Model-Driven Engineering

An introduction to the EMF facilities

F. Mallet

Frederic.Mallet@unice.fr

Université Nice Sophia Antipolis

Model-Driven Engineering

- Software Engineering
  - Build software useful to end-users to solve a particular problem

- **M**odel-**D**riven **E**ngineering
  - Build models to help software engineer to build **faster**, **better** software able to handle **more complex** problems

Modularity vs. Composition

- Two steps to solve a problem
  - Modularity: *“Divide and Conquer”*
    - Decompose into (simpler) sub-systems
  - Composition
    - Compose partial solutions to build an integrated solution
- Manual or automatic
  - Manual approaches are not scalable
  - Need tools to decompose **and** integrate

The software crisis is still on !

“The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!” Edsger Dijkstra, 1972

- Structured and object-oriented programming aimed at solving the crisis

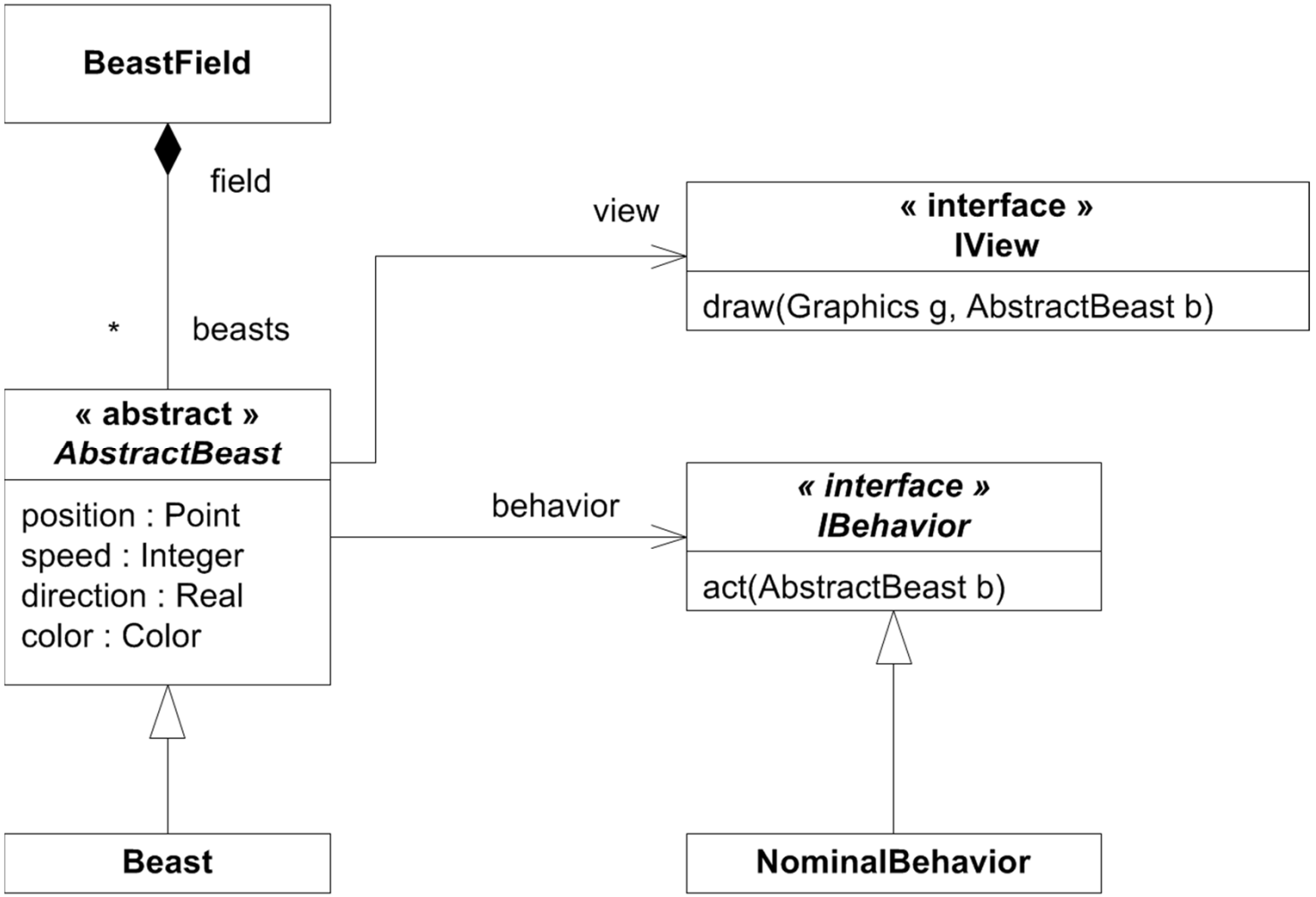
- Since the 70s, computer complexity has kept increasing very quickly (Moore’s law)
  - The gap between what need to be built and what can be built with today abstractions is increasing !

Program or model

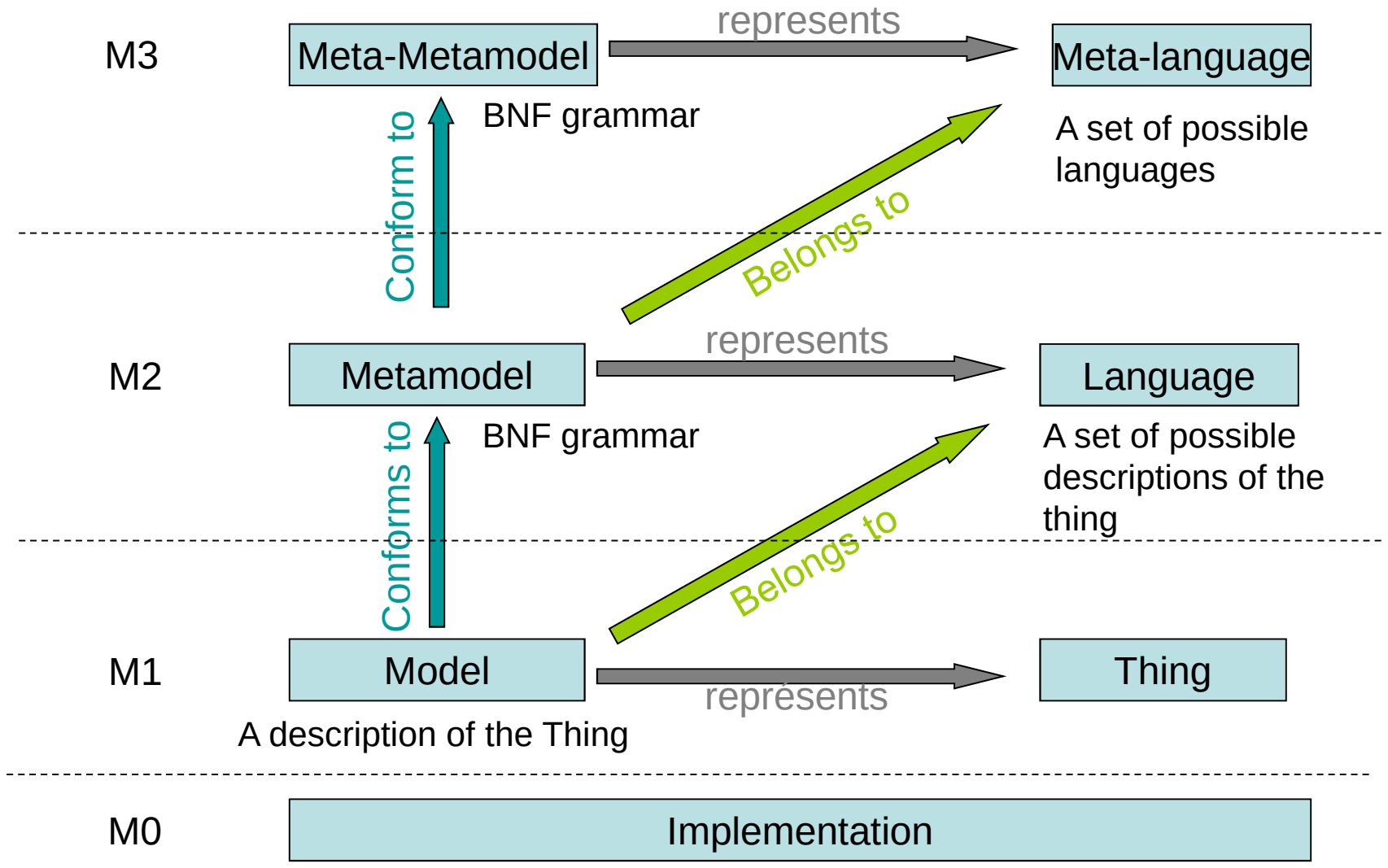
```
abstract class AbstractBeast {
   protected int x, y;              // position on the screen
   int speed;                       // speed [pix/s]
   double direction;                // radians [0 - 2 PI[
   protected Color color;           // Filling color
   protected BeastField field;      // the field
   static final int SIZE = 10;

   protected AbstractBeast(BeastField field, int x, int y,
      Color color) {
      this.field = field;
      this.x = x;
      this.y = y;
      this.color = color;

      Random gen = new Random();
      direction = gen.nextFloat() * 2 * Math.PI;
      speed = gen.nextInt(field.maxSpeed);
   }

   public abstract void act();

   protected IBehavior behavior;
```

```
   public boolean see(AbstractBeast b) {
      double angle = Math.atan2 (b.getY()-y, b.getX()-x);
      double diff = Math.abs(angle-direction)%(2*Math.PI);
      if (diff>Math.PI) diff=2*Math.PI-diff;
      return diff<champDeVue/2;
   }

   public double getDistance(IBeast b) {
      return distanceFromAPoint(b.getX(), b.getY());
   }

   double distanceFromAPoint(double x1, double y1){
         // @returns distance between the beast and a point
      return Math.sqrt((x1 - x)*(x1-x) + (y1 - y)*(y1 - y));
   }

   protected IView view = null;
   void drawYourself(Graphics g) {
      if (this.view != null)
         this.view.draw(g, this);
   }

   final public void translate(double dx, double dy) {
      this.x += dx;
      this.y += dy;
   }
}
```

Program or model



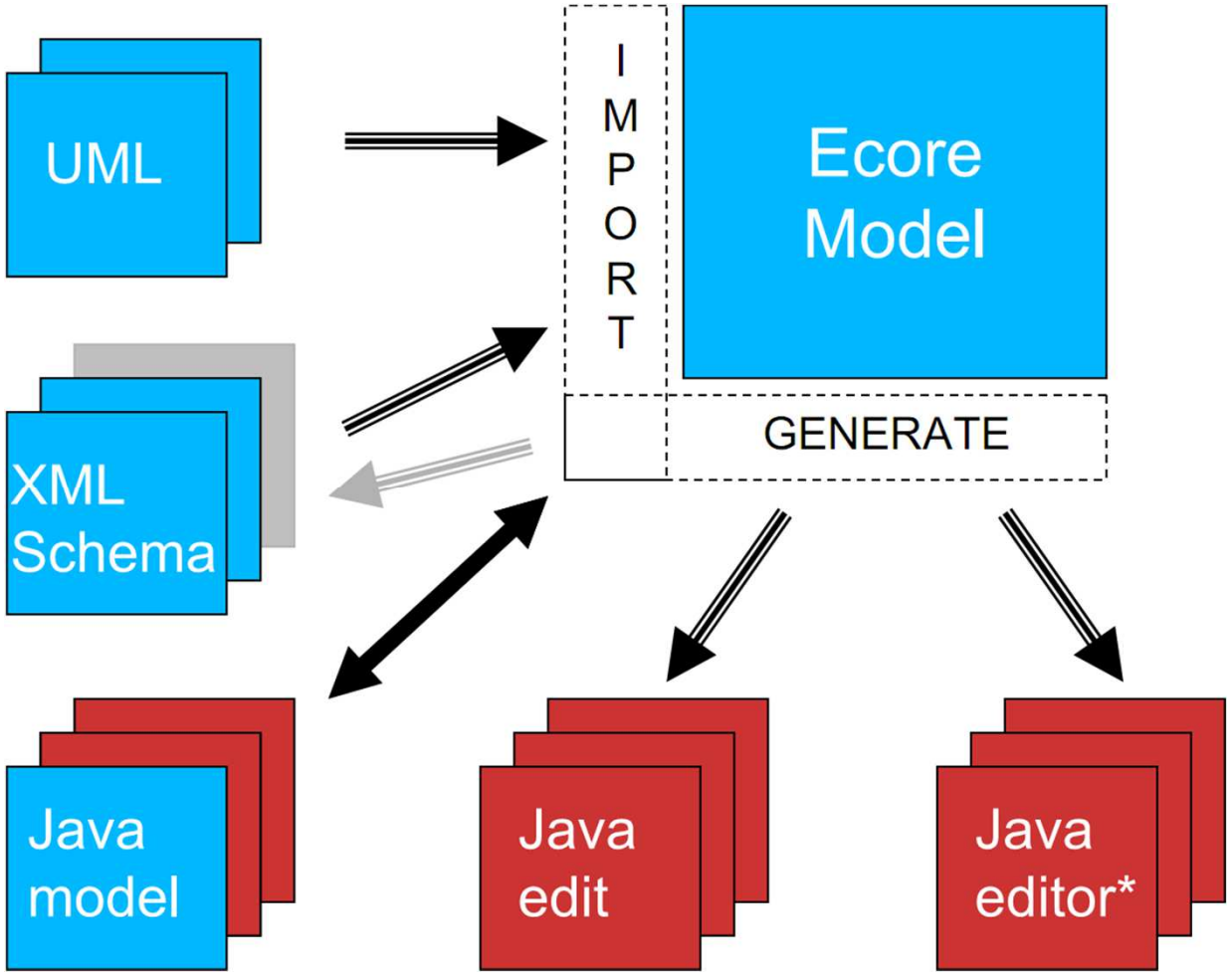
Models/Metamodels/Languages



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 (**GMF** – Graphical Modeling Framework)
  - Generate a parser, syntax highlighting (XText, TCS, Sintaks)

EMF import/export

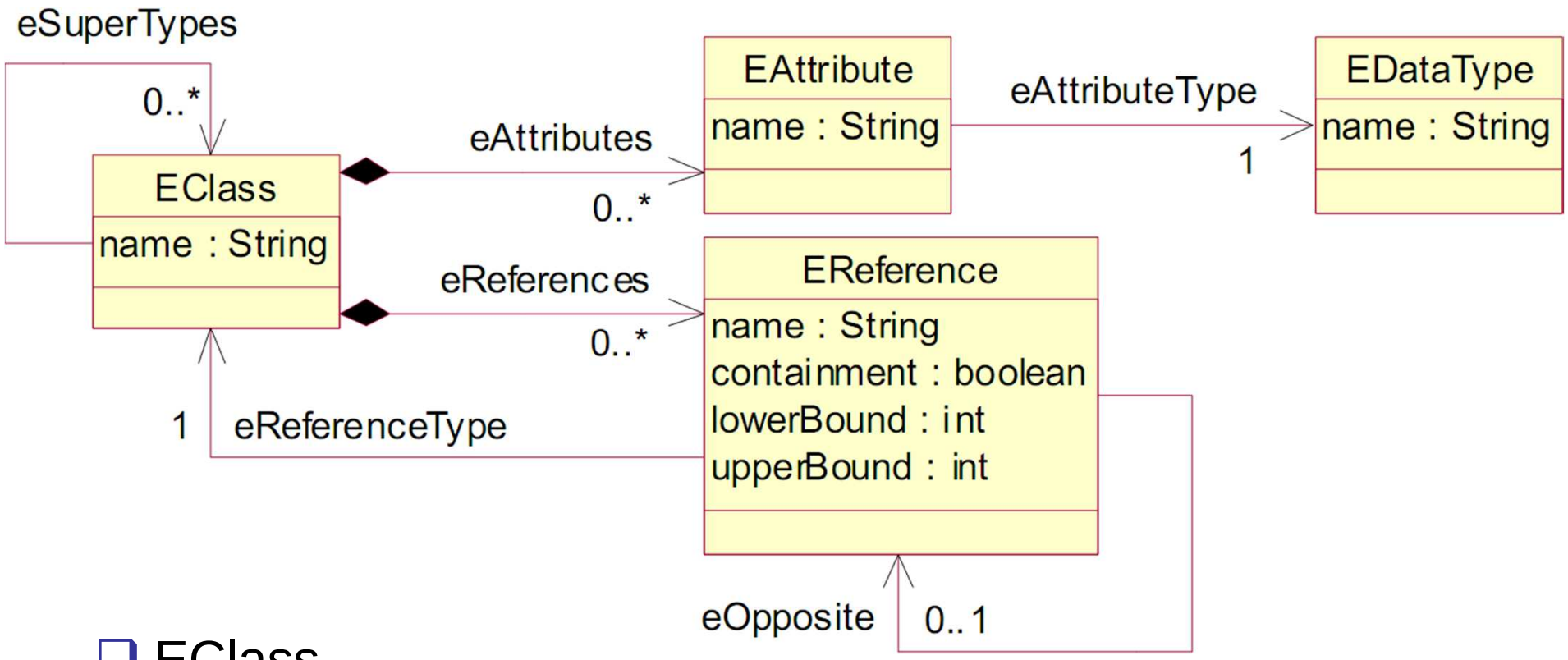


IBM, Ed. Merks & D. Steinberg, EclipseCon 2005

EMF and UML ?

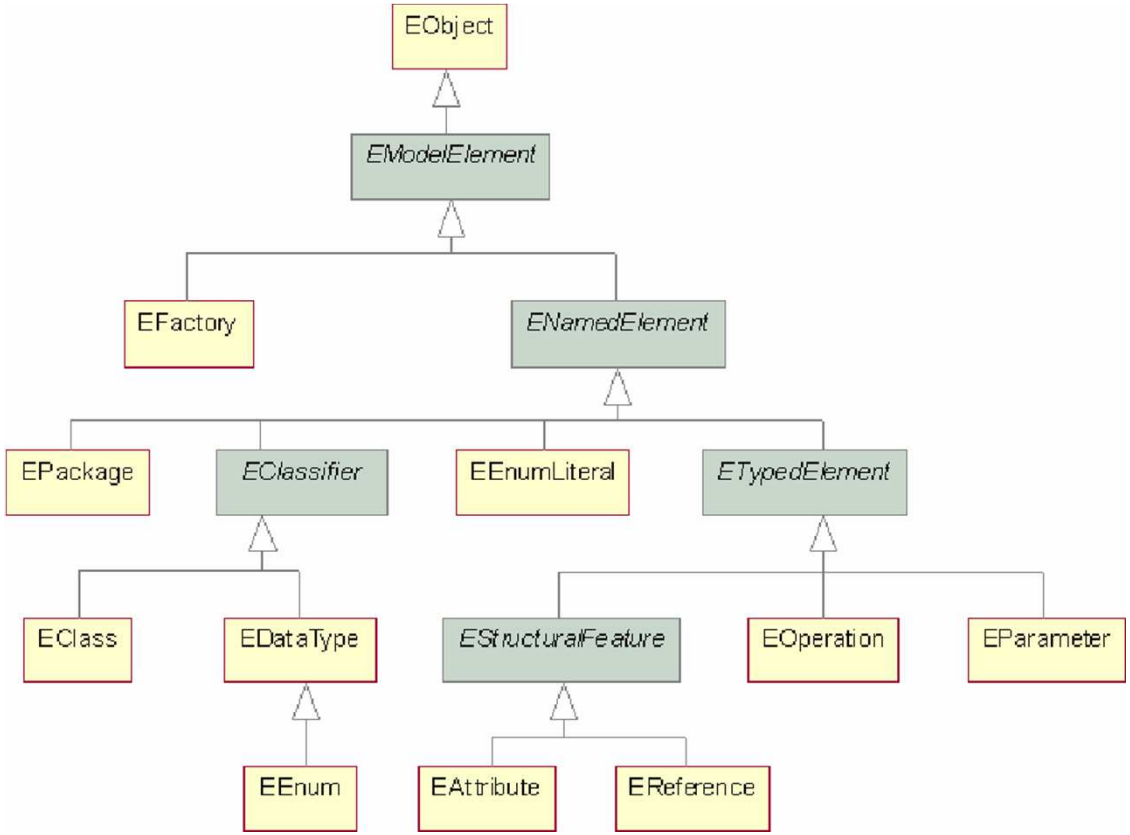
- **MOF**: Meta-metamodel (M3) by OMG
  - **M**eta-**O**bject **F**acilities
  - EMOF (Essential MOF) is a subset
- EMF
  - Was initially just an implementation of the MOF
  - Has evolved to become **ECORE**
- Two very different business models
  - OMG # Eclipse Foundation

Ecore: EMF meta-metamodel (M3)



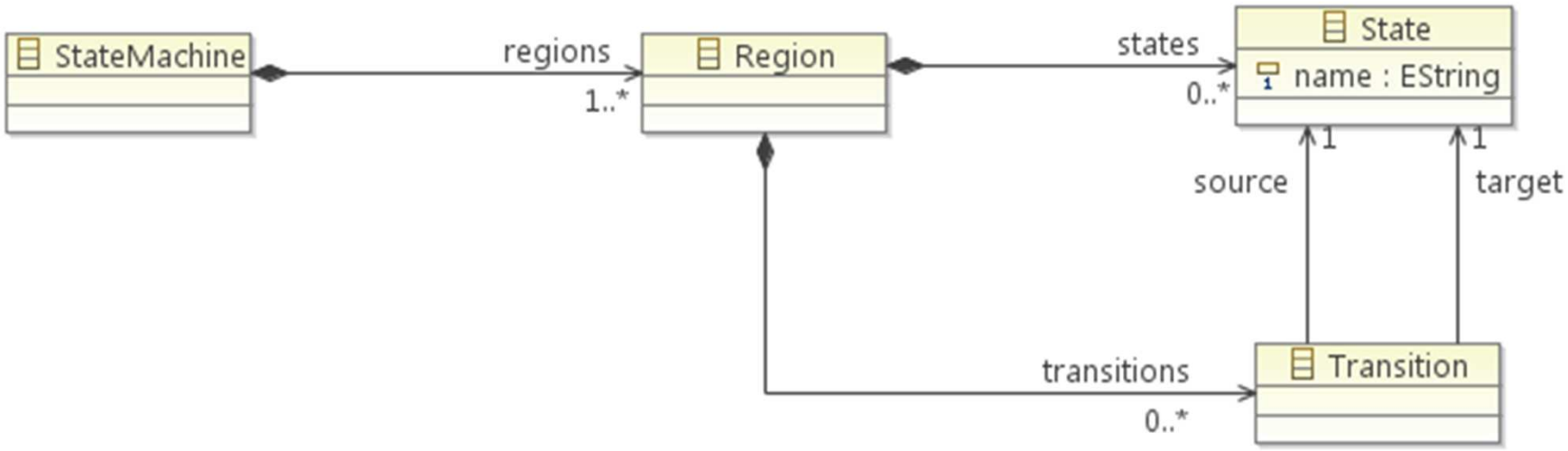
- EClass
  - Own attributes (data types) and typed references (classes)
  - Can have a super type (autres EClass) **[specialization]**

Ecore: EMF meta-metamodel (M3)



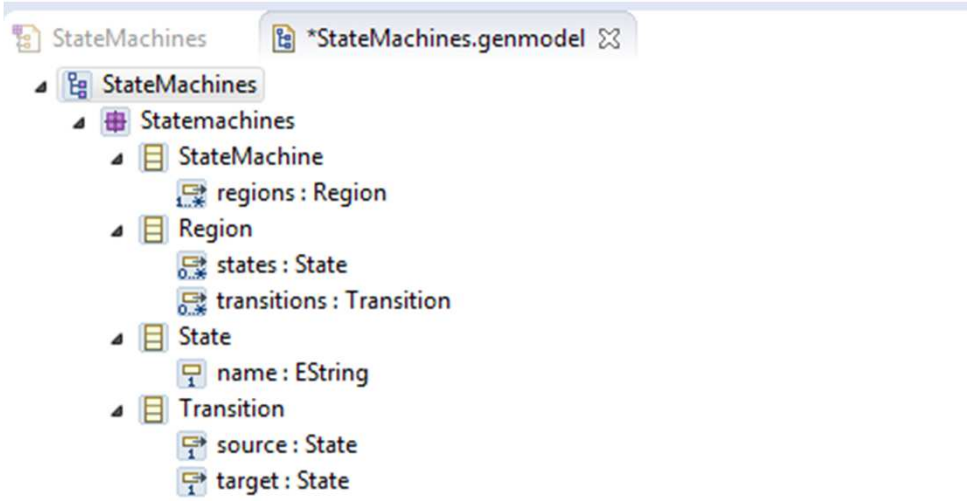
- Small meta-metamodel
  - Enhance the native introspection of Java
  - Only structural information (no access to the code)

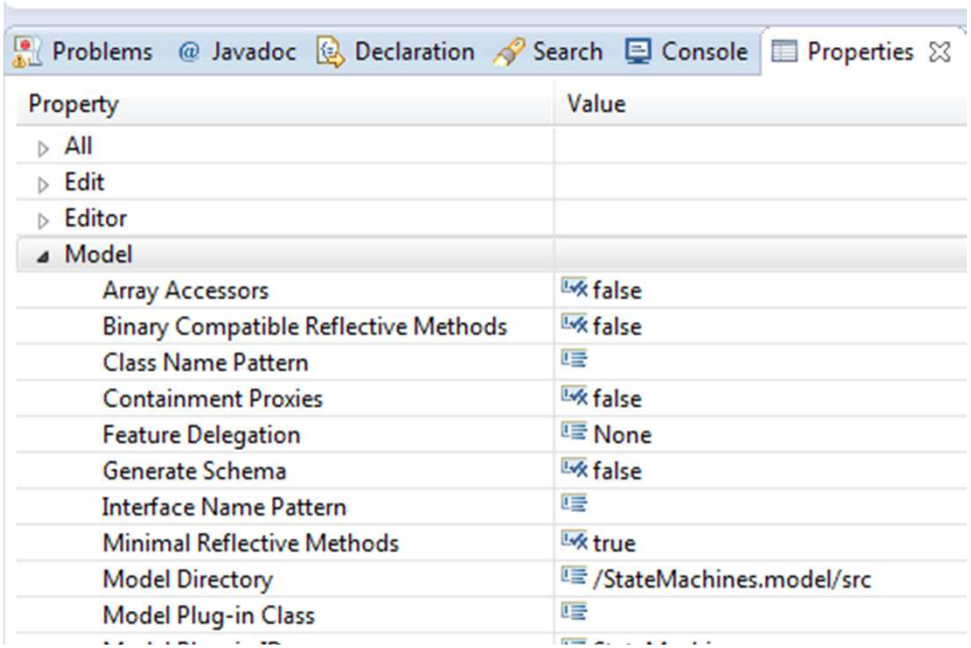
Example – State Machines



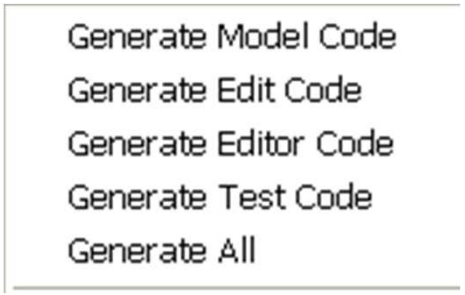
- Ecore Diagrams
  - StateMachines.ecorediag + StateMachines.ecore

Generation model: .genmodel





- Import Ecore model
- Create a .genmodel
  - What to generate
  - Where to generate
  - How to generate



Generate code/Edit/Editor

Abstract model

- Java interfaces

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

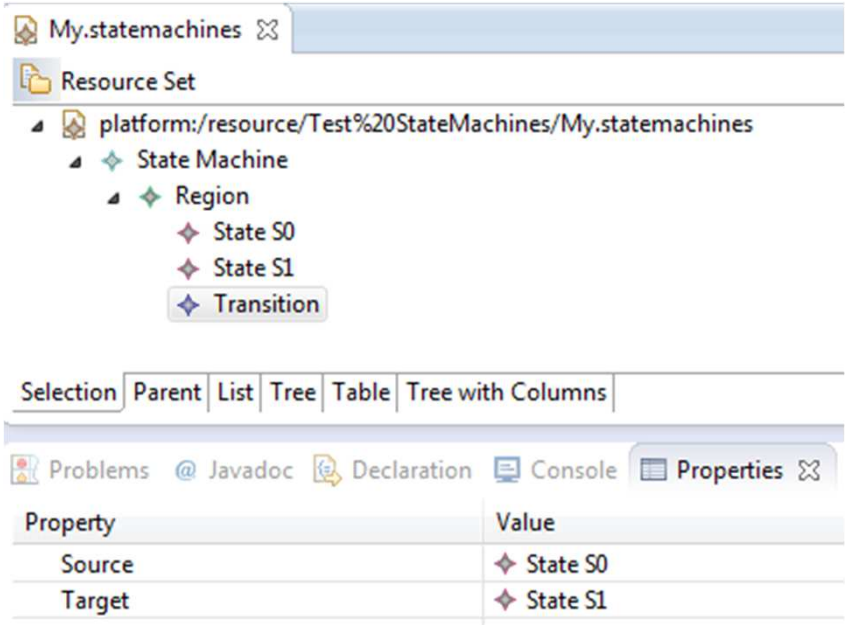
Implementation

- 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



MODEL TRANSFORMATION

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

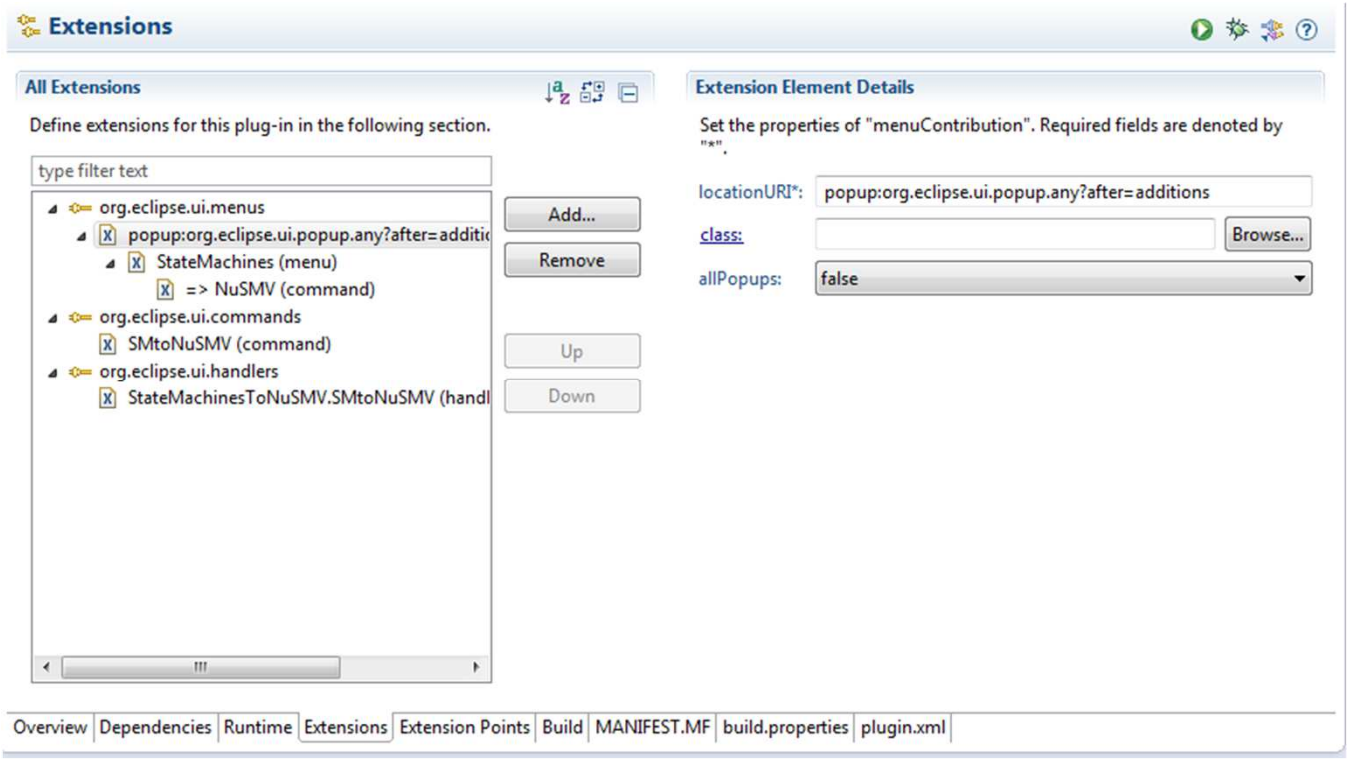
- Standalone applications
  - EMF generates a set of helpers to access/parse/generate models
- Through an eclipse plugin
  - Small Java program that augments Eclipse
    - Add menu, button, editors, …
  - Better/easier integration with other tools
  - Needs *Eclipse Modeling (Juno)*
  - File/New/Plug-in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

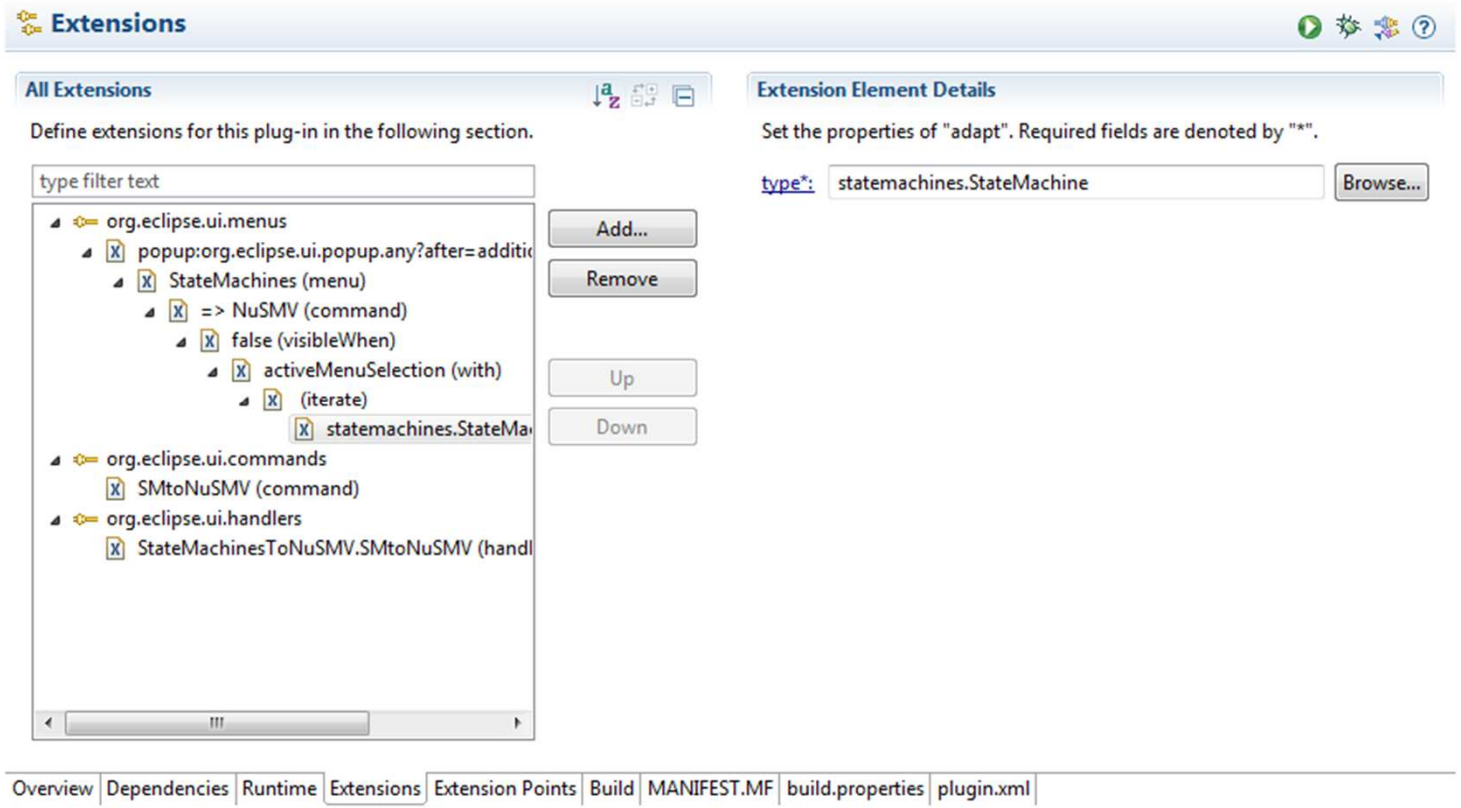
org.eclipse.ui.menus

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



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



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

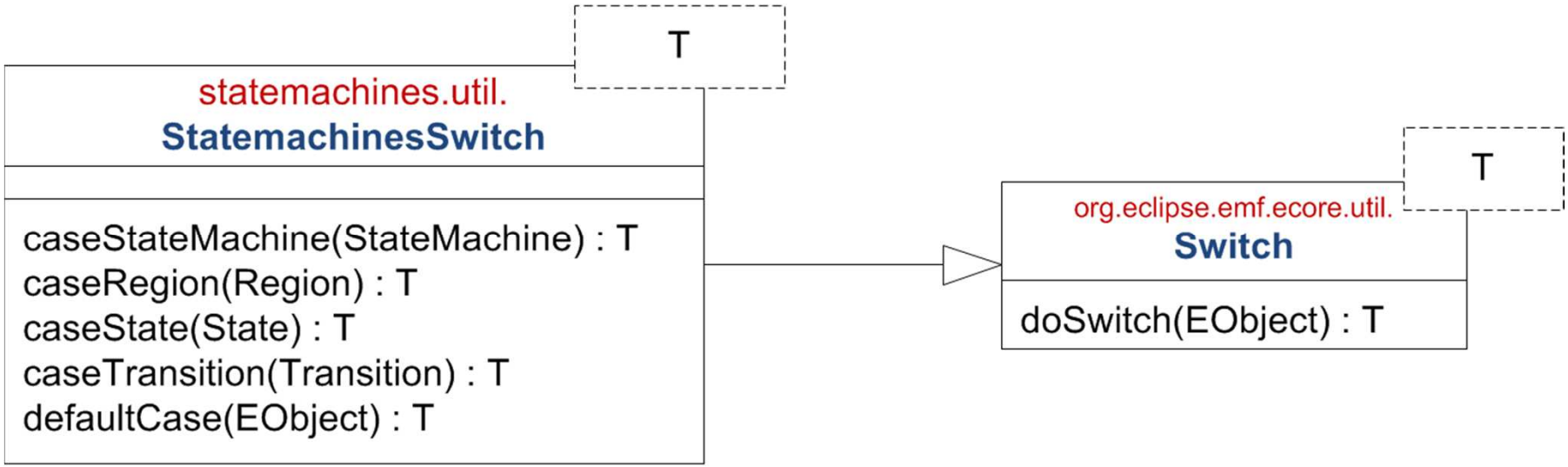
org.eclipse.ui.handlers

- 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) {}
}
```

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;
  }
}
```

Practical

- Generate a NuSMV code from
  - A UML state machine
  - A dedicated model
- Bring your laptops
  - Install Eclipse Modeling (Kepler)