Linguistic architecture

University of Koblenz-Landau
Department of Computer Science
Andrei Varanovich
Software Languages Team

Based on the original joint work with Ralf Lämmel and Jean-Marie Favre

http://softlang.uni-koblenz.de/mega/
A typical software project

- **programming languages** (e.g., Java, PHP, JavaScript, or Python)
- **library-based languages** (e.g., JQuery, DOM API)
- **models**
- **schemata**
- **configurations** (e.g., Hibernate mapping)
- **domain-specific languages** (e.g., CSS, XSLT, or SQL)

“many” languages and technologies “somehow” related
Software architecture

- The software architecture of a system is the set of structures needed to reason about the system, which comprise *software elements, relations among them, and properties of both* [Clements, Paul; Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford (2010). Documenting Software Architectures: Views and Beyond, Second Edition. Boston: Addison-Wesley.]

*viewpoints* in the IEEE 1471/42010 ontology:

- Functional/logical viewpoint
- Code/module viewpoint
- Development/structural viewpoint
- Concurrency/process/runtime/thread viewpoint
- Physical/deployment/install viewpoint
- User action/feedback viewpoint
- Data view/data model

None of the viewpoints focuses on the “linguistic” aspect
Goal: Develop a megamodelling approach that is useful for understanding the linguistic architecture of software products supported by the MegaL language and the tool suite

[Some] motivations:

• megamodelling must help with managing diversity and heterogeneity of software technologies

• cognitive value is important!
A megamodel is a model of which at least some elements represent and/or refer to models or metamodels.

The notion of megamodel

• Megamodels are (ER) models.
• Entities of interest
  ‣ Languages
  ‣ Technologies (components thereof)
  ‣ Programs
  ‣ ...
• Relationships of interest
  ‣ Conformance
  ‣ Transformation
  ‣ ...

That’s a megamodel, too!

Bootstrapping a C compiler written in C, by compiling it using another compiler written in machine code.

More examples:
http://userpages.uni-koblenz.de/~laemmel/slecour/slides/mega.pdf
X/O Mapping in .NET

XML Schema Definition Tool (Xsd.exe)
generates XML schema or common language runtime classes from XDR, XML, and XSD files, or from classes in a runtime assembly.

XSD to Classes
Generates runtime classes from an XSD schema file. The generated classes can be used in conjunction with System.XML.Serialization.XmlSerializer to read and write XML code that follows the schema.
Megamodel for X/O mapping in .NET

Instances

doc → de-serialize → obj

Schemas

xmlTypes → xsd.exe → ooTypes

Languages

ClrObj → C# → XSD → XML

Formally valid. But comprehensive?

doc and xmlTypes hold the same “information capacity”
both (metamodel and model) are transformed
obj and ooTypes hold the same “information capacity”
Megamodel of O/X mapping with xsd.exe

**Language** XSD, CSharp, XML, ClrObjectGraphs.
**local Language** CSharpFromXsd subsetOf CSharp.
**variable** File+ xmlTypes elementOf XSD.
**variable** File+ ooTypes elementOf CSharpFromXsd.
**variable** File xmlDoc elementOf XML.
**variable** ObjectGraph clrObj elementOf ClrObjectGraphs.
**local Function** classgen : XSD -> CSharpFromXsd.
**local Function** deserialize : XML -> ClrObjectGraphs.
xmlDoc conformsTo xmlTypes.
clrObj conformsTo ooTypes.
classgen(xmlTypes) |-> ooTypes.
deserialize(xmlDoc) |-> clrObj.
Megamodel of a product using xsd.exe
The upper frame uses the MegaL/yEd visual notation for megamodeling. The lower frame shows some linked artifacts explained later in the paper.

Fig. 1. The linguistic architecture of a software product when displayed with the MegaL/Explorer tool.

Consider, for example, the fact that the class generator is not described as generating 'arbitrary' C\texttt{w}. Instead, a designated subset, \texttt{CSharpFromXsd}, is used because the generator indeed produces very regular code whose regularity helps with understanding Object/XML mapping, as we discuss later.

http://worker.101companies.org/MegaModels/implementations/xsdClasses/
Linked megamodels

An entity is linked to a unique resource that can be browsed and examined.
MegaL

Design goal: general-purpose megamodelling language

https://github.com/avaranovich/megal
Key design principles

• Entities and relationships are extendable
• Everything is a resource
• Relationships are executable
MegaL prelude

// Entity types
Set < Entity .
Artifact < Entity .
Language < Set .
Technology < Entity .
Concept < Entity .
File < Artifact .
Folder < Artifact .

// Relationship types
elementOf < Entity * Set .
subsetOf < Set * Set .
partOf < Artifact * Artifact .
instanceOf < Concept * Concept .
isA < Concept * Concept .
conformsTo < Artifact * Artifact .
definitionOf < Artifact * Language .
Some megamodels

/* Models, metamodels, and conformance */
Model : Artifact .
Metamodel : Model .
conformsTo : Model * Metamodel .

Java : Language .
extern JavaProgram : File .
JavaProgram elementOf Java .

/*
The MVC concept.
MVC consists of model, view, and controller.
Those subconcepts depend on the main concept.
MVC is a design pattern.
*/

MVC : Concept .
Model @ MVC : Concept .
View @ MVC : Concept .
Controller @ MVC : Concept .
DP ["Design pattern"] : Concept .

Model partOf MVC .
View partOf MVC .
Controller partOf MVC .
MVC instanceOf DP .
Design challenges

- Many source of (partial) errors
- Extensibility of the language, controlled by the type system
MegaL pipeline

Parse ➔ Extend ➔ Analyze ➔ Link ➔ Check

Entity/Relationship plugins

Resources

event bus

report ➔ event recorder
Config

{
"relationships": [
{
   "relationship": "megal.relationships.FileElementOfLanguage",
   "config": {
      "checkers": [
         {
            "resource": "http://101companies.org/resources/languages/Java",
            "checker": "megal.checkers.languages.Java",
            "type": "class"
         }]
   }
},
"linking": [
   {
      "type": "File",
      "name": "JavaProgram",
      "resource": "https://raw.github.com/avaranovich/megal/master/eclipse/megal/src/test/resources/files/basic.java"
   }
]
}
MegaL server

- python -m SimpleHTTPServer 8000
Questions?
Thank you!