Model Driven Engineering (MDE)

(a.k.a formal model-driven software development)

Juan de Lara, Elena Gómez, Esther Guerra
{Juan.deLara, MariaElena.Gomez, Esther.Guerra}@uam.es

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Masters: I2ICSI and formal methods (shared)

Objectives of the course

- Techniques for model driven engineering
  - Modelling and Meta-modelling
  - Design of domain-specific (visual, textual) languages
  - Model transformation
  - Code generation
  - Model-based analysis and verification

- Combine practice and theory

Course Organization

Syllabus

- ***Teaching load:***
  - 6 ECTS credits.
  - 14 weeks of classes (last week for project preparation).
  - Theory/practical sessions. Presentations.

1. Introduction to Model Driven Engineering
2. Software Modelling

   UML, OCL, validation (model finding)
3. Meta-modelling and Domain Specific Languages

   Meta-modelling, multi-level modelling, analysis

   External DSLs. Textual and graphical concrete syntaxes
4. Model Transformations

   In-place, model-to-model. Graph transformation
5. Code generation
6. Model analysis and verification

   Petri nets, others

Course Organization

Evaluation

- 5 exercises (individually **35%**)
- 1 paper presentation (individually **15**%)
- 1 practical project (groups of 2-3 people**, 50**%)
  - Define a language (meta-model)
  - Build a textual syntax
  - Build a transformation or a code generator
  - Make some analysis
  - Presentation: 19th January 2023
- All need to be passed separately (>=5)
- Material in: http://posgrado.uam.es
  - send me your (uam) e-mail address if you do not have access, or use: estudiante.metodosformalesii@estudiante.uam.es password Mformalesii%2018
- Talk to us if you cannot assist regularly to classes (>70%), as you'd enter in "non-continuous evaluation mode".

Model Driven Engineering

Model Driven Engineering

What is it?

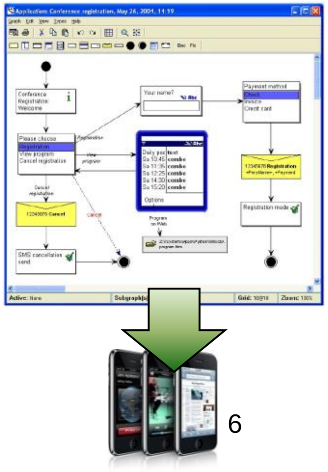
- Develop software using high-level models, instead of coding

- Implies designing languages (sometimes graphical) for specific tasks or domains

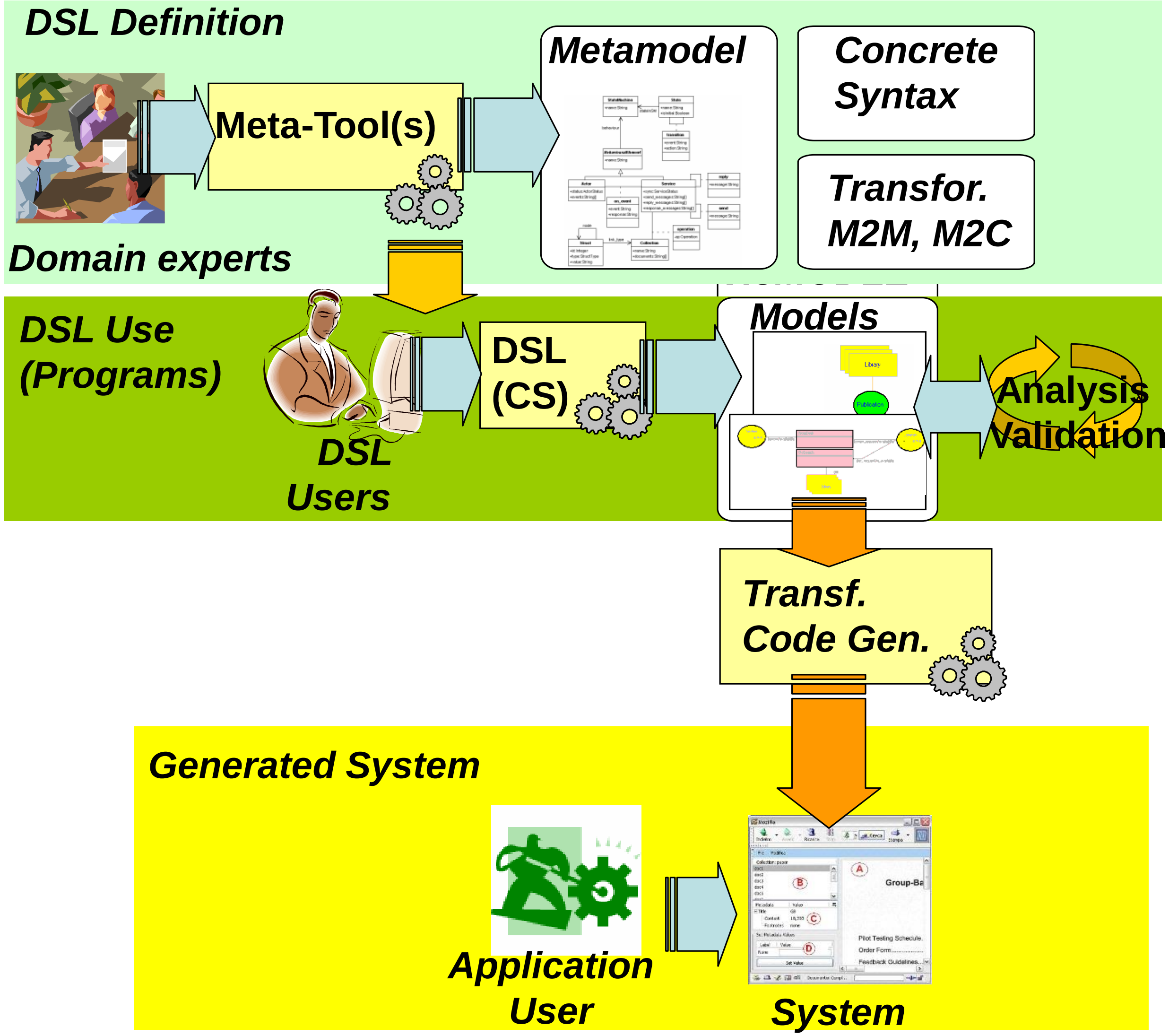
- Models are more comprehensible for developers (not necessarily computer scientists). Less accidental complexity

- Model transformation is an essential activity in MDE

- Combination with code generators



Basic Scheme



DSL Definition

Domain experts

Meta-Tool(s)

Metamodel

Concrete Syntax

Transfor. M2M, M2C

DSL Use (Programs)

DSL Users

DSL (CS)

Models

Analysis Validation

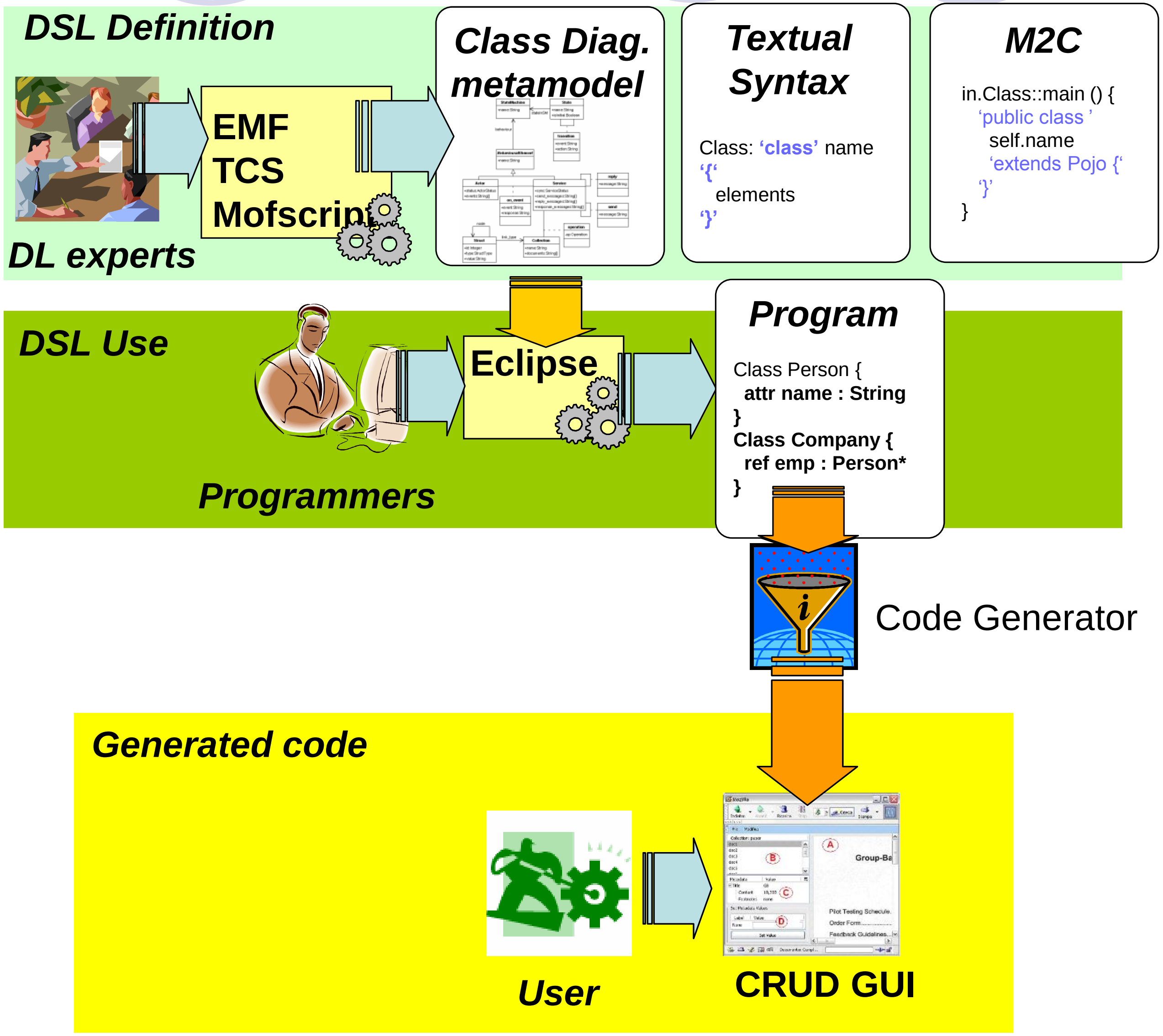
Transf. Code Gen.

Generated System

Application User

System

Example: CRUD Generation



Textual Syntax

```
Class: 'class' name
'{'
  elements
'}'
```

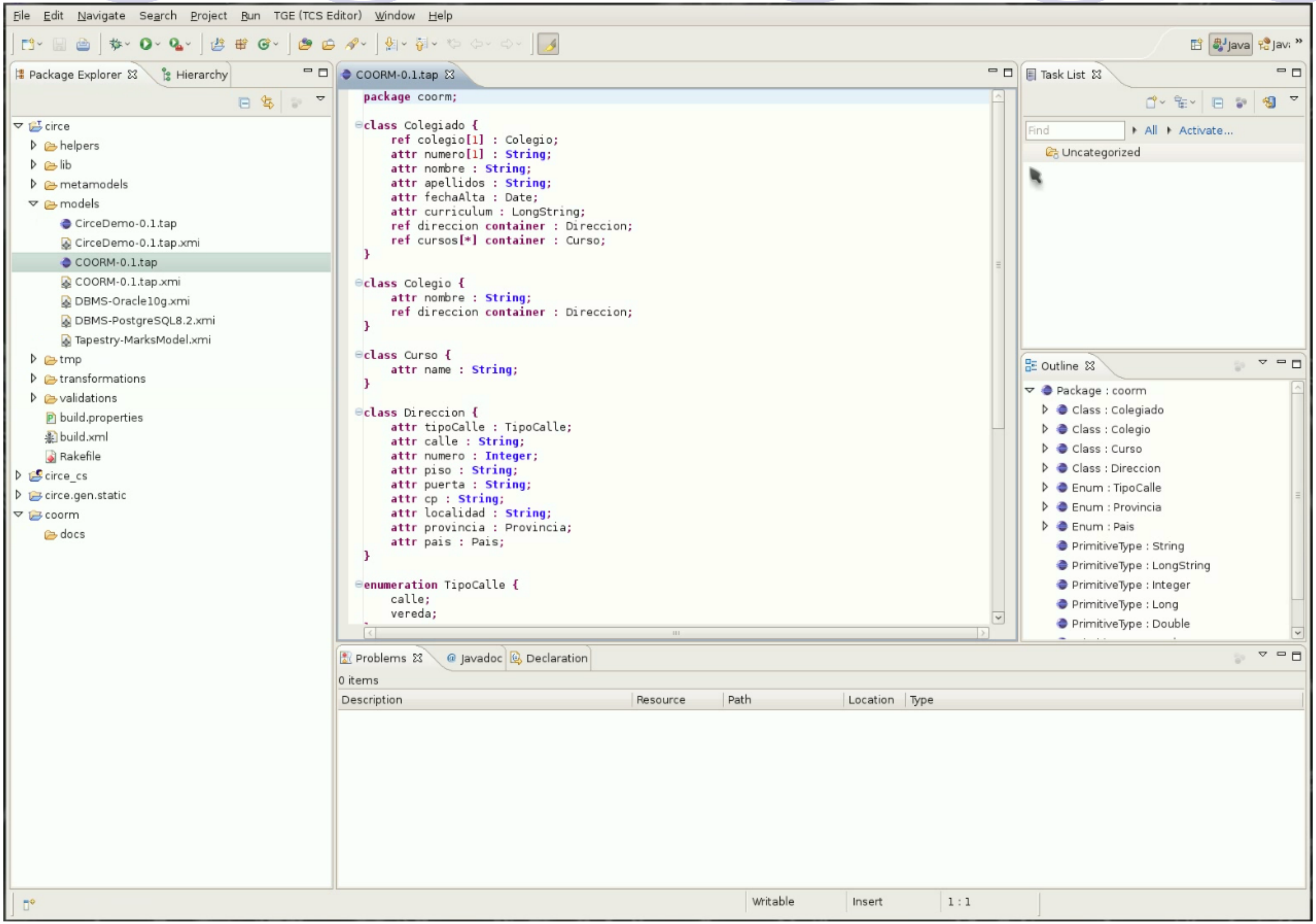
M2C

```
in.Class::main () {
  'public class '
   self.name
   'extends Pojo {'
  '}'
}
```

Program

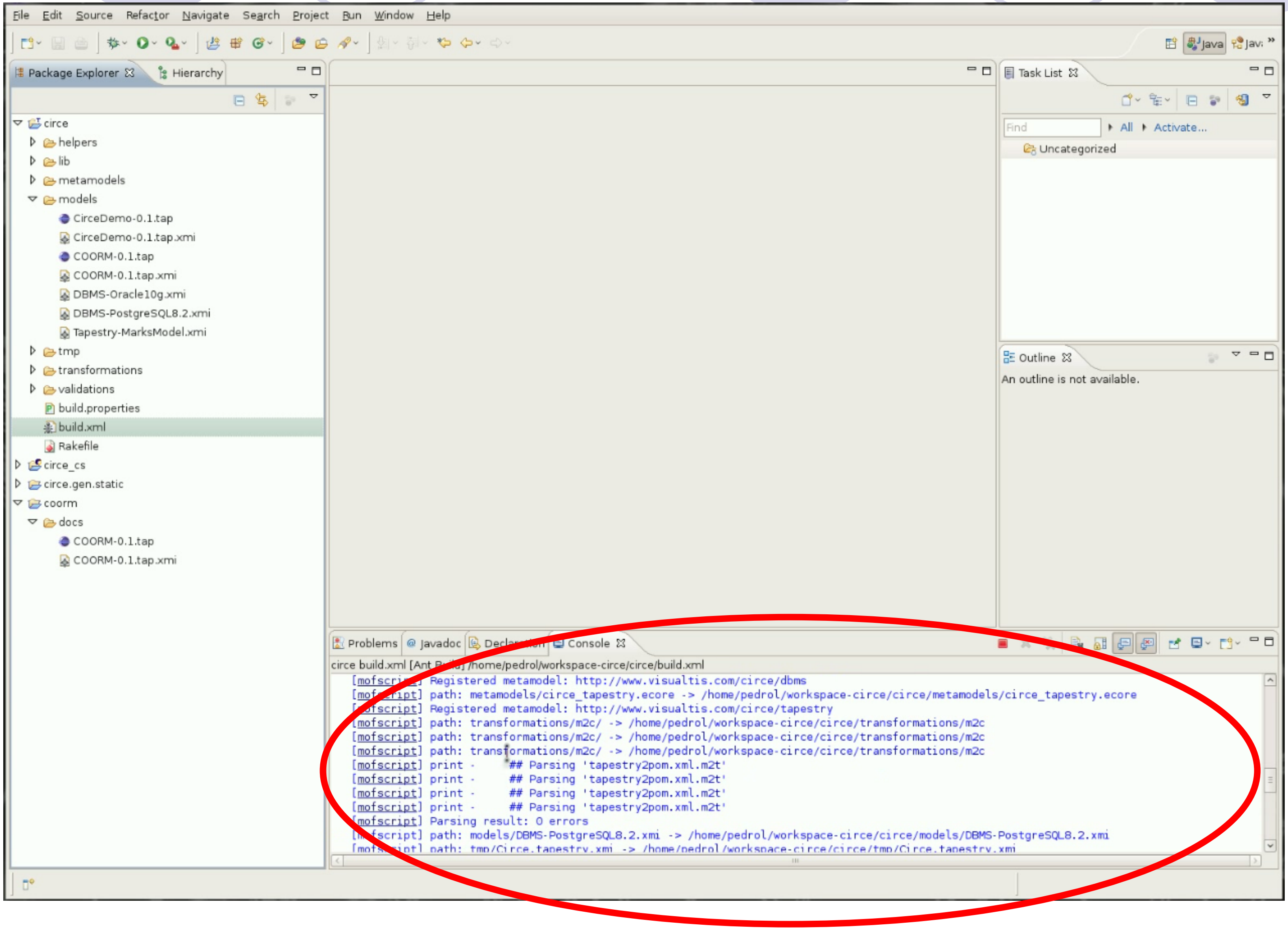
```
Class Person {
  attr name : String
}
Class Company {
  ref emp : Person*
}
```

Example: CRUD Generation



```
package coorm;

class Colegiado {
    ref colegio[1] : Colegio;
    attr numero[1] : String;
    attr nombre : String;
    attr apellidos : String;
    attr fechaAlta : Date;
    attr curriculum : LongString;
    ref direccion container : Direccion;
    ref cursos[*] container : Curso;
}

class Colegio {
    attr nombre : String;
    ref direccion container : Direccion;
}

class Curso {
    attr name : String;
}

class Direccion {
    attr tipoCalle : TipoCalle;
    attr calle : String;
    attr numero : Integer;
    attr piso : String;
    attr puerta : String;
    attr cp : String;
    attr localidad : String;
    attr provincia : Provincia;
    attr pais : Pais;
}

enumeration TipoCalle {
    calle;
    vereda;
```

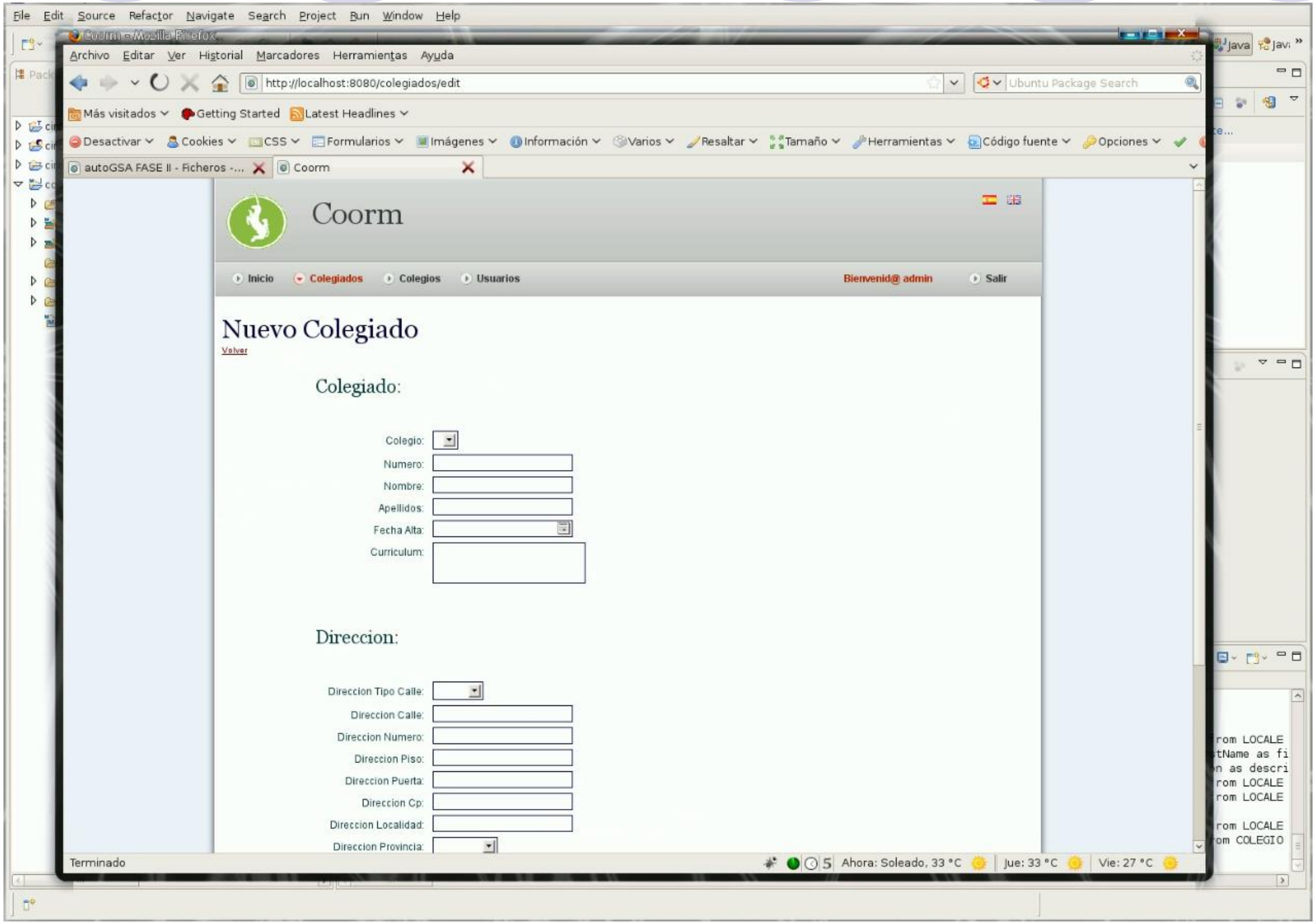
Example: CRUD Generation



circe build.xml [Ant Build] /home/pedrol/workspace-circe/circe/build.xml

```
[mofscript] Registered metamodel: http://www.visualtis.com/circe/dbms
[mofscript] path: metamodels/circe_tapestry.ecore -> /home/pedrol/workspace-circe/circe/metamodels/circe_tapestry.ecore
[mofscript] Registered metamodel: http://www.visualtis.com/circe/tapestry
[mofscript] path: transformations/m2c/ -> /home/pedrol/workspace-circe/circe/transformations/m2c
[mofscript] path: transformations/m2c/ -> /home/pedrol/workspace-circe/circe/transformations/m2c
[mofscript] path: transformations/m2c/ -> /home/pedrol/workspace-circe/circe/transformations/m2c
[mofscript] print -      ## Parsing 'tapestry2pom.xml.m2t'
[mofscript] print -      ## Parsing 'tapestry2pom.xml.m2t'
[mofscript] print -      ## Parsing 'tapestry2pom.xml.m2t'
[mofscript] print -      ## Parsing 'tapestry2pom.xml.m2t'
[mofscript] Parsing result: 0 errors
[mofscript] path: models/DBMS-PostgreSQL8.2.xmi -> /home/pedrol/workspace-circe/circe/models/DBMS-PostgreSQL8.2.xmi
[mofscript] path: tmp/Circe.tapestry.xmi -> /home/pedrol/workspace-circe/circe/tmp/Circe.tapestry.xmi
```

Example: CRUD Generation



File Edit Source Refactor Navigate Search Project Run Window Help
Coorm - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://localhost:8080/colegiados/edit
Ubuntu Package Search
Más visitados Getting Started Latest Headlines
Desactivar Cookies CSS Formularios Imágenes Información Varios Resaltar Tamaño Herramientas Código fuente Opciones
autoGSA FASE II - Ficheros -...
Coorm
Inicio Colegiados Colegios Usuarios
Bienvenid@ admin
Salir
Terminado
Ahora: Soleado, 33 °C Jue: 33 °C Vie: 27 °C

Coorm

Nuevo Colegiado

Volver

Colegiado:

Colegio:

Numero:

Nombre:

Apellidos:

Fecha Alta:

Curriculum:

Direccion:

Direccion Tipo Calle:

Direccion Calle:

Direccion Numero:

Direccion Piso:

Direccion Puerta:

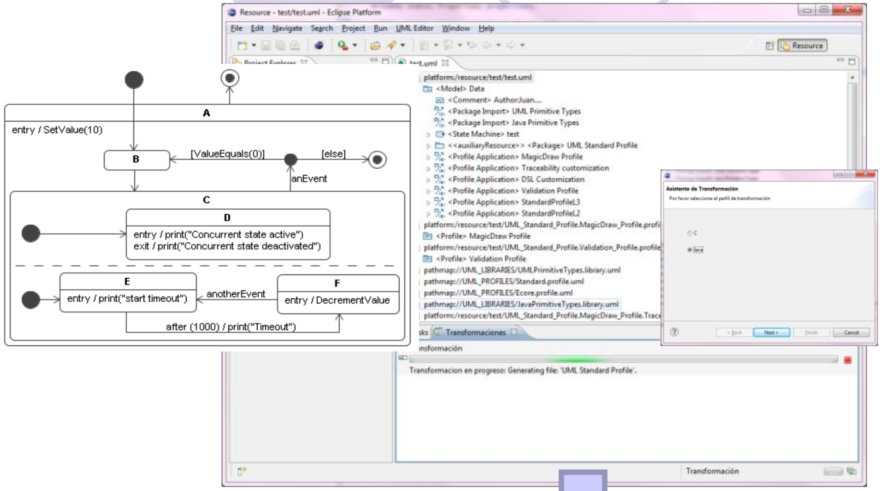
Direccion Cp:

Direccion Localidad:

Direccion Provincia:

```
rom LOCALE
tName as fi
n as descri
rom LOCALE
rom LOCALE
rom LOCALE
om COLEGIO
```

Some industrial projects



A
entry / SetValue(10)
B
[ValueEquals(0)]
[else]
anEvent
C
D
entry / print("Concurrent state active")
exit / print("Concurrent state deactivated")
E
entry / print("start timeout")
anotherEvent
F
entry / DecrementValue
after (1000) / print("Timeout")
Resource - test/test.uml - Eclipse Platform
File Edit Navigate Search Project Run UML Editor Window Help
Resource
Project Explorer
test.uml
Transformaciones
< Back
Next >
Finish
Cancel
Transformación

platform:/resource/test/test.uml
<Model> Data
<Comment> Author:Juan....
<Package Import> UML Primitive Types
<Package Import> Java Primitive Types
<State Machine> test
<<auxiliaryResource>> <Package> UML Standard Profile
<Profile Application> MagicDraw Profile
<Profile Application> Traceability customization
<Profile Application> DSL Customization
<Profile Application> Validation Profile
<Profile Application> StandardProfileL3
<Profile Application> StandardProfileL2
platform:/resource/test/UML_Standard_Profile.MagicDraw_Profile.profi
<Profile> MagicDraw Profile
platform:/resource/test/UML_Standard_Profile.Validation_Profile.profile
<Profile> Validation Profile
pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml
pathmap://UML_PROFILES/Standard.profile.uml
pathmap://UML_PROFILES/Ecore.profile.uml
pathmap://UML_LIBRARIES/JavaPrimitiveTypes.library.uml
platform:/resource/test/UML_Standard_Profile.MagicDraw_Profile.Trace

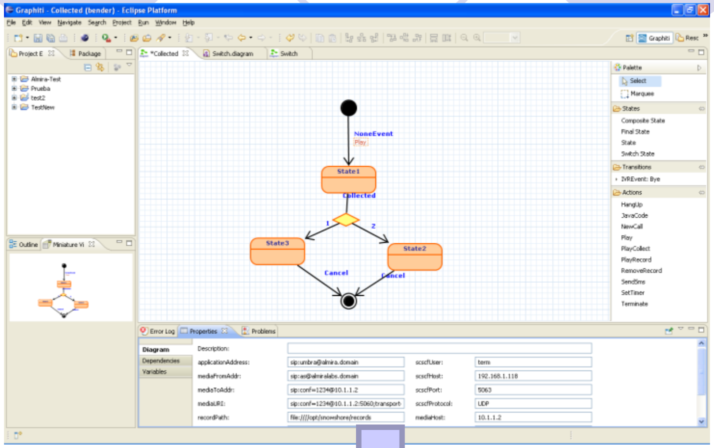
Asistente de Transformación
Por favor seleccione el perfil de transformación
○ C
● Java

Transformacion en progreso: Generating file: 'UML Standard Profile'.

Code generator

- Code generation from State-Machines for Railway Signaling Systems.

CAF Signalling



Description:
applicationAddress: sip:umbra@almira.domain
mediaFromAddr: sip:as@almiralabs.domain
mediaToAddr: sip:conf=1234@10.1.1.2
mediaURI: sip:conf=1234@10.1.1.2:5060;transport
recordPath: file:////opt/snowshore/records
scscfUser: tem
scscfHost: 192.168.1.118
scscfPort: 5063
scscfProtocol: UDP
mediaHost: 10.1.1.2

Code generator

- Modelling, validation and automatic code generation of telephony services

Almira Labs
Quality Engineering

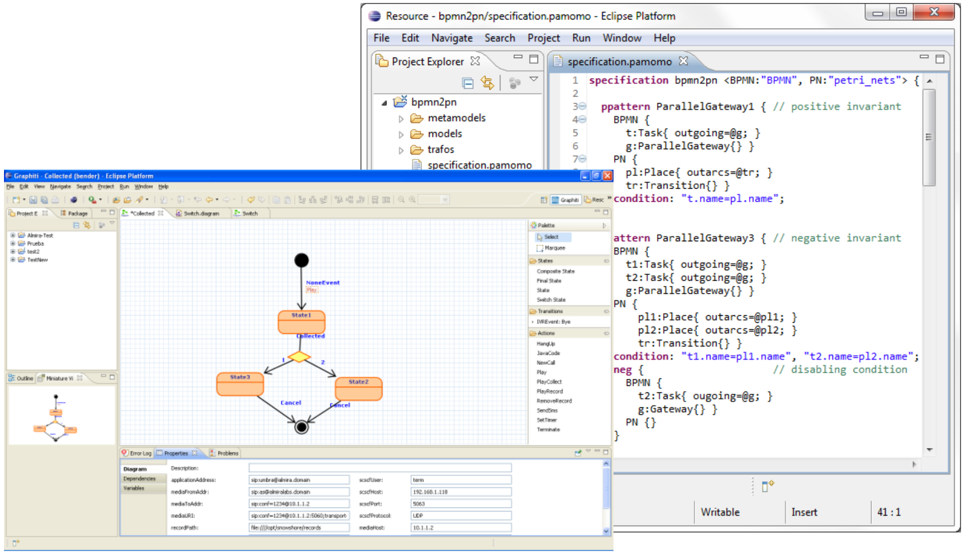
Meta-Modelling

- We will study the basic principles and concepts of meta-modelling.
- MOF is the standard language to build meta-models.
- *eCore* and *EMF* are (de-facto) standard implementations over Eclipse.
- http://www.eclipse.org/modeling/emf/

Domain-Specific Languages

- How to design visual or textual syntaxes for modelling languages?

- Pre-EMF era:
  - AToM^3
  - Diagen/Diameta
  - GenGed
  - …



```
specification bpmn2pn <BPMN:"BPMN", PN:"petri_nets"> {

  ppattern ParallelGateway1 { // positive invariant
    BPMN {
      t:Task{ outgoing=@g; }
      g:ParallelGateway{} }
    PN {
      pl:Place{ outarcs=@tr; }
      tr:Transition{} }
condition: "t.name=pl.name";


attern ParallelGateway3 { // negative invariant
BPMN {
  t1:Task{ outgoing=@g; }
  t2:Task{ outgoing=@g; }
  g:ParallelGateway{} }
PN {
    pl1:Place{ outarcs=@pl1; }
    pl2:Place{ outarcs=@pl2; }
    tr:Transition{} }
condition: "t1.name=pl1.name", "t2.name=pl2.name";
neg {                          // disabling condition
  BPMN {
    t2:Task{ ougoing=@g; }
    g:Gateway{} }
  PN {}
}
```

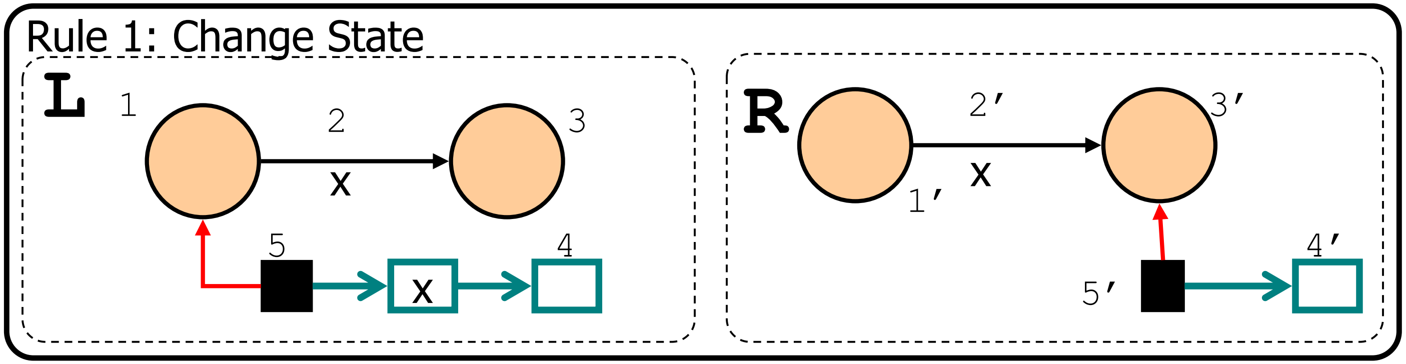
- EMF-based:
  - xText (for textual languages)
  - Sirius, EUGENIA, GMF, Graphiti (for visual languages)

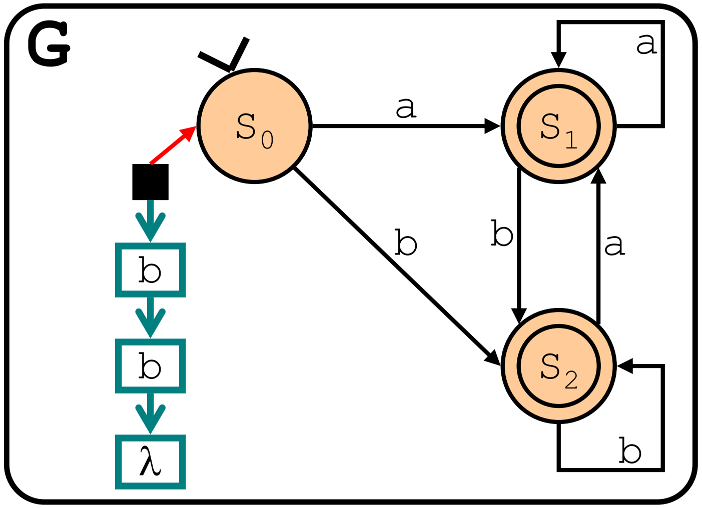
Model Transformations

- High-level languages to manipulate models:
  - Simulation or animation
  - Optimization (refactoring, etc.)
  - Transformation into another language (e.g. a semantic domain for analysis)

Model Transformation

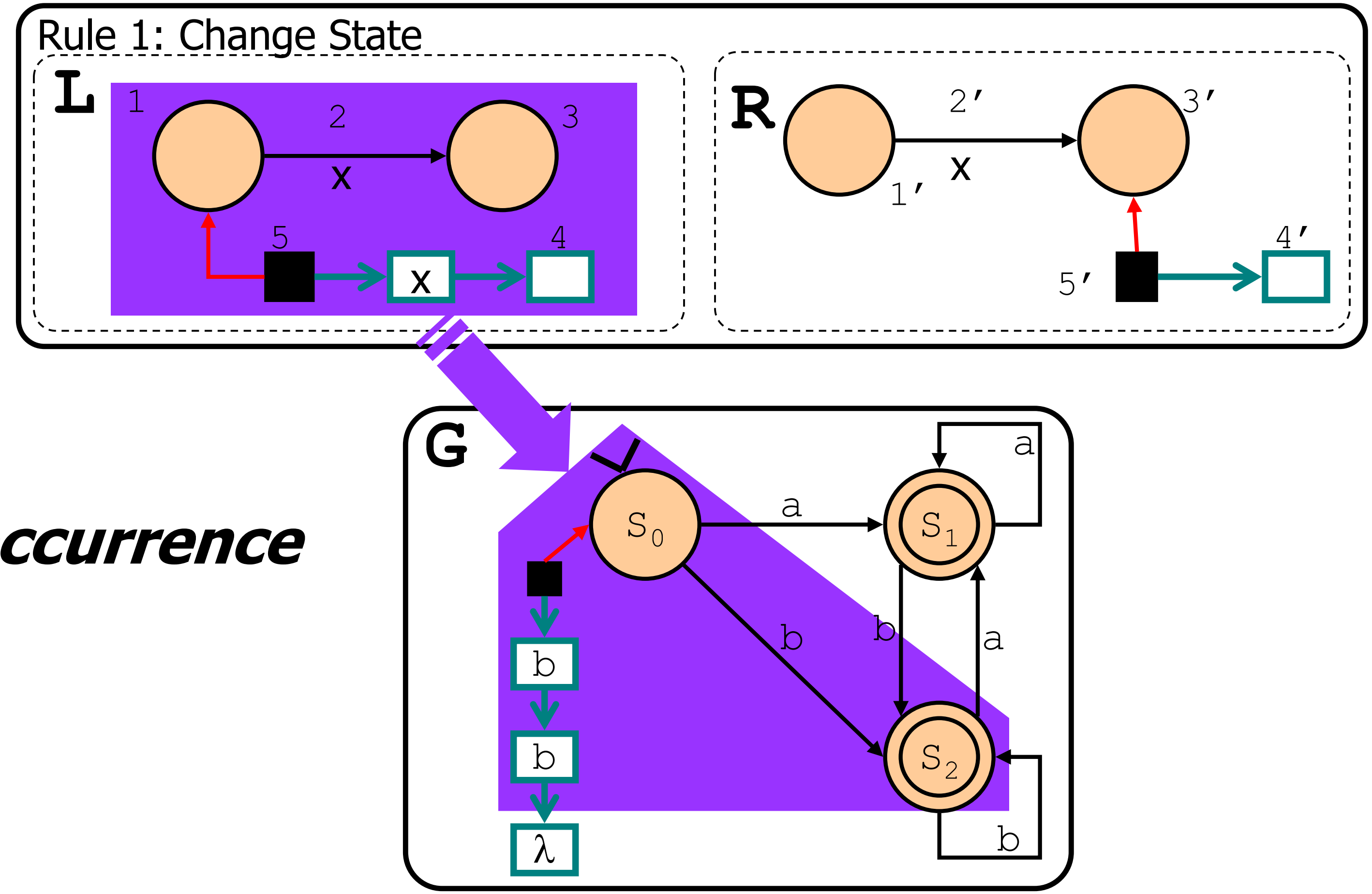
Graph Grammars





Model Transformation

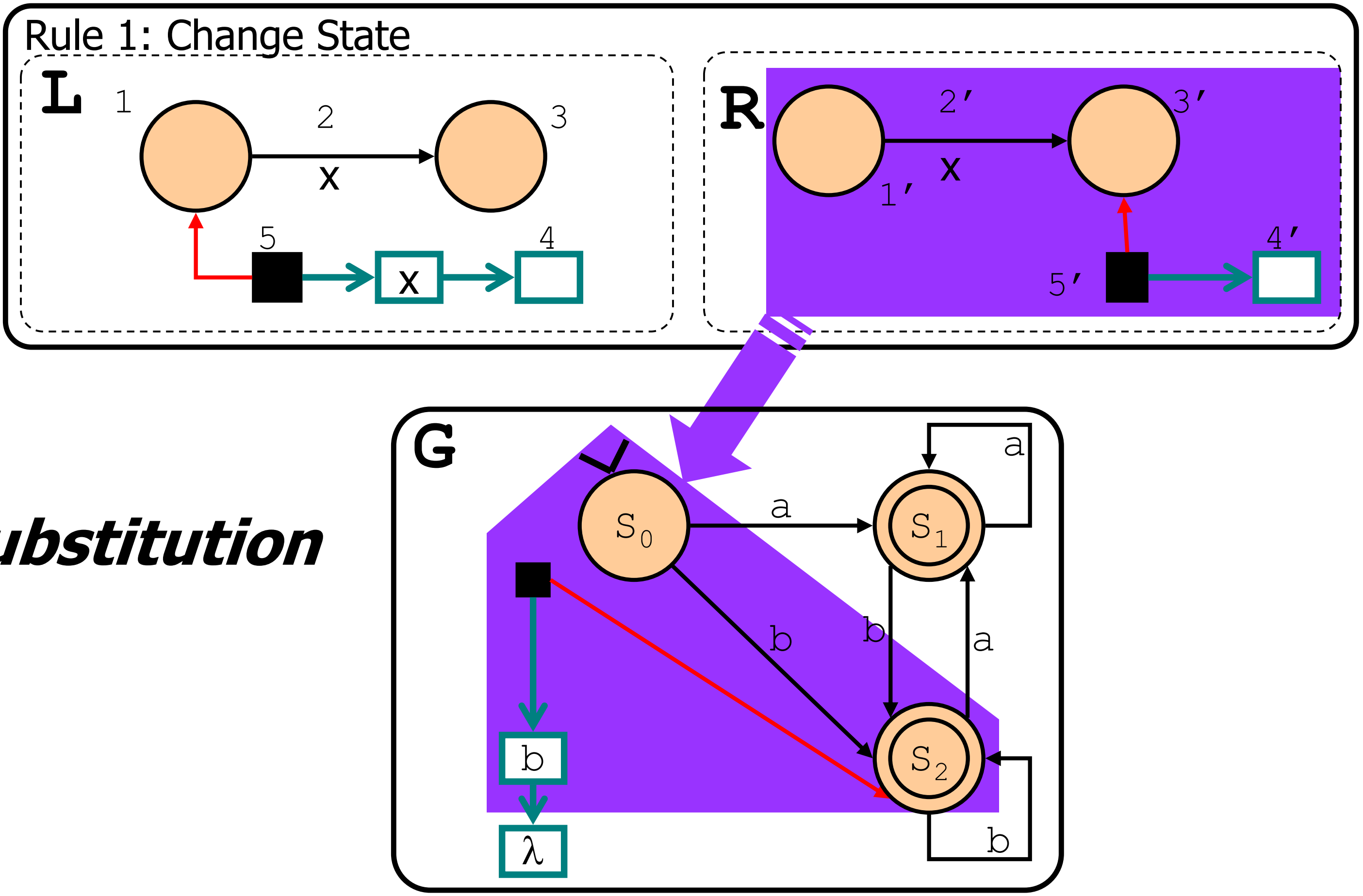
Graph Grammars



occurrence

Model Transformation

Graph Grammars



substitution

Model Transformation

Graph Grammars

- Useful to:
  - Specify simulators.
  - Specify transformations
    - Between formalisms (TGGs)
    - Optimizations
  - Specify the dialog with the generated environment
  - ...
- Formal Technique:
  - Based on category theory
  - Analysis:
    - Termination (partially)
    - Confluence
    - Dependencies/Conflicts/Rule concurrency

Model Transformation: ATL

- We will also study other transformation languages, like ATL

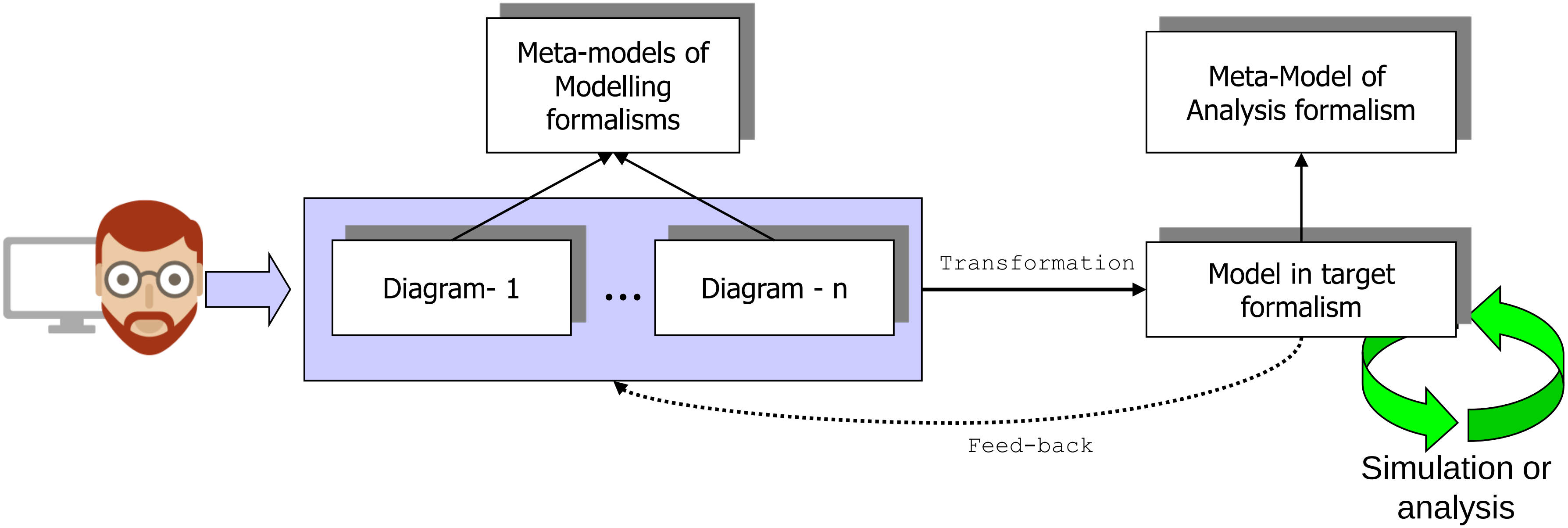
- For transformations between two different languages

```
rule Operator2Place {
  from o : FAC!Operator ( o.state <> #off)
  to   p : CPN!Place (
             name <- 'Operator '+o.name,
             author <- thisModule.author,
             tokens <- Set{thisModule.TokenCreation(o)}
    )
}
```

Model analysis

- How do we know if the models we have built are ***correct***? (not only from a structural, static viewpoint).
- How do we know if they ***satisfy*** certain behavioural or efficiency ***properties***?
- Methods enabling model analysis ***before*** code generation.
- Models are higher-level, more “intensional” than code, hence easier to analyse.

Model analysis


Meta-models of Modelling formalisms
Meta-Model of Analysis formalism
Diagram- 1
...
Diagram - n
Transformation
Model in target formalism
Feed-back
Simulation or analysis

- We will introduce a few formalisms with rich analysis and verification support, like Petri nets, process algebra, etc.

Summary

Learning goals

- Modelling
  - UML, OCL
- Meta-Modelling
  - EMF
  - MetaDepth (Multi-level)
- Define a visual or textual concrete syntax
  - Sirius, xText
- Define manipulations over these models, including code generation
  - Graph Grammars
  - ATL
  - Acceleo
- Analyse behavioural and structural aspects of the models