Markus Voelter
Independent/itemis
voelter@acm.org

Textual DSLs and Code Generation with Eclipse and oAW

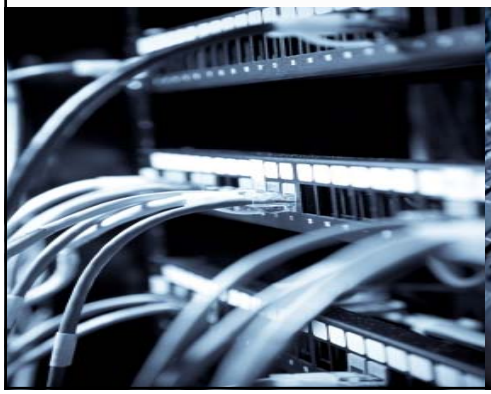
1

Context: Architecture DSLs

As you
understand
and
develop
your architecture…

…develop a
language
to express it

The language resembles
architectural
concepts…



and we describe
the application(s)
with the language

Architecture
DSL

2

Architecture DSL Example

Components

```
component DelayCalculator {
  provides IDelayCalculator
  requires IInfoScreen
}
component InfoScreen {
  provides IInfoScreen
}
component AircraftModule {
  provides IAircraftModule
  requires IDelayCalculator
}

interface IDelayCalculator {}
interface IInfoScreen {}
interface IAircraftModule {}
```

```
namespace com.mycompany.test {
  system testSystem {
    instance dc: DelayCalculator
    instance screen1: InfoScreen
    instance screen2: InfoScreen
    connect dc.screens
         to (screen1.default, screen2.default)
  }
}
```

Data Replication

```
struct FlightInfo {
  from: Airport
  to: Airport
  scheduled: Time
  expected: Time
  …
}

replicated singleton flights {
  flights: FlightInfo[]
}

component DelayCalculator {
  publishes flights
}

component InfoScreen {
  consumes flights
}
```

Pre- and Post-Conditions

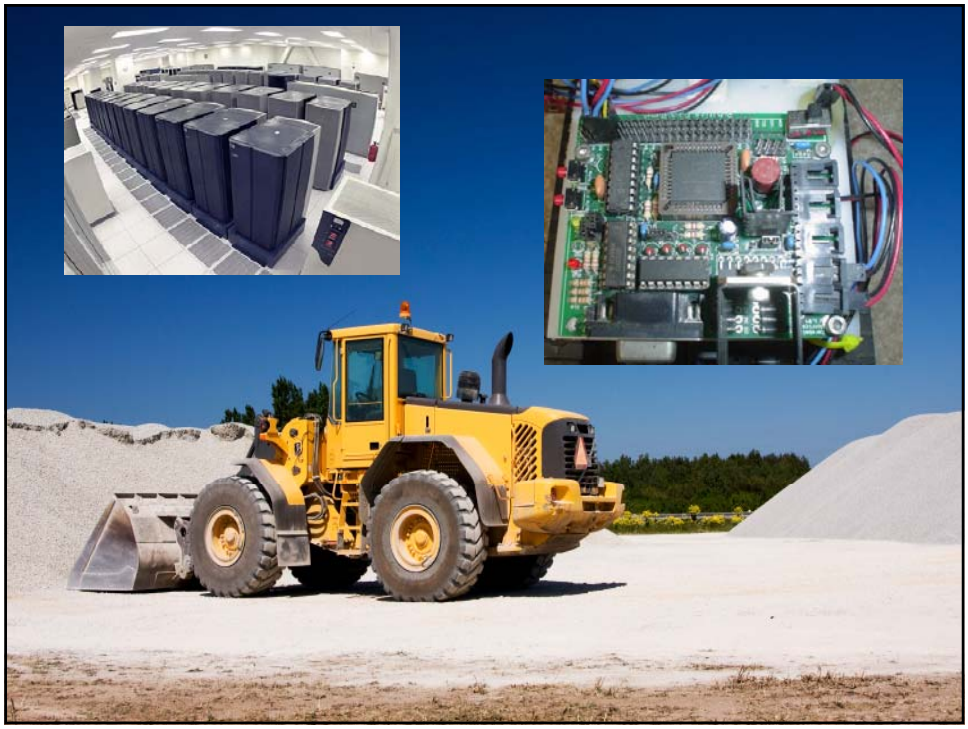
```
interface IAircraftStatus {

  oneway message reportPosition(aircraft: ID,
                                pos: Position ) {
    pre aircraft != null: “aircraft not specified”
    pre pos != null: “position not specified”
  }

  request-reply message reportProblem {
    request (aircraft: ID, problem: Problem,
                           comment: String)
    reply (repairProcedure: ID)
    pre aircraft != null: “aircraft not specified”
    pre problem != null: “problem not specified”
    post repairProcedure != null
  }

}
```

Message Sequences: Protocol State Machines

```
interface IAircraftStatus {
  oneway message registerAircraft(aircraft: ID! )
  oneway message unregisterAircraft(aircraft: ID! )
  oneway message reportPosition(aircraft: ID!,
                pos: Position! )
  request-reply message reportProblem {
    request (aircraft: ID!, problem: Problem!,
             comment: String!)
    reply (repairProcedure: !ID)
  }
  protocol initial = new {
    state new {
      registerAircraft => registered
    }
    state registered {
      unregisterAircraft => new
      reportPosition
      reportProblem
    }
  }
}
```

Based on actual practical experience

Currently in use with four of my customers



Why Textual?

… or: why not graphical?

Languages and Editors are easier to build

Languages and Editors are easier to build

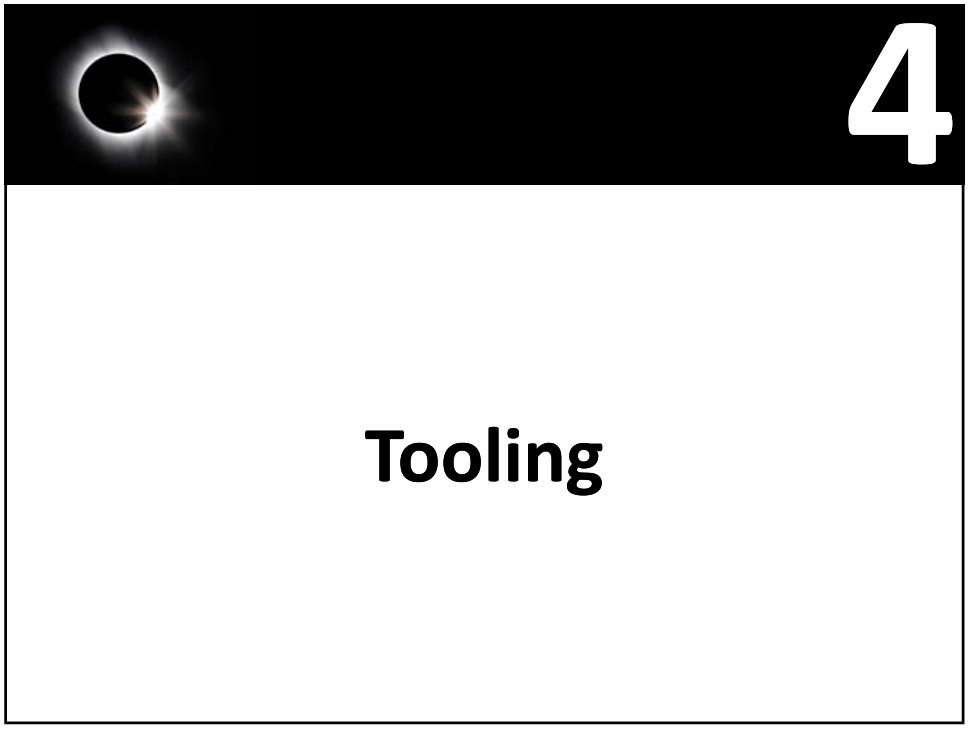
Evolve Language and simple editor as you understand and discuss the architecture, in real time!

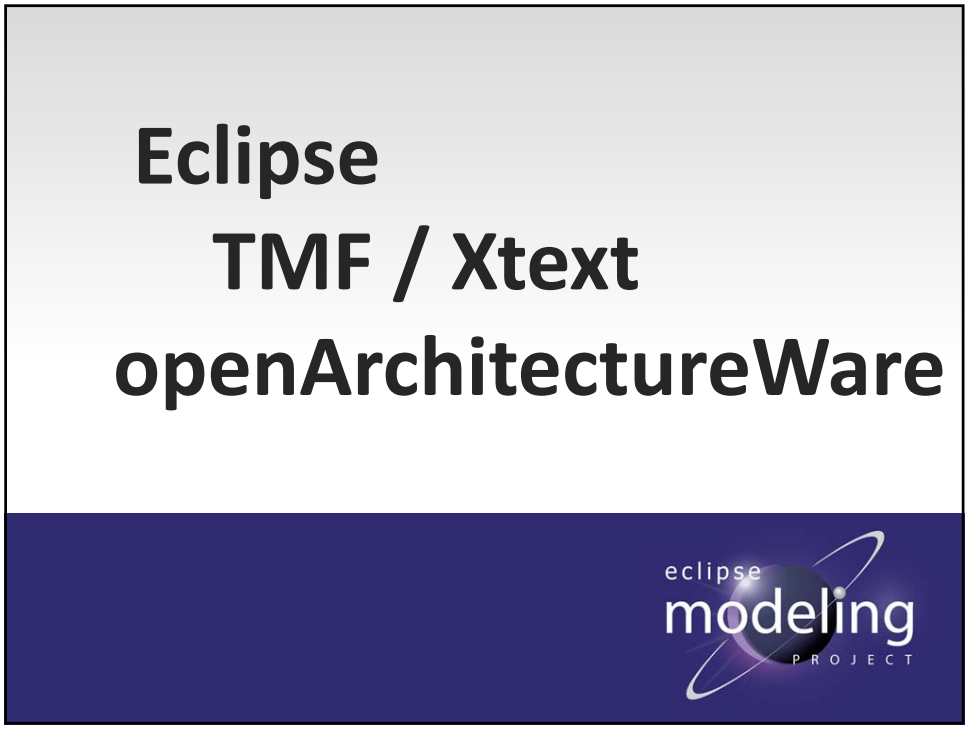
Integrates easily with current infrastructure: CVS/SVN diff/merge

adapting existing models as the DSL evolves

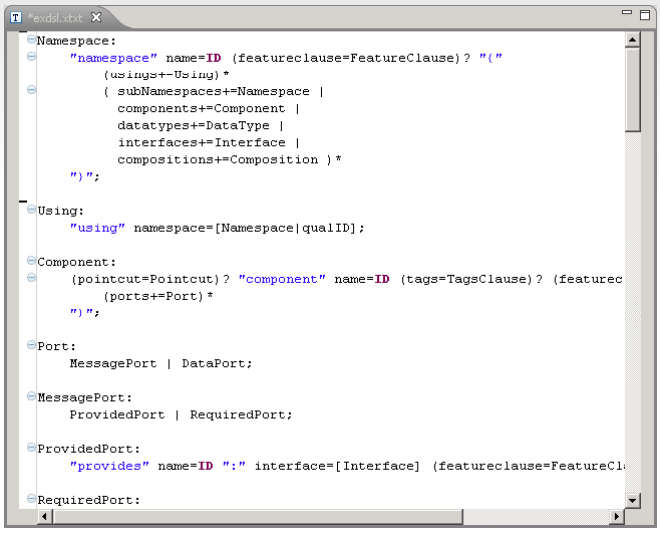
Model evolution is trivial, you can always use *grep*.

Many developers prefer textual notations


4
Tooling

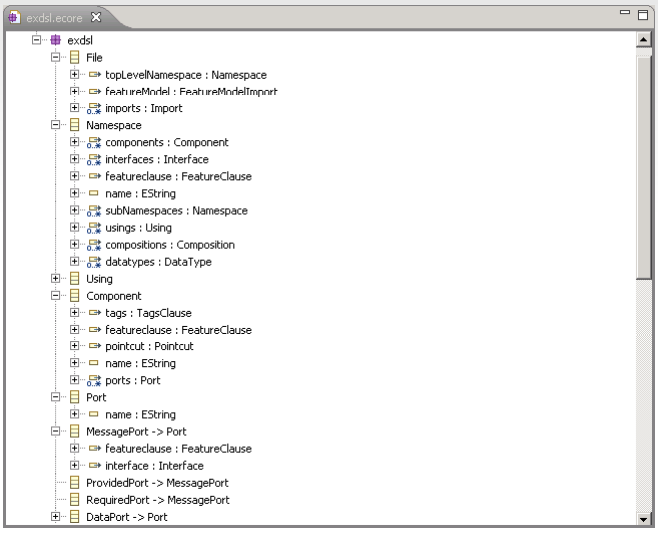

Eclipse
TMF / Xtext
openArchitectureWare
eclipse
modeling
PROJECT

Specify Grammar

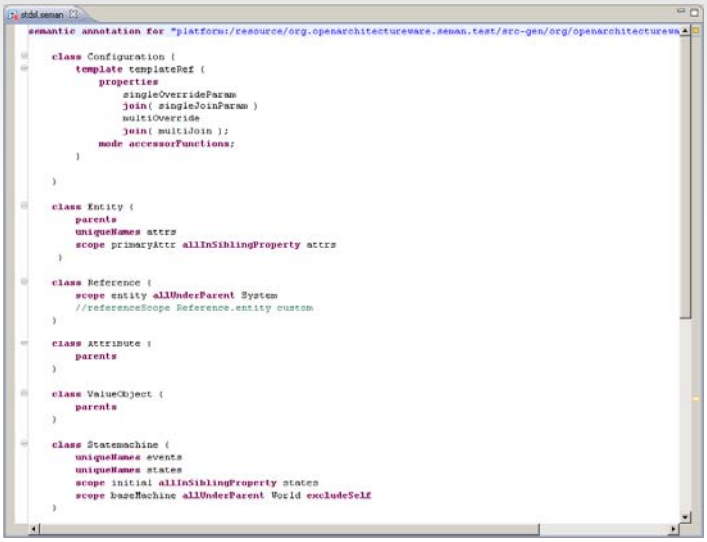


Antlr Grammar and Parser is generated from this specification

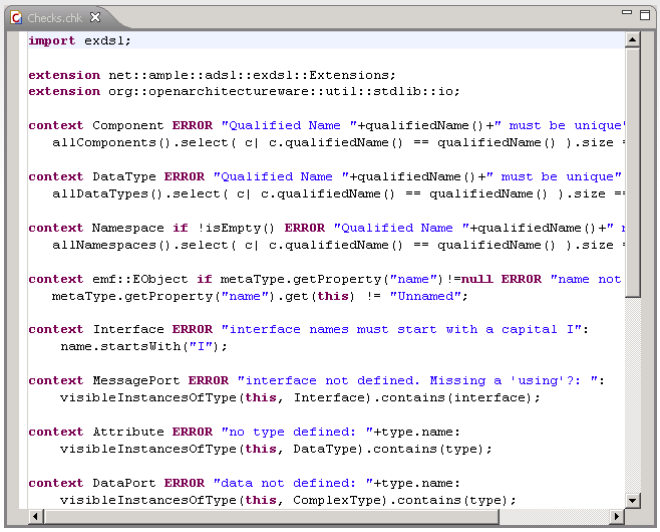
Generated Metamodel



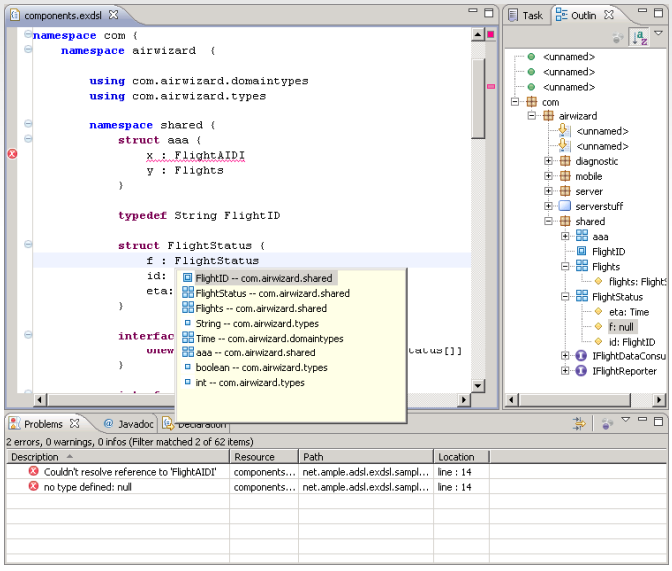
Specify Semantics



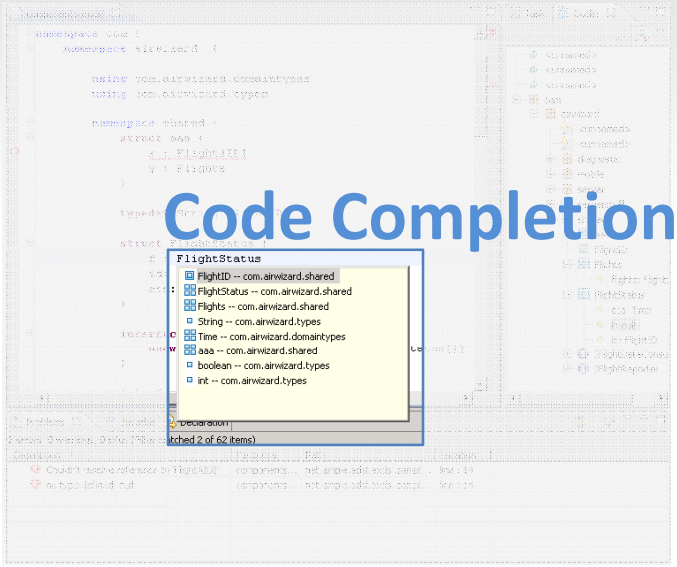
Specify Constraints



Generated Editor

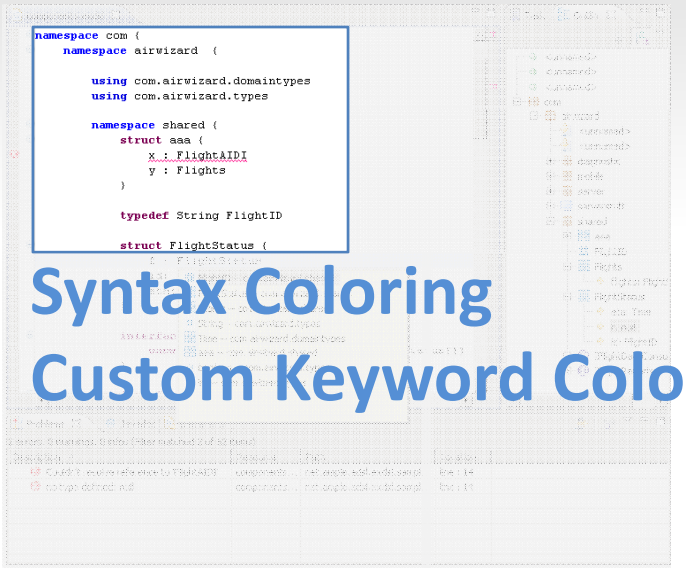


Generated Editor



Code Completion

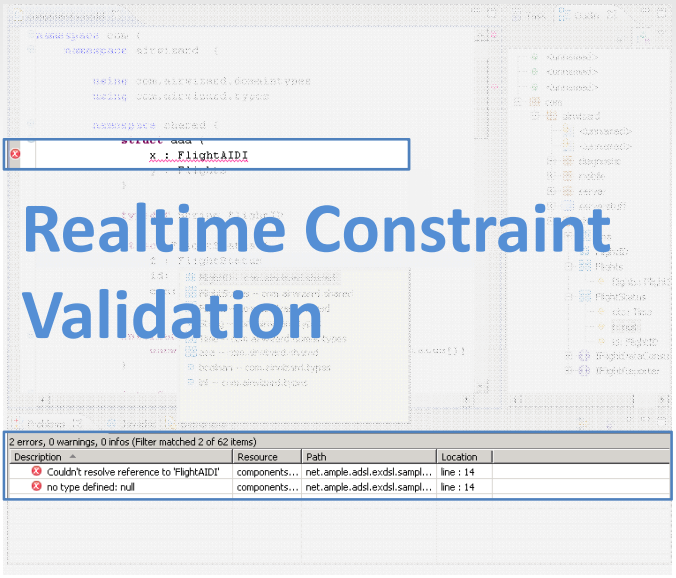
Generated Editor



Syntax Coloring

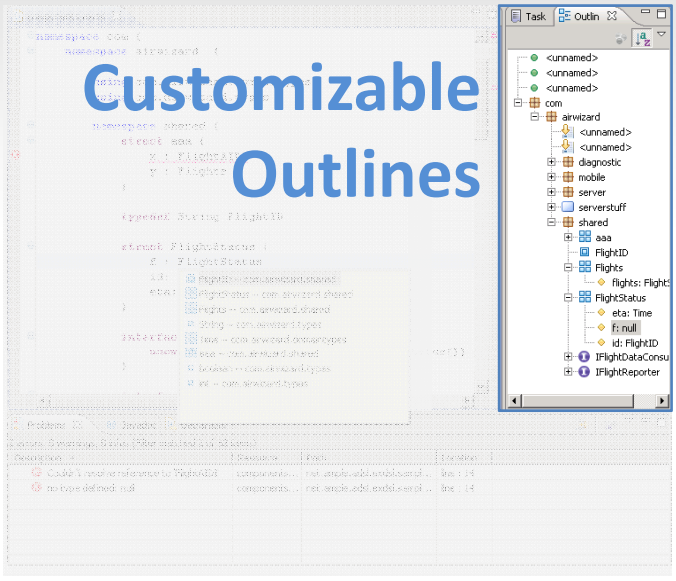
Custom Keyword Coloring

Generated Editor



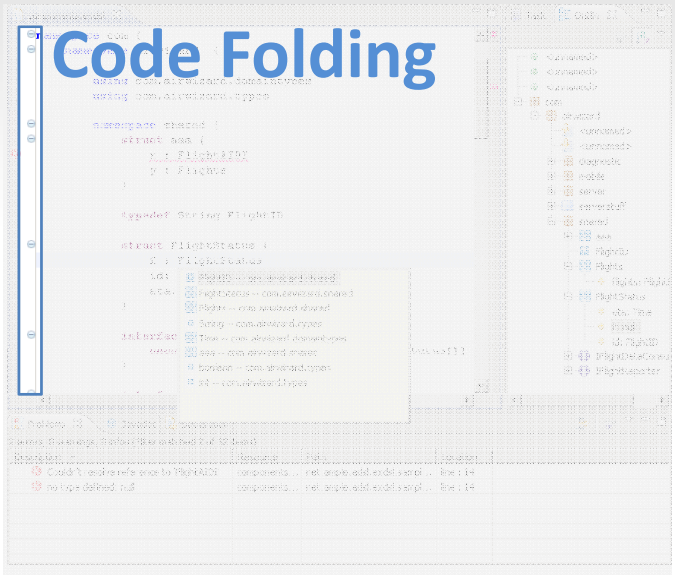
Realtime Constraint Validation

Generated Editor



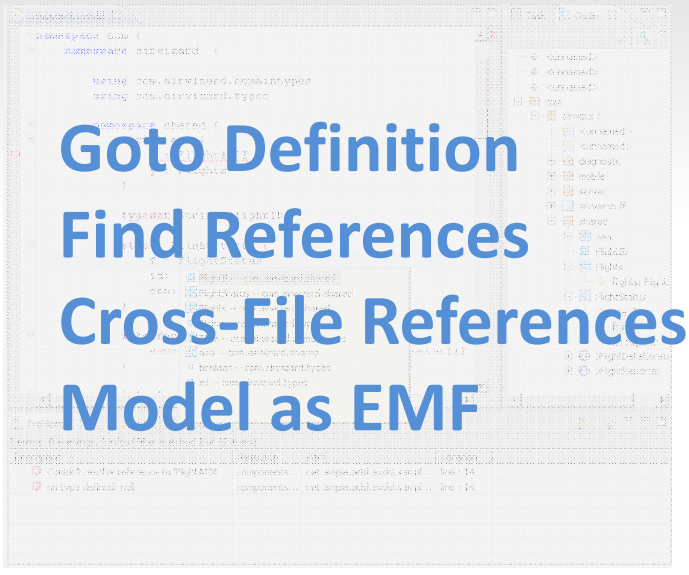
Customizable Outlines

Generated Editor



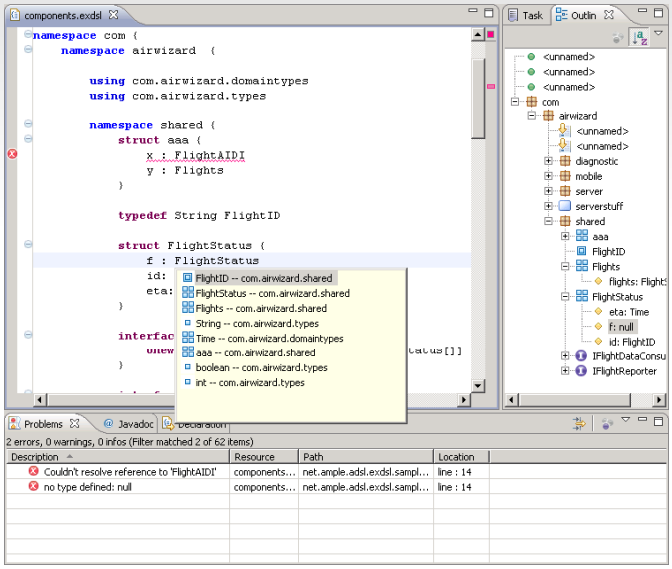
Code Folding

Generated Editor

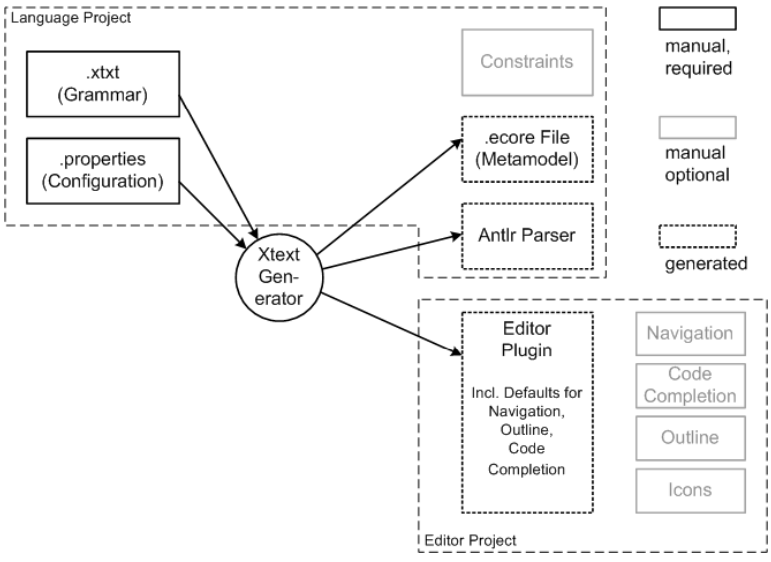


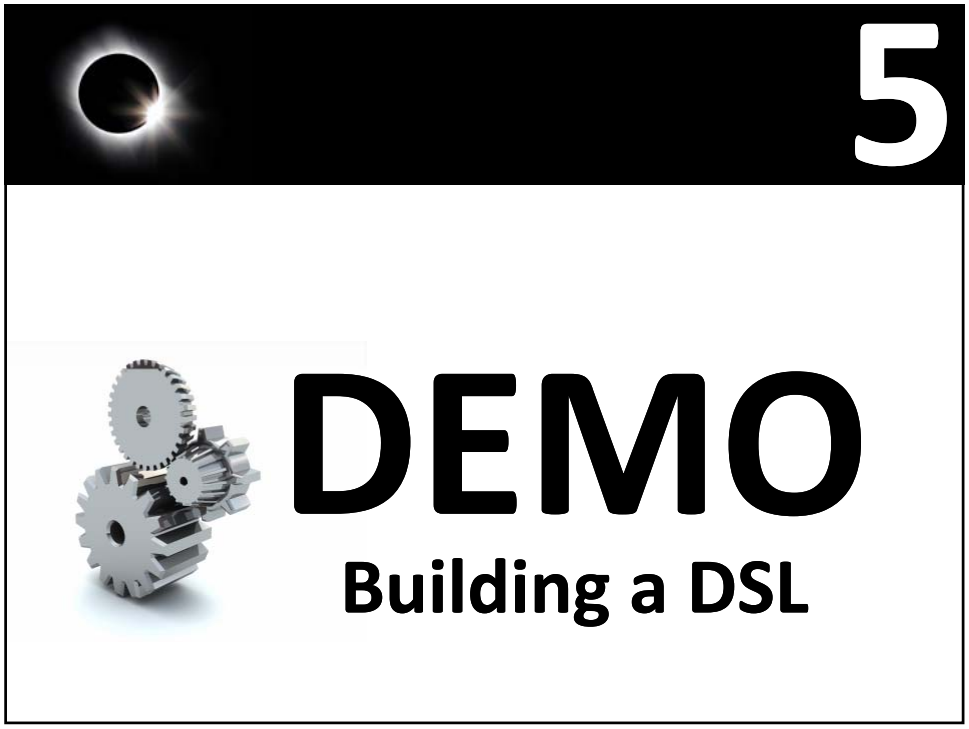
Goto Definition
Find References
Cross-File References
Model as EMF

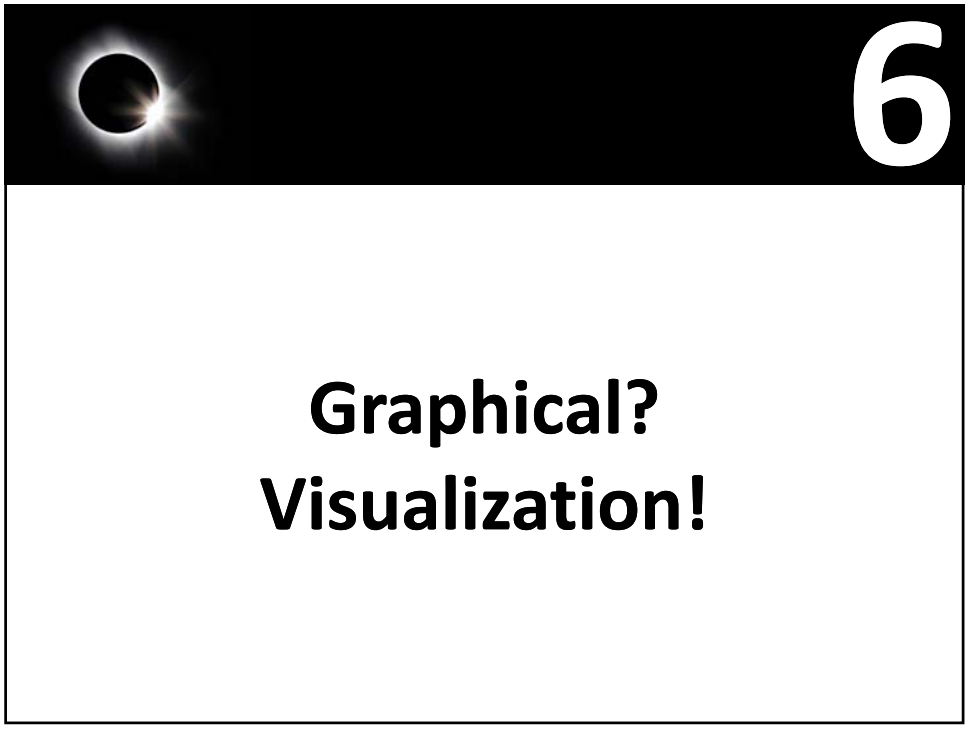
Generated Editor

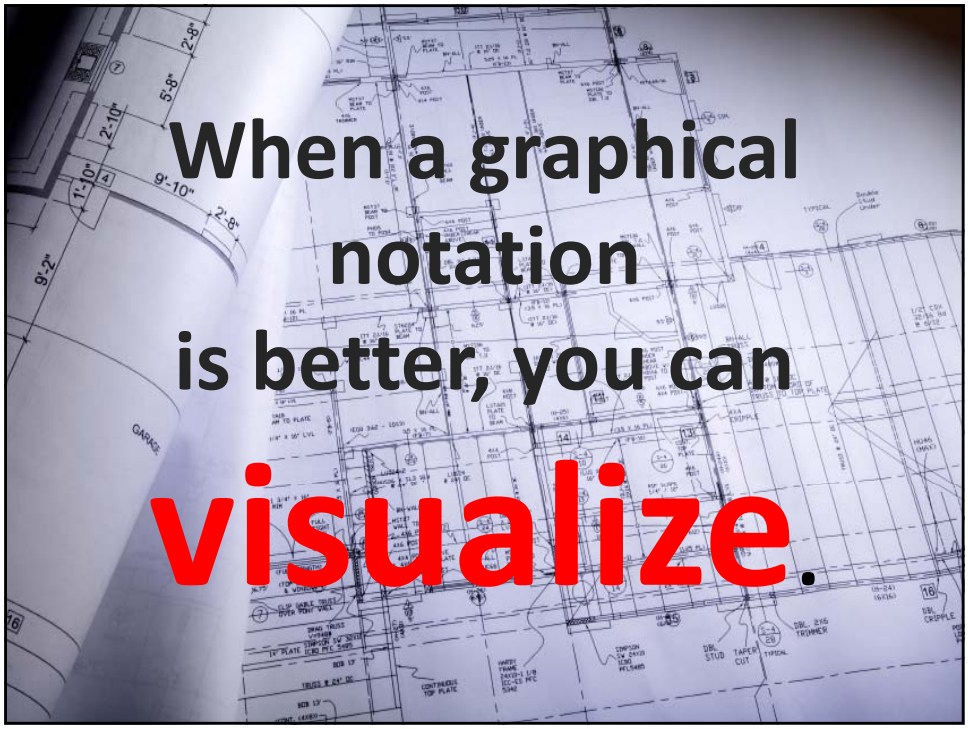


Xtext Overview




5
DEMO
Building a DSL

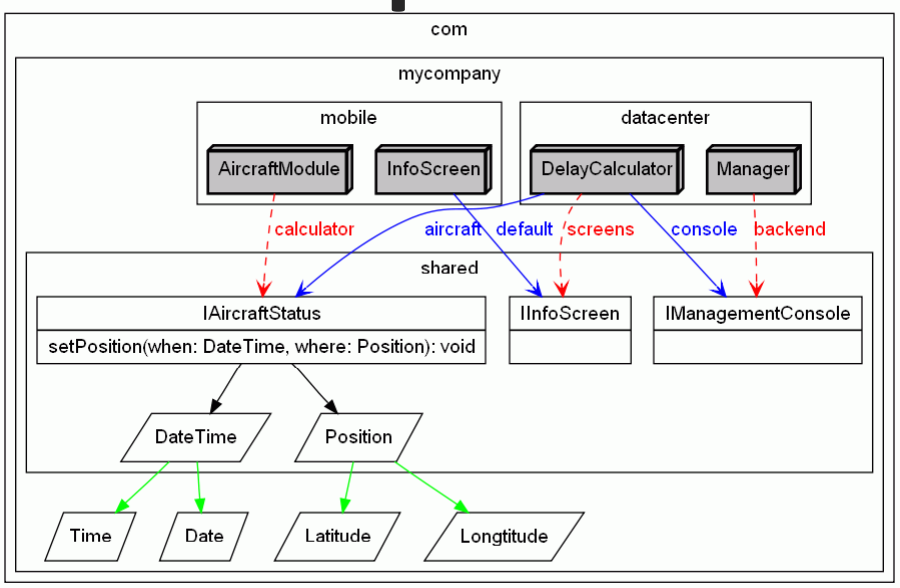

6
Graphical?
Visualization!


When a graphical
notation
is better, you can
visualize.

Via M2M
Read-Only
Auto-Layout
Drill-Down

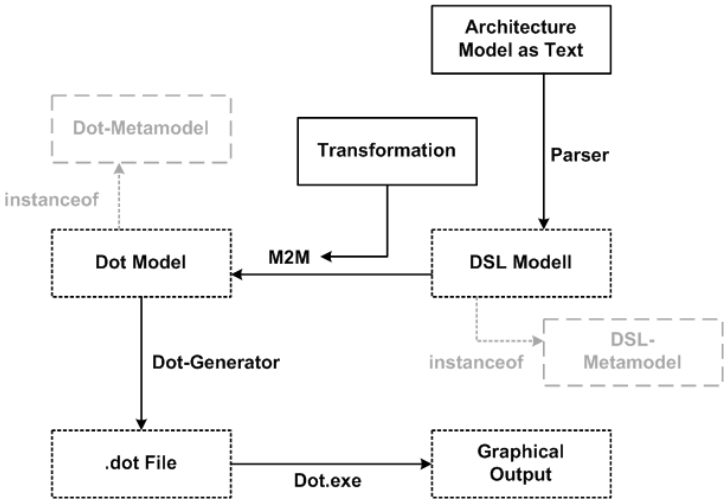
Textual DSLs
vs.
Graphical
vs.
Visualization

Graphviz



Prefuse

GrapViz Overview



GrapViz Trafo Code

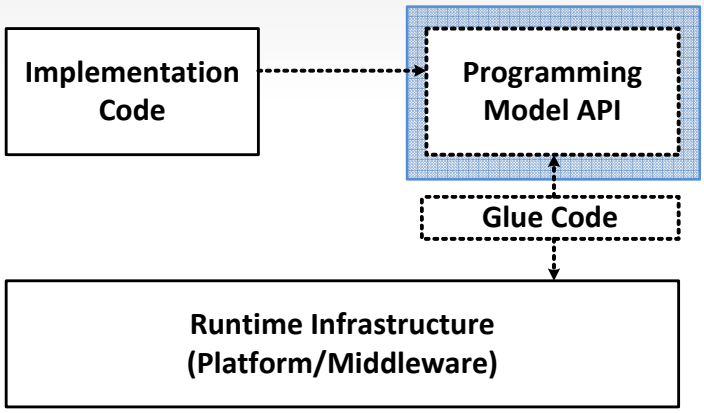
```
import xpand;
import dot;

extension org::openarchitectureware::util::stdlib::io;
extension org::openarchitectureware::util::stdlib::naming;
extension org::openarchitectureware::util::stdlib::elementprops;

extension dotlib;

Object top(File file):
    toGraph(file);

create dot::graph toGraph(File file):
    setType(dot::graphtype::digraph) ->
    setName(this.metaType.toString()) ->
    stmts.add( file.topLevelNamespace.toSubgraph() ) ->
    setFont( "arial" );


create dot::subgraph toSubgraph( Namespace ns ):
    setName( ns.qualifiedName() )
    .setLabel( ns.name )
    .addStatements( ns.subNamespaces.toSubgraph() )
    .addStatements( ns.components.toNode() )
    .addStatements( ns.interfaces.toNode() )
    .addStatements( ns.datatypes.typeSelect(ComplexType).toNode() );

connectPort( Port p ):
    directedEdge( p.eContainer, p.interface, p )
        .setArrowHead("vee")
        .setStyle( ProvidedPort.isInstance(p) ? "solid" : "dashed" )
        .setColor( ProvidedPort.isInstance(p) ? "blue" : "red" )
        .setFontColor(  ProvidedPort.isInstance(p) ? "blue" : "red" )
        .setLabel( p.name );

create dot::node_stmt this toNode(ComplexType t):
    setLabel( t.name )
    .setShape( "parallelogram" )
    .setName( t.qualifiedName() );

create dot::node_stmt this toNode(Component ns):
    setName( ns.qualifiedName() )
    .setLabel( ns.name )
    .setShape( "box3d" )
    .setFillColor( "grey" )
    .setStyle("filled, bold");
```

7

Generating Code

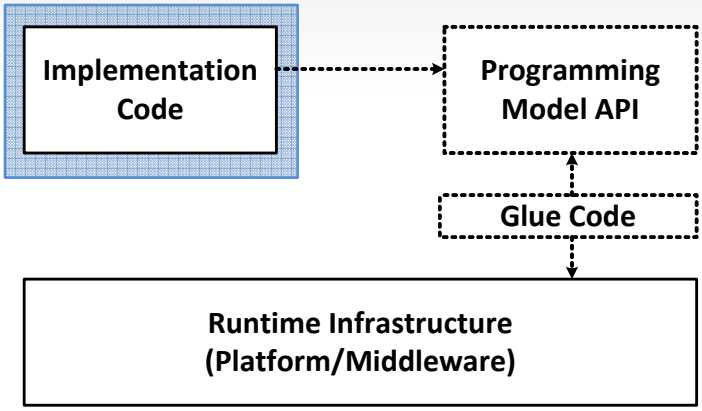
Generate API

Maps Architectural Concepts to Implementation language (non-trivial!)



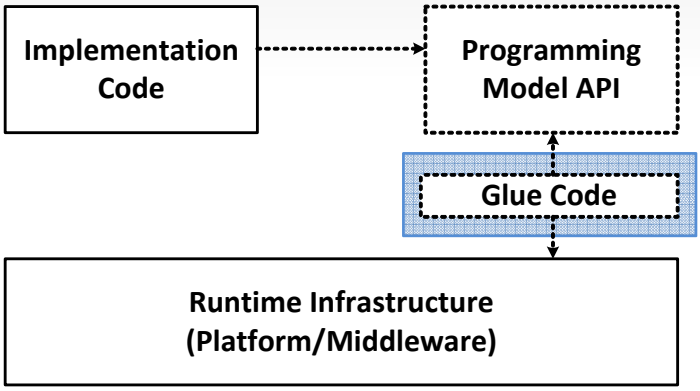
Implementation

Implementation only depends on the generated programming model API



Glue Code

Aka Technology Mapping Code
Maps API to selected platform



Templates

data.xpt

```
1 «IMPORT imp»
2
3 «EXTENSION dataimport::dsl::impUtil»
4
5 «DEFINE root FOR DataStructure»
6 «FILE filename()»
7
8 import java.util.List;
9 import java.util.ArrayList;
10 import dataimport.platform.DataBase;
11
12
13 public class «classname()» extends dataimport.platform.DataBase
14
15     «FOREACH attributes AS a»
16         private «a.type» «a.name»;
17     «ENDFOREACH»
18
19     «FOREACH references AS r»
20         «IF r.ismulti»
21             private List<«r.type.fqClassname()»> «r.name»List =
22         «ELSE»
23             private «r.type.fqClassname()» «r.name»;
24         «ENDIF»
25     «ENDFOREACH»
26
27     «FOREACH attributes AS a»
28         public void set«a.name.toFirstUpper()»( «a.type» value )
```

Extensions

Extensions.ext

```
1  import imp;
2
3  extension dataimport::dsl::GenExtensions reexport;
4
5  Collection[Instance] instancesInScope( RecordHandler this ):
6      parentHandler() != null ? instances.union( parentHandler().instancesInScope() ) : instances;
7
8  parentHandler( RecordHandler this ):
9      eContainer != null ? ( RecordHandler.isInstance(eContainer) ? ((RecordHandler)eContainer) : null ) :
10         null;
11
```

stdutil

arrayindex.xpt

```
«EXTENSION templates::util»

«DEFINE roo
    «FILE i
        pac
```

notaus
org::openarchitectureware::util::stdlib::io
org::openarchitectureware::util::stdlib::naming
templates::util

8

DEMO

A Code Generator

9

Tools Summary

EMF

Ecore meta meta model

+

Editing

Transactions

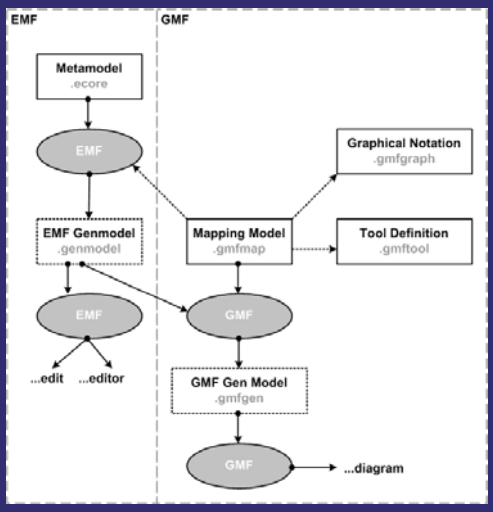
Validation

Query

Distribution/Persistence

GMF

Graphical Box/Line editors based on EMF



TMF

Building Textual Editors

currently being built from oAW Xtext

M2M

Model-to-Model Transformations

INRIA's ATL
Several QVT implementations

M2T

Model-to-Text Transformations

JET: Java Emitter Templates
Xpand: oAW's template engine

openArchitectureWare

www.openarchitectureware.org

eclipse modeling PROJECT

One Stop Toolkit for DSLs + X

Version **4.3.1** is current

Lively ecosystem of tools and extensions

Proven track record in various domains & project contexts

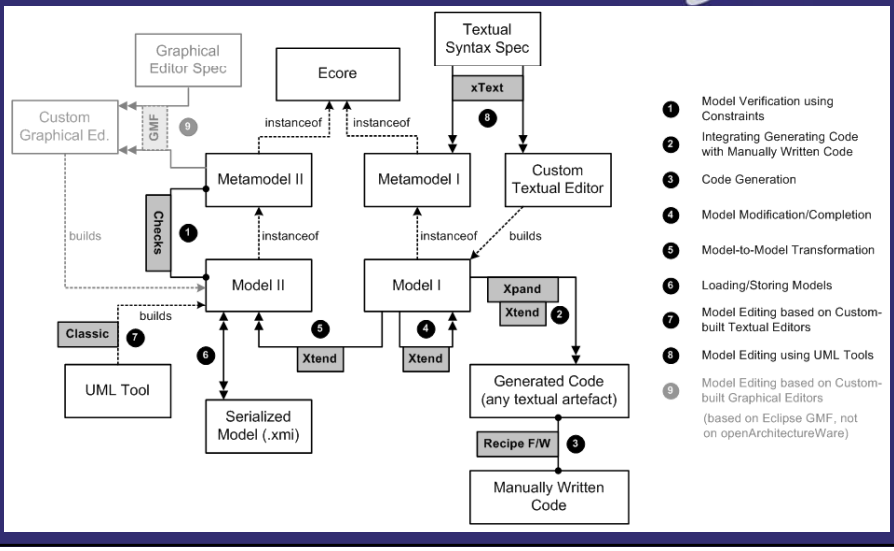
Stable, productive and helpful developer, support and user **communities**

Integration with Eclipse:

- Uses EMF as a basis
- Graphical editors based on GMF
- All editors and tooling based on Eclipse

openArchitectureWare

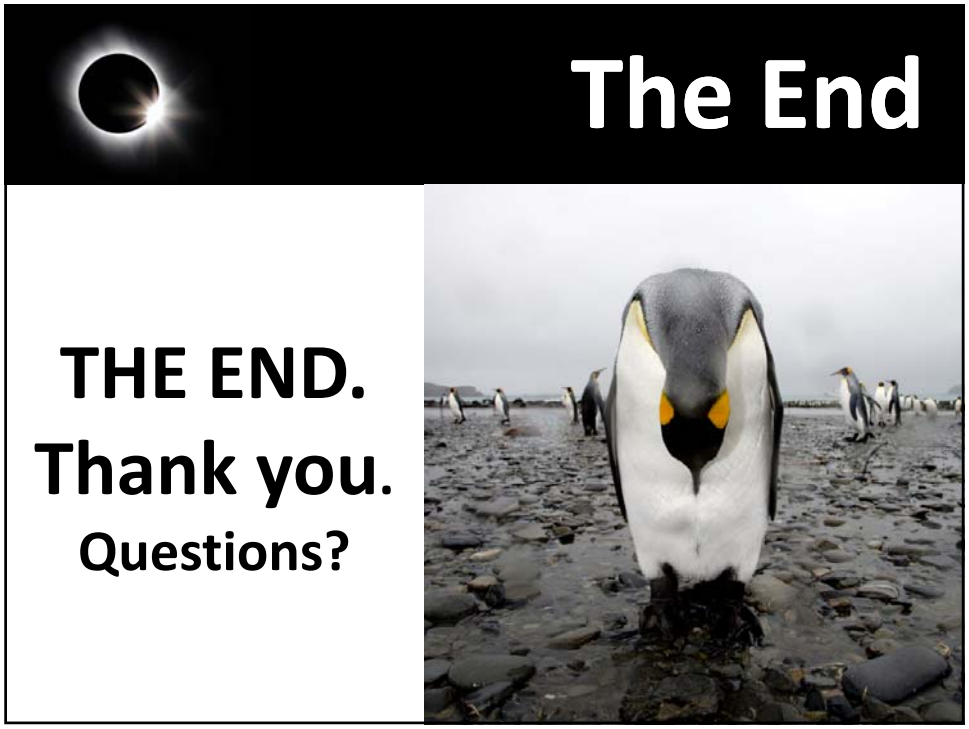
eclipse modeling PROJECT

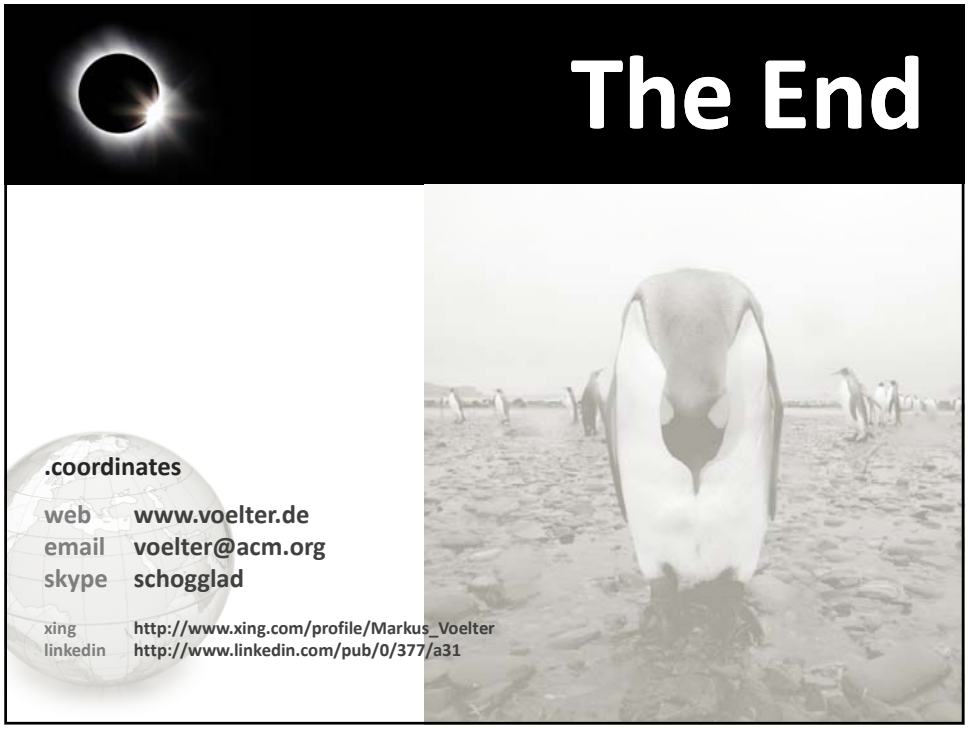


oAW 5

To be released in June

- Xpand, Xtend, Check, MWE migrated to Eclipse projects
- Completely new, much more powerful Xtext/TMF


The End
THE END.
Thank you.
Questions?


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


© 2009 Markus Völter