
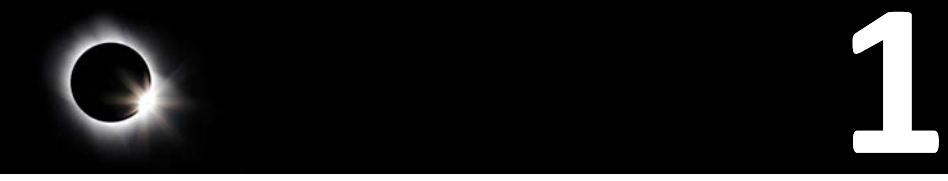


Markus Voelter
Independent/itemis
voelter@acm.org



Textual DSLs
and
Code Generation
with
Eclipse and **oAW**

A black header bar containing a white image of a solar eclipse on the left and a large white number '1' on the right.

Context: Architecture DSLs

A photograph of network cables plugged into a server rack, showing a dense array of connections.

As you
understand
and
develop
your architecture...

...develop a
language
to express it



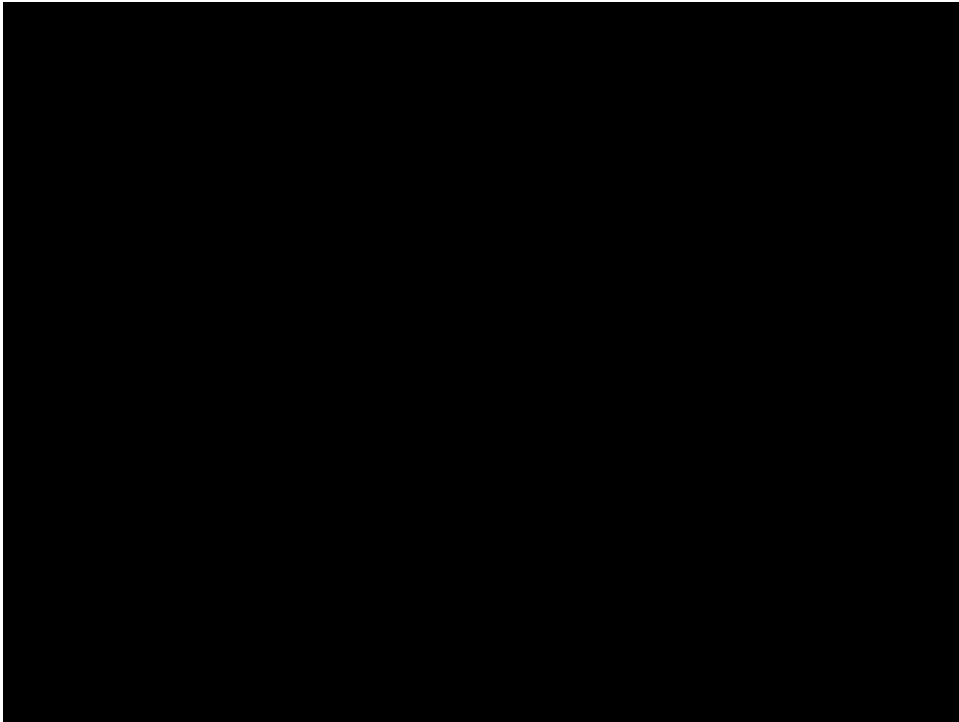
The language resembles
architectural
concepts...



and we describe
the application(s)
with the language



Architecture
DSL



2

**Architecture DSL
Example**

A decorative header bar with a black background. On the left, there is a glowing planet with a bright ring. On the right, the number '2' is displayed in a large, white, sans-serif font. Below the header bar, the text 'Architecture DSL Example' is centered in a bold, black, sans-serif font.

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

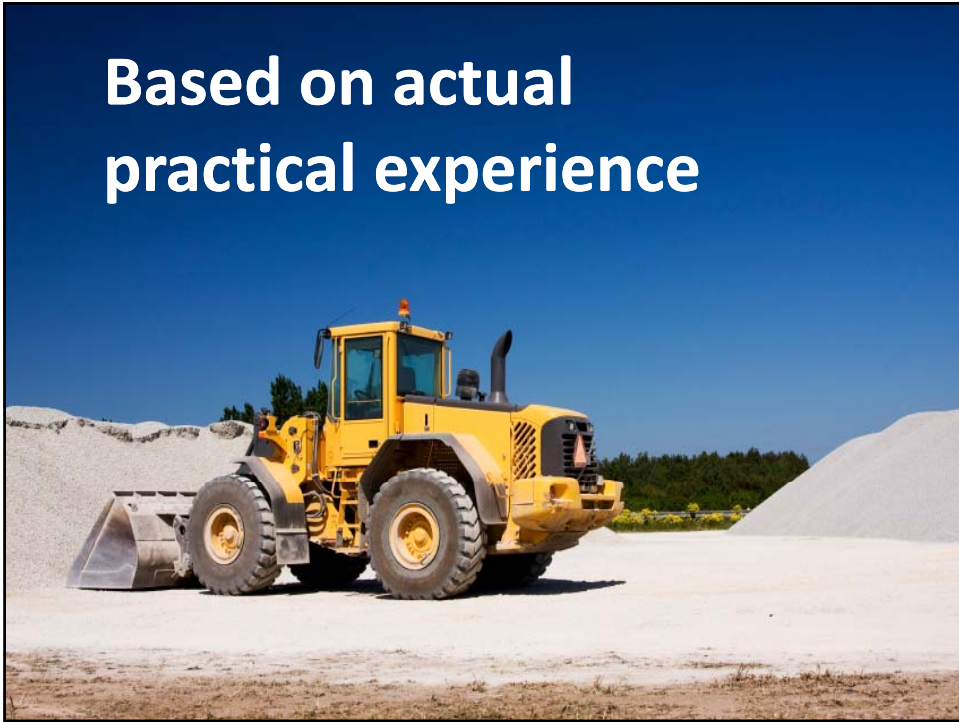

```
interface IAircraftStatus {  
  
    oneway message reportPosition(aircraft: ID,  
                                   pos: Position ) {  
        pre aircraft != null: "aircraft not specified"  
        pre pos != null: "position not specified"  
    }  
  
    request-reply message reportProblem {  
        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

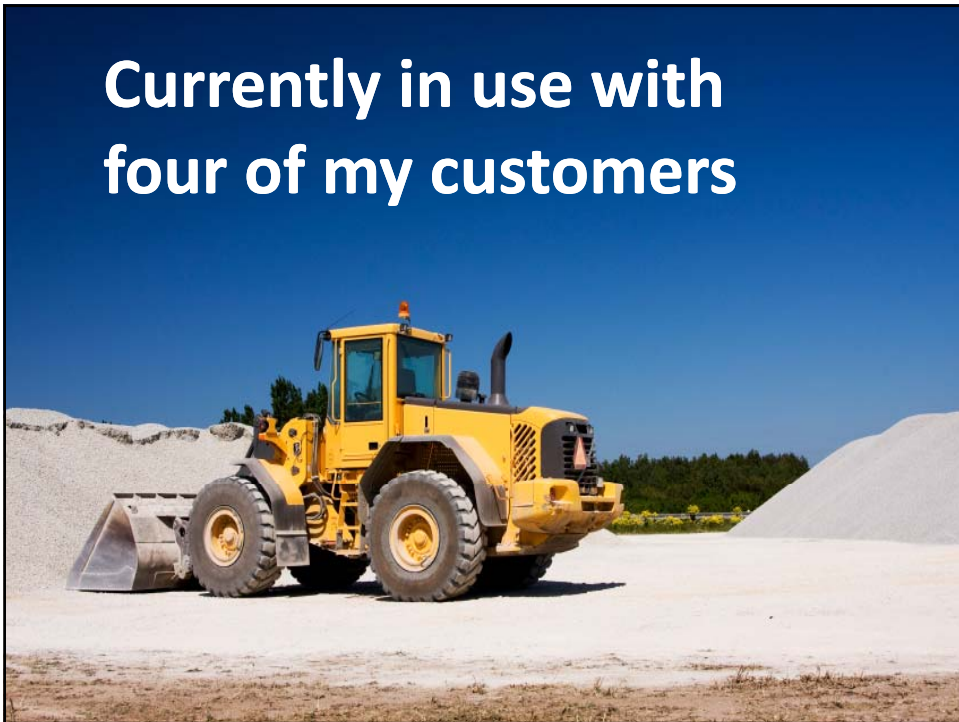
```
interface IAircraftStatus {
  oneway message registerAircraft(aircraft: ID! )
  oneway message unregisterAircraft(aircraft: ID! )
  oneway message reportPosition(aircraft: ID!,
    pos: Position! )
  request-reply message reportProblem {
    request (aircraft: ID!, problem: Problem!,
      comment: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

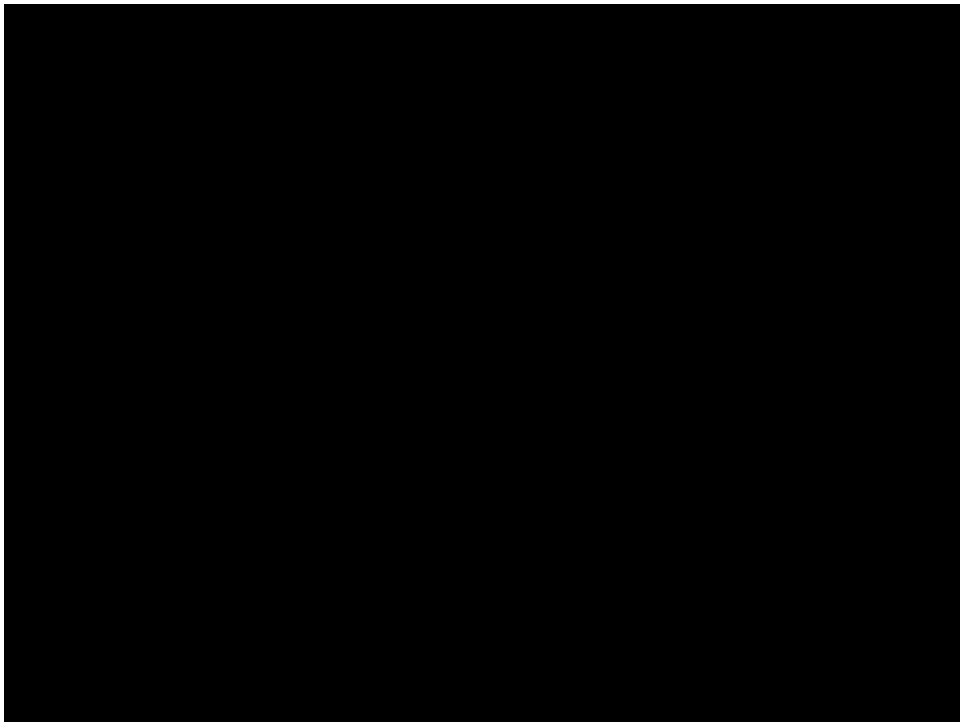
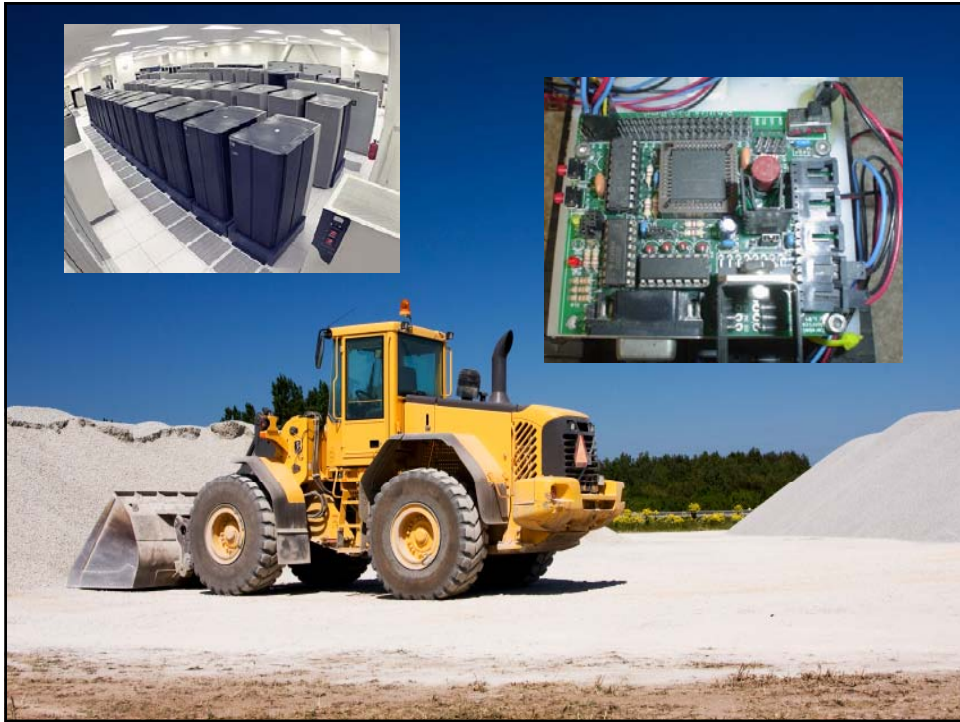
protocol initial = new {
  state new {
    registerAircraft => registered
  }
  state registered {
    unregisterAircraft => new
    reportPosition
    reportProblem
  }
}
}
```


**Based on actual
practical experience**



**Currently in use with
four of my customers**








3

Why Textual?



3

... or: why not graphical?

**Languages and Editors
are easier to build**

**Languages and Editors
are easier to build**

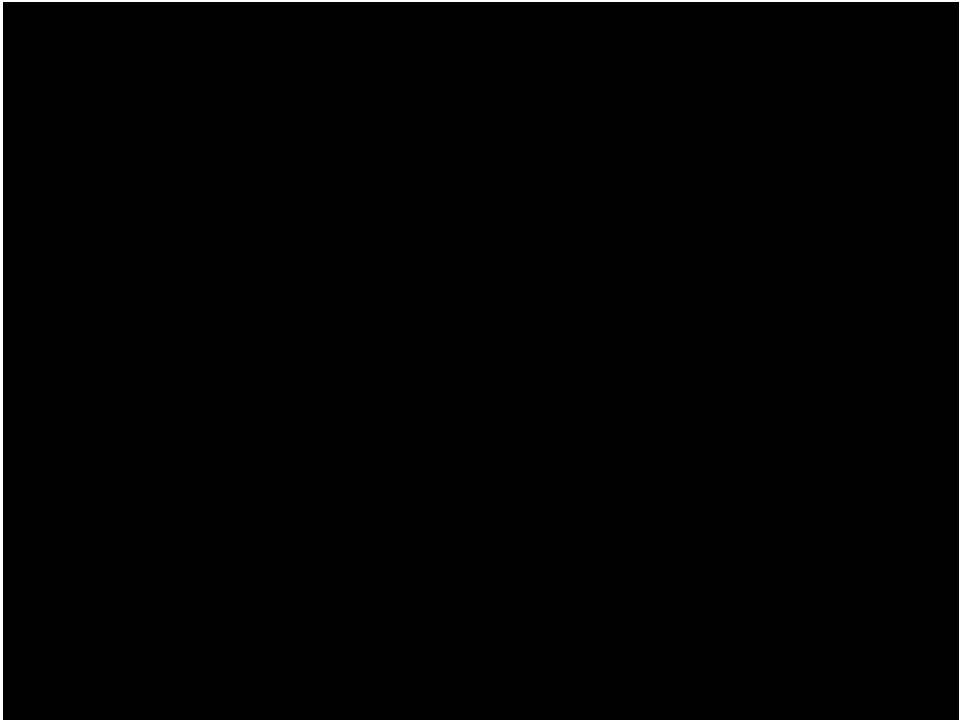
**Evolve Language and simple editor
as you understand and discuss the
architecture, in real time!**

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**

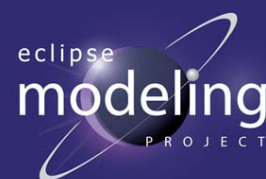




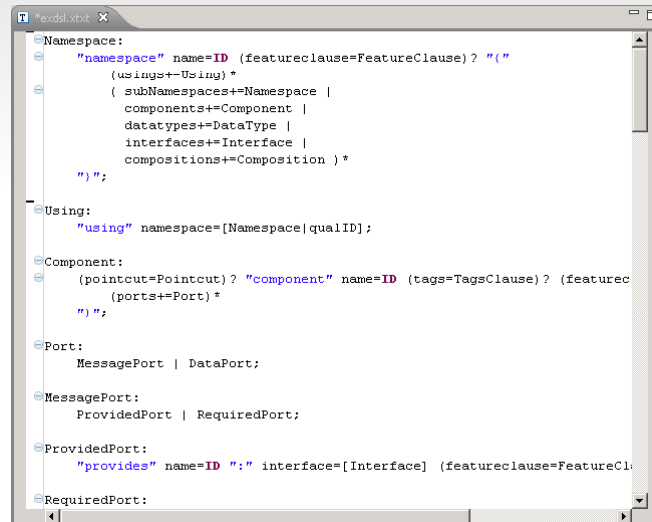
4

Tooling

Eclipse
TMF / Xtext
openArchitectureWare



Specify Grammar



```

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;

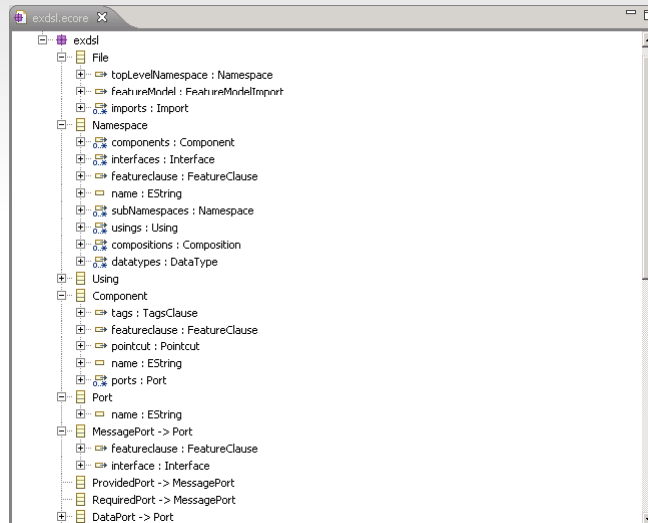
MessagePort:
  ProvidedPort | RequiredPort;

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

RequiredPort:
  
```

**Antlr Grammar and
Parser is generated
from this specification**

Generated Metamodel



Specify Semantics

```

semantic annotation for "platform:/resource/org.openarchitectureaware.seman.test/src-gen/org.openarchitectureaware
class Configuration {
    template templateRef (
        properties
        singleOverrideParam
        join( singleJoinParam )
        multiOverride
        join( multiJoin );
        mode accessorFunctions;
    )
}

class Entity {
    parents
    uniqueNames attrs
    scope primaryAttr allInSiblingProperty attrs
}

class Reference {
    scope entity allUnderParent System
    //referenceScope Reference.entity custom
}

class Attribute {
    parents
}

class ValueObject {
    parents
}

class StateMachine {
    uniqueNames events
    uniqueNames states
    scope initial allInSiblingProperty states
    scope baseMachine allUnderParent World excludeSelf
}

```

Specify Constraints

```

Checks.chk
import exdsl;

extension net::ample::adsl::exdsl::Extensions;
extension org::openarchitectureware::util::stdlib::io;

context Component ERROR "Qualified Name "+qualifiedName()+" must be unique"
  allComponents().select( c | c.qualifiedName() == qualifiedName() ).size == 1;

context DataType ERROR "Qualified Name "+qualifiedName()+" must be unique"
  allDataTypes().select( c | c.qualifiedName() == qualifiedName() ).size == 1;

context Namespace if !isEmpty() ERROR "Qualified Name "+qualifiedName()+" must be unique"
  allNamespaces().select( c | c.qualifiedName() == qualifiedName() ).size == 1;

context emf::EObject if metaType.getProperty("name") != null ERROR "name not defined"
  metaType.getProperty("name").get(this) != "Unnamed";

context Interface ERROR "interface names must start with a capital I":
  name.startsWith("I");

context MessagePort ERROR "interface not defined. Missing a 'using'?":
  visibleInstancesOfType(this, Interface).contains(interface);

context Attribute ERROR "no type defined: "+type.name:
  visibleInstancesOfType(this, DataType).contains(type);

context DataPort ERROR "data not defined: "+type.name:
  visibleInstancesOfType(this, ComplexType).contains(type);
  
```

Generated Editor

The screenshot shows an IDE window titled 'components.exdsl'. The main editor displays XSD code with the following structure:

```

namespace com (
  namespace airwizard (
    using com.airwizard.domaintypes
    using com.airwizard.types

    namespace shared (
      struct aaa (
        x : FlightAIDI
        y : Flights
      )

      typedef String FlightID

      struct FlightStatus (
        f : FlightStatus
        eta:
      )
    )
  )
  interface Time (
    aaa -- com.airwizard.shared
    boolean -- com.airwizard.types
    int -- com.airwizard.types
  )
)
  
```

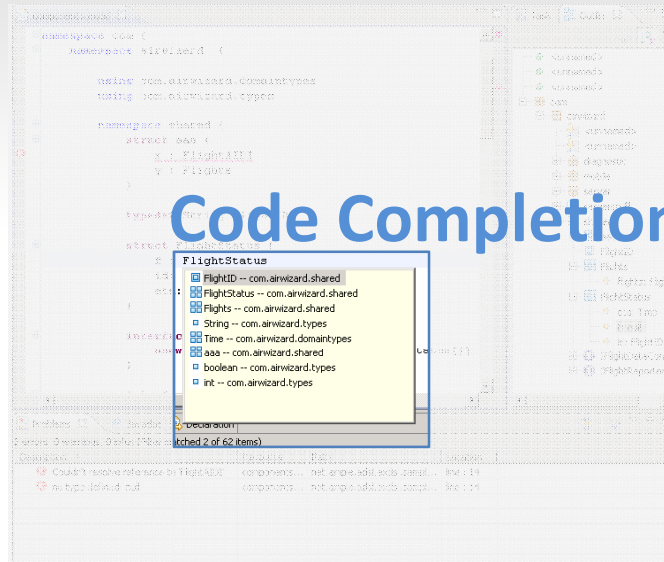
A tooltip is visible over the 'FlightStatus' struct definition, showing its fields and their types:

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

At the bottom, the 'Problems' window shows two errors:

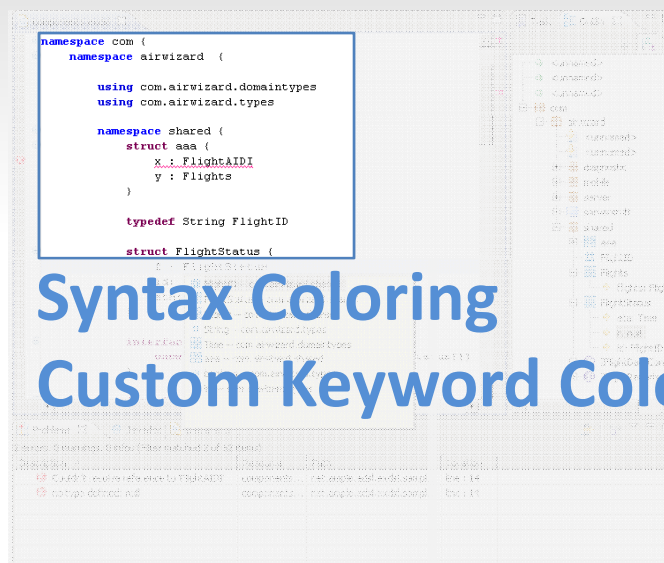
Description	Resource	Path	Location
Couldn't resolve reference to 'FlightAIDI'	components...	net.ample.adsl.exdsl.samp...	line : 14
no type defined: null	components...	net.ample.adsl.exdsl.samp...	line : 14

Generated Editor



Code Completion

Generated Editor



Syntax Coloring
Custom Keyword Coloring

Generated Editor

Realtime Constraint Validation

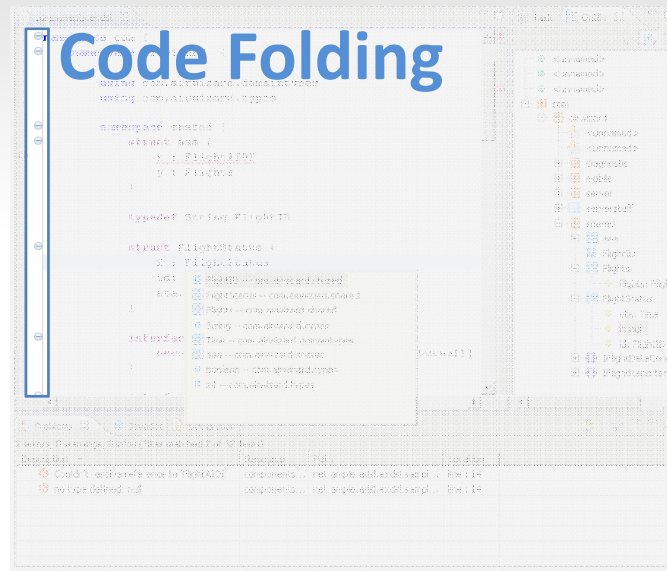
Description	Resource	Path	Location
Couldn't resolve reference to "FlightAIDI"	components...	net:ampl:addl:exdsl:sampl...	line : 14
no type defined: null	components...	net:ampl:addl:exdsl:sampl...	line : 14

Generated Editor

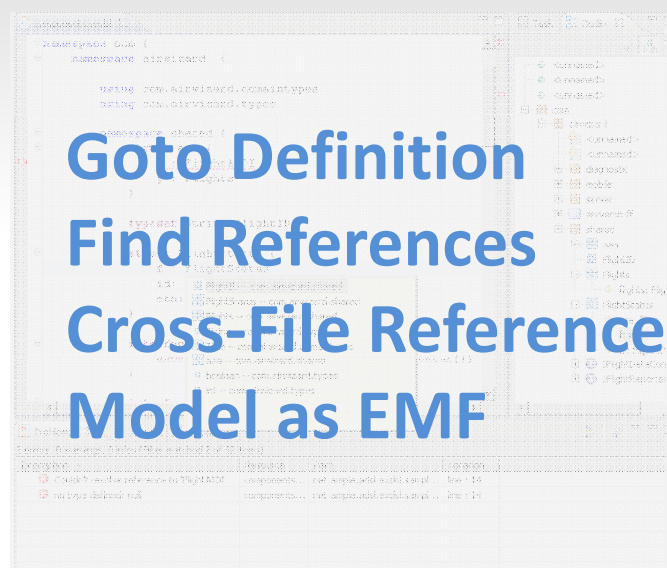
Customizable Outlines

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

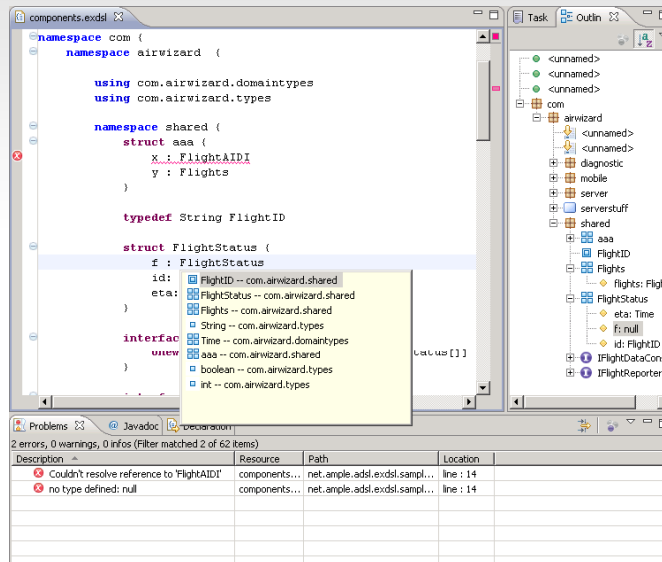
Generated Editor



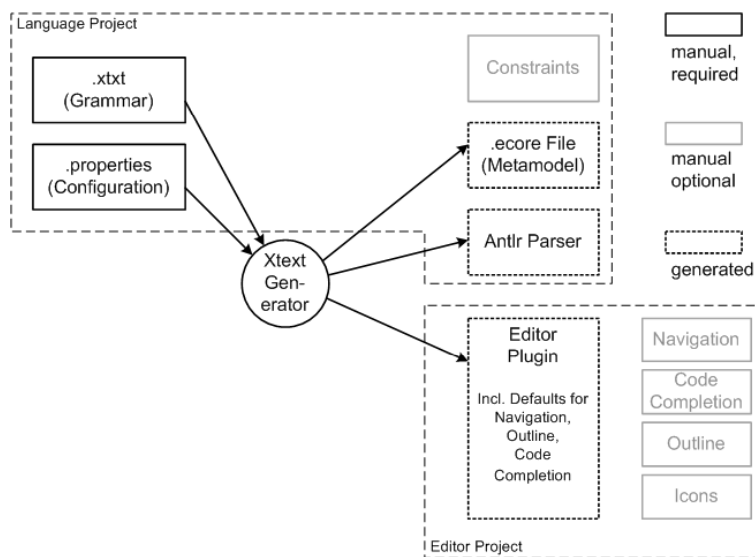
Generated Editor




Generated Editor



Xtext Overview



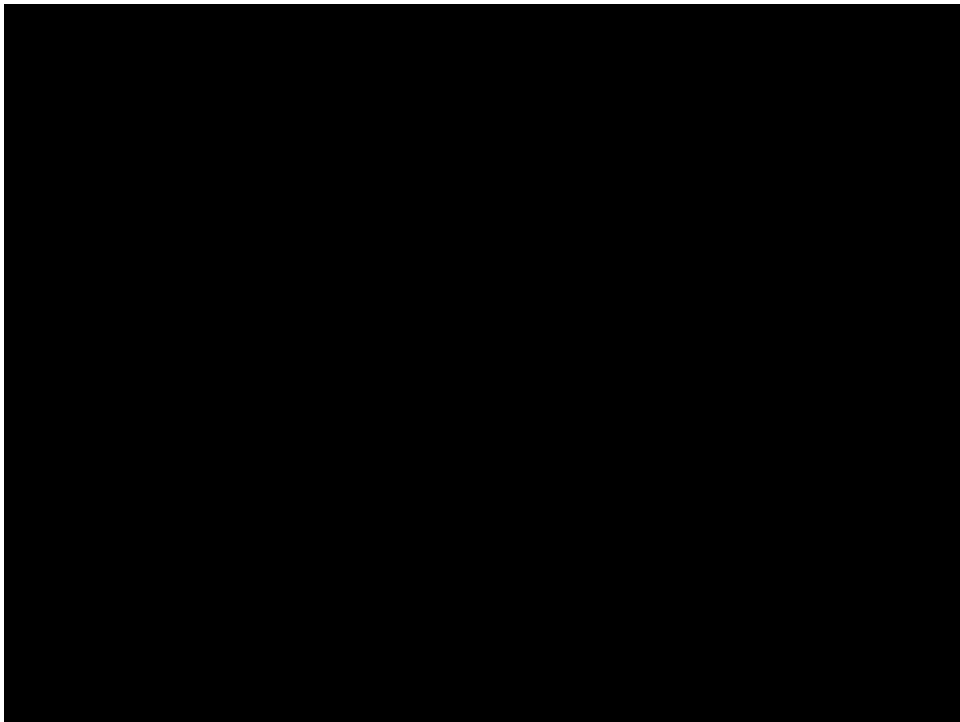


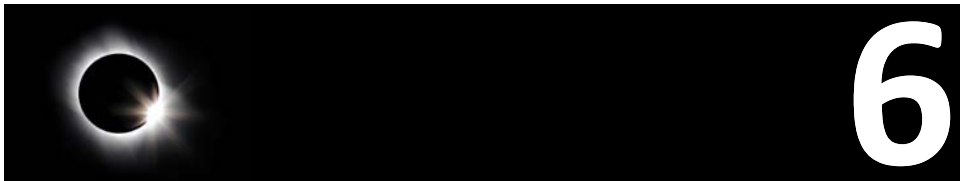
5



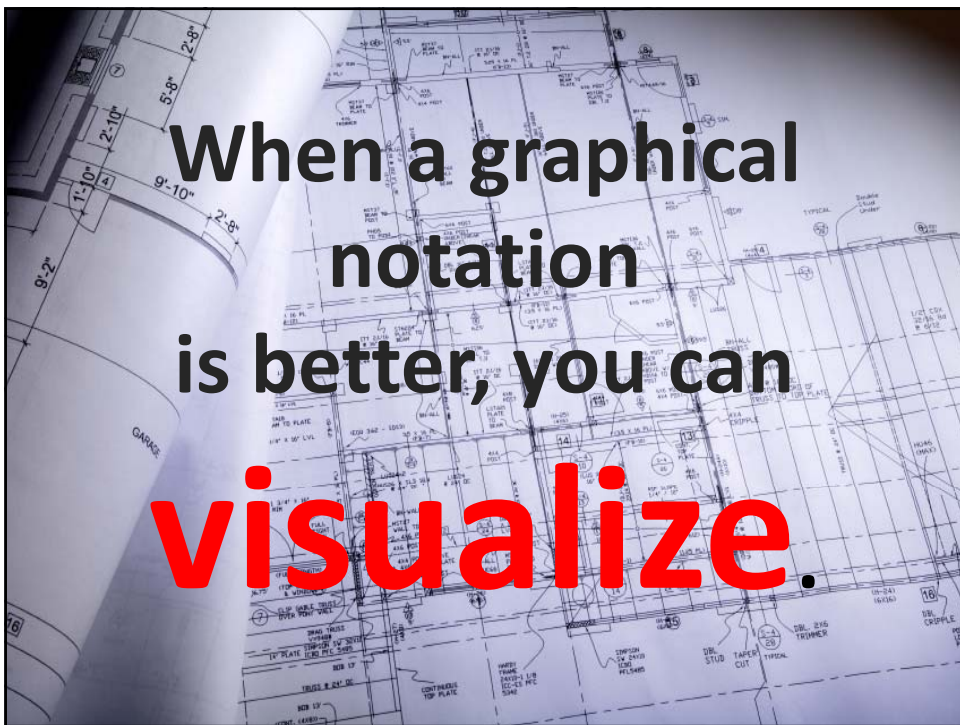
DEMO

Building a DSL



A black horizontal bar at the top of the slide. On the left side, there is a small, bright image of a solar eclipse. On the right side, the number '6' is written in a large, white, sans-serif font.

Graphical? Visualization!

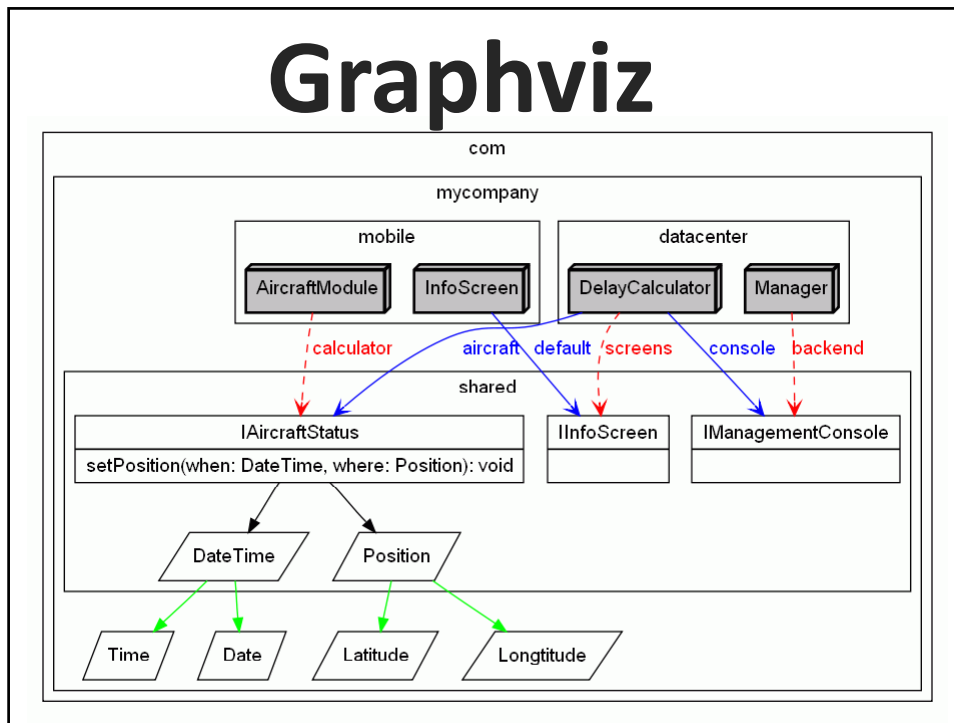
A detailed architectural floor plan or technical drawing is shown, featuring various lines, dimensions, and annotations. The drawing is partially unrolled from the left side, creating a sense of depth and perspective.

**When a graphical
notation
is better, you can
visualize.**

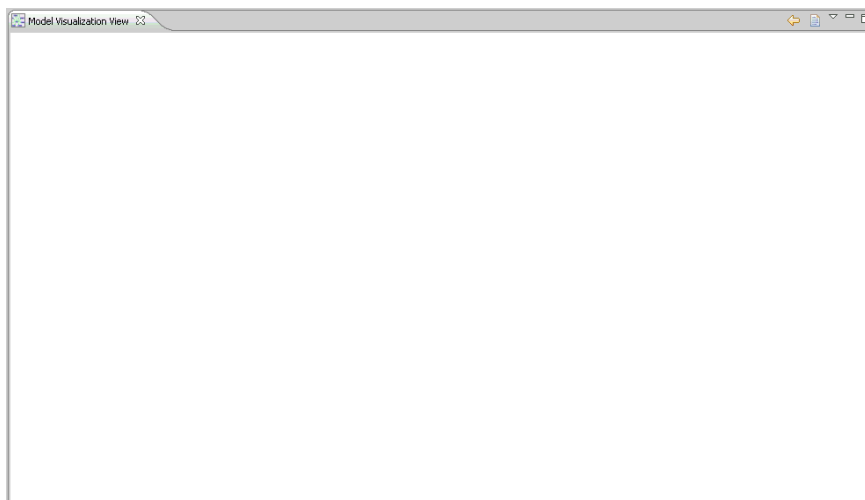
**Via M2M
Read-Only
Auto-Layout
Drill-Down**

**Textual DSLs
vs.
Graphical
vs.
Visualization**

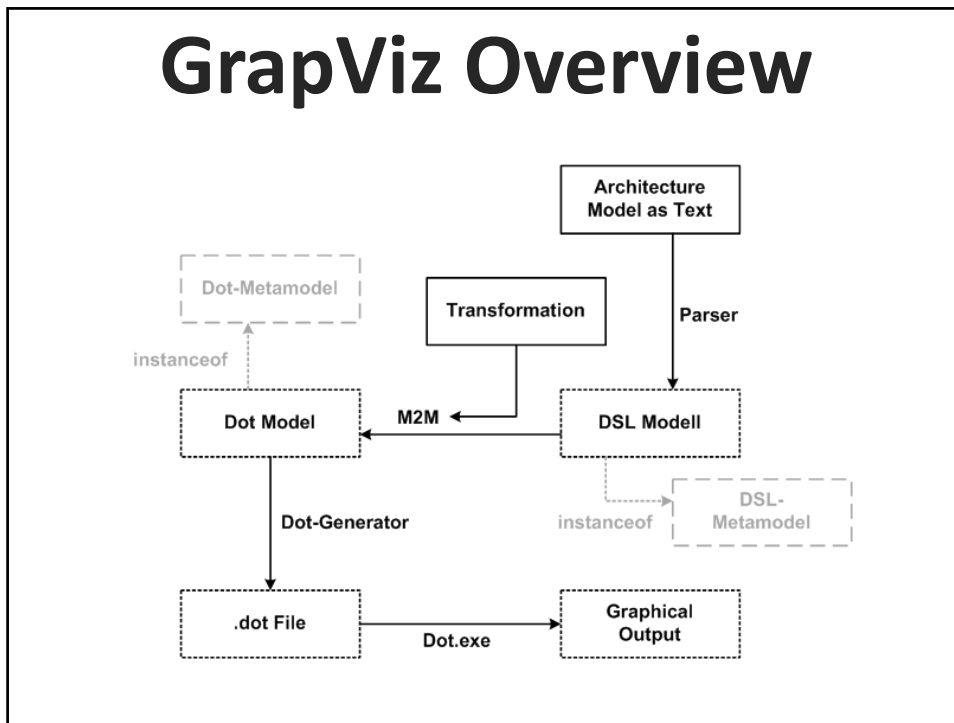
Graphviz



Prefuse



GrapViz Overview



GrapViz Trafo Code

```

import adsl;
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);

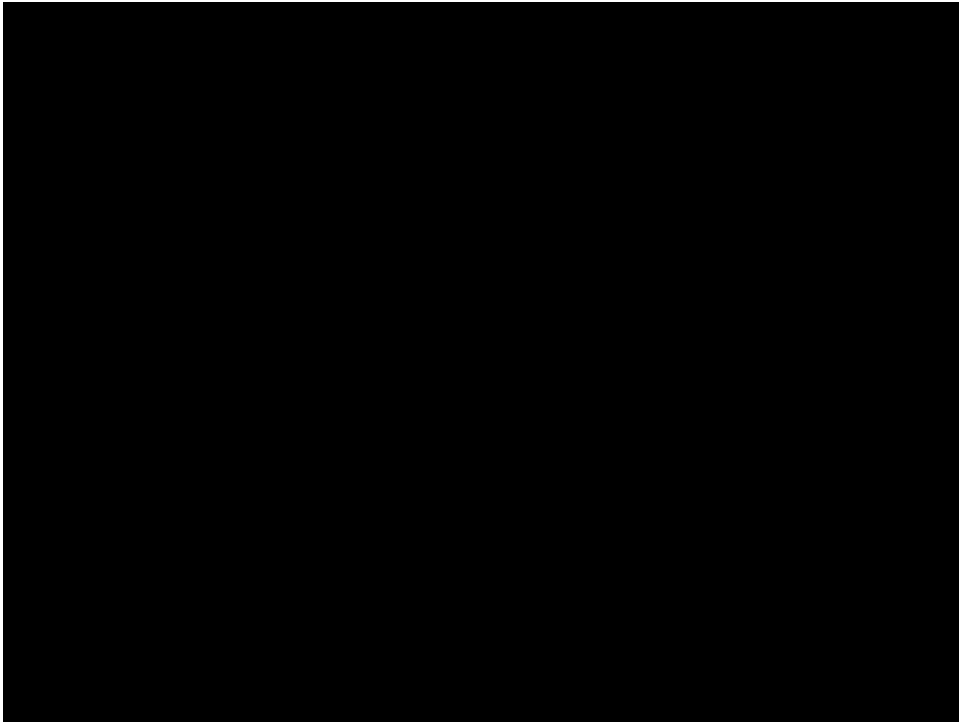
create dot::graph toGraph(File file):
  setType(dot::graphType::digraph) ->
  setName(this.metaType.toString()) ->
  state.add( file.toLevelNamespace.toSubgraph() ) ->
  setFont( "arial" );

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.container, p.interface, p )
  .setArrowhead( "tee" )
  .setStyle( ProvidesPort.isInstance(p) ? "solid" : "dashed" )
  .setColor( ProvidesPort.isInstance(p) ? "blue" : "red" )
  .setFontColor( ProvidesPort.isInstance(p) ? "blue" : "red" )
  .setLabel( p.name );

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

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

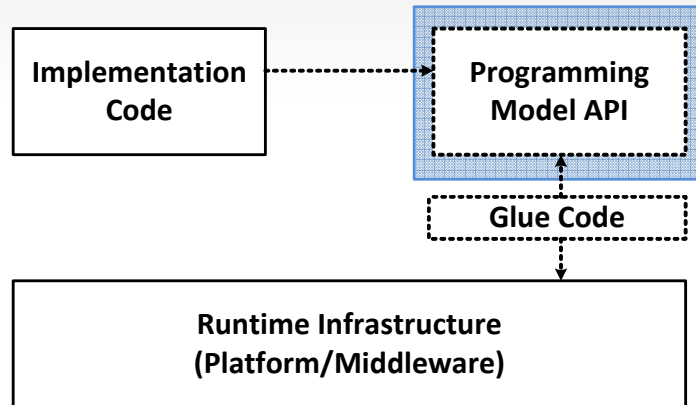
A decorative header bar with a black background. On the left side, there is a glowing planet with a bright ring of light around it. On the right side, the number '7' is written in a large, white, sans-serif font.

7

Generating Code

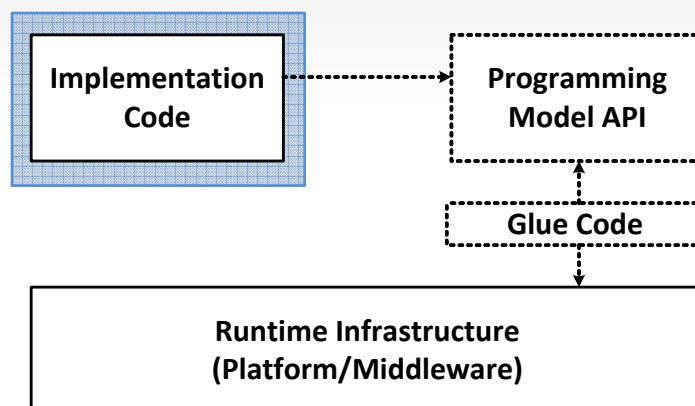
Generate API

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



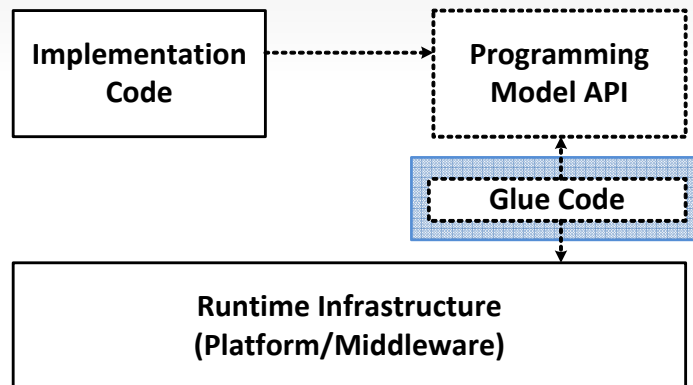
Implementation

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
8import java.util.List;
9import java.util.ArrayList;
10import dataimport.platform.DataBase;
11
12
13public 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 )
29
30
  
```


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

```

arrayindexopt
<<EXTENSION templates::util>>
<<DEFINE roo
<<FILE 1
pac
pub
>> extends s

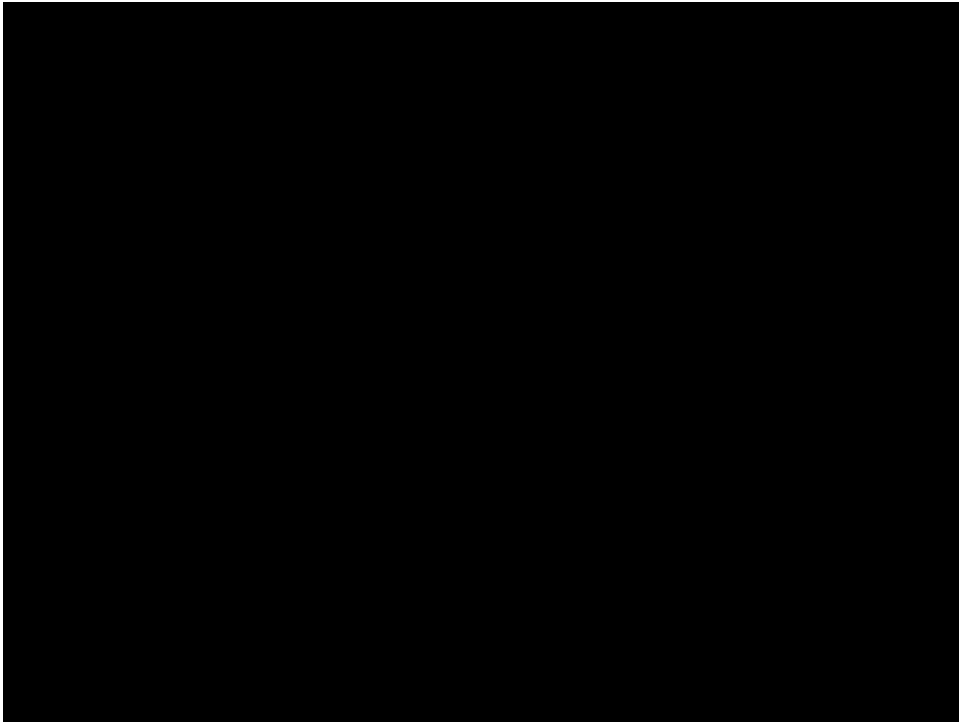
```



DEMO

A Code Generator

8

A slide header consisting of a black horizontal bar. On the left side of the bar is a glowing planet with a bright ring of light around its equator. On the right side of the bar is a large, white, bold number '9'. Below the black bar is a white rectangular area with a thin black border.

Tools Summary

EMF



Ecore meta meta model

+

Editing

Transactions

Validation

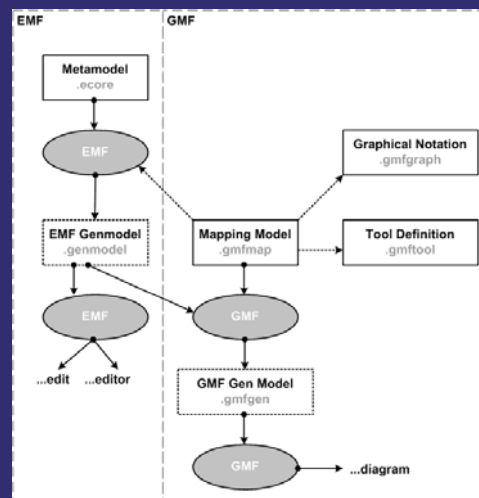
Query

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



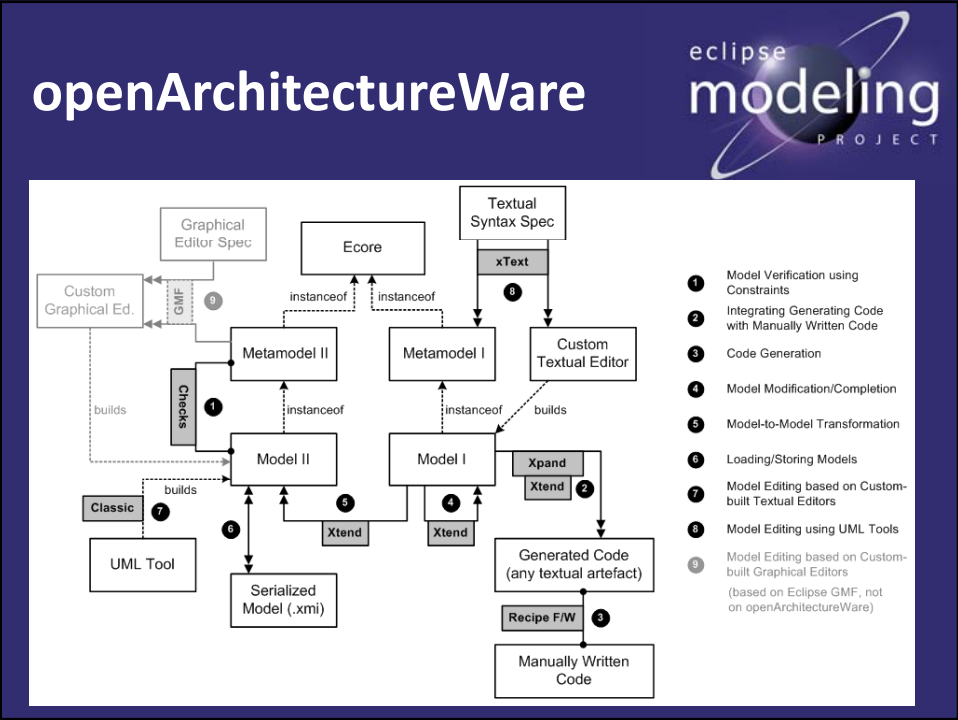
One Stop Toolkit for DSLs + X

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



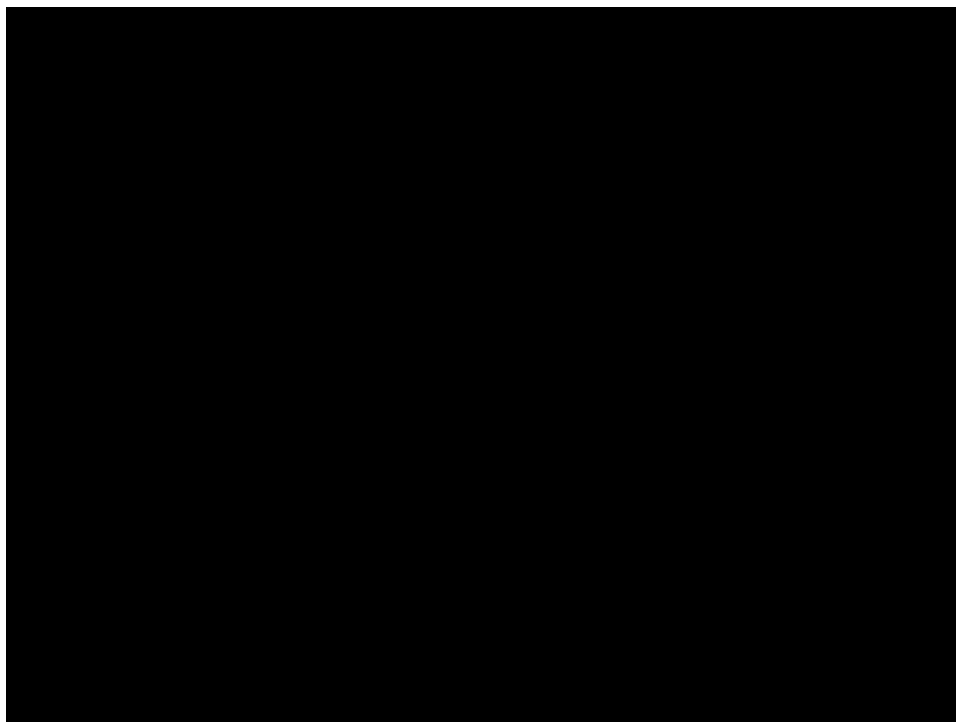
oAW 5



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

