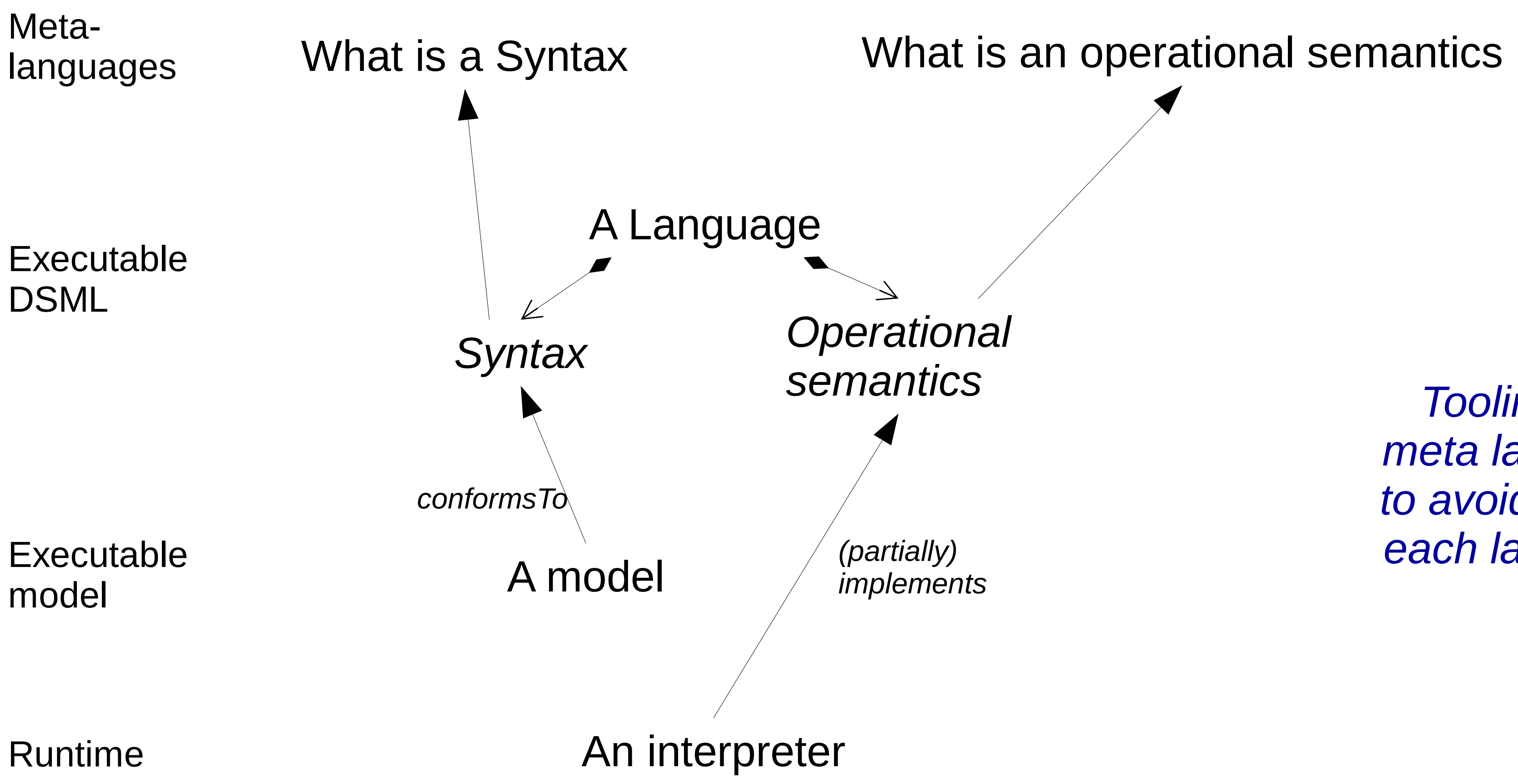
Meta-Languages and the GEMOC studio

Julien Deantoni

GEMOC approach : context

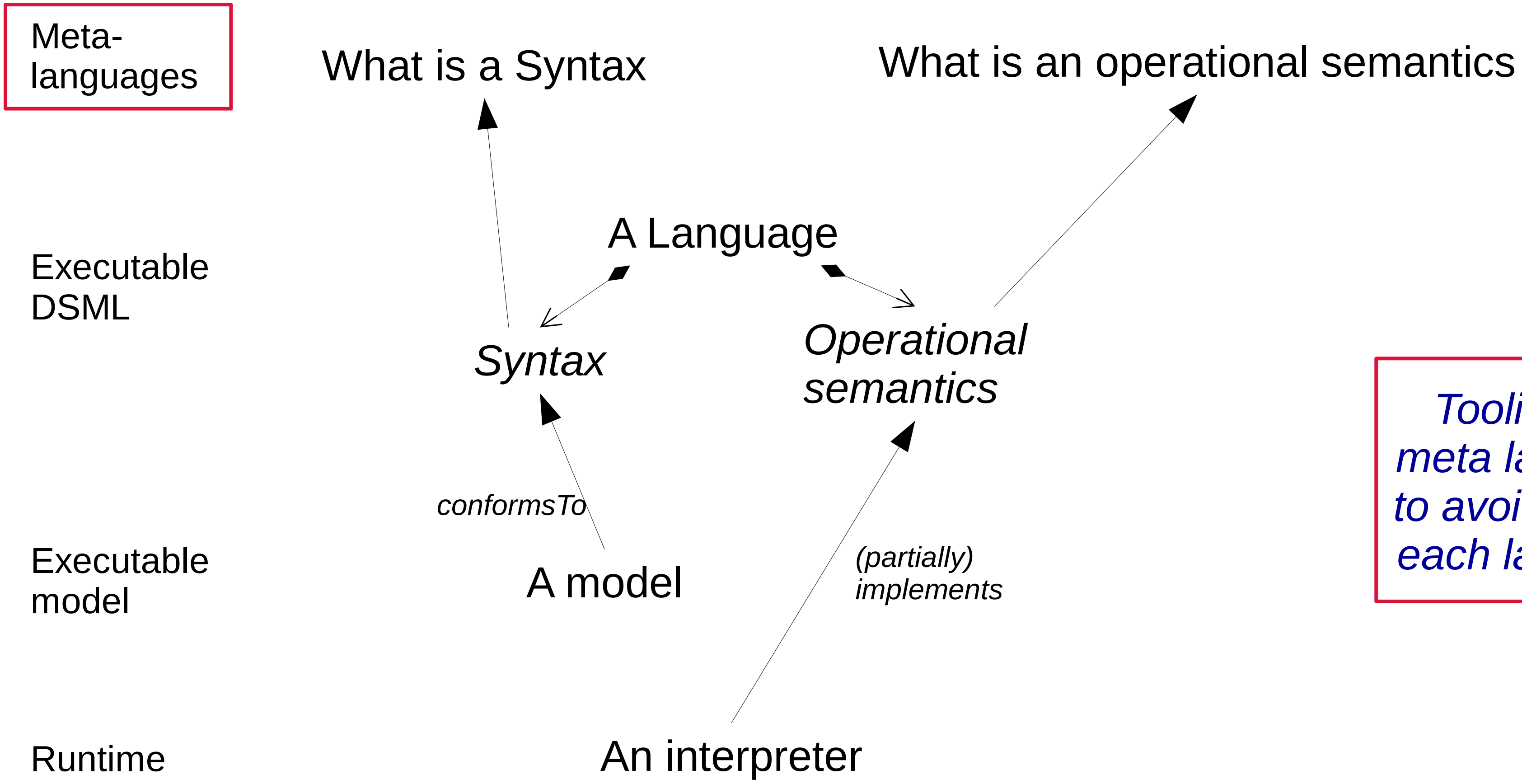
- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



Tooling the meta language to avoid tooling each language

GEMOC approach : context

- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



Meta-languages

What is a Syntax

What is an operational semantics

Executable DSML

A Language

Syntax

Operational semantics

conformsTo

Executable model

A model

(partially) implements

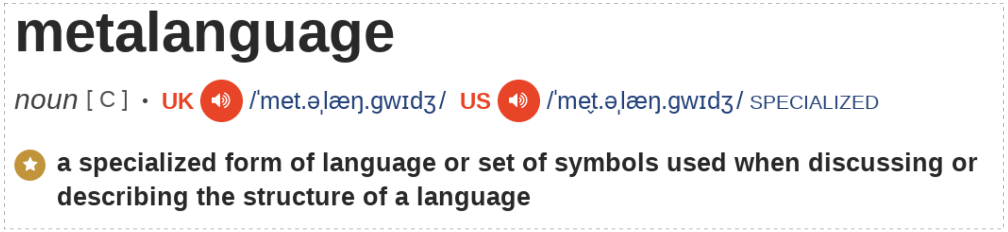
Runtime

An interpreter

Tooling the meta language to avoid tooling each language

Meta Languages

- A MetaLanguage is a Language.
- It’s a langage to specify (part of) a language.

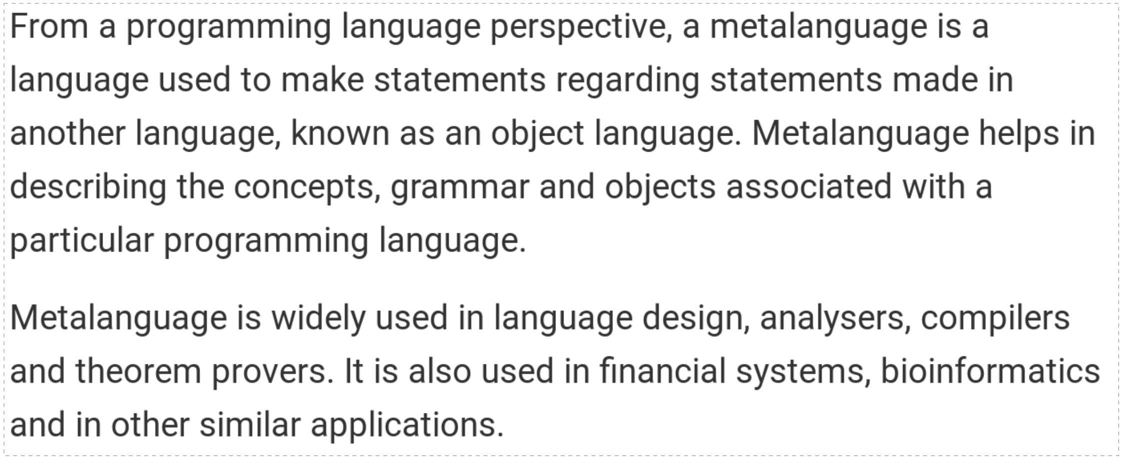


metalanguage

noun [C] • **UK** /ˈmet.əˌlæŋ.ɡwɪdʒ/ **US** /ˈmet̬.əˌlæŋ.ɡwɪdʒ/ SPECIALIZED

a specialized form of language or set of symbols used when discussing or describing the structure of a language

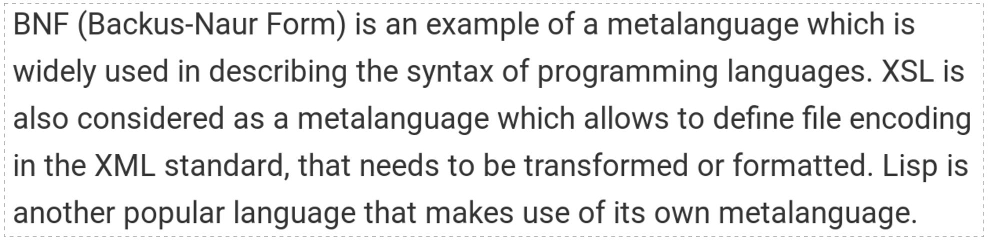
https://dictionary.cambridge.org/fr/dictionnaire/anglais/metalanguage



From a programming language perspective, a metalanguage is a language used to make statements regarding statements made in another language, known as an object language. Metalanguage helps in describing the concepts, grammar and objects associated with a particular programming language.

Metalanguage is widely used in language design, analysers, compilers and theorem provers. It is also used in financial systems, bioinformatics and in other similar applications.

https://www.techopedia.com/definition/8316/metalanguage

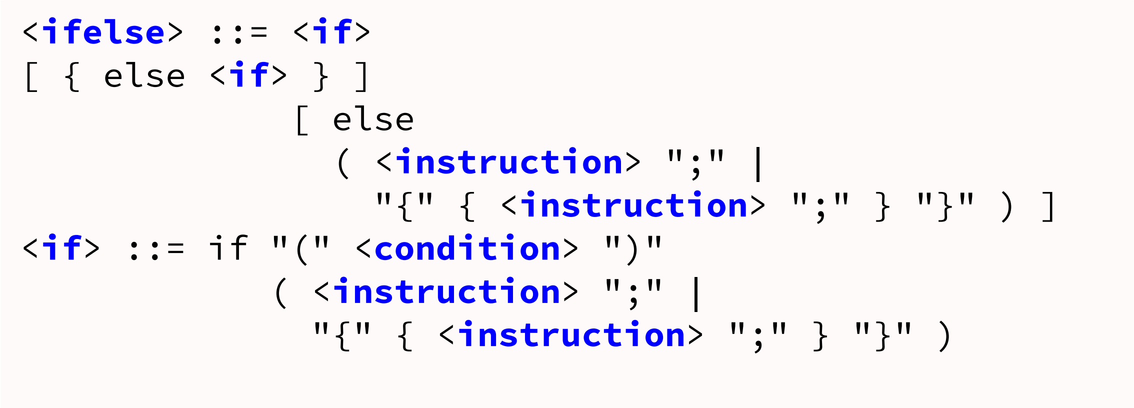


BNF (Backus-Naur Form) is an example of a metalanguage which is widely used in describing the syntax of programming languages. XSL is also considered as a metalanguage which allows to define file encoding in the XML standard, that needs to be transformed or formatted. Lisp is another popular language that makes use of its own metalanguage.

https://www.techopedia.com/definition/8316/metalanguage

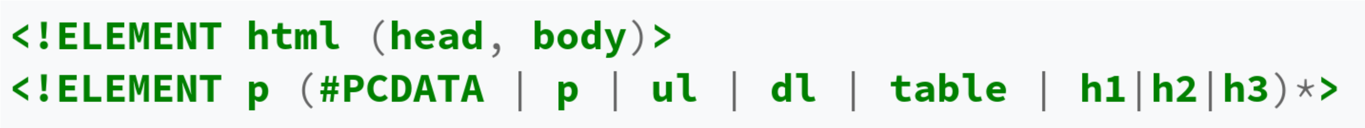
Meta Languages for syntax

- BNF

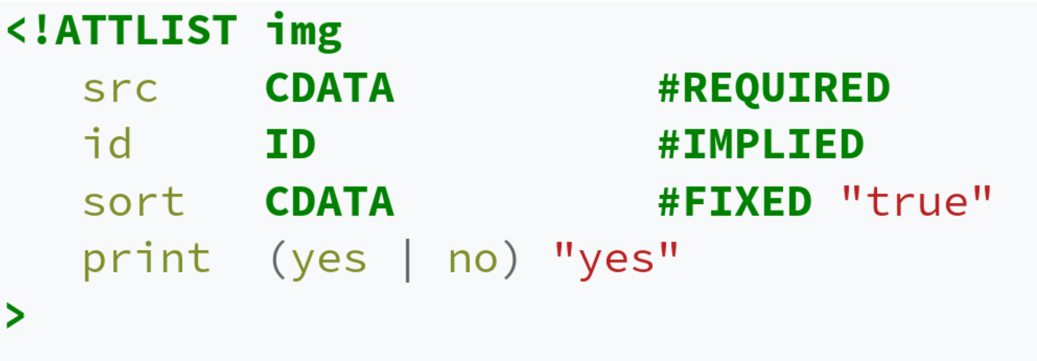


```
<ifelse> ::= <if>
[ { else <if> } ]
              [ else
                ( <instruction> ";" |
                  "{" { <instruction> ";" } "}" ) ]
<if> ::= if "(" <condition> ")"
             ( <instruction> ";" |
               "{" { <instruction> ";" } "}" )
```

- DTD



```
<!ELEMENT html (head, body)>
<!ELEMENT p (#PCDATA | p | ul | dl | table | h1|h2|h3)*>
```

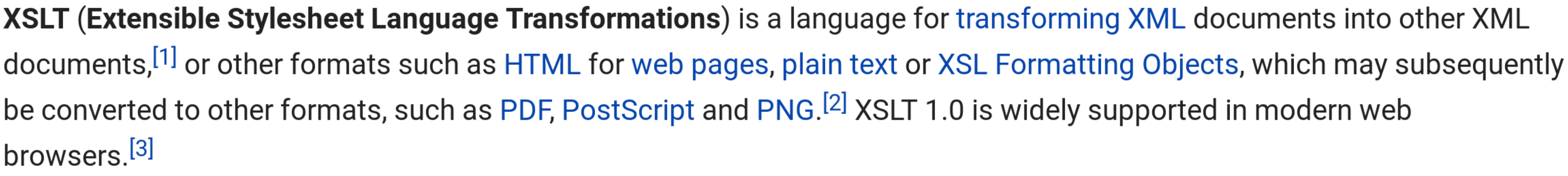


```
<!ATTLIST img
   src    CDATA          #REQUIRED
   id     ID             #IMPLIED
   sort   CDATA          #FIXED "true"
   print  (yes | no) "yes"
>
```

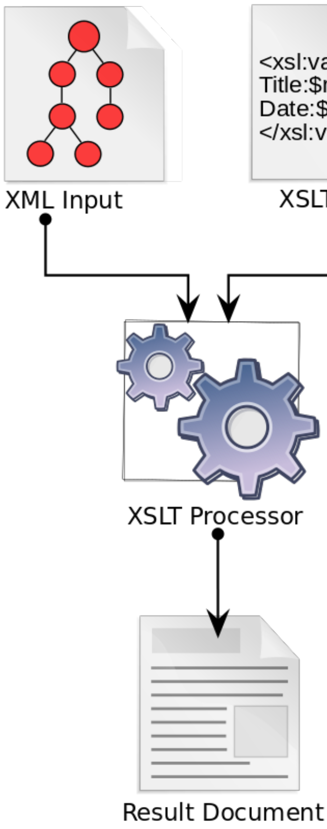
Examples taken from wikipedia

Meta Languages for semantics

- XSLT : specified by a specific DTD, it is consequently an XML document.



XSLT (**Extensible Stylesheet Language Transformations**) is a language for transforming XML documents into other XML documents,[1] or other formats such as HTML for web pages, plain text or XSL Formatting Objects, which may subsequently be converted to other formats, such as PDF, PostScript and PNG.[2] XSLT 1.0 is widely supported in modern web browsers.[3]



```
<xsl:value-of>
Title:$name
Date:$curdat
</xsl:value-o
```

Examples and pictures taken from wikipedia

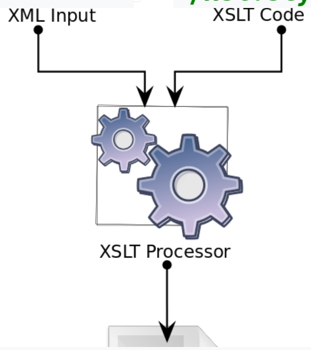
Meta Languages for semantics

- XSLT : specified by a specific DTD, it is consequently an XML document.

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="/persons">
    <root>
      <xsl:apply-templates select="person"/>
    </root>
  </xsl:template>

  <xsl:template match="person">
    <name username="{@username}">
      <xsl:value-of select="name" />
    </name>
  </xsl:template>

</xsl:stylesheet>
```

Based on templates

```xml
<?xml version="1.0" ?>
<persons>
  <person username="JS1">
    <name>John</name>
    <family-name>Smith</family-name>
  </person>
  <person username="MI1">
    <name>Morka</name>
    <family-name>Ismincius</family-name>
  </person>
</persons>
```

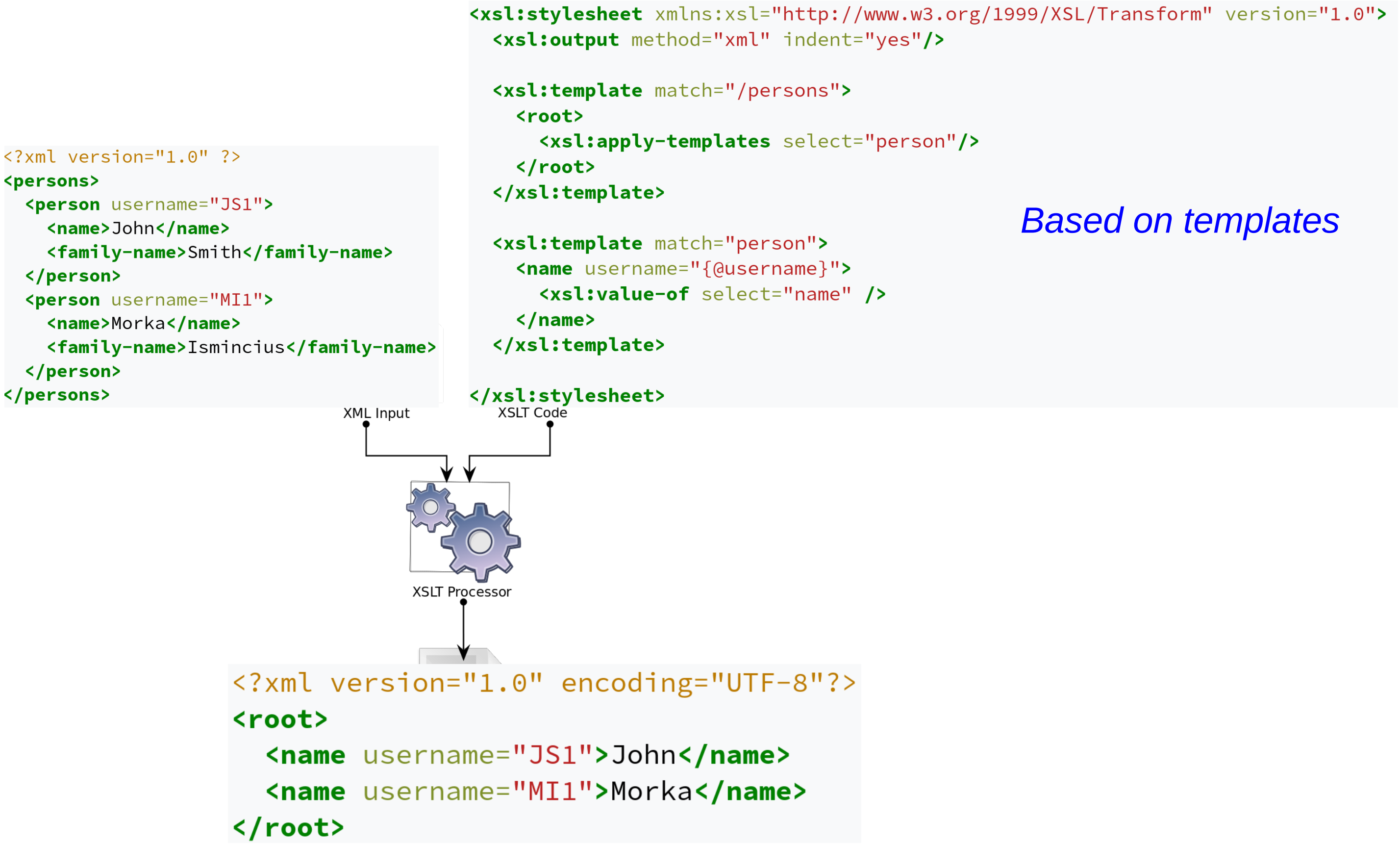


```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name username="JS1">John</name>
  <name username="MI1">Morka</name>
</root>
```

Examples and pictures taken from wikipedia

Meta Languages for semantics

- XSLT : specified by a specific DTD, it is consequently an XML document.



```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="/persons">
    <root>
      <xsl:apply-templates select="person"/>
    </root>
  </xsl:template>

  <xsl:template match="person">
    <name username="{@username}">
      <xsl:value-of select="name" />
    </name>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<persons>
  <person username="JS1">
    <name>John</name>
    <family-name>Smith</family-name>
  </person>
  <person username="MI1">
    <name>Morka</name>
    <family-name>Ismincius</family-name>
  </person>
</persons>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name username="JS1">John</name>
  <name username="MI1">Morka</name>
</root>
```

Based on templates

- For examples based on « BNF » like langage, check ANTLR translation rules (*e.g.*, https://theantlrguy.atlassian.net/wiki/spaces/ST/pages/1409118/Language+Translation+Using+ANTLR+and+StringTemplate)

Examples and pictures taken from wikipedia

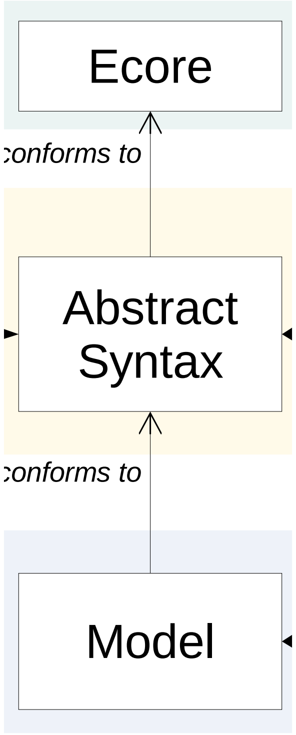
The GEMOC approach

Meta-languages

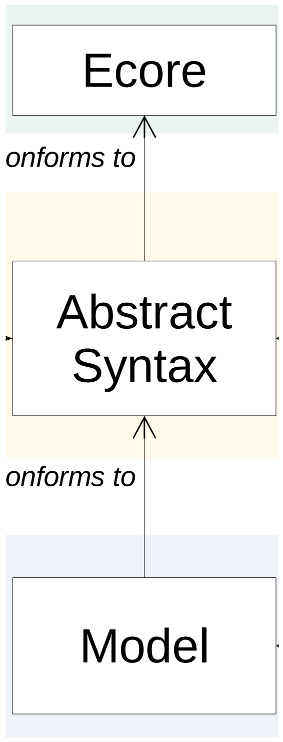
Executable DSML

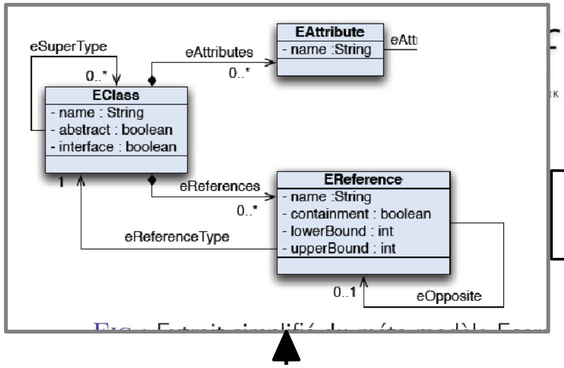
Executable model

Runtime



Tooling a Meta Language, simplified example



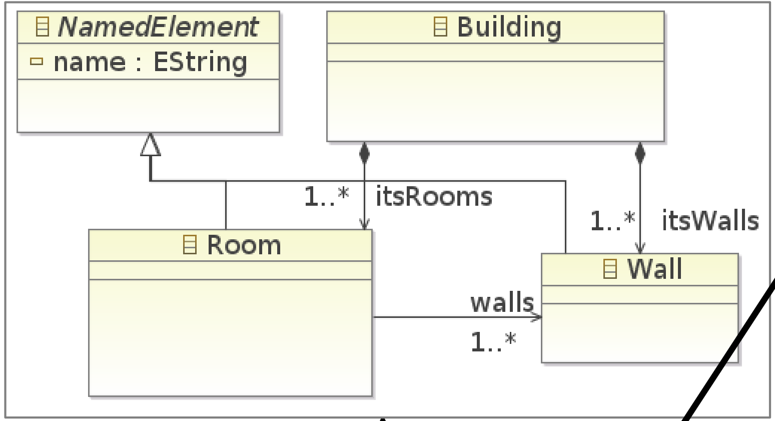


.XML

compiler

```
for(ea : allEClass){
 print(‘public class ’+ea.name)
 print(‘ extends ’
    +ea.eSuperType.get(0).name)
 print(‘{\n‘)
 for(er : eReferences){
    if(er.upperBound > 1){
     print(‘ArrayList<’+er.eReferenceType.name
```

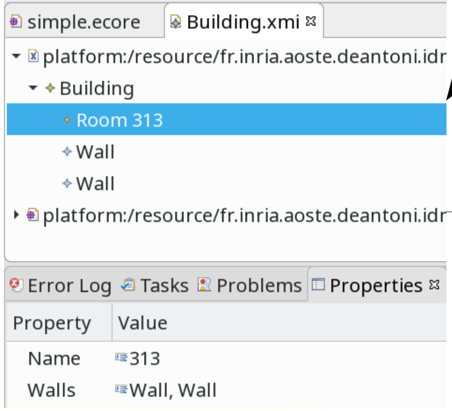
conformsTo



```
public class Room extends NamedElement {
     ArrayList<Wall> walls ;
}
```

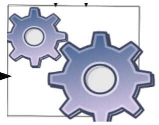
.java

conformsTo

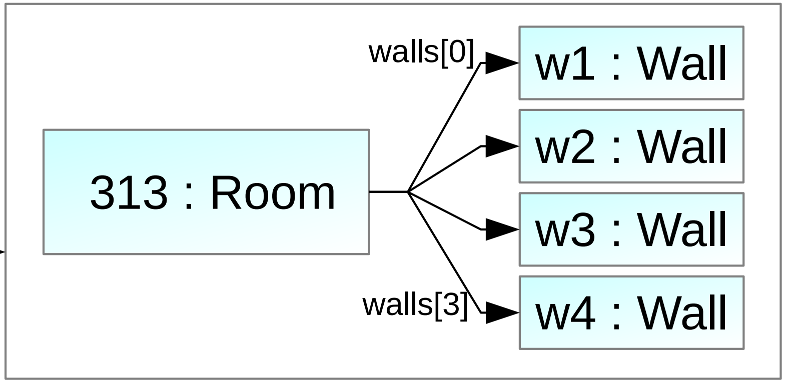


```
▾ platform:/resource/fr.inria.aoste.deantoni.idr
  ▾ Building
      Room 313
      Wall
      Wall
▸ platform:/resource/fr.inria.aoste.deantoni.idr
```

Property	Value
Name	313
Walls	Wall, Wall

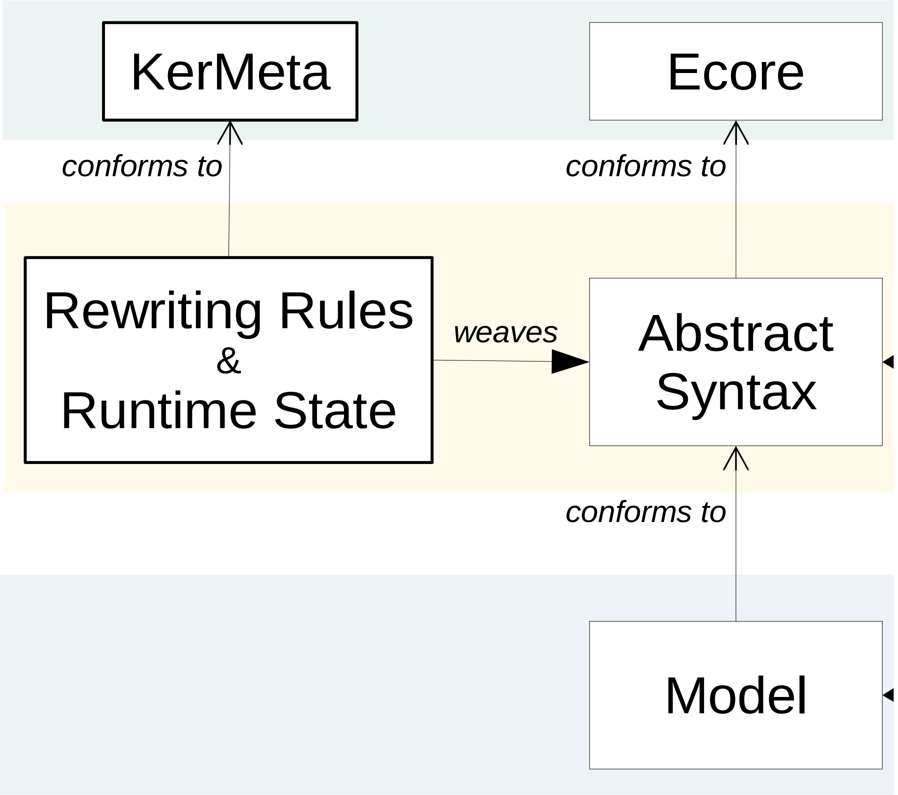


parser



Object graph

The GEMOC Approach



KerMeta

Ecore

conforms to

conforms to

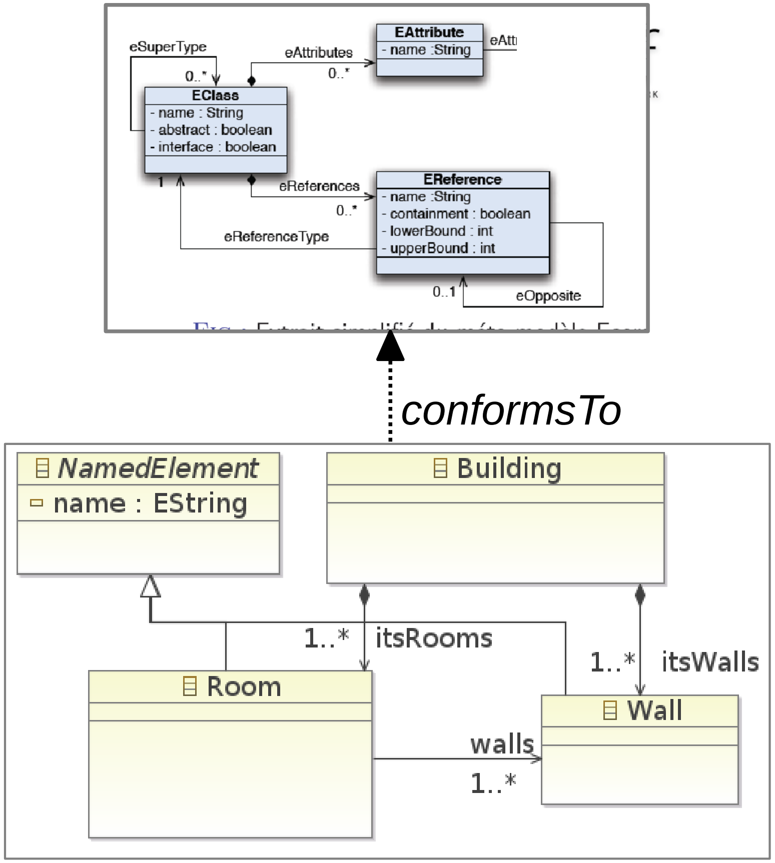
Rewriting Rules & Runtime State

weaves

Abstract Syntax

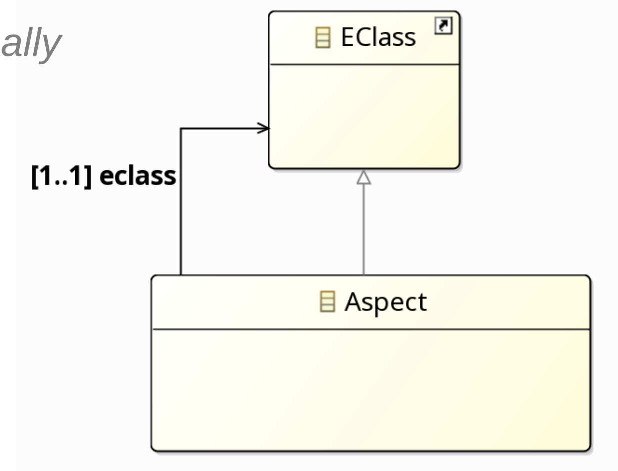
conforms to

Model

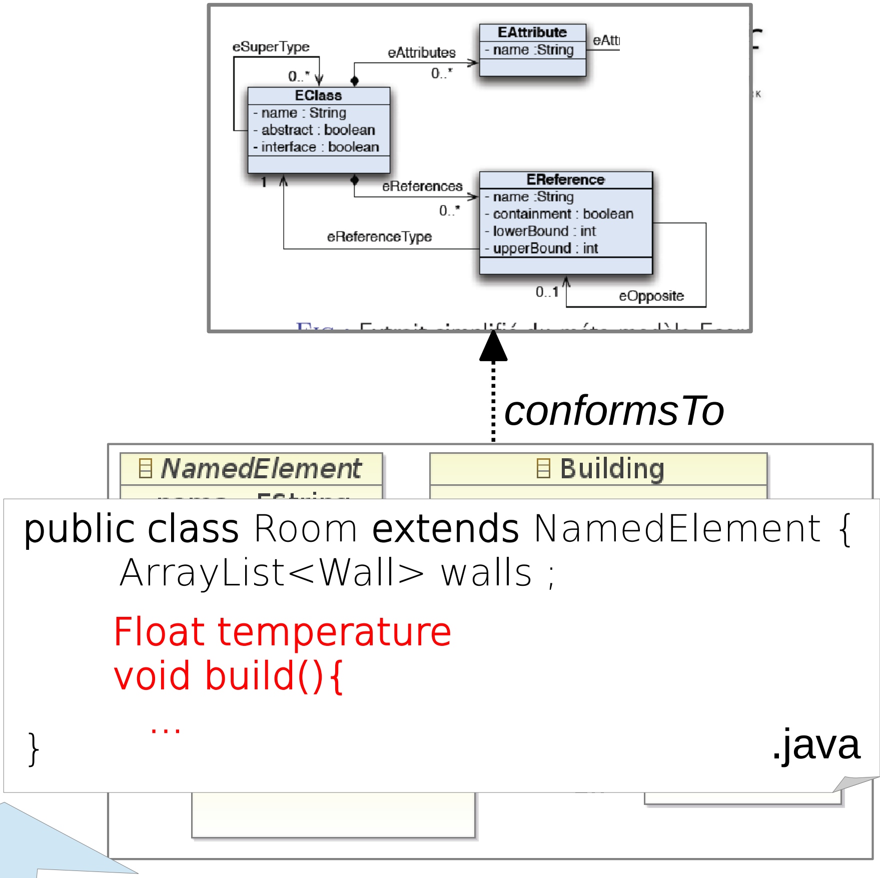


Tooling of meta langages : k3

conceptually



```
@Aspect(class=Room)
class RoomAspect {
        Float temperature
        void build() {
        …
      }
}
```



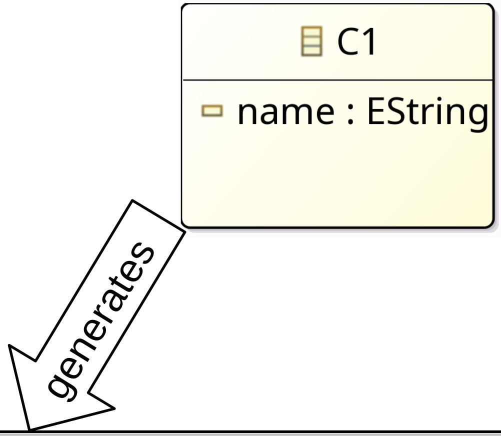
```
public class Room extends NamedElement {
    ArrayList<Wall> walls ;
    Float temperature
    void build(){
      …
}
```

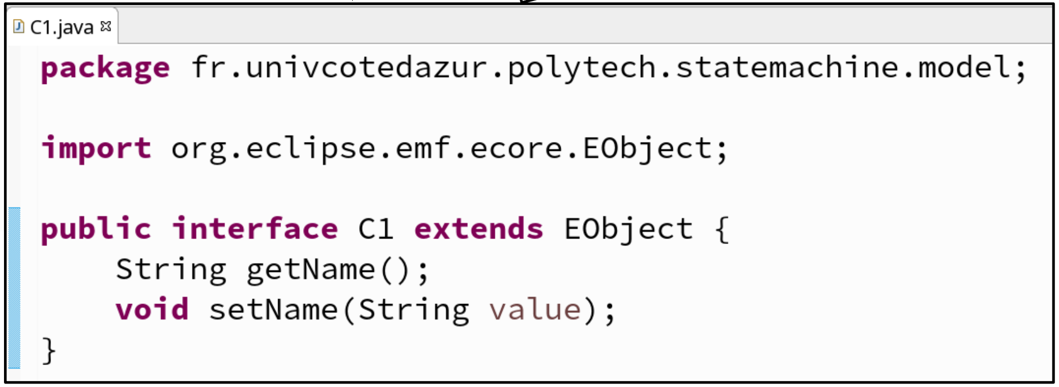
.java

Several strategies here along the development.
Conceptually it always *reopens* the class...

K3 generated code

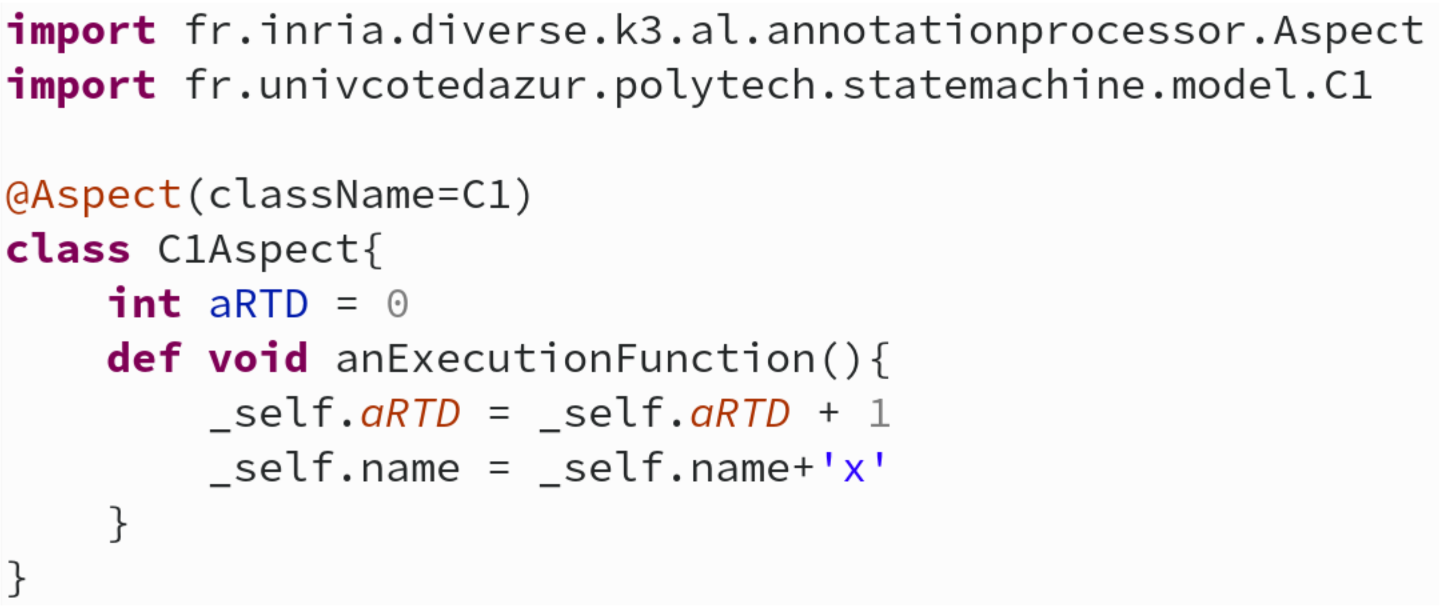
```
import fr.inria.diverse.k3.al.annotationprocessor.Aspect
import fr.univcotedazur.polytech.statemachine.model.C1

@Aspect(className=C1)
class C1Aspect{
    int aRTD = 0
    def void anExecutionFunction(){
        _self.aRTD = _self.aRTD + 1
        _self.name = _self.name+'x'
    }
}
```



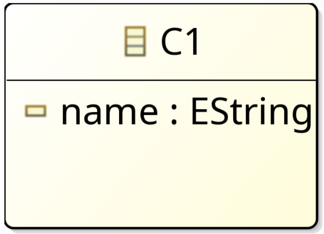


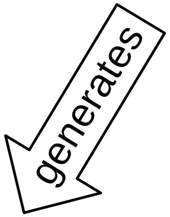
```
package fr.univcotedazur.polytech.statemachine.model;

import org.eclipse.emf.ecore.EObject;

public interface C1 extends EObject {
    String getName();
    void setName(String value);
}
```

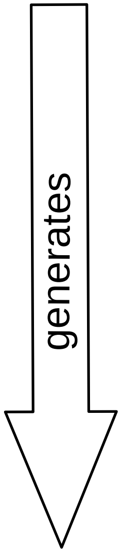
K3 generated code

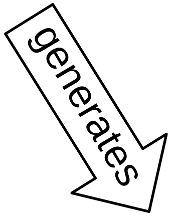


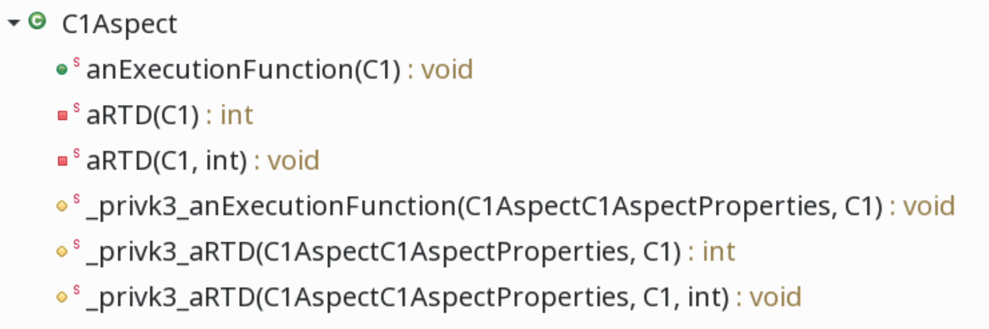
```
import fr.inria.diverse.k3.al.annotationprocessor.Aspect
import fr.univcotedazur.polytech.statemachine.model.C1

@Aspect(className=C1)
class C1Aspect{
    int aRTD = 0
    def void anExecutionFunction(){
        _self.aRTD = _self.aRTD + 1
        _self.name = _self.name+'x'
    }
}
```



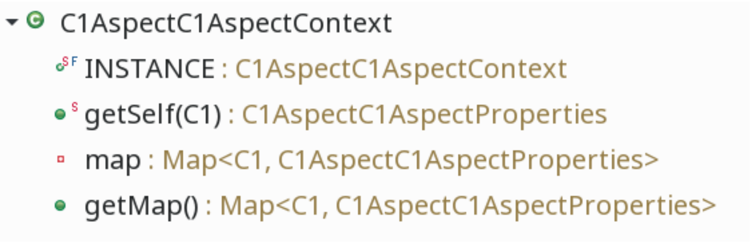




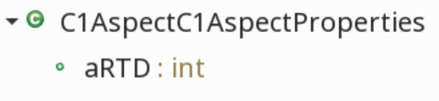




- C1Aspect
  - anExecutionFunction(C1) : void
  - aRTD(C1) : int
  - aRTD(C1, int) : void
  - _privk3_anExecutionFunction(C1AspectC1AspectProperties, C1) : void
  - _privk3_aRTD(C1AspectC1AspectProperties, C1) : int
  - _privk3_aRTD(C1AspectC1AspectProperties, C1, int) : void

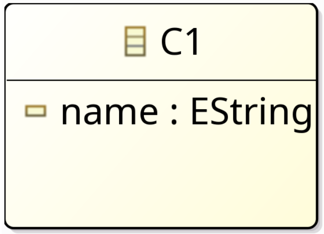


- C1AspectC1AspectContext
  - INSTANCE : C1AspectC1AspectContext
  - getSelf(C1) : C1AspectC1AspectProperties
  - map : Map<C1, C1AspectC1AspectProperties>
  - getMap() : Map<C1, C1AspectC1AspectProperties>



- C1AspectC1AspectProperties
  - aRTD : int

K3 generated code

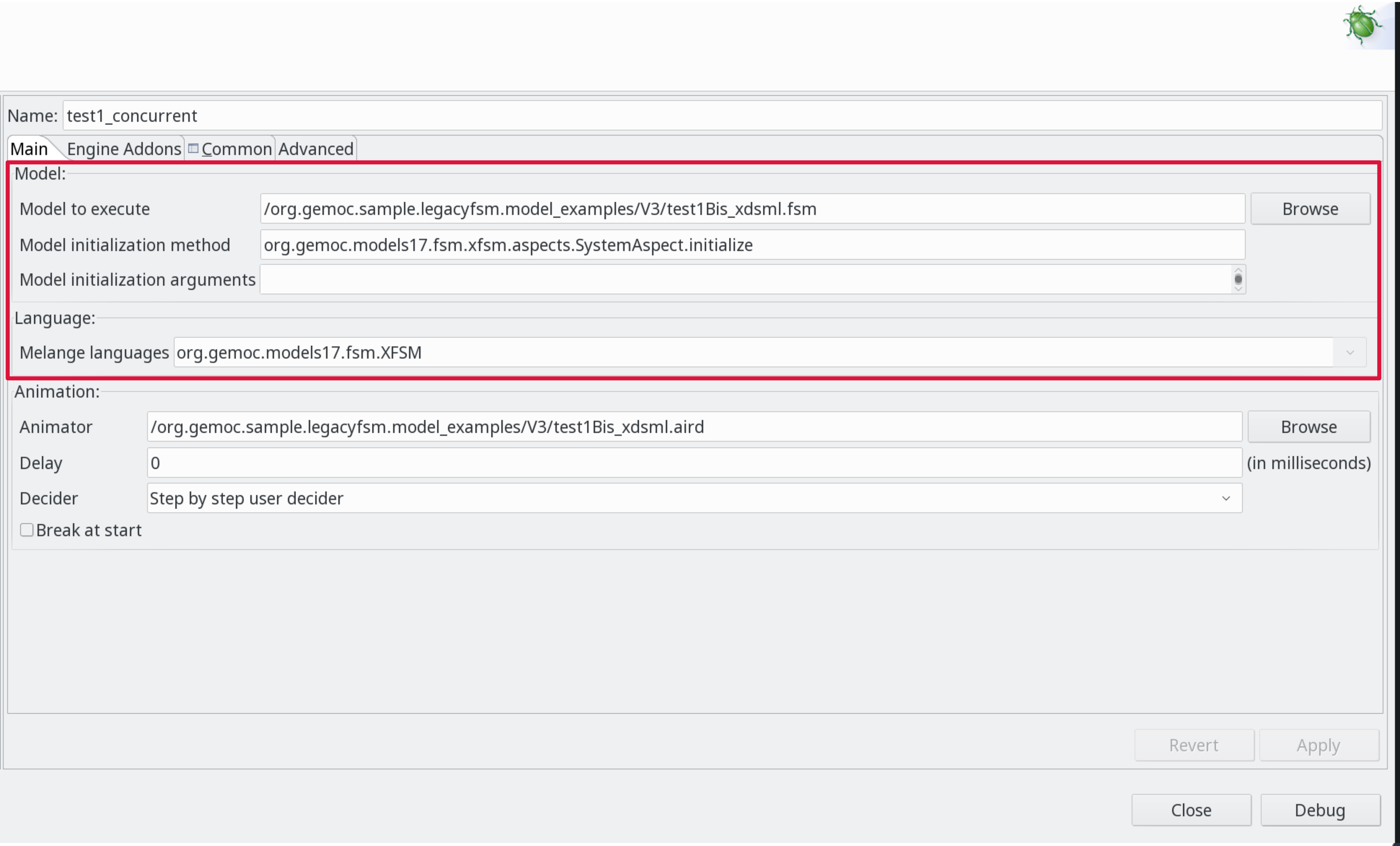


```
@Aspect(className=C1)
class C1Aspect{
    int aRTD = 0
    def void anExecutionFunction(){
        _self.aRTD = _self.aRTD + 1
        _self.name = _self.name+'x'
    }
}
```

```
@Aspect(className = C1.class)          // the annotation is preserved in the generated code
public class C1Aspect {
  public static void anExecutionFunction(final C1 _self) {
    final C1AspectC1AspectProperties _self_ = C1AspectC1AspectContext.getSelf(_self);
    C1Aspect._privk3_anExecutionFunction(_self_, (C1)_self);
  }
```

```
protected static void _privk3_anExecutionFunction(
  final C1AspectC1AspectProperties _self_,
  final C1 _self)       // the compiler chose between _self or _self_
  {
    _self_.aRTD = _self_.aRTD + 1;
    _self.setName(_self.getName()+'x');
  }
```

Use of K3 generated code



Name: test1_concurrent

Model:

Model to execute: /org.gemoc.sample.legacyfsm.model_examples/V3/test1Bis_xdsml.fsm

Model initialization method: org.gemoc.models17.fsm.xfsm.aspects.SystemAspect.initialize

Model initialization arguments:

Language:

Melange languages: org.gemoc.models17.fsm.XFSM

Animation:

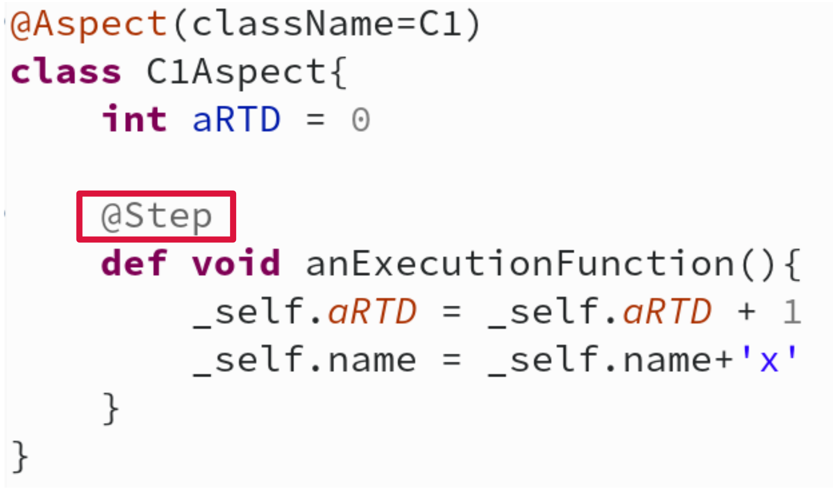
Animator: /org.gemoc.sample.legacyfsm.model_examples/V3/test1Bis_xdsml.aird

Delay: 0 (in milliseconds)

Decider: Step by step user decider

□ Break at start

Usage of K3 generated code

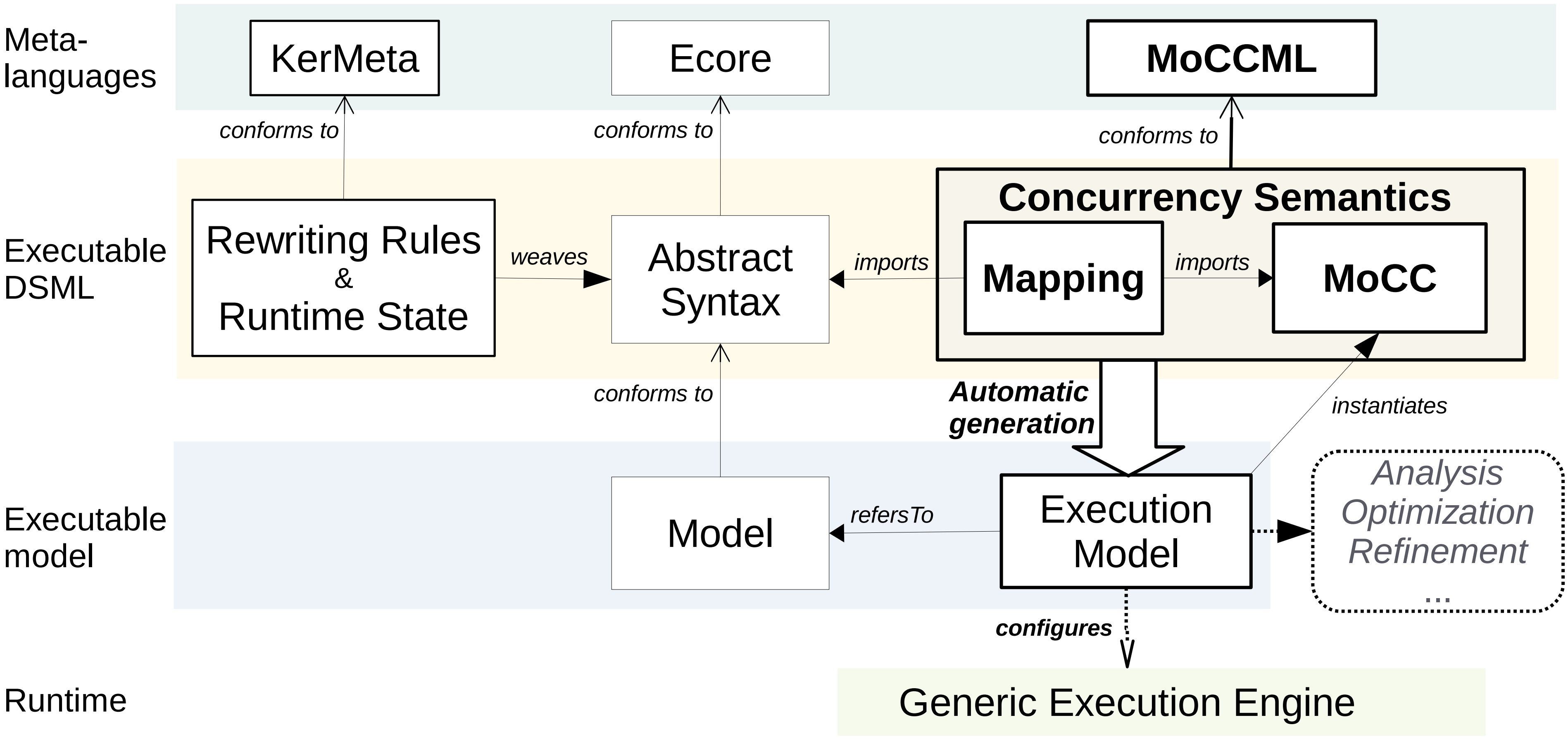


```
@Aspect(className=C1)
class C1Aspect{
    int aRTD = 0

    @Step
    def void anExecutionFunction(){
        _self.aRTD = _self.aRTD + 1
        _self.name = _self.name+'x'
    }
}
```

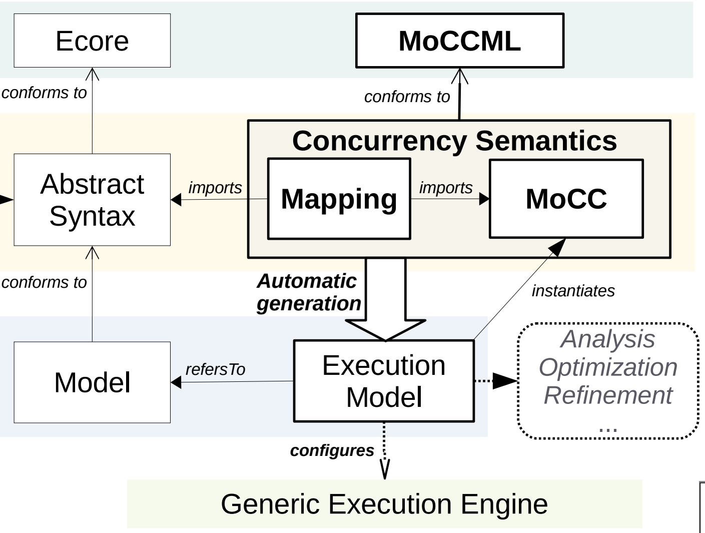
The annotation is defined in the metalanguage and used during the model execution

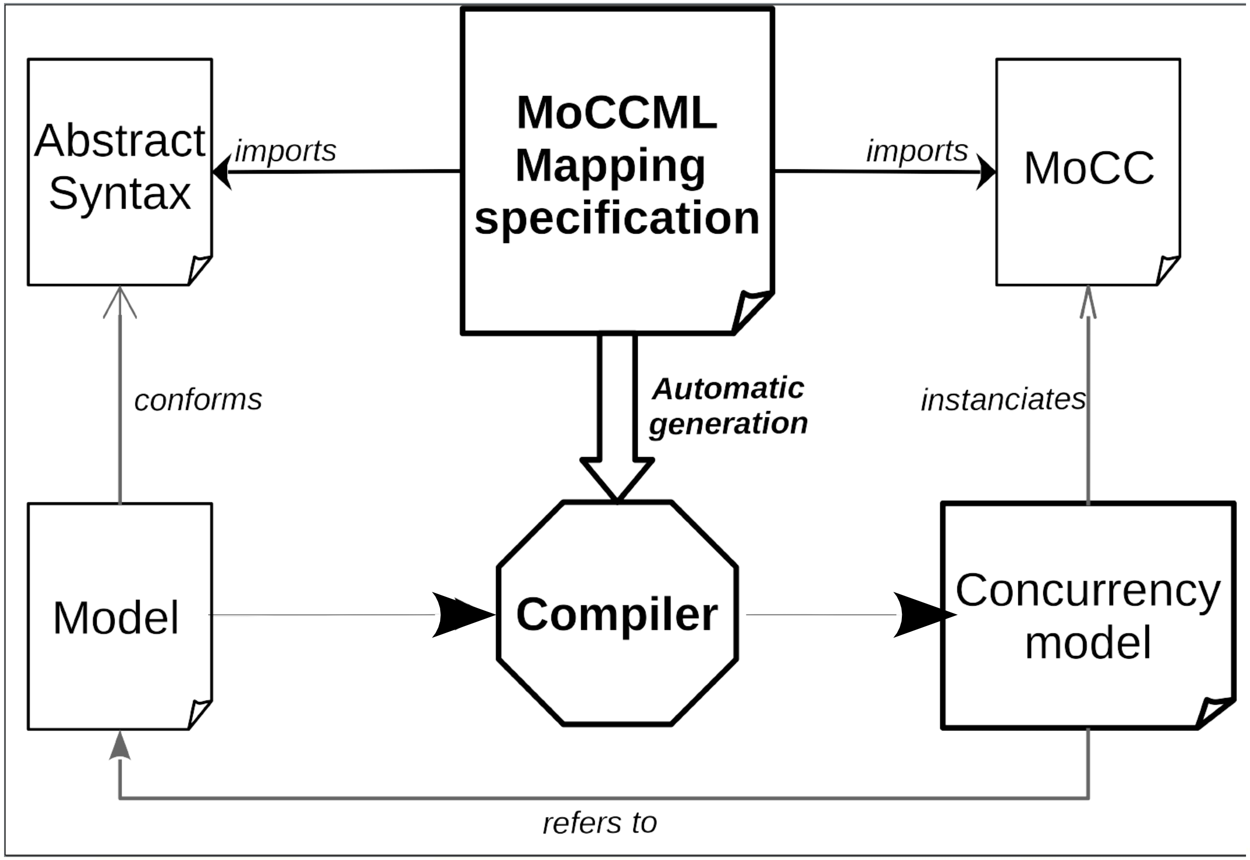
```
final boolean isStepMethod = initializeMethod
.isAnnotationPresent(fr.inria.diverse.k3.al.annotationprocessor.Step.class);
if (isStepMethod) {
    StepCommand command = new StepCommand() {
        @Override
        public void execute() {
            callInitializeModel();
        }
    };
    IStepManager stepManager = PlainK3ExecutionEngine.this;
    stepManager.executeStep(entryPointMethodParameters.get(0),
                            command,
                            entryPointClass.getName(),
                            initializeMethod.getName());
} else {
    callInitializeModel();
}
```

The GEMOC approach

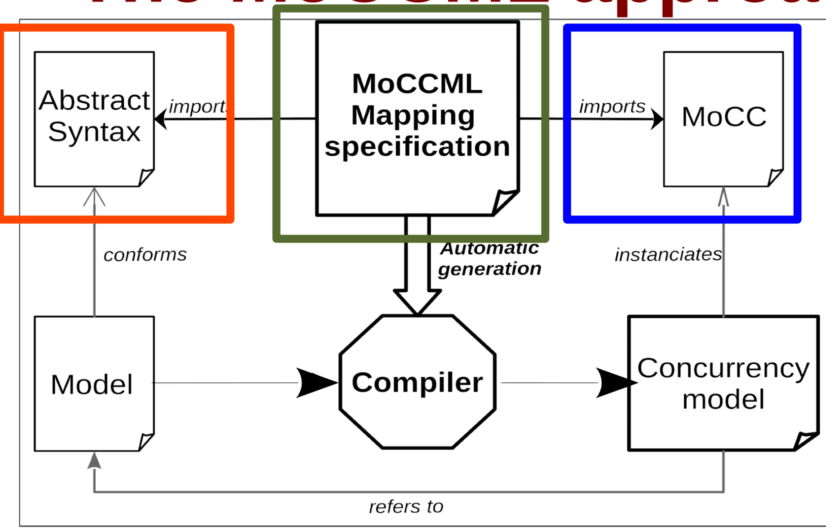


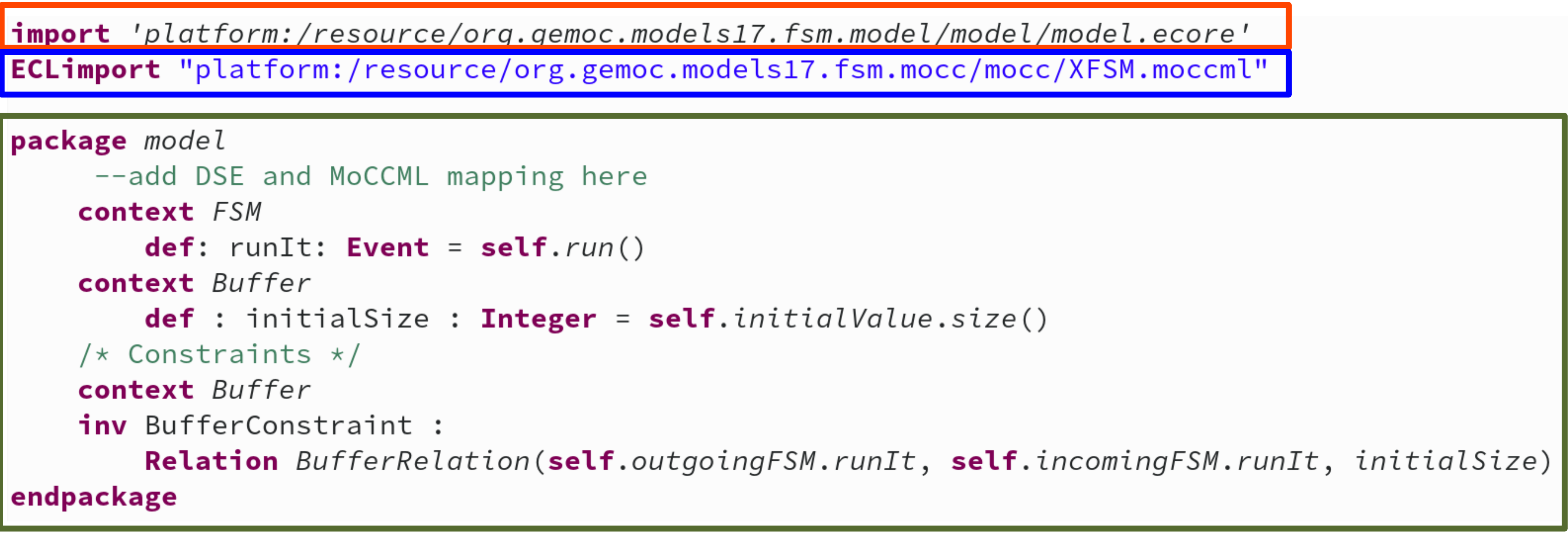
The MoCCML approach





The MoCCML approach

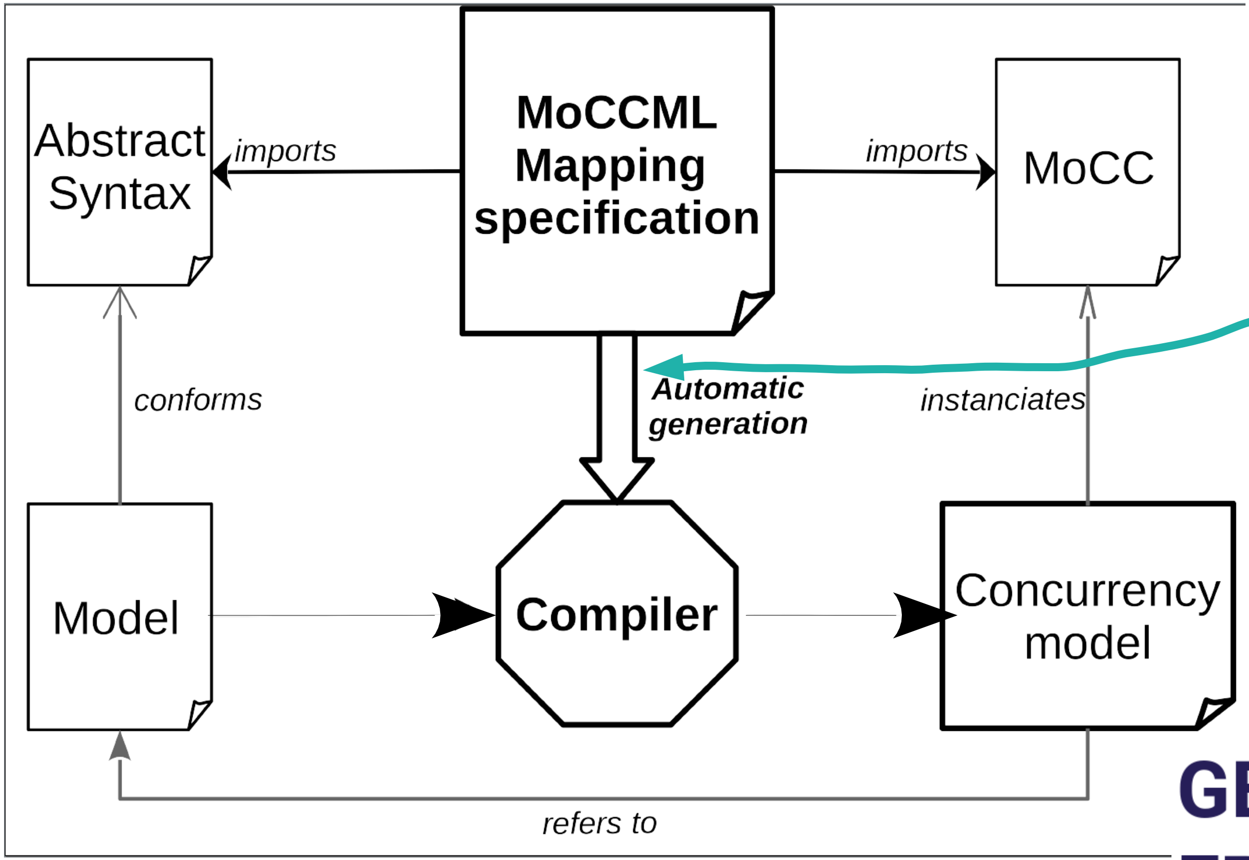




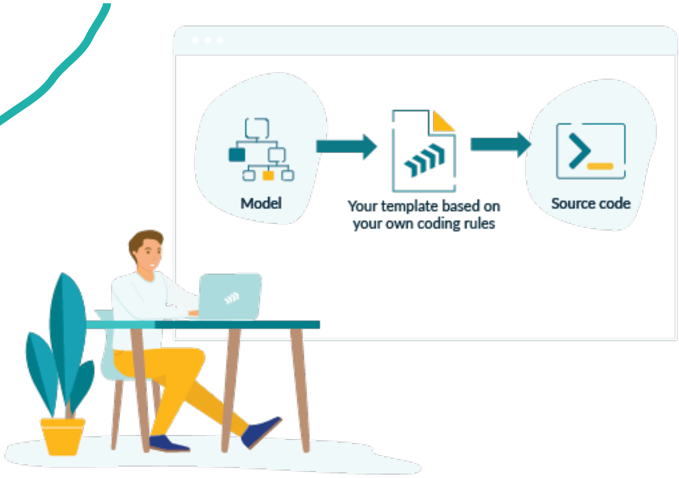
```
import 'platform:/resource/org.gemoc.models17.fsm.model/model/model.ecore'
ECLimport "platform:/resource/org.gemoc.models17.fsm.mocc/mocc/XFSM.moccml"
```

```
package model
     --add DSE and MoCCML mapping here
    context FSM
        def: runIt: Event = self.run()
    context Buffer
        def : initialSize : Integer = self.initialValue.size()
    /* Constraints */
    context Buffer
    inv BufferConstraint :
        Relation BufferRelation(self.outgoingFSM.runIt, self.incomingFSM.runIt, initialSize)
endpackage
```

The MoCCML approach



written in acceleo

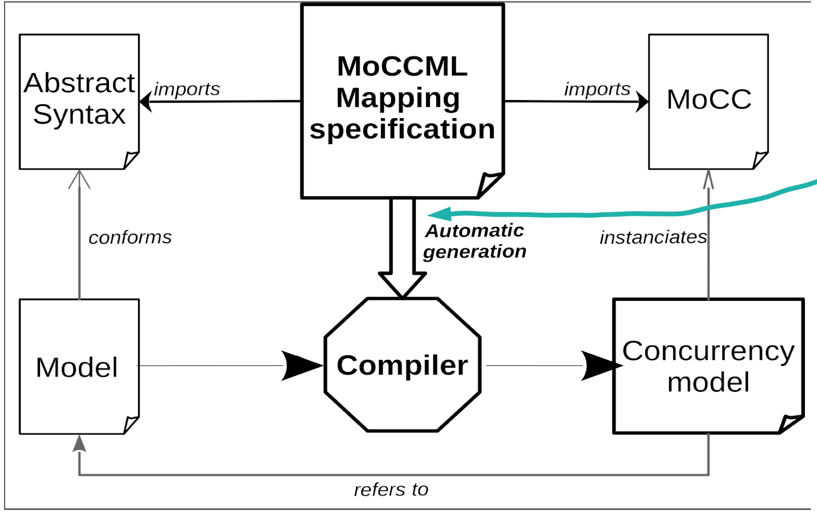


From https://www.eclipse.org/acceleo/#

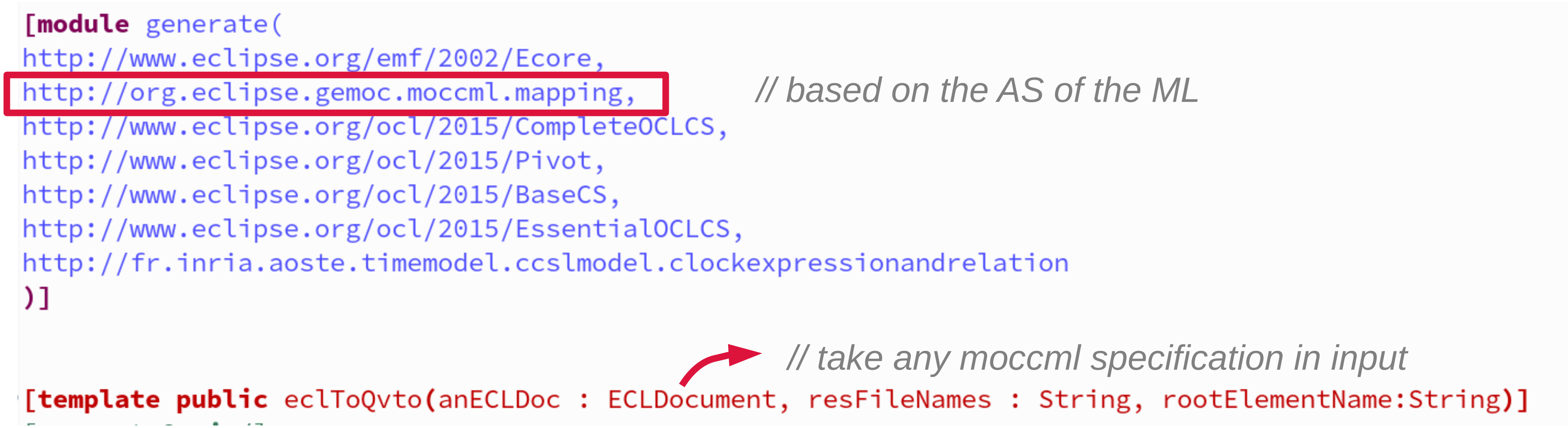
GENERATE ANYTHING FROM ANY EMF MODEL

Acceleo is the result of several man-years of R&D started in the French company Obeo.
Junction between the OMG MTL standard, its team's experience with industrial code generation and the latest research advances into the M2T field, it offers outstanding advantages : High ability to customize, Interoperability, Easy kick off, and much more!

The MoCCML approach



written in acceleo

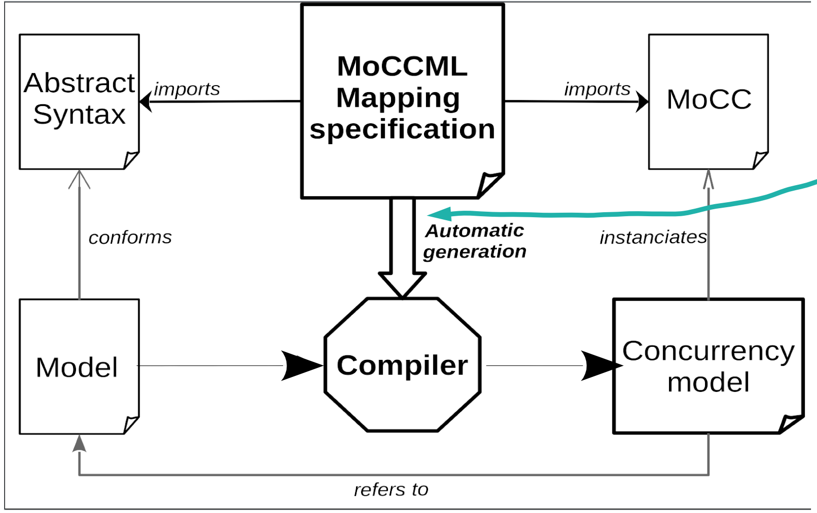


```
[module generate(
http://www.eclipse.org/emf/2002/Ecore,
http://org.eclipse.gemoc.moccml.mapping,
http://www.eclipse.org/ocl/2015/CompleteOCLCS,
http://www.eclipse.org/ocl/2015/Pivot,
http://www.eclipse.org/ocl/2015/BaseCS,
http://www.eclipse.org/ocl/2015/EssentialOCLCS,
http://fr.inria.aoste.timemodel.ccslmodel.clockexpressionandrelation
)]


[template public eclToQvto(anECLDoc : ECLDocument, resFileNames : String, rootElementName:String)]
```

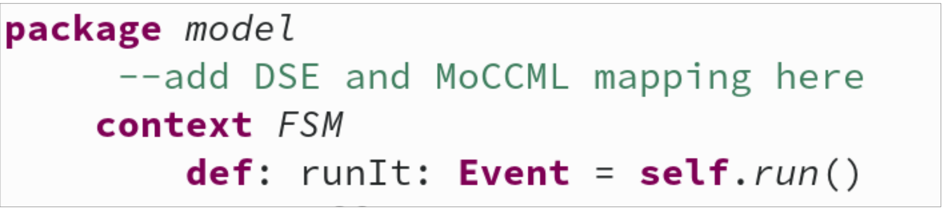
// based on the AS of the ML

// take any moccml specification in input

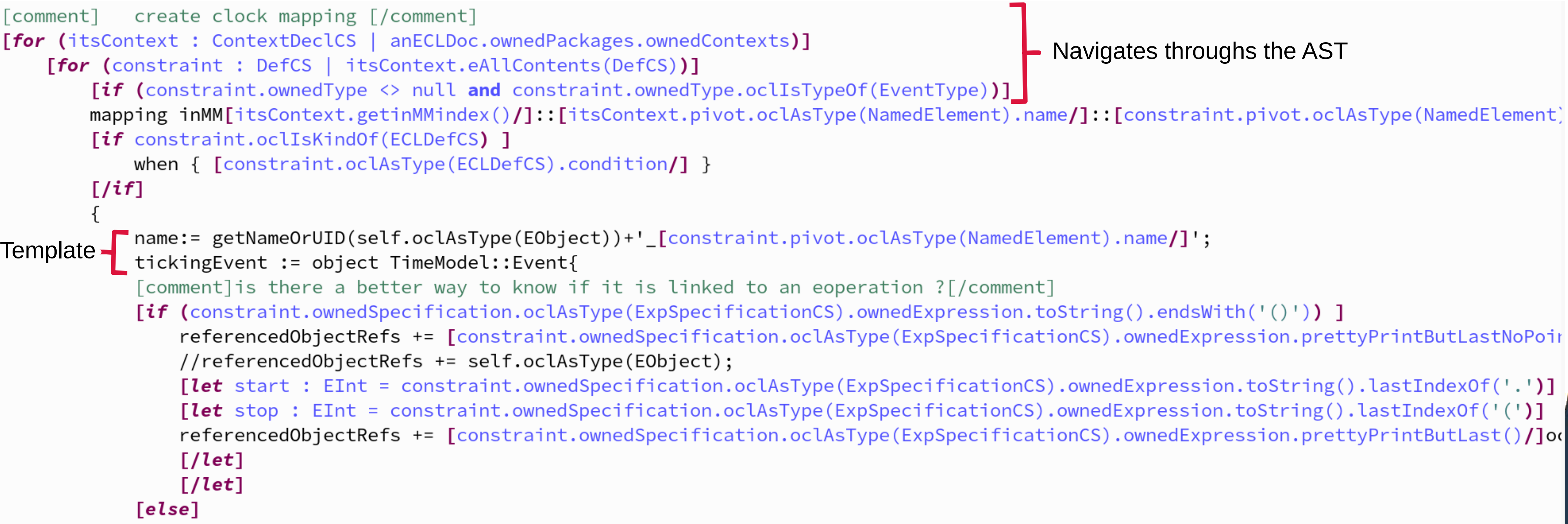
The MoCCML approach



written in acceleo



```
package model
   --add DSE and MoCCML mapping here
  context FSM
    def: runIt: Event = self.run()
```

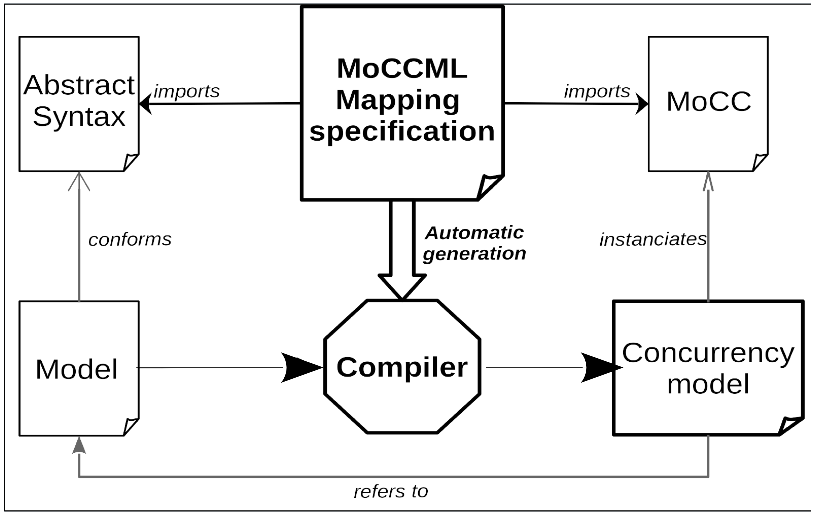


```
[comment]   create clock mapping [/comment]
[for (itsContext : ContextDeclCS | anECLDoc.ownedPackages.ownedContexts)]
  [for (constraint : DefCS | itsContext.eAllContents(DefCS))]
    [if (constraint.ownedType <> null and constraint.ownedType.oclIsTypeOf(EventType))]
    mapping inMM[itsContext.getinMMindex()/]::[itsContext.pivot.oclAsType(NamedElement).name/]::[constraint.pivot.oclAsType(NamedElement
    [if constraint.oclIsKindOf(ECLDefCS) ]
      when { [constraint.oclAsType(ECLDefCS).condition/] }
    [/if]
    {
      name:= getNameOrUID(self.oclAsType(EObject))+'_[constraint.pivot.oclAsType(NamedElement).name/]';
      tickingEvent := object TimeModel::Event{
      [comment]is there a better way to know if it is linked to an eoperation ?[/comment]
      [if (constraint.ownedSpecification.oclAsType(ExpSpecificationCS).ownedExpression.toString().endsWith('()')) ]
        referencedObjectRefs += [constraint.ownedSpecification.oclAsType(ExpSpecificationCS).ownedExpression.prettyPrintButLastNoPoi
        //referencedObjectRefs += self.oclAsType(EObject);
        [let start : EInt = constraint.ownedSpecification.oclAsType(ExpSpecificationCS).ownedExpression.toString().lastIndexOf('.')]
        [let stop : EInt = constraint.ownedSpecification.oclAsType(ExpSpecificationCS).ownedExpression.toString().lastIndexOf('(')]
        referencedObjectRefs += [constraint.ownedSpecification.oclAsType(ExpSpecificationCS).ownedExpression.prettyPrintButLast()/]o
        [/let]
        [/let]
      [else]
```

Navigates throughs the AST

Template

The MoCCML approach

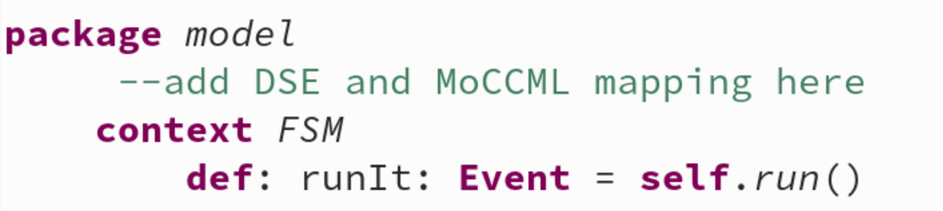


Result of acceleo is a transformation in QVTo :

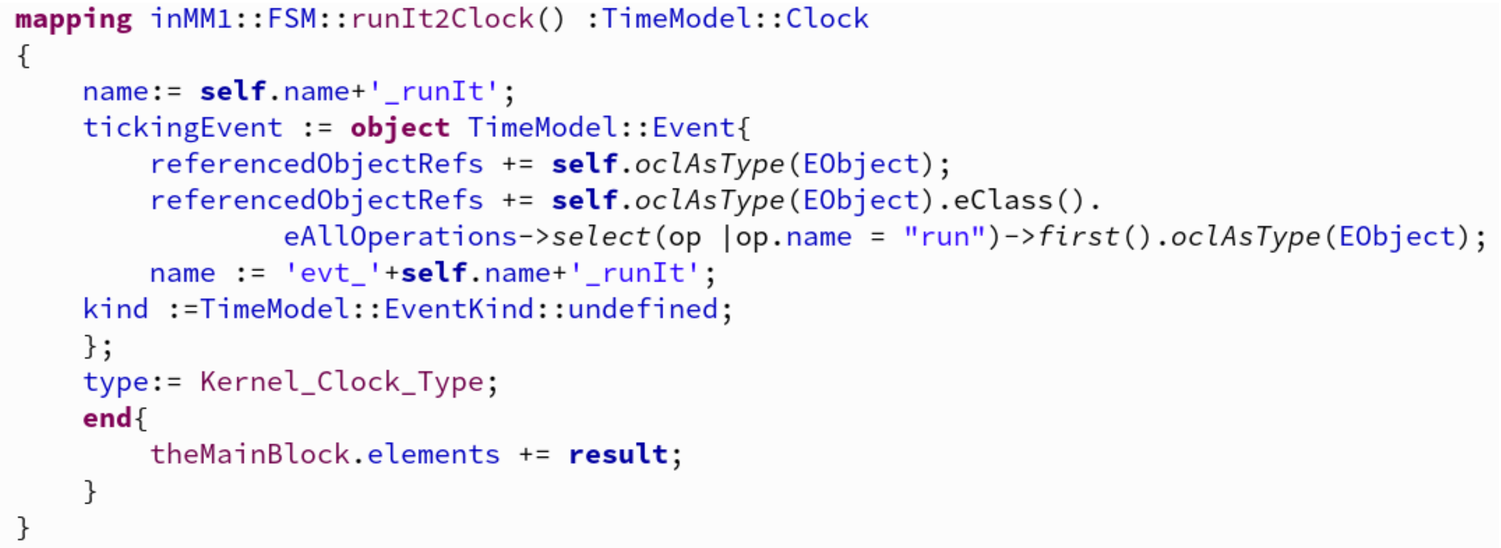
- *The Eclipse QVT Operational component is an implementation of the Operational Mappings Language defined by Meta Object Facility™ (MOF™) 2.0 Query/View/Transformation™ (QVT™).*

This is consequently a *HOT*

(High Order Transformation)



```
package model
    --add DSE and MoCCML mapping here
  context FSM
     def: runIt: Event = self.run()
```



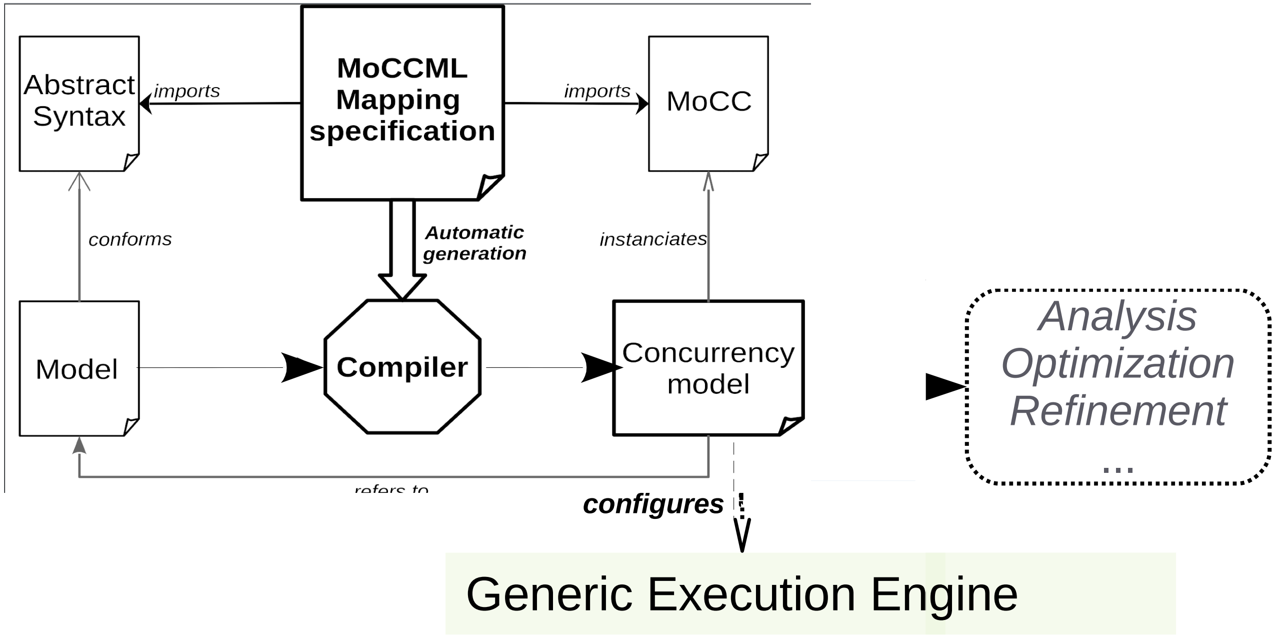
```
mapping inMM1::FSM::runIt2Clock() :TimeModel::Clock
{
    name:= self.name+'_runIt';
    tickingEvent := object TimeModel::Event{
        referencedObjectRefs += self.oclAsType(EObject);
        referencedObjectRefs += self.oclAsType(EObject).eClass().
                eAllOperations->select(op |op.name = "run")->first().oclAsType(EObject);
        name := 'evt_'+self.name+'_runIt';
    kind :=TimeModel::EventKind::undefined;
    };
    type:= Kernel_Clock_Type;
    end{
        theMainBlock.elements += result;
    }
}
```

The MoCCML approach

Result of acceleo is a transformation in QVTo :

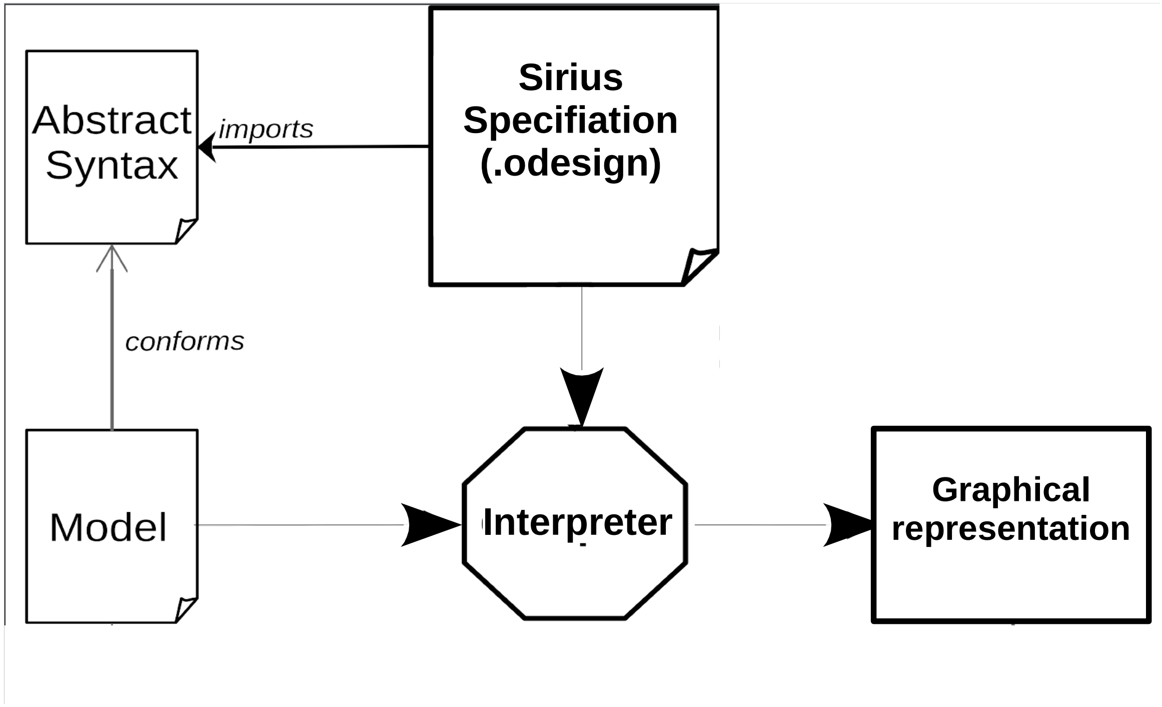
- *The Eclipse QVT Operational component is an implementation of the Operational Mappings Language defined by Meta Object Facility™ (MOF™) 2.0 Query/View/Transformation™ (QVT™).*

This is consequently a HOT, whose result parametrizes a generic interpreter or can be used for analysis



The Sirius approach

The interpreter takes two inputs, a model that conforms a L and the sirius specification written in the ML (which imports the L)

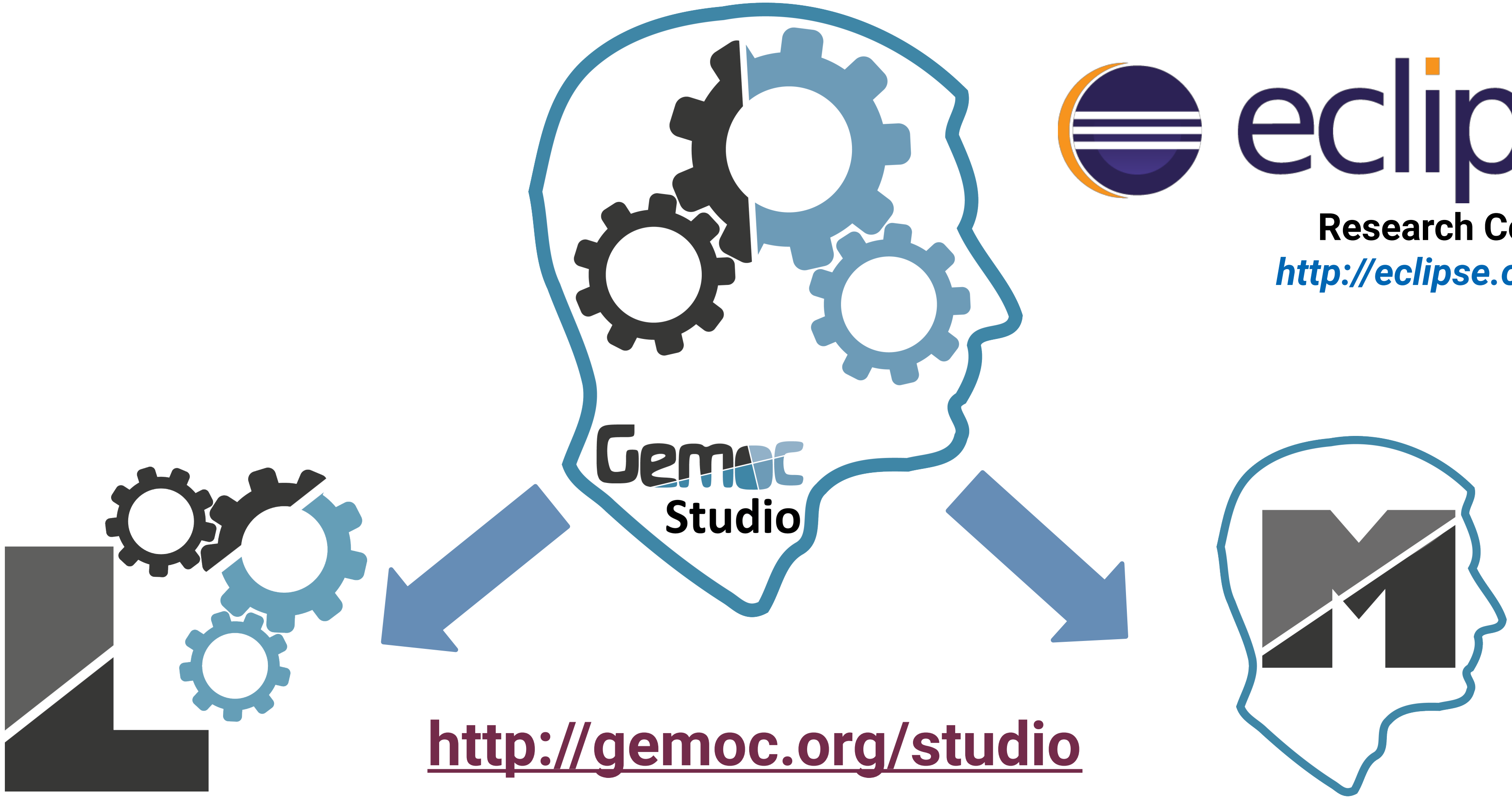


The GEMOC Studio

eclipse

Research Consortium

http://eclipse.org/gemoc



http://gemoc.org/studio

Language Workbench

Modeling Workbench

Design and compose your executable DSMLs

Edit and debug your heterogeneous models