(Towards) a megamodel of the ATL model transformation language and toolkit

Ralf Lämmel
Software Languages Team, University of Koblenz-Landau
Joint work with Jean-Marie Favre (Visiting Researcher at Softlang) and Martin Leinberger, Thomas Schmorleiz, and Andrei Varanovich

http://softlang.wikidot.com/
Preamble
Ralf Lämmel introduced

- W3 Professor of CS at University of Koblenz-Landau
- Leader of the Software Languages Team @ Koblenz
- Co-founder of SLE conference series
- Co-founder of GTTSE summer school series
- Previous jobs: MSFT, VU (A’dam), CWI, Uni Rostock
- Interests: languages, grammars, software language engineering, software linguistics, transformations, automation, lambdas, programs, technologies, understanding, ...
Topics for collaboration

- Megamodeling (today)
- API analysis and migration
- Software language comprehension
- Software technology comprehension
to buttons cannot be easily migrated to does not extend and can have child widgets. Since matching covariance indicates a potentially serious API mismatch. In this case, wrappers must register a counterpart in the target API. In this case, wrappers must register a counterpart in the target API. In this case, wrappers must register a counterpart in the target API. In this case, wrappers must register a counterpart in the target API.

For instance, if any visible type of the source API, clients such as Swing-SWT are covariant mappings. Hence, Swing-SWT is mapped to a counterpart in the target API. In this case, wrappers must register a counterpart in the target API. In this case, wrappers must register a counterpart in the target API. In this case, wrappers must register a counterpart in the target API.

Second, wrappers can take advantage of the source API’s exposed references its superclasses. Clients that attach widgets become wrappers that attach widgets. When there is a correspondence between callbacks of both APIs, then one can delegate suitable classes of the source API. When there is a correspondence between callbacks of both APIs, then one can delegate suitable classes of the source API. When there is a correspondence between callbacks of both APIs, then one can delegate suitable classes of the source API.

E. Inversion of control

Wrappers may need to delegate control from the target API to the source API. For example, when a surrogate for JButton is instantiated, it cannot immediately construct the corresponding adaptee because the parent widget is unknown. This also means that wrappers may need to delay adaptee construction. Thus, classification time presents a challenge to wrappers. Surrogates may need to delay adaptee construction. Thus, classification time presents a challenge to wrappers. Surrogates may need to delay adaptee construction. Thus, classification time presents a challenge to wrappers.

D. Varying creation and wiring protocols

Clients are more constrained. In contrast, clients are more constrained. In contrast, clients are more constrained. In contrast, clients are more constrained.

C. Mapping varying type hierarchies

Note that the mappings of the open-source wrappers do not establish a isomorphism to the target API. For example, when a surrogate for JButton is instantiated, it cannot immediately construct the corresponding adaptee. This also means that wrappers may need to delay adaptee construction. Thus, classification time presents a challenge to wrappers. Surrogates may need to delay adaptee construction. Thus, classification time presents a challenge to wrappers. Surrogates may need to delay adaptee construction. Thus, classification time presents a challenge to wrappers.

• Megamodelling (today)

• API analysis and migration

• Software language comprehension

• Software technology comprehension

© 2012, Software Languages Team @ University of Koblenz-Landau & Collaborators
Fig. 17 Coverage of P'P's Base Data Schema

Each data reference must be resolvable to a data item in a P'P data schema. Second, the latest and provisional specification stipulates sanity checking rules—say, coherence constraints. We also obtained constraints from Yu et al.'s article on the 'relational semantics' of P'P. This article covers key constraints as well as coherence constraints.

4.2.1 Key constraints

The key constraints of the relational schema for P'P's normal form, as of §-, immediately constrain the abstract syntax of §-. We check these constraints naturally as we normalize policies by deriving relations from the P'P statements. Whenever we insert tuples into the relations, we admit identical tuples, but we do not admit tuples that violate key constraints. This is part of the normalization algorithm.

We refer to Table for violation counts for the different key constraints. We note again that the key constraints are potentially debatable; see the discussion in §-

http://www.w3.org/TR/P3P11/#ua_sanity

Topics for collaboration

- Megamodeling (today)
- API analysis and migration
- **Software language comprehension**
- Software technology comprehension
Topics for collaboration

- Megamodeling (today)
- API analysis and migration
- Software language comprehension
- Software technology comprehension

http://101companies.org/
Collaboration indicators

$ pwd
/Users/laemmel/projects/misc/talks/120215-nantes/dl

$ ls
BridgingEclipseMicrosoftModeling.pdf  MoScript.pdf
CorrectATL.pdf  OnModelTyping.pdf
GeneralCompositionSemantics.pdf  TraceabilityWebApplications.pdf
ImprovingHigherOrderInATL.pdf  TypingArtifactsInMegamodeling.pdf
IncrementalExecutionATL.pdf  TypingInModelManagement.pdf
LazyATLPaperModels2011.pdf
(Towards) a megamodel of the ATL model transformation language and toolkit

Ralf Lämmel
Software Languages Team, University of Koblenz-Landau
Joint work with Jean-Marie Favre (Visiting Researcher at Softlang) and Martin Leinberger, Thomas Schmorleiz, and Andrei Varanovich
What’s a megamodel?

Java

HelloWorld.java  conformsto  Java Grammar

elementOf

models

JMF style; see the pyramids papers.
That’s a megamodel, too!

http://en.wikipedia.org/wiki/Tombstone_diagram
Yet another megamodel!

Figure 1. An overview of model transformation

Figure 1 summarizes the full model transformation process. A model $M_a$, conforming to a metamodel $MM_a$, is here transformed into a model $M_b$ that conforms to a metamodel $MM_b$. The transformation is defined by the model transformation model $M_t$ which itself conforms to a model transformation metamodel $MM_t$. This last metamodel, along with the $MM_a$ and $MM_b$ metamodels, has to conform to a metametamodel (such as MOF or Ecore).

3 A simple transformation example

This section introduces the transformation example that is going to be developed in the document. The aim of this first example is to introduce users with the basic concepts of the ATL programming. To this end, this example considers two similar metamodels, Author (Figure 2) and Person (Figure 3), that both encode data relative to persons.

Figure 2. The Author metamodel

Figure 3. The Person metamodel

Both metamodels are composed of a single eponym element: Author for the Author metamodel and Person for the Person metamodel. Both entities are characterized by the same couple of string properties (name and surname).

The objective is here to design an ATL transformation enabling to generate a Person model from an Author model. The transformation to be designed will have to implement the following (obvious) semantics:

- A distinct Person element is generated for each source Author element;
- The name of the generated Person has to be initialized with the name of the source Author;
- The surname of the generated Person has to be initialized with the name of the source Author.

That’s nearly the same megamodel.


© 2012, Software Languages Team @ University of Koblenz-Landau & Collaborators
We discuss the languages first. The C# language is mentioned because derived object models are represented in C#. Other .NET languages could be used here as well. The XSD language, the language of XML schemas, is mentioned because an XML schema is the major point of reference: the schema to which XML data has to conform and the schema from which to generate an object model. The XML language is mentioned because it denotes the primary representation format for data. Finally, we use ClrObj to refer to the language of all object graphs, say in the CLR of the type system of .NET.

There are two schemas involved in such OrX mapping: the XML schema, xmlTypes, for XML data and the object model, ooTypes, for objects. Both schemas are associated with the corresponding language by edges denoting membership. The schemas are connected to each other through the technology node xsd.exe proxying for the class generation mode of using the command line tool xsd.exe. There are two instances—one for each schema; the associations edges denote conformance. The instances are connected to each other through the technology node de-serialize proxying for the API function for deserialization as provided by the API System.Xml.Serialization.

At this point, the notion of a megamodel becomes clear. That is a megamodel identifies entities such as languages, schemata, instances, and technologies as well as relationships such as membership, conformance, and dataflow. Naturally, the following questions arise:

- Did we model all crucial aspects of OrX mapping?
- Did we model all interesting specifics of the .NET technology at hand?

We lack completeness in both dimensions. For instance, we did not yet model the fact that OrX mapping is carried out 'for a purpose': some OO program is meant.

http://softlang.uni-koblenz.de/mega/
Research questions

• Can we do *heavy lifting* with megamodeling?
• Does a *general megamodeling language* exist?
  ‣ What are the *entities* of linguistic architecture?
  ‣ What are the *relationships* of interest?
  ‣ (What is a good visual syntax?)
• How to *validate megamodels*?
Heavy lifting with megamodeling

Claim by this speaker:
Megamodeling lifts heavily once it can explain, for example, **Object/Relational/XML mapping** at a high level of abstraction in a comprehensible and falsifiable manner.

More generally, megamodeling must help with managing diversity and heterogeneity of software technologies.
We have a problem.
Issues with software technologies

• Silos of knowledge
• Combining technologies
• Complexity of technologies
• Entering a new space
• Teaching technologies
Issues with software technologies

• Silos of knowledge
• Combining technologies
• Complexity of technologies
• Entering a new space
• Teaching technologies?

In need of ...

• analogies
• examples
• abstractions
Analogies, examples, abstractions

Meta language
- MOF
- XSD
- RDFS
- SQL.DDL
- EBNF

Navigation
- OCL
- XPath
- SPARQL
- SQL
- TXL

Query
- XQuery
- XSLT
- Protégé
- MySQL
- ASF

Transfo.
- QVT
- ArgoUML
- XMLSpy
- Topbeard
- MetaEnv.

Toolkit
- Rose
- VS-XML
- XMLSpy
- VS-XML
- MetaEnv.

Conferences
- MoDELS
- XML
- ICSW
- VLDB
- CC
- ECMDA
- VLDB
- ESWC
- SIGMOD
- POPL

© 2012, Software Languages Team @ University of Koblenz-Landau & Collaborators
Analogies, examples, abstractions

```prolog
company "meganalysis" {
    department "Research" {
        manager "Craig" {
            address "Redmond"
            salary 123456
        }
        employee "Erik" {
            address "Utrecht"
            salary 12345
        }
        employee "Ralf" {
            address "Koblenz"
            salary 1234
        }
    }
    department "Development" {
        manager "Ray" {
            address "Redmond"
            salary 234567
        }
    }
    ...  
}
```

© 2012, Software Languages Team @ University of Koblenz-Landau & Collaborators
Analogies, examples, abstractions

What’s the taxonomy of technologies?

What’s the essence of technology $xyz$?

This is where megamodeling kicks in!
Towards a general megamodeling language
Entities of megamodels

- **Sets**
  - Languages
  - Domains
  - Relations
    - e.g., meanings of models
    - Functions
      - e.g., meaning of tools

- **Elements**
  - Strings, trees, graphs
  - Models (values, instances)
  - Metamodels (types, schemas)
  - Pairs (related elements)

- **Singletons**
  - Tools
  - Other “black boxes”
Relationships of megamodels

- Membership ("elementsOf")
- Conformance ("conformsTo")
- Modeling ("models" / "representationOf")
- Correspondence ("correspondsTo")
- Reference ("refersTo")
- ...

The black hole of modeling.
(Towards) a megamodel of the ATL model transformation language and toolkit
Validation of megamodels:

How to know that we understand?
Associate each entity with "evidence".
<?xml version="1.0" encoding="UTF-8"?>
  <eClassifiers xsi:type="ecore:EClass" name="Company">
    <eStructuralFeatures xsi:type="ecore:EReference" name="depts" ordered="false" upperBound="-1" eType="#//Dept" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Dept">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="manager" ordered="false" lowerBound="1" eType="#//Employee" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="subunits" ordered="false" upperBound="-1" eType="#//Subunit" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Employee">
    <eStructuralFeatures xsi:type="ecore:EReference" name="person" ordered="false" lowerBound="1" eType="#//Person" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="salary" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Person">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="address" ordered="false" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Subunit">
    <eStructuralFeatures xsi:type="ecore:EReference" name="pu" ordered="false" eType="#//Employee" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="du" ordered="false" eType="#//Dept" containment="true"/>
  </eClassifiers>
</ecore:EPackage>
<?xml version="1.0" encoding="UTF-8"?>
nsURI="http://www.company.com" nsPrefix=""/>
<eClassifiers xsi:type="ecore:EClass" name="Company">
  <eClassifiers xsi:type="ecore:EClass" name="Dept">
    <eStructuralFeatures xsi:type="ecore:EReference" name="manager" ordered="false"
      lowerBound="1" eType="#//Employee" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="depts" ordered="false"
      upperBound="-1" eType="#//Dept" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Employee">
    <eStructuralFeatures xsi:type="ecore:EReference" name="person" ordered="false"
      lowerBound="1" eType="#//Person" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Person">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" lowerBound="1"
      eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="address" ordered="false"
      lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Subunit">
    <eStructuralFeatures xsi:type="ecore:EReference" name="pu" ordered="false" eType="#//Employee"
      containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="du" ordered="false" eType="#//Dept"
      containment="true"/>
  </eClassifiers>
</eClassifiers>
<?xml version="1.0" encoding="ISO-8859-1"?>
  <depts name="Research">
    <manager salary="123456.0">
      <person name="Craig" address="Redmond"/>
    </manager>
    <subunits>
      <pu salary="12345.0">
        <person name="Erik" address="Utrecht"/>
      </pu>
    </subunits>
    <subunits>
      <pu salary="1234.0">
        <person name="Ralf" address="Koblenz"/>
      </pu>
    </subunits>
  </depts>
  <depts name="Development">
    <manager salary="234567.0">
      <person name="Ray" address="Redmond"/>
    </manager>
    <subunits>
      <du name="Dev1">
        <manager salary="23456.0">
          <person name="Klaus" address="Boston"/>
        </manager>
        <subunits>
          <du name="Dev1.1">
            <manager salary="2345.5">
              <person name="Karl" address="Riga"/>
            </manager>
            <subunits>
              <pu salary="2344.0">
                <person name="Joe" address="Wifi City"/>
              </pu>
            </subunits>
          </du>
        </subunits>
      </du>
    </subunits>
  </depts>
</Company>
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
namespace="http://www.total.com">
  <eClassifiers xsi:type="ecore:EClass" name="TotalWrapper">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="total" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#EDouble"/>
  </eClassifiers>
</ecore:EPackage>
module Total;
create OUT: Total from IN: Company;

rule Company2Total {
    from company : Company!Company
to t : Total!TotalWrapper (total <- Company!Employee.allInstances() -> collect(e | e.salary) -> sum())
}
module Total;
create OUT: Total from IN: Company;

rule Company2Total {
  from
    company : Company!Company
  to
t : Total!TotalWrapper (t : Total!TotalWrapper (total <- Company!Employee.allInstances()
                     -> collect(e | e.salary) -> sum())
  )
}
Ecore Model

Source Metamodel

Source Model

conformsTo

ATL Metamodel

Target Metamodel

models

ATL Model

Target Model

conformsTo

ATL Transformation

conformsTo

?
What about the relationships?
Leveling expectations:
By no means, we are ATL experts.

Slogan for the rest:
Please help me to get these slides right (so that I can publish a great paper some day).
Discussion:

The complex nature of Ecore
Discussion:

*The complex nature of Ecore*

- XML
  - subsetOf
  - subsetOf
  - XMLish Ecore
    - correspondsTo(eachOther)
  - Ecore
  - elementOf
  - conformsTo
  - Ecore Model

*Italics* means “abstract”
Discussion: The role of ASM

- ASM
- XML
- ASM DTD
- ATL Library
- ATL Model

- subsetOf
- models
- conformsTo
- elementOf
- refersTo
Discussion:

The issue of compilation
Discussion:

The issue of compilation

Oops: the compiler does not process source and target metamodels (even though it could or perhaps should). Thanks to AtlanMod for pointing this out.
Discussion: The issue of interpretation
Discussion:

The issue of interpretation

Oops: the virtual machine processes source and target metamodels so that it can de-/serialize and type-check. Thanks to AtlanMod for pointing this out.
Further discussion topics

• KM3
• Compiler internals
• Eclipse support
• ...

© 2012, Software Languages Team @ University of Koblenz-Landau & Collaborators
Conclusion

• Megamodels model linguistic architecture.
• ATL requires a non-trivial megamodel.
• First ideas for such a megamodel were presented.
• Let’s work on the ultimate ATL megamodel.

Thanks!
Questions?