Software Engineering
Tools for Code Generation

F. Mallet

Frederic.Mallet@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR

Introduction

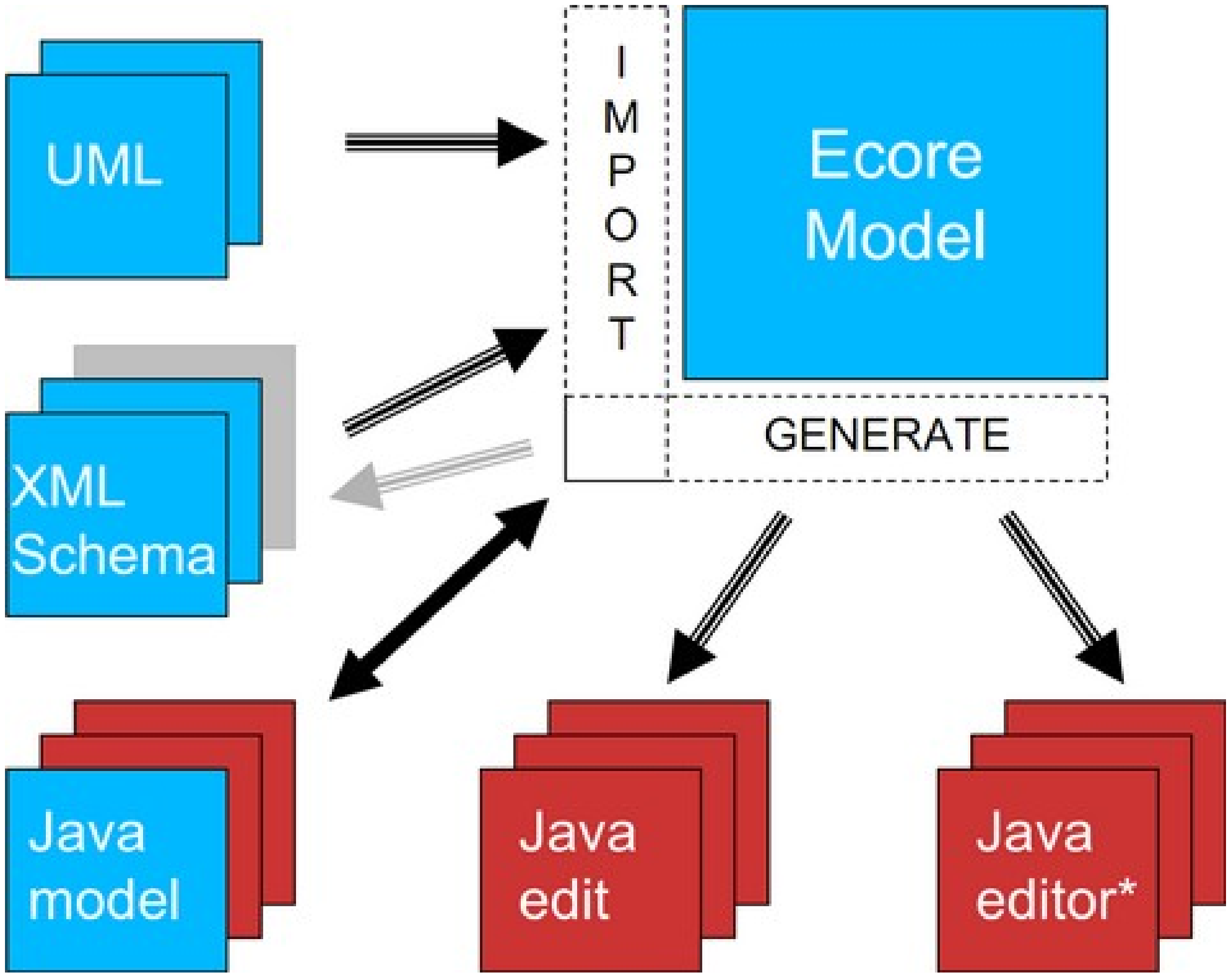
- Outline
  - MDE & Meta-Model
  - Meta-Modeling & EMF
- Application
  - Textual Concrete Syntax
  - Graphical Concrete Syntax

ECLIPSE META-MODELING FRAMEWORK

Eclipse Modeling Framework

- Modeling Framework with code generation
  - To build tools based on data models
  - The model is captured as a **XMI** (XML Metadata Interchange) file
- Import existing code to build the model
  - Java code with annotations
  - XML documents (**XSD** – XML Schema Definition)
  - UML tools (e.g., Rational Rose)
- Code generation from the model
  - Set of Java classes and interfaces
  - An Edit/Editor environment (editing tree)
- Many (and many more coming) extensions
  - Generate a graphical editor (Graphical Modeling Framework, Sirius)
  - Generate a parser, syntax highlighting (XText, TCS, Sintaks)

EMF import/export

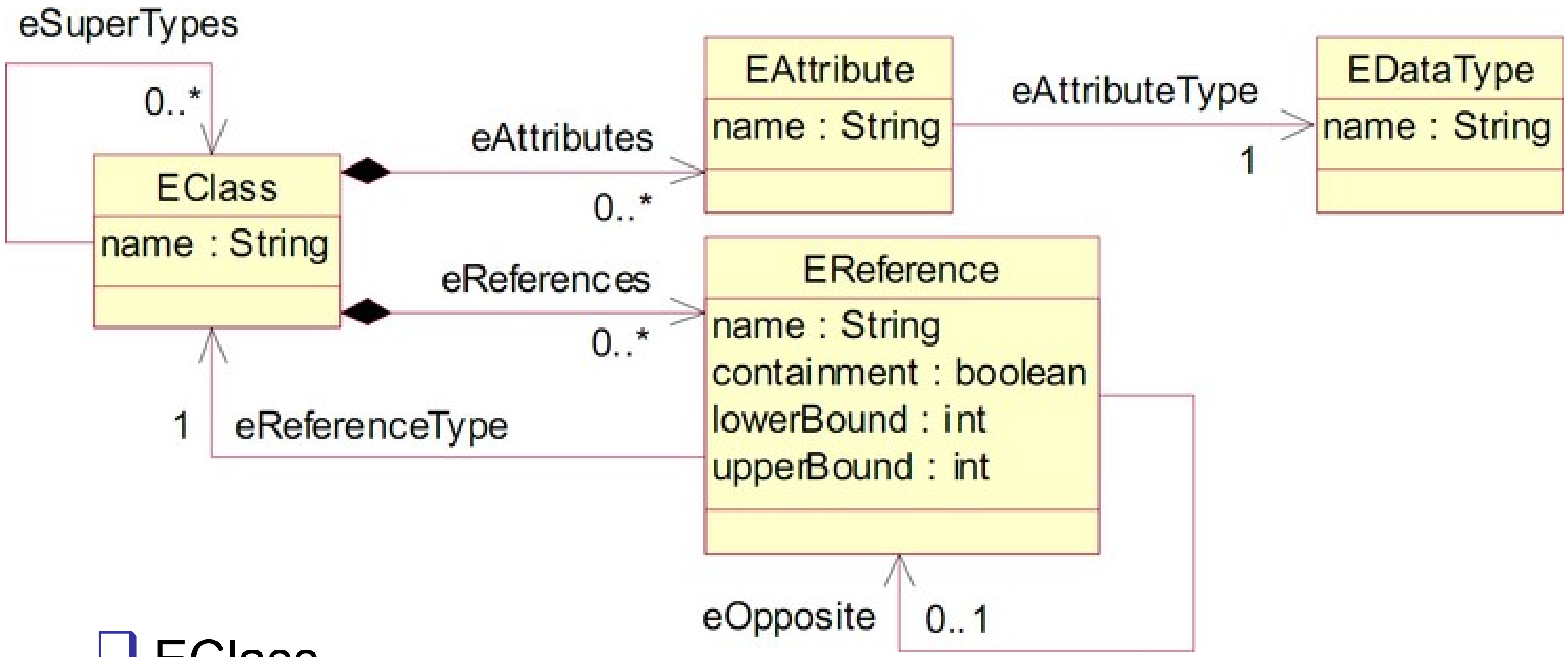


IBM, Ed. Merks & D. Steinberg, EclipseCon 2005

EMF and UML ?

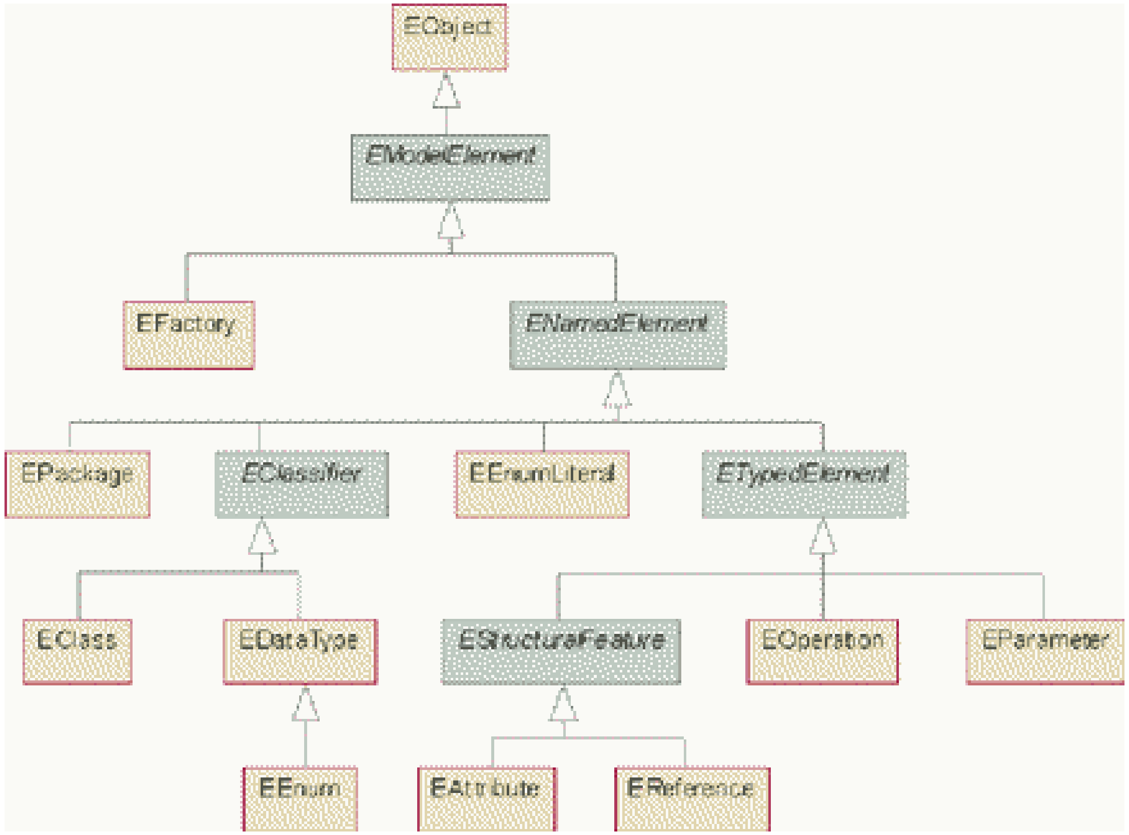
- **MOF**: Meta-metamodel (M3) by OMG
  - **M**eta-**O**bject **F**acilities
  - EMOF (Essential MOF) is a subset
- EMF
  - Was initially just an implementation of the MOF
  - Has evolved to become **ECORE**
- Two very different business models
  - OMG # Eclipse Foundation

Ecore: EMF meta-metamodel (M3)



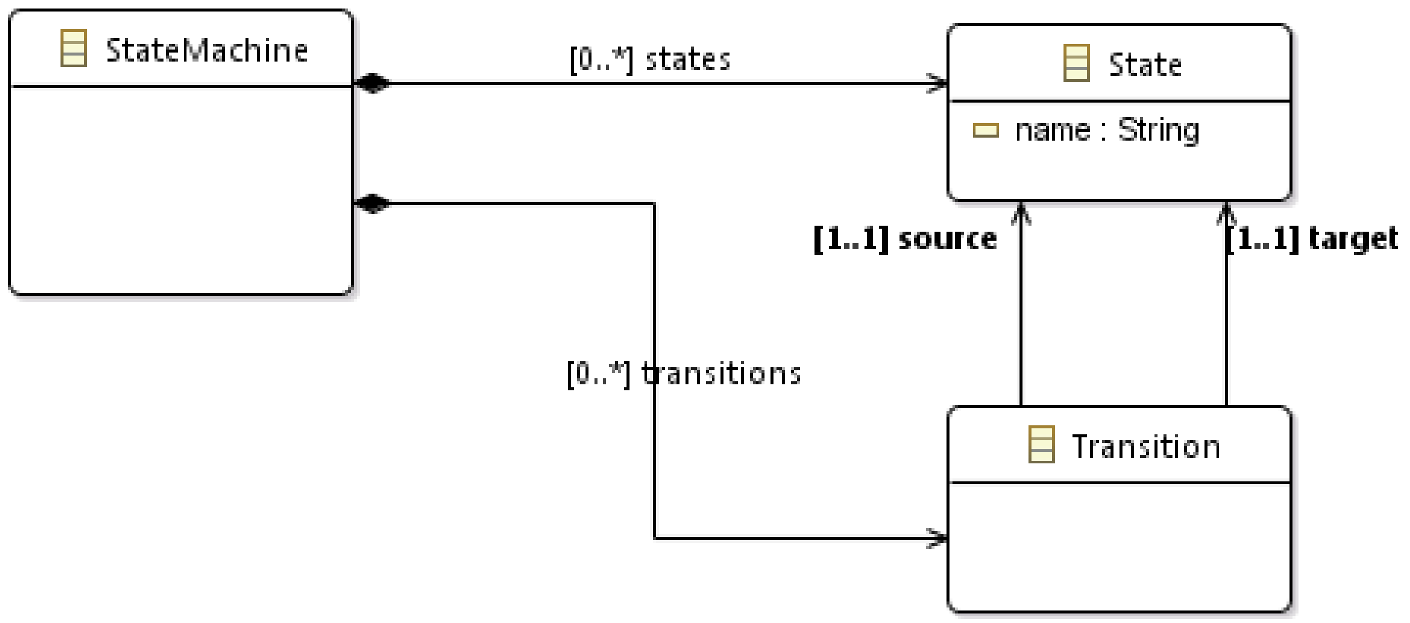
- EClass
  - Own attributes (data types) and typed references (classes)
  - Can have a super type (autres EClass) **[specialization]**

Ecore: EMF meta-metamodel (M3)



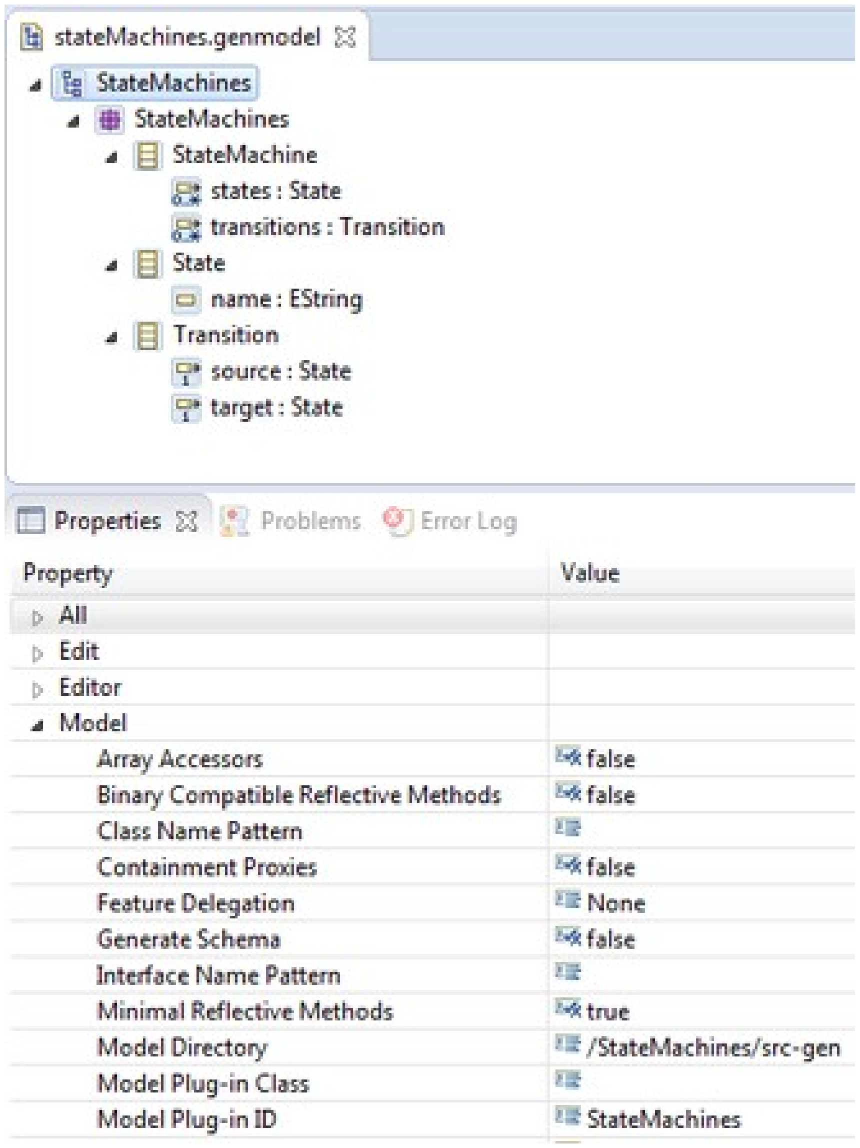
- Small meta-metamodel
  - Enhance the native introspection of Java
  - Only structural information (no access to the code)

Example – State Machines



- Ecore Modeling Project (Eclipse Modeling)
  - StateMachines.aird
  - StateMachines.ecore

Generation model: .genmodel



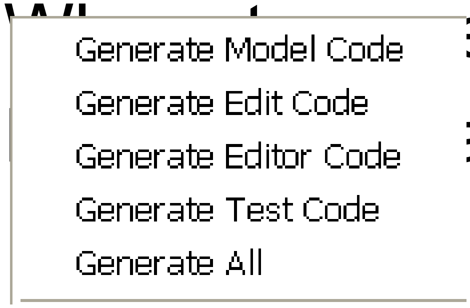
stateMachines.genmodel

- StateMachines
  - StateMachines
    - StateMachine
      - states : State
      - transitions : Transition
    - State
      - name : EString
    - Transition
      - source : State
      - target : State

Properties | Problems | Error Log

Property	Value
All	
Edit	
Editor	
Model	
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	
Containment Proxies	false
Feature Delegation	None
Generate Schema	false
Interface Name Pattern	
Minimal Reflective Methods	true
Model Directory	/StateMachines/src-gen
Model Plug-in Class	
Model Plug-in ID	StateMachines

- ❑ Import Ecore model
- ❑ Create a .genmodel
  - What to generate
  - erate
  - ate



Generate code/Edit/Editor

- Abstract model
  - Java interfaces

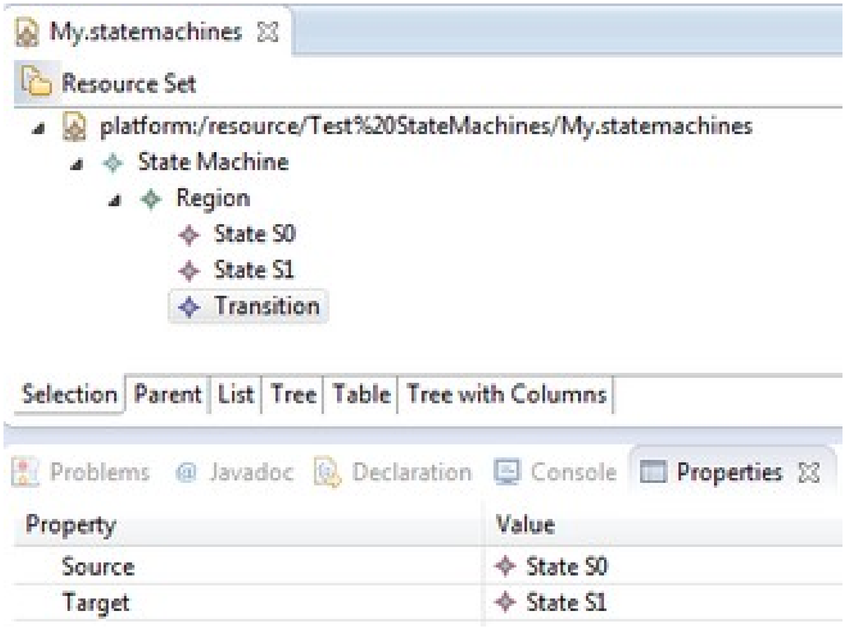
```
public interface State extends EObject {
 /** @generated */  String getName();
 /** @generated */  void setName(String value);
```

- Implementation
  - With listeners

```
public abstract class StateImpl extends EObjectImpl implements State {
  protected static final String NAME_EDEFAULT = null;
  protected String name = NAME_EDEFAULT;
  public void setName(String newName) {
   String oldName = name;
   name = newName;
   if (eNotificationRequired())
     eNotify(new ENotificationImpl(this, Notification.SET, StsPackage.STATE__NAME, oldName, name));
  }
```

Automatic code generation

- Automatic generation
  - Tree editor
  - XML Marshalling/Unmashalling
  - XML Validator
  - Wizard for creating new models



Property	Value
Source	State S0
Target	State S1

CODE GENERATION

Kinds of model transformations

- **Model to text**
  - Generate text (or code) from a model
  - Dedicated languages: XSLT
  - Manual: in Java through the Ecore API
- **Model to model**
  - Transform a model into another model
    - Ex: UML State Machines into NuSMV files
  - Dedicated transformation languages
    - ATL, Kermeta, QVTo

Accessing the model

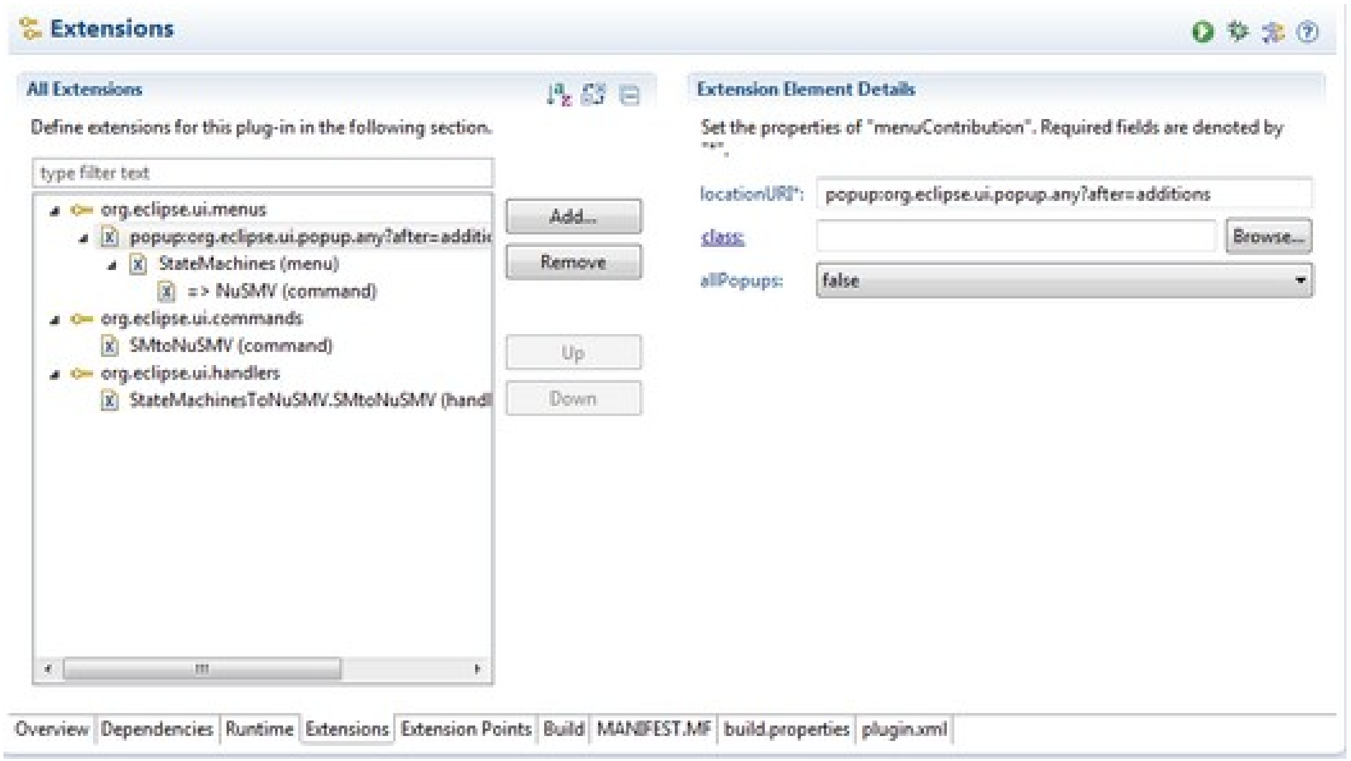
- Standalone applications
  - EMF generates a set of helpers to access/parse/generate models
- Through an eclipse plugin
  - Small Java program that augments Eclipse
    - Add menu, button, editors, …
  - Better/easier integration with other tools
  - Needs *Eclipse Modeling*
  - File/New/Plug-in project…

An example of plug-in

- Add a menu to Eclipse (3 extensions needed)
  - org.eclipse.ui.menus
    - Add a menu and menu item into Eclipse
  - org.eclipse.ui.commands
    - Add a command: can be (un)done through menus or toolbars
  - org.eclipse.ui.handlers
    - Attach a handler to a command (code to be executed)

org.eclipse.ui.menus

- 3 stages
  - menuContribution: popup:org.eclipse.ui.popup.any?after=additions
  - menu: with a label
  - command: MenuItem that references a command



Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

```
org.eclipse.ui.menus
  popup:org.eclipse.ui.popup.any?after=additi
    StateMachines (menu)
      => NuSMV (command)
org.eclipse.ui.commands
  SMtoNuSMV (command)
org.eclipse.ui.handlers
  StateMachinesToNuSMV.SMtoNuSMV (handl
```

Extension Element Details

Set the properties of "menuContribution". Required fields are denoted by "*".

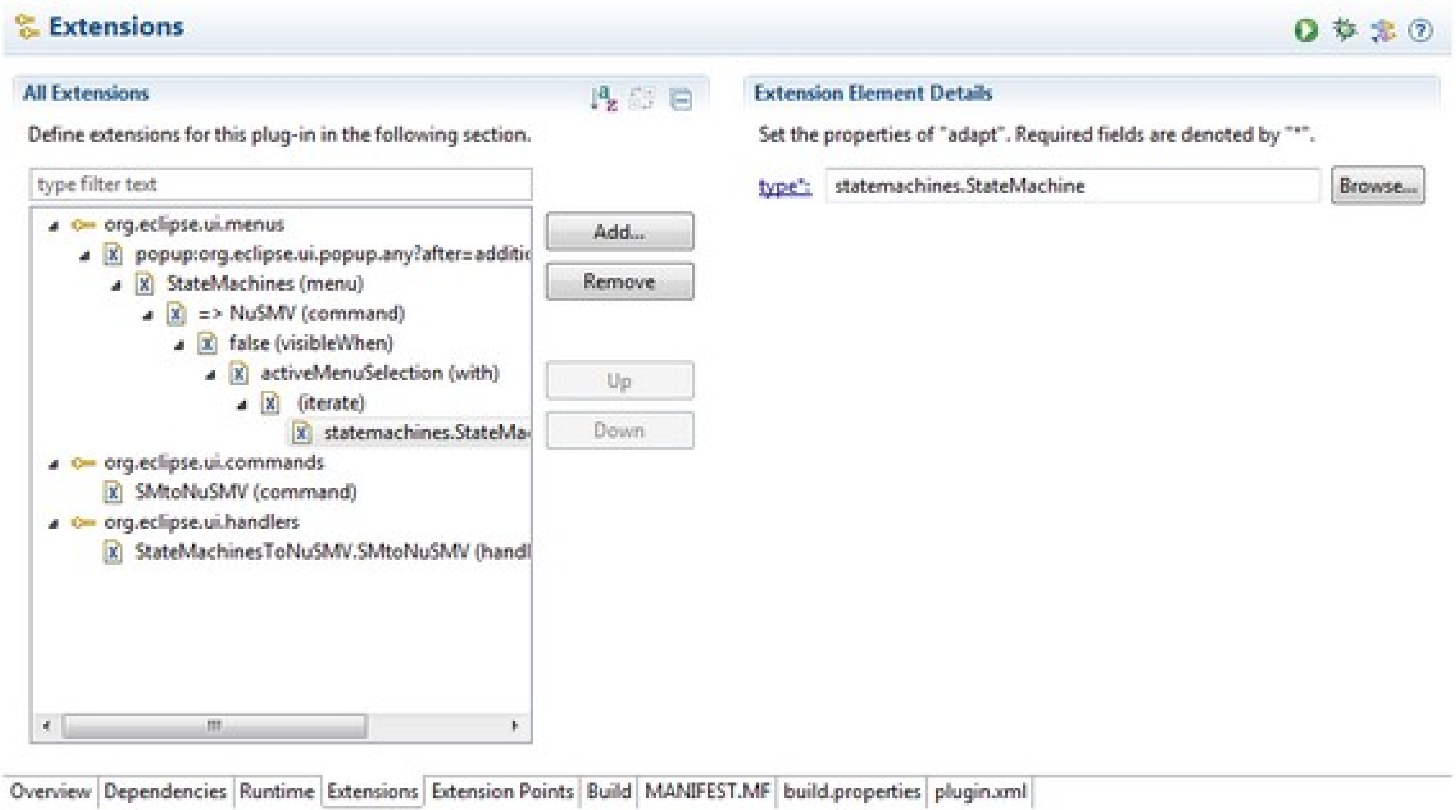
locationURI*: popup:org.eclipse.ui.popup.any?after=additions

class:

allPopups: false

org.eclipse.ui.menus

- Select when the menu is visible
  - Ex1: only when a statemachines.StateMachine is selected
    - Requires a **dependency** to the code generated by EMF
  - Ex2: org.eclipse.uml2.uml.StateMachine
    - Requires a **dependency** to org.eclipse.uml2.uml



Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

```
org.eclipse.ui.menus
  popup:org.eclipse.ui.popup.any?after=additic
    StateMachines (menu)
      => NuSMV (command)
        false (visibleWhen)
          activeMenuSelection (with)
            (iterate)
              statemachines.StateMa
org.eclipse.ui.commands
  SMtoNuSMV (command)
org.eclipse.ui.handlers
  StateMachinesToNuSMV.SMtoNuSMV (handl
```

Extension Element Details

Set the properties of "adapt". Required fields are denoted by "*".

type*: statemachines.StateMachine

Browse...
Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF | build.properties | plugin.xml

Visibility of the menu

- Either programmatically in the code
- Through the extension interface
  - menuContribution locationURI: popup:org.eclipse.ui.popup.any?after=additions
  - menu label: « Name of the menu »
  - Command
    - Label: « Name of the command »
    - Id: same as the id of the command
  - isVisibleWhen: checkifEnabled=true
  - With variable: activeMenuSelection
  - Iterate operator: and isEmpty:false
  - Adapter(depends on where to integrate)
    - Type: org.eclipse.core.resources.IFile (File in the Navigator)
    - Type: org.eclipse.jdt.core.ICompilationUnit (Java file in Package explorer)

org.eclipse.ui.commands

- Allows for the creation of commands
  - A command can be done/undone by clicking a menu or a toolbar icon or by code
  - Give a unique id
    - Ex: fr.unice.m1.SMtoNuSMV.command
  - Must be referenced by menus, toolbars, handlers

org.eclipse.ui.handlers

- Specify what code should be executed/attached to a command
  - Reference a command through its id
  - Define a class that must implement **org.eclipse.core.commands.IHandler**

```
public class SMToNuSMV implements IHandler {
  public void addHandlerListener(IHandlerListener handlerListener) {}
  public void dispose() {}
  public Object execute(ExecutionEvent event) throws ExecutionException {
    // TODO Auto-generated method stub
    return null;
  }
  public boolean isEnabled() { return true; }
  public boolean isHandled() { return true; }
  public void removeHandlerListener(IHandlerListener handlerListener) {}
}
```

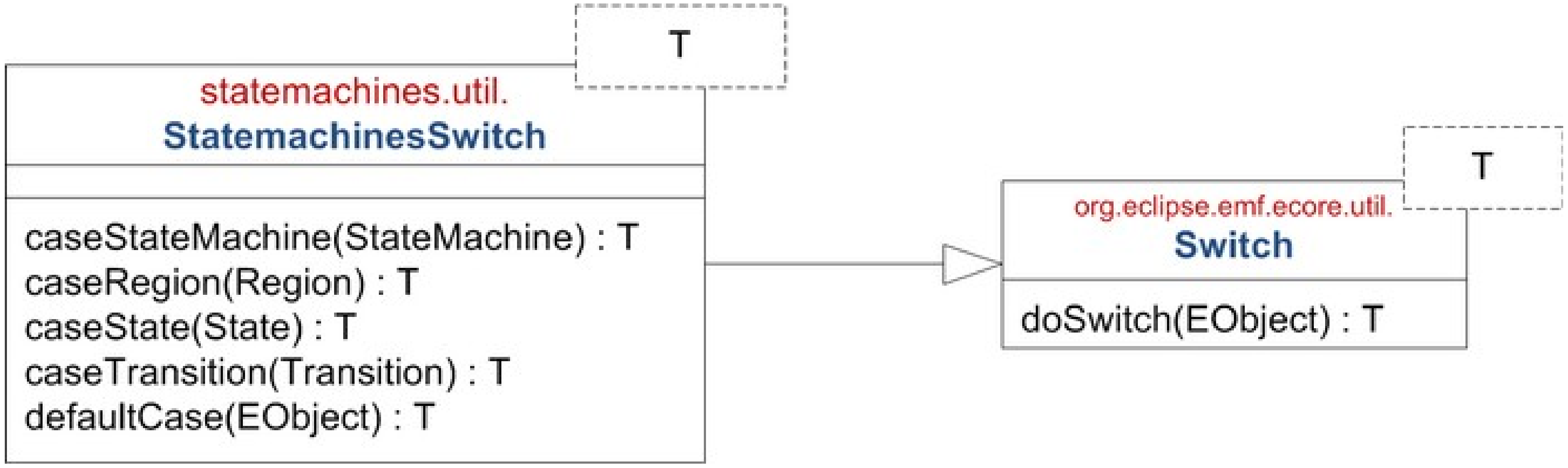
Use the Switch

- Define the right handler

```
public class SMToNuSMVHandler extends AbstractHandler {
  @Override
  public Object execute(ExecutionEvent event) throws ExecutionException {
    ISelection selection = PlatformUI.getWorkbench().getActiveWorkbenchWindow()
                                     .getActivePage().getSelection();
    if (!(selection instanceof StructuredSelection)) return null;
    Object selected = ((StructuredSelection)selection).getFirstElement();

    // The type should be guaranteed by the "isVisibleWhen"
    assert(selected instanceof StateMachine);

    // do something with (StateMachine)selected

    return null;
  }
}
```

EMF Switches

- Realize the **visitor** design patterns
  - Automatically generated by EMF
  - Allows for *visiting* a complex hierarchical structure



Switch: do it yourself

- Example that counts the number of elements

```
public class SMCountElements extends StatemachinesSwitch<Boolean> {
  private int nbStatemachines = 0;
  private int nbStates = 0;
  private int nbTransitions = 0;

  public Boolean caseStateMachine(StateMachine object) {
    nbStatemachines ++;
    for(Region region : sm.getRegions()) doSwitch(region);
    return true;
  }
  public Boolean caseRegion(Region region) {
    for(State state : region.getStates()) doSwitch(state);
    for(Transition transition : region.getTransitions()) doSwitch(transition);
    return true;
  }
  public Boolean caseState(State object) {
    nbStates++;
    return true;
  }
  public Boolean caseTransition(Transition object) {
    nbTransitions++;
    return true;
  }
```

Use the Switch

- Define the right handler

```
public class SMToNuSMVHandler extends AbstractHandler {
  @Override
  public Object execute(ExecutionEvent event) throws ExecutionException {
    ISelection selection = PlatformUI.getWorkbench().getActiveWorkbenchWindow()
                                    .getActivePage().getSelection();
    if (!(selection instanceof StructuredSelection)) return null;
    Object selected = ((StructuredSelection)selection).getFirstElement();

    // The type should be guaranteed by the "isVisibleWhen"
    assert(selected instanceof StateMachine);
    SMCountElements counter = new SMCountElements();
    counter.doSwitch((StateMachine)selected);
    JOptionPane.showMessageDialog(null, counter.getNbStatemachines()+" state machines\n"+
                                    counter.getNbStates()+" states\n"+
                                    counter.getNbTransitions()+" transitions",
                              "State Machines", JOptionPane.INFORMATION_MESSAGE);
    return null;
  }
}
```