Making EMF polyglot

Virtual Meetup Strumenta community, 25 February 2021

DSL Consultancy

Making EMF polyglot

Agenda

1. Introduction to / recap of EMF (Meinte, 10')
2. Horacio (JSOI), and Vincent (PyEcore) talk about their work (10' each)
3. Discussion about which feature to make polyglot first (15')
   Contenders:
   - Ecore as metamodeling standard
   - Definition and standardization of JSON-variant of XMI
   - JVM-agnostic standardization of EMF
4. Discussion about the way forward (15')

DSL Consultancy

Making EMF polyglot

Introduction and recap of EMF

- EMF = Eclipse Modeling Framework, alternatively the “Ed Merks Framework”
- Used in lots of software:

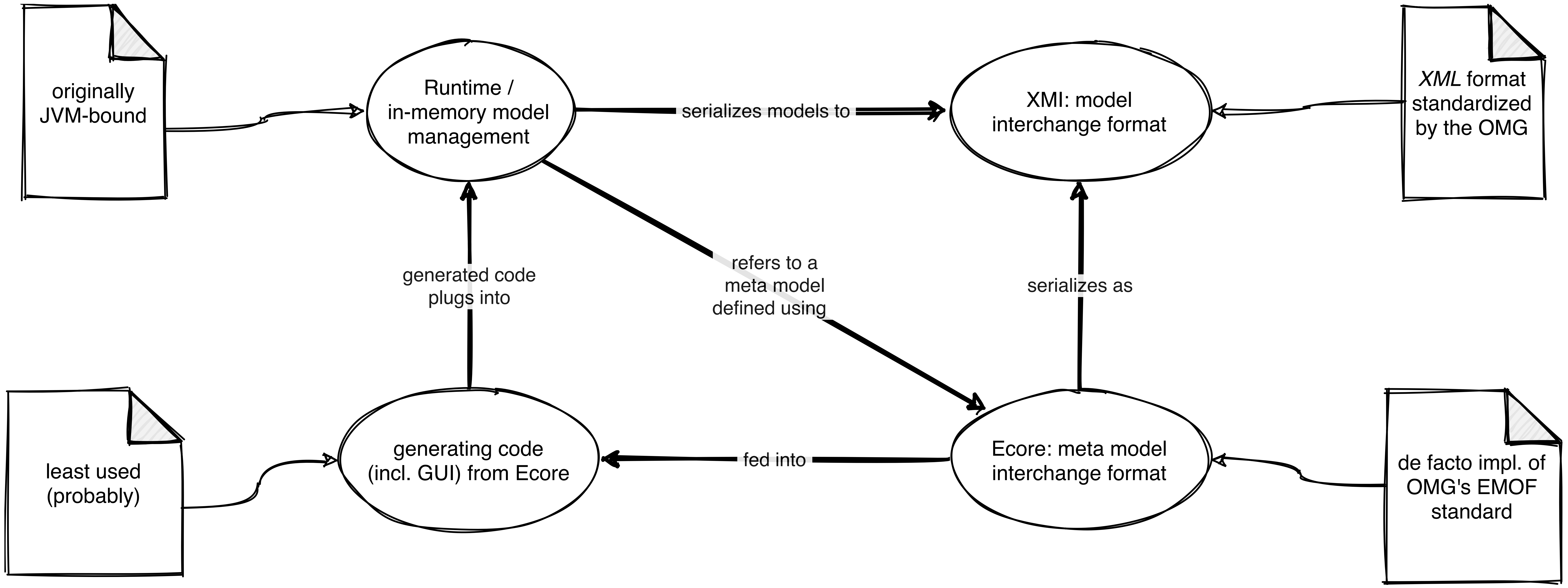
  Acceleo, Obeo, Xtext, CDO, GMF, ATL, Sirius, Rational Software Modeler, EMF.cloud, Sprotty, etc.

- Mostly bound to the JVM.

DSL CONSULTANCY

Making EMF polyglot

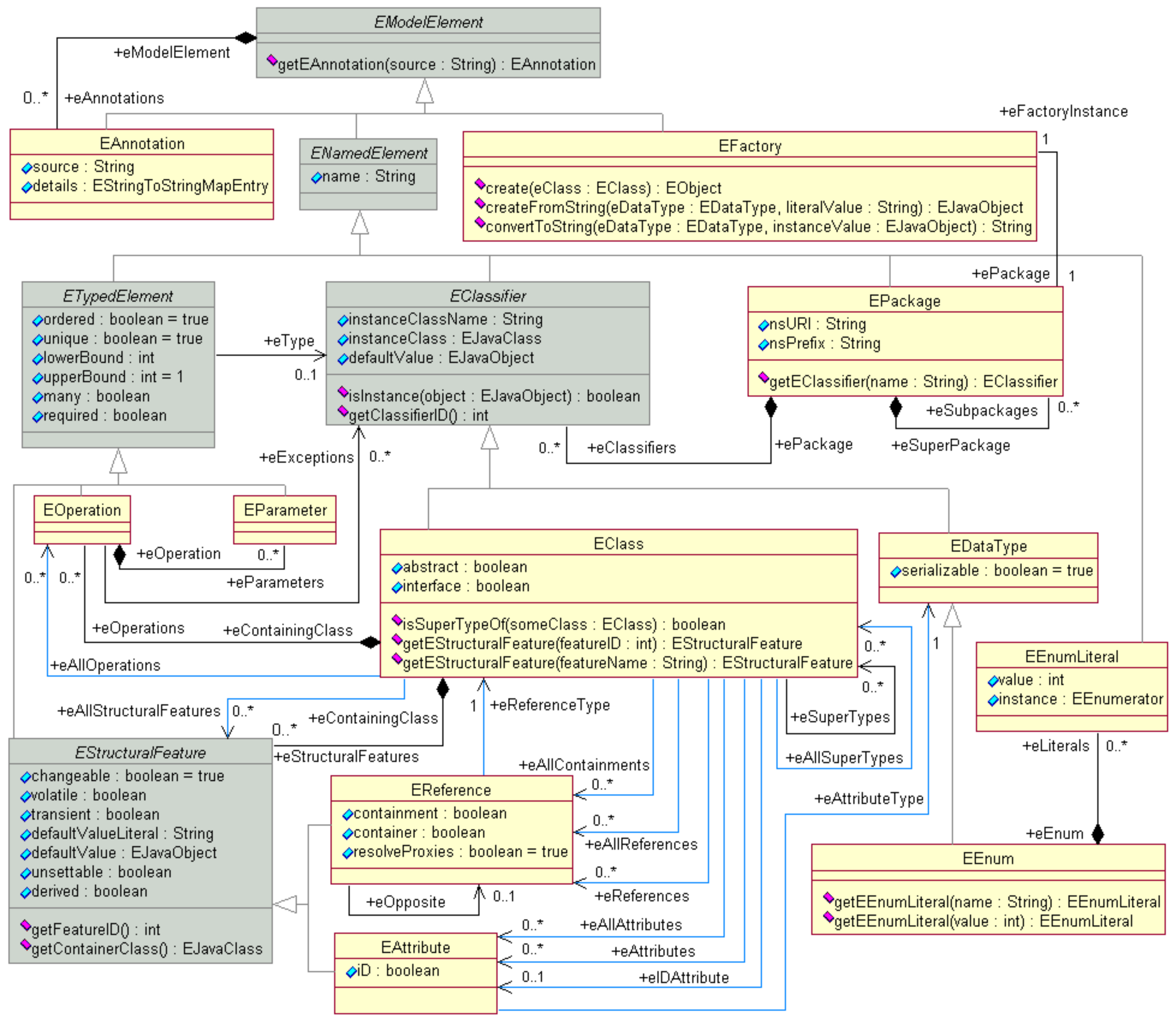
Introduction and recap of EMF


originally JVM-bound
Runtime / in-memory model management
serializes models to
XMI: model interchange format
XML format standardized by the OMG
generated code plugs into
refers to a meta model defined using
serializes as
least used (probably)
generating code (incl. GUI) from Ecore
fed into
Ecore: meta model interchange format
de facto impl. of OMG's EMOF standard

DSL Consultancy

Making EMF polyglot

Meta modeling with Ecore



Ecore as Ecore model

DSL Consultancy

Making EMF polyglot

Meta modeling with Ecore

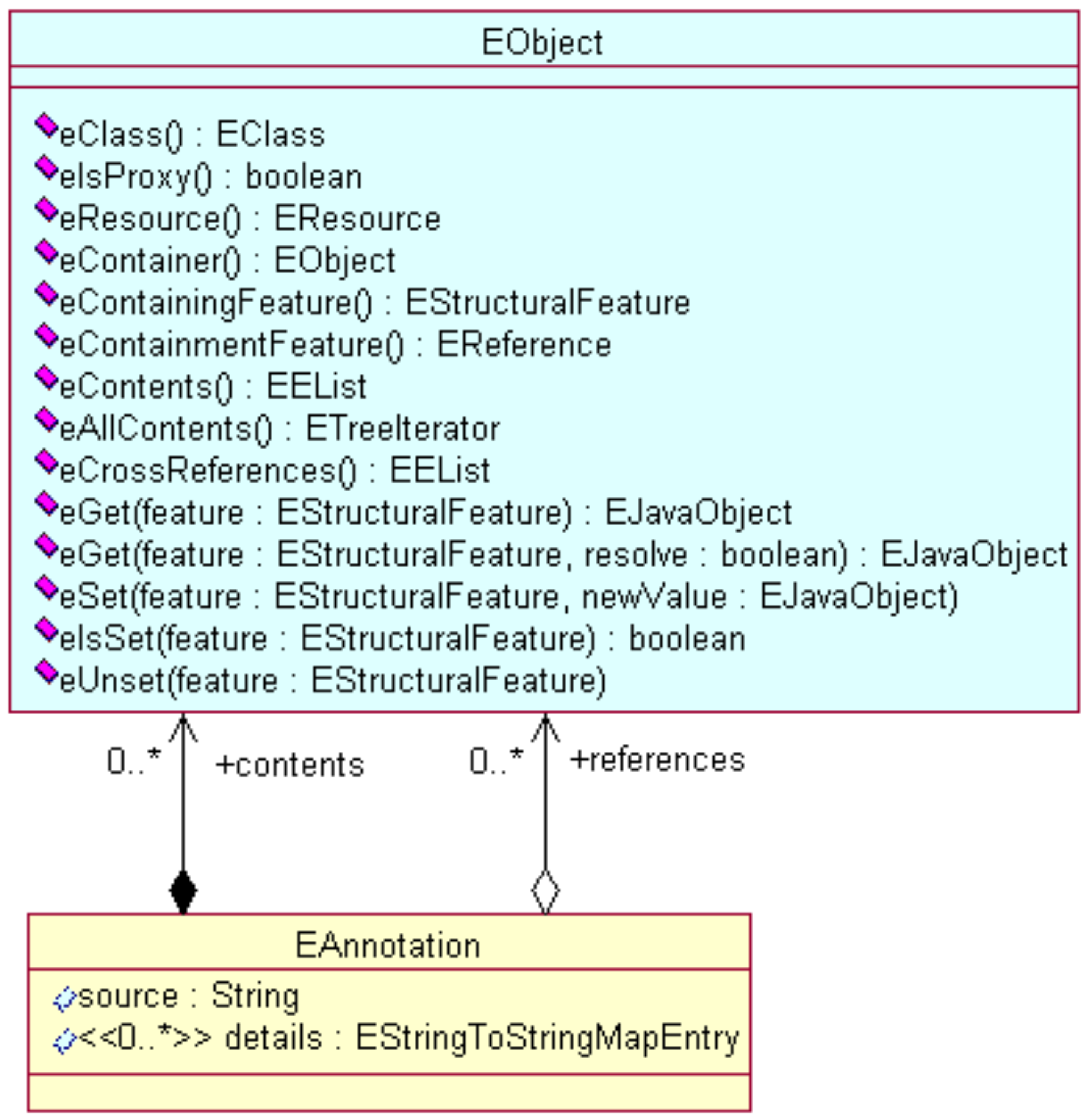
- It's “precisely enough”.
- Persisted in XMI format.
- Already has a degree of interoperability: tool support, generators.
- (Textual variant: Xcore).

Making EMF polyglot

Runtime model management - typical usage

1. Load model (`EResource`) from XMI file into memory.
2. Model is backed by an Ecore model.
3. An `EResource` is a collection of `EObjects`.
4. Save `EResource` back to XMI again.

- Features: reverse references, notifications, annotations, dynamic usage, etc.
- Large-scale solution: CDO = Connected Data Objects.


EObject
eClass() : EClass
eIsProxy() : boolean
eResource() : EResource
eContainer() : EObject
eContainingFeature() : EStructuralFeature
eContainmentFeature() : EReference
eContents() : EEList
eAllContents() : ETreeIterator
eCrossReferences() : EEList
eGet(feature : EStructuralFeature) : EJavaObject
eGet(feature : EStructuralFeature, resolve : boolean) : EJavaObject
eSet(feature : EStructuralFeature, newValue : EJavaObject)
eIsSet(feature : EStructuralFeature) : boolean
eUnset(feature : EStructuralFeature)
0..*
+contents
0..*
+references
EAnnotation
source : String
<<0..*>> details : EStringToStringMapEntry

Ecore representation of EObject

DSL Consultancy

Making EMF polyglot

XMI: model interchange format

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.1" xmlns:uml="http://schema.omg.org/spec/UML/2.0" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">
    <xmi:Documentation exporter="StarUML" exporterVersion="2.0"/>
    <uml:Model xmi:id="AAAAAAFjzEc8LY39RtY=" xmi:type="uml:Model" name="RootModel">
        <packagedElement xmi:id="AAAAAAFF+qBWK6M3Z8Y=" name="Model" visibility="public" xmi:type="uml:Model">
            <packagedElement xmi:id="AAAAAAFjy7/FJY0kKh8=" name="StateMachine1" visibility="public" isReentrant="true" xmi:type="uml:StateMachine">
                <region xmi:id="AAAAAAFjy7/FJo0lkWM=" visibility="public" xmi:type="uml:Region">
                    <subvertex xmi:id="AAAAAAFjy7/ZDo0reHg=" name="A" visibility="public" xmi:type="uml:State">
                        <entry xmi:id="AAAAAAFjy7/ssI1RuOM=" name="eA" visibility="public" isReentrant="true" xmi:type="uml:OpaqueBehav
                        <exit xmi:id="AAAAAAFjy8ASyY1f3qw=" name="xA" visibility="public" isReentrant="true" xmi:type="uml:OpaqueBehavi
                        <doActivity xmi:id="AAAAAAFjy7//q41YmXc=" name="dA" visibility="public" isReentrant="true" xmi:type="uml:Opaque
                    </subvertex>
                    <subvertex xmi:id="AAAAAAFjy8AqOI1m7+A=" visibility="public" xmi:type="uml:Pseudostate" kind="initial"/>
                    <subvertex xmi:id="AAAAAAFjy8A7RY2ICY8=" visibility="public" xmi:type="uml:FinalState"/>
                    <subvertex xmi:id="AAAAAAFjzEbnL43DlLY=" name="B" visibility="public" xmi:type="uml:State"/>
                    <transition xmi:id="AAAAAAFjy8AqzY13yCY=" visibility="public" xmi:type="uml:Transition" source="AAAAAAFjy8AqOI1m7+A=" t
                    <transition xmi:id="AAAAAAFjzEXaD42uNpg=" visibility="public" xmi:type="uml:Transition" source="AAAAAAFjy7/ZDo0reHg=" t
                        <effect xmi:id="AAAAAAFjzEXmBo3Atuk=" name="final" visibility="public" isReentrant="true" xmi:type="uml:OpaqueB
                    </transition>
                    <transition xmi:id="AAAAAAFjzEb74o3pJ2c=" visibility="public" xmi:type="uml:Transition" source="AAAAAAFjy7/ZDo0reHg=" t
                        <ownedMember xmi:id="AAAAAAFjzEcBq437tuw=" name="EV" visibility="public" xmi:type="uml:AnyReceiveEvent"/>
                        <trigger xmi:id="AAAAAAFjzEc8Lo3+Y3Q=" xmi:type="uml:Trigger" name="EV" event="AAAAAAFjzEcBq437tuw="/>
                        <trigger xmi:id="AAAAAAFjzEcBq437tuw=" name="EV" visibility="public" xmi:type="uml:AnyReceiveEvent"/>
                    </transition>
                </region>
            </packagedElement>
        </packagedElement>
    </uml:Model>
</xmi:XMI>
```

Some UML as XMI

DSL Consultancy

Making EMF polyglot

XMI: model interchange format

- OMG standard.
- Used in many modeling tools, e.g. for UML.
- *Con:* it's XML...
  (JSON is more prevalent these days).

Making EMF polyglot

Existing work outside of Eclipse

- PyEcore (Python) by *Vincent Aranega* (see next slides)
- JSOI (JVM) by *Horacio Hoyos Rodríguez*
- ecore.js (JS) and emfjson-jackson (JVM) by Guillaume Hillairet

PyEcore

Ecore in Python

https://github.com/pyecore/pyecore

Quick stuffs about me

- Assistant professor at University of Lille -- 2018 - now
  - Working on Pharo/Smalltalk about dynamic language, language interoperability, debugging, virtual machines
  - Software evolution, MDE and code/AST transformations
- R&D Project Manager at GenMyModel -- 2011 - 2018
  - worked on real-time modeling collaboration, modeling, codegen and language engineering in the cloud.
- PhD on test/debugging models and transformation chains -- 2008 - 2011

Why PyEcore

WARNING

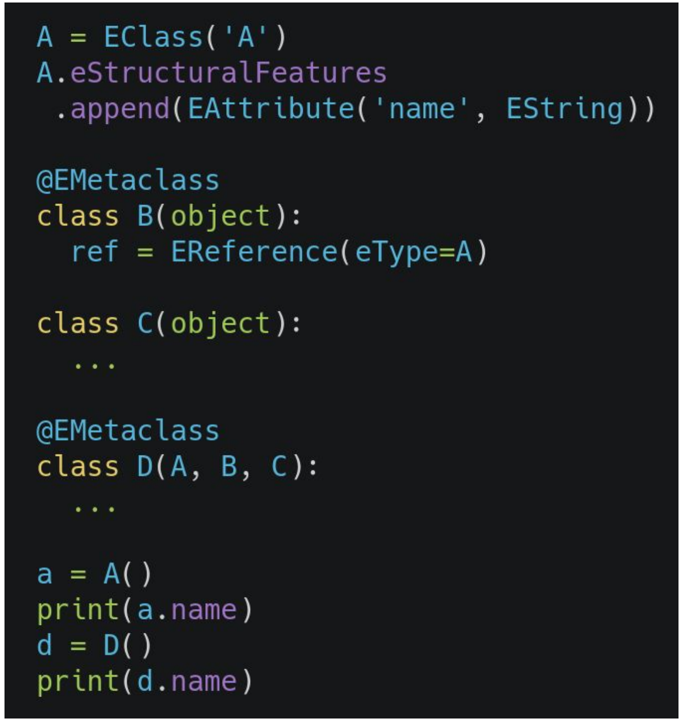
With real non-objective and non-relevant opinions

- EMF is great but
  - Heavy to use (Eclipse, dependency, Java)
  - Static and dynamic flavours induce different way of coding
  - Doing quick experiments with is painful
  - Not suited for some experiments relying on dynamicity (because of Java)
  - Eclipse EMF tooling sometimes painful to use or to reuse in a non-Eclipse environment
  - Back in the days (2011 to 2018...), hell of multiple same? artifacts in maven central
- Python
  - Suits well for software development and quick scripts
  - OCL like syntax
  - Dynamic language, reflexive layer (introspection/intercession), open classes, ...
  - Huge collection of libraries
  - Easy to deploy

PyEcore

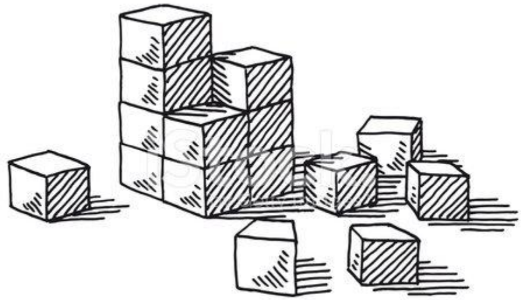
- First version in 2017
- Full Python with few dependencies
- 11 direct or indirect contributors

- Ease manipulation of EMF models
- Align dynamic and static metamodels
- Keep dynamic Python nature (model DU)
- See what dynamic language can bring to MDE
- Experiment with Models/Metamodels
- XMI/JSON (json emf-Jackson format)



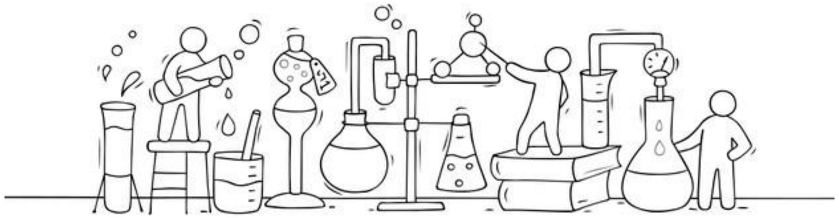
```
A = EClass('A')
A.eStructuralFeatures
 .append(EAttribute('name', EString))

@EMetaclass
class B(object):
  ref = EReference(eType=A)

class C(object):
  ...

@EMetaclass
class D(A, B, C):
  ...

a = A()
print(a.name)
d = D()
print(d.name)
```

Built on or use PyEcore



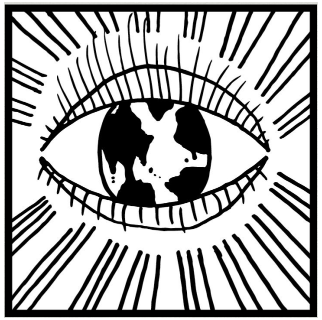
- PyEcoregen: code generator from Ecore to Python
- PyUML2: implementation of UML2 compatible with Eclipse UML2
- PyGeppetto (MetaCell): brain-cells simulation (serialization and API)
- NetPyNE (MetaCell): development, parallel simulation and analysis of biological neuronal networks
- ESDL: modelling language created to describe energy systems (serialization and API)
- Essential Object Query (EOQ): a language to interact remotely with object-oriented models
- Bridge between Fame MetaMeta in Smalltalk and Ecore (gen using Mako)
- Internal tooling in some companies of the automotive/aeronautic sector

Experimental projects



- TextX + PyEcore
- Motra: Model to model transformation lib for Python inspired by QVTo
- Advanced Traceability for Motra
- PyEcoreOCL: a library giving a more OCL like feeling for Python and
  a transpiler from OCL to Python
- PlantUML2UML2: create your UML model from plantUML syntax
- TUI Generic Editor: a generic Textual User Interface tree like editor
- SysML implementation
- Generic codegen lib
- Experimentations around self-modifying metamodels/models (for fun)

Future features



- Copy-paste
- Effective clone and deep-clone (already prototypes from contributors)
- JSOI and EMF binary resource implementation (tricky)
- Reimplementation of a concept I provided for GMM for command/model manipulation by transactions
- Stabilizing Motra (M2M library) for a first open release
- Better M2T library/framework (currently using either Jinja2 or Mako)
- Waiting to play with new pattern matching feature from Python
- Look at Python dataclasses
- Always improving performances

Making EMF polyglot

Idea / motivation

- EMF works very well, but is bound mostly to JVM.
- Idea: make EMF more polyglot.
- Questions:
  - Does this idea make sense?
  - What use cases would be served?
  - What aspect of EMF provides the most value?

DSL Consultancy

Making EMF polyglot

Topics “wall”

Which topic would we like to discuss:

1. Ecore as metamodeling standard
2. Definition and standardization of JSON-variant of XMI
3. JVM-agnostic standardization of EMF

The meeting voted for #3

DSL Consultancy

Making EMF polyglot

Links to other links

- The Strumenta forum thread that started it all
- GitHub repo with lots of links: https://github.com/dslmeinte/polyglot-emf/

DSL Consultancy

Making EMF polyglot

Post-meeting recap

(This is quite a subjective, personal recap!)

- There's a lot of value created through the ability to exchange models, including their meta models, across multiple languages/ecologies/platforms, especially by combining the features of the language with accessing exchanged models in an standard “EMF-idiomatic” way.

- Idea: have a list/“prioritized Backlog” of requirements, and/or a comprehensive test suite, to help with porting the most important parts of EMF to other languages.

DSL Consultancy

Making EMF polyglot

Imploration

Please share your experiences with exchanging with and consuming EMF models on other platforms than the JVM/Eclipse.

You can share them in the Strumenta forum thread that started it all.

I'll make sure to record a link (to the post or a blog post) in the polyglot-emf GitHub repo.

DSL Consultancy