editor ◆ Typesystem ◆ Generator



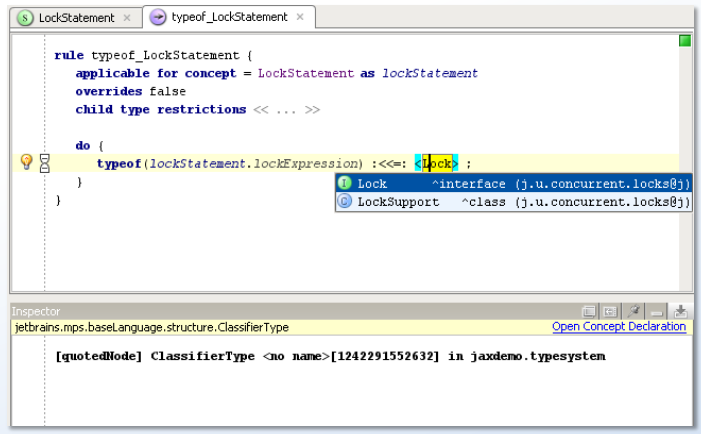
Structure ◆ Editor ◆ Typesystem ◆ Generator



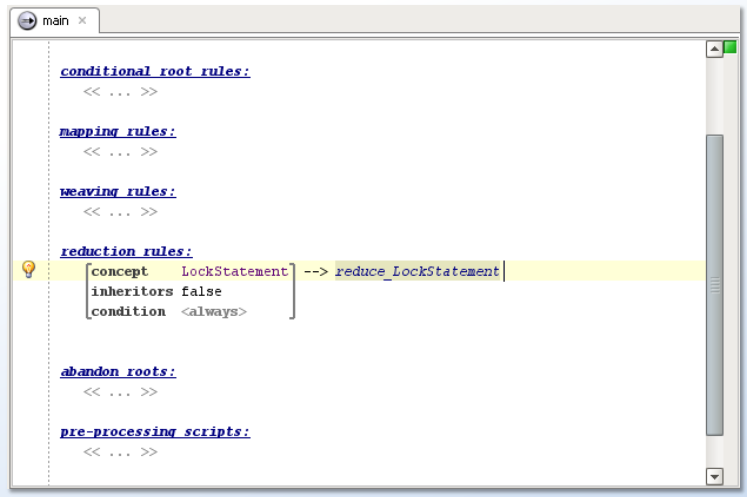
Structure ◆ Editor ◆ Typesystem ◆ Generator



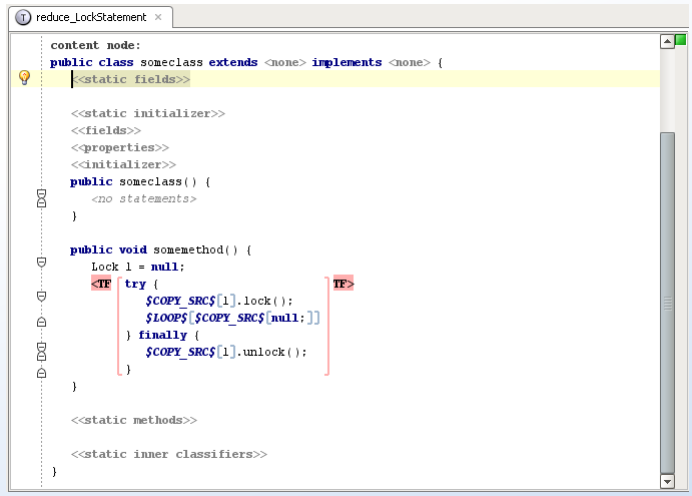
Structure ◆ Editor ◆ Typesystem ◆ Generator



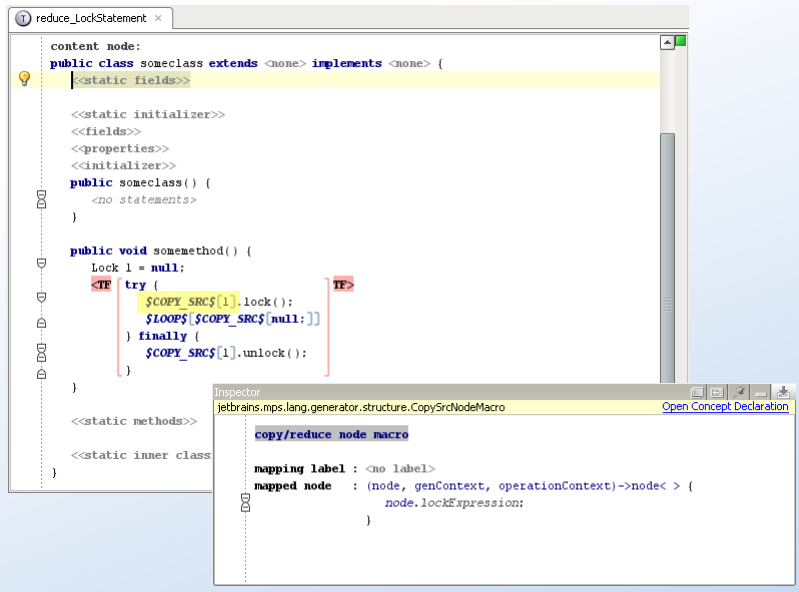
Structure ◆ Editor ◆ Typesystem ◆ Generator



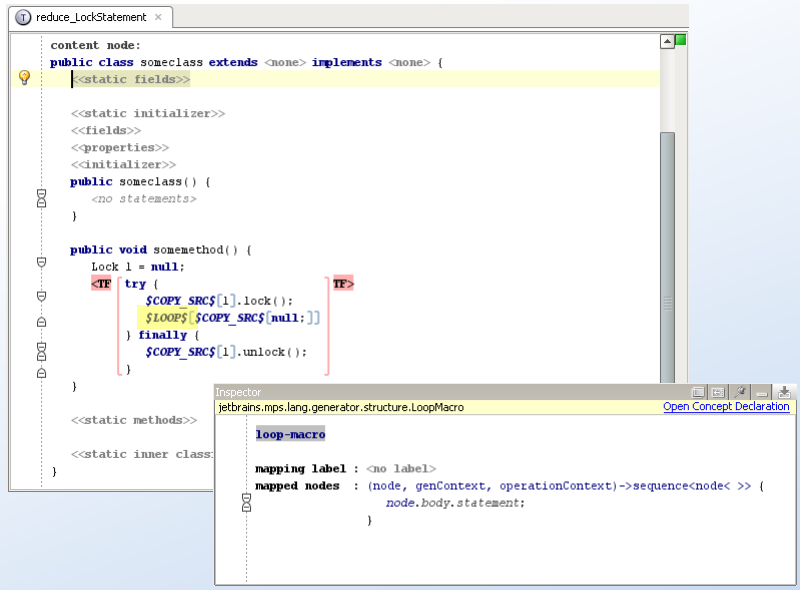
Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator



Structure ◆ Editor ◆ Typesystem ◆ Generator

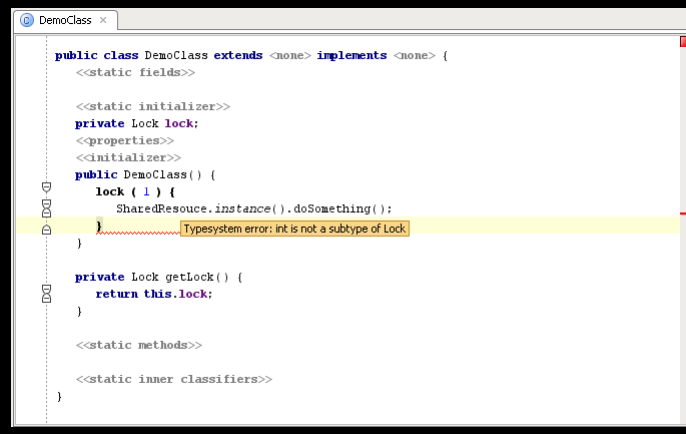


Language Extension Example

Result behaves like a native base language construct

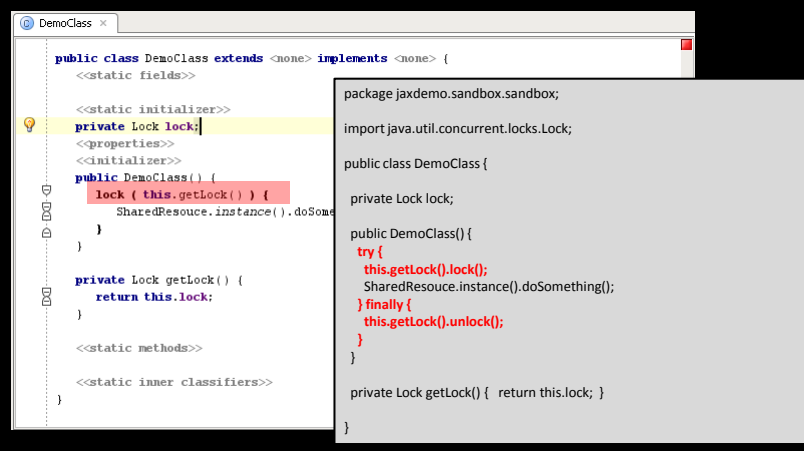
Language Extension Example

Result behaves like a native base language construct



Language Extension Example

Translated to regular Java code based on the generator



DEMO II

Building DSLs with
JetBrains MPS

THE END.