# Active transformation within Kermeta

Olivier Beaudoux

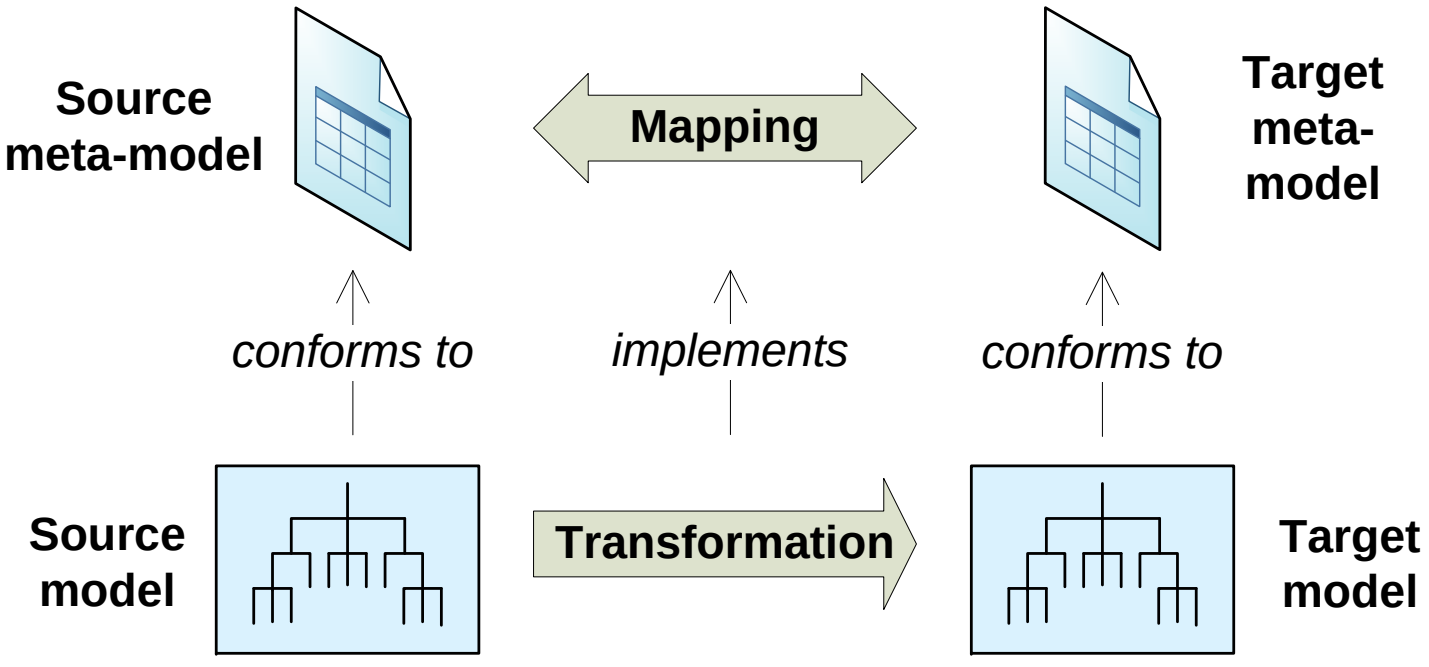
GRI - ESEO

Kermeta Day 2009

# Plan

1. Mapping versus Transformation
   - Overview
   - Solutions
2. Application to GUI
   - MVC
   - From data to GUI
3. Active Transformation: AcT.Kermeta
   - Observables and Observable Sets
   - Active sets
   - Mappings
4. Conclusion
   - Contribution
   - Perspectives

# Overview (1)


Source meta-model
Mapping
Target meta-model
conforms to
implements
conforms to
Source model
Transformation
Target model

# Overview (2)

## Definition

- A mapping **declares** a relation between model elements.
- A transformation **implements** this relation at the model level.

# Overview (2)

## Definition

- A mapping **declares** a relation between model elements.
- A transformation **implements** this relation at the model level.

## Problem

- Transformation = batch process
- Does not update the target $T$ while the source $S$ changes

# Solutions

## Incremental transformations

1. Computation of the source change $\triangle S = S(t+1) - S(t)$
2. Elicitation of a rule that matches $\triangle S$
3. Re-evaluation of the rule in order to produce $T(t+1)$

# Solutions

## Incremental transformations

1. Computation of the source change $\triangle S = S(t+1) - S(t)$
2. Elicitation of a rule that matches $\triangle S$
3. Re-evaluation of the rule in order to produce $T(t+1)$

$\rightarrow$ **Not a trivial algorithm...**

# Solutions

## Incremental transformations

1. Computation of the source change $\triangle S = S(t+1) - S(t)$
2. Elicitation of a rule that matches $\triangle S$
3. Re-evaluation of the rule in order to produce $T(t+1)$

$\rightarrow$ **Not a trivial algorithm...**

## Active transformations

Use of the observable/observer design pattern:

- The observable is the source (can also be the target).
- The observer is the transformation.

# Solutions

## Incremental transformations

1. Computation of the source change $\triangle S = S(t+1) - S(t)$
2. Elicitation of a rule that matches $\triangle S$
3. Re-evaluation of the rule in order to produce $T(t+1)$

$\rightarrow$ **Not a trivial algorithm...**

## Active transformations

Use of the observable/observer design pattern:

- The observable is the source (can also be the target).
- The observer is the transformation.

$\rightarrow$ **A more complex transformation...**

# Solutions

## Incremental transformations

1. Computation of the source change $\triangle S = S(t+1) - S(t)$
2. Elicitation of a rule that matches $\triangle S$
3. Re-evaluation of the rule in order to produce $T(t+1)$

$\rightarrow$ **Not a trivial algorithm...**

## Active transformations

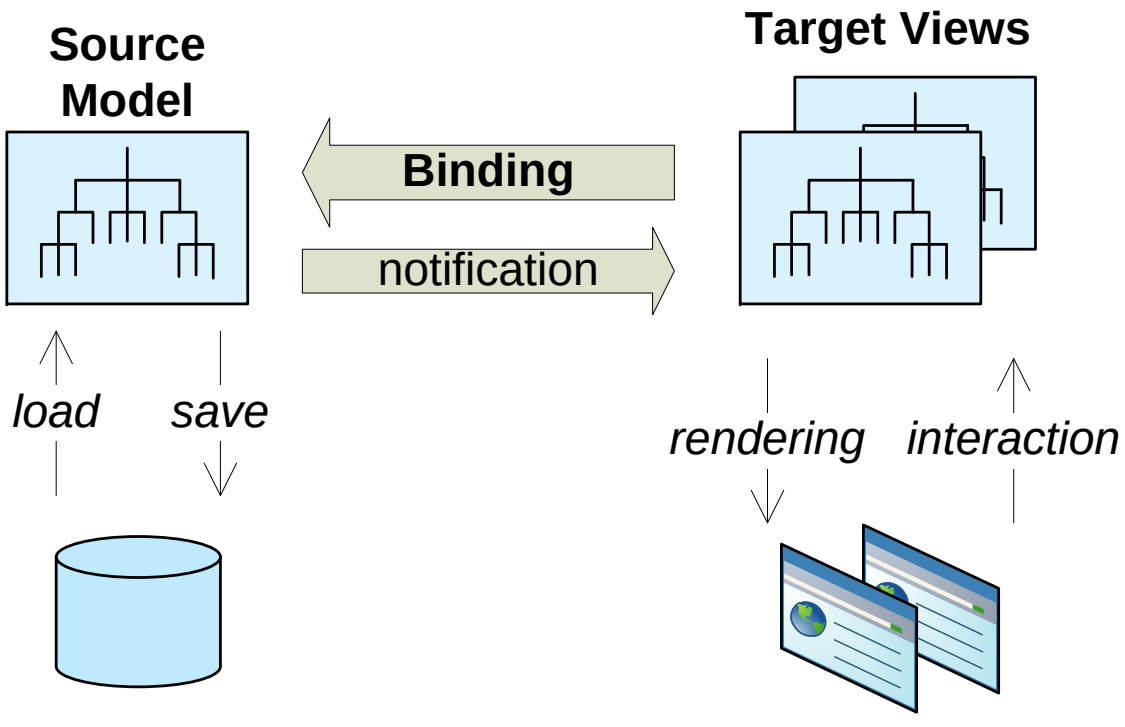
Use of the observable/observer design pattern:

- The observable is the source (can also be the target).
- The observer is the transformation.

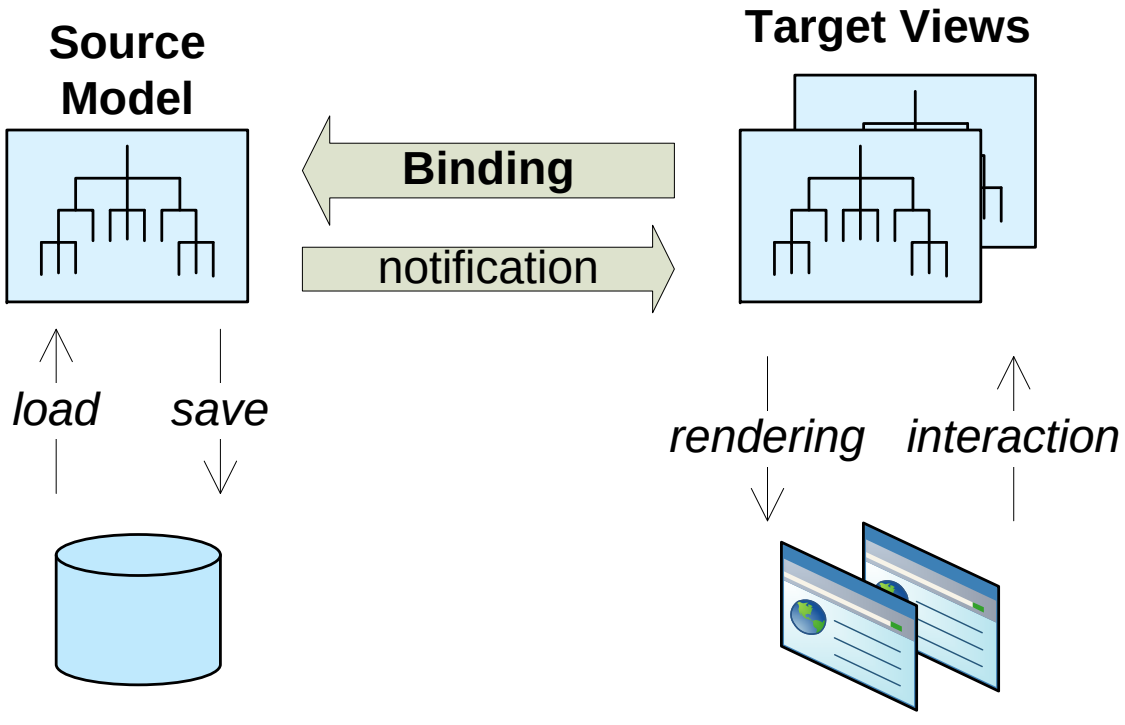
$\rightarrow$ **A more complex transformation...**

**... but generated from the mapping**

# MVC (1)


Source Model
Target Views
Binding
notification
load
save
rendering
interaction

# MVC (1)




- The observable is the source model (M).
- The observer is the target view (V).

# MVC (2)

## Pros

- A clear separation between models and views
- Multiple synchronized views
- Sharable models

# MVC (2)

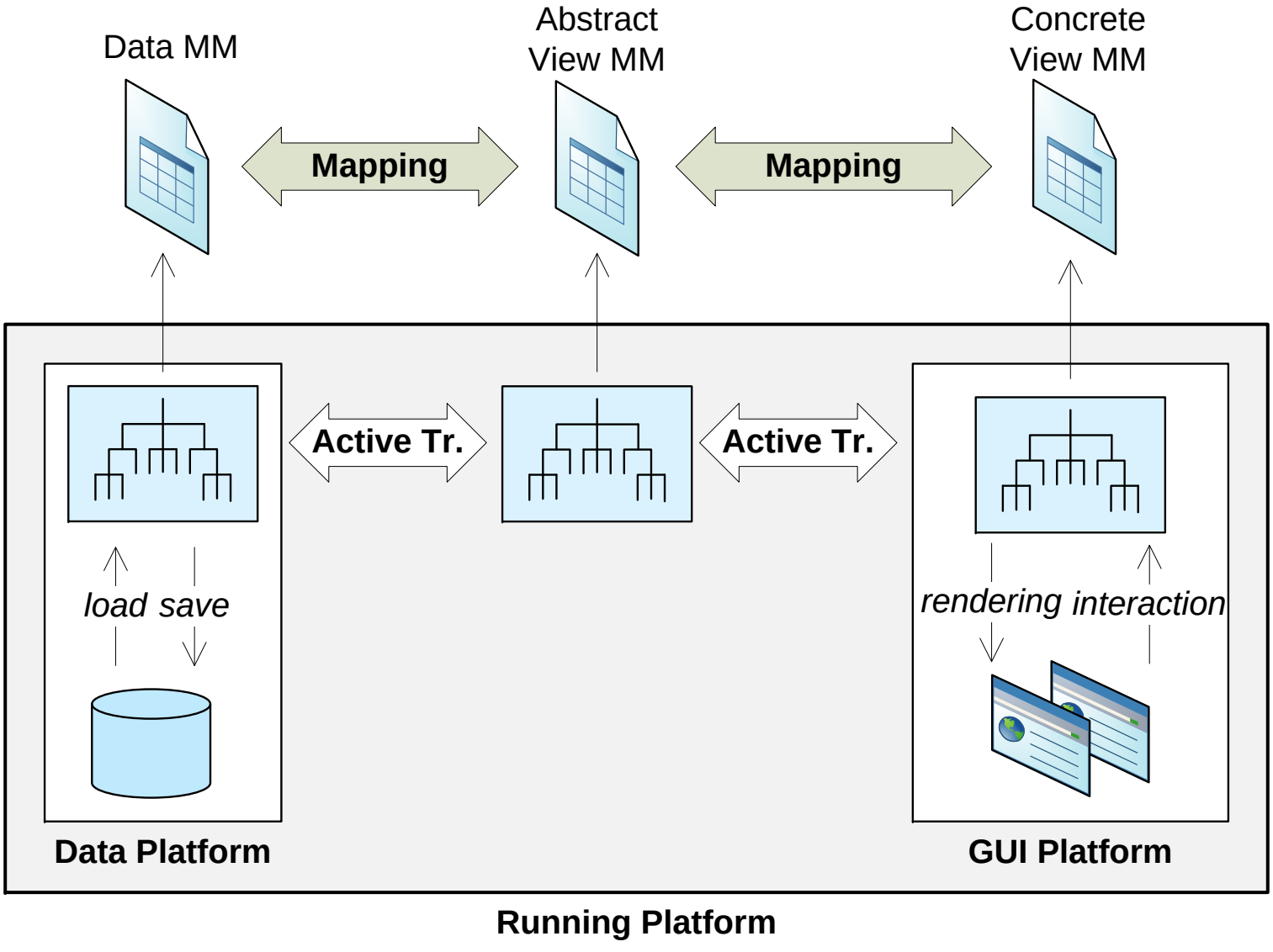
## Pros

- A clear separation between models and views
- Multiple synchronized views
- Sharable models

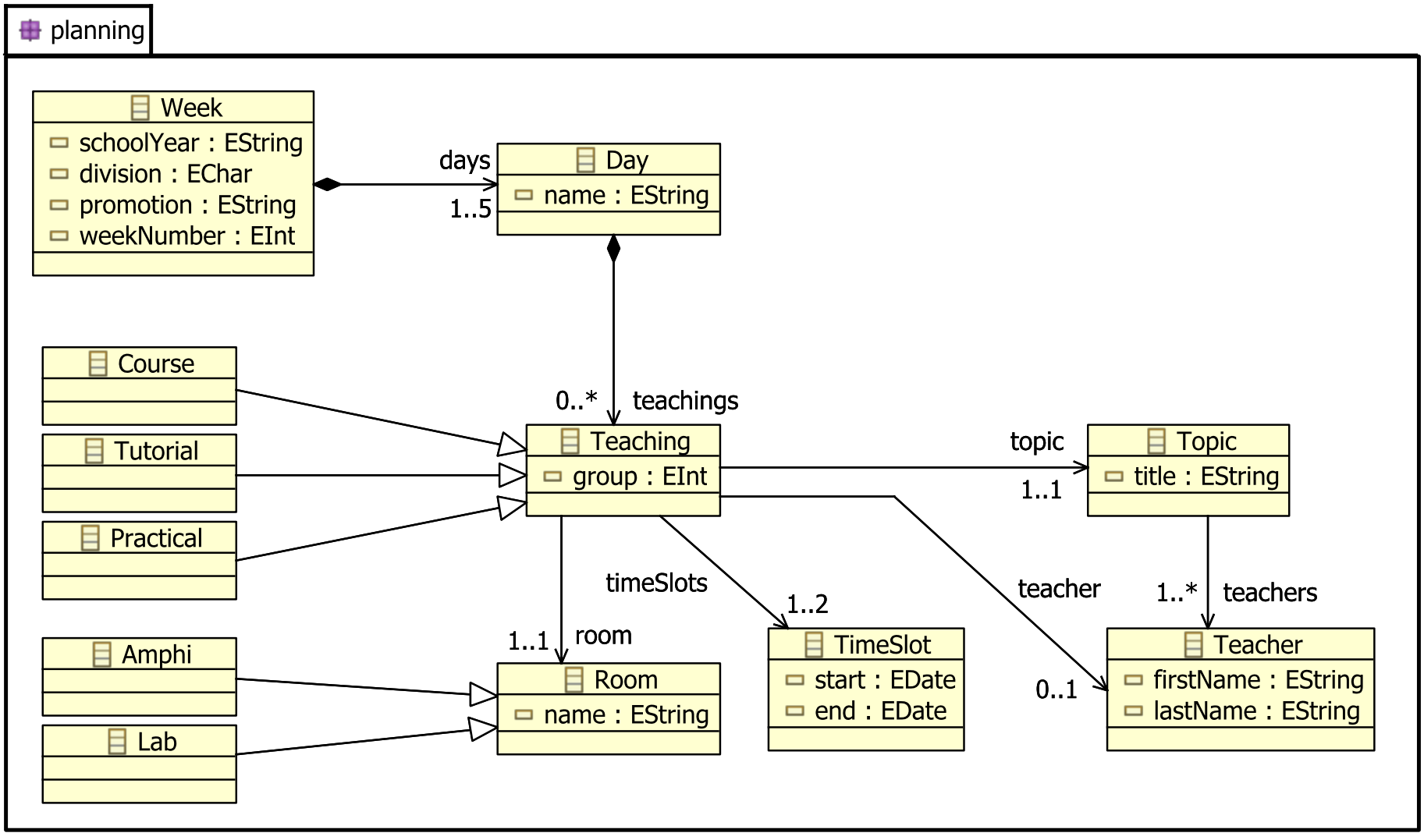
## Cons

- Complex implementation
- Depends on the GUI toolkit
- **M is the abstract model of V, not of the application data!**
  (*e.g.* the *TableModel* is the model of the widget *JTable*)

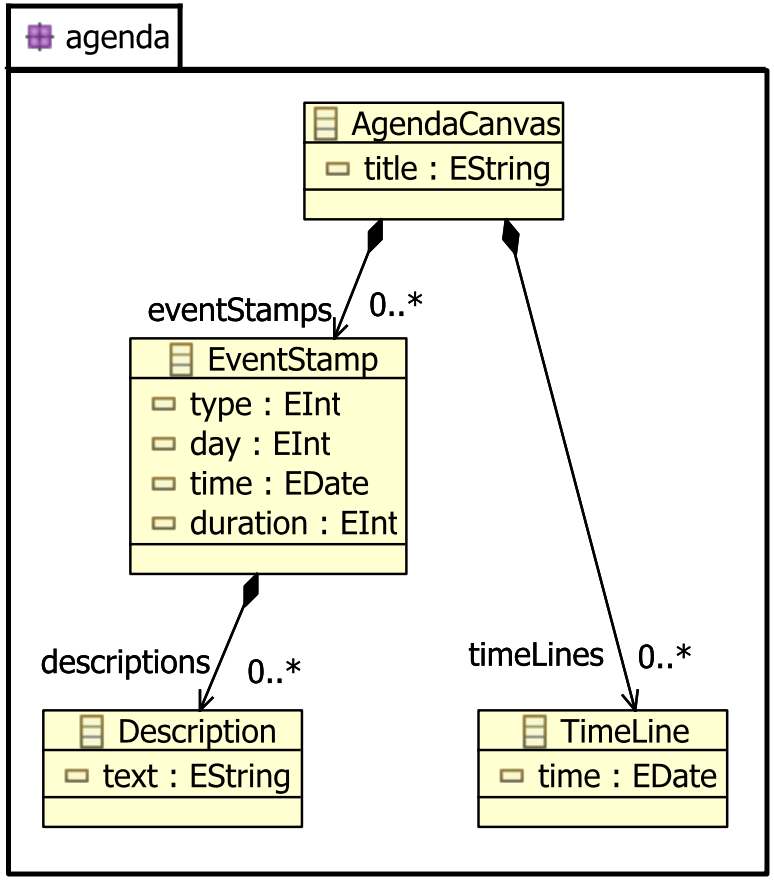
# From data to GUI



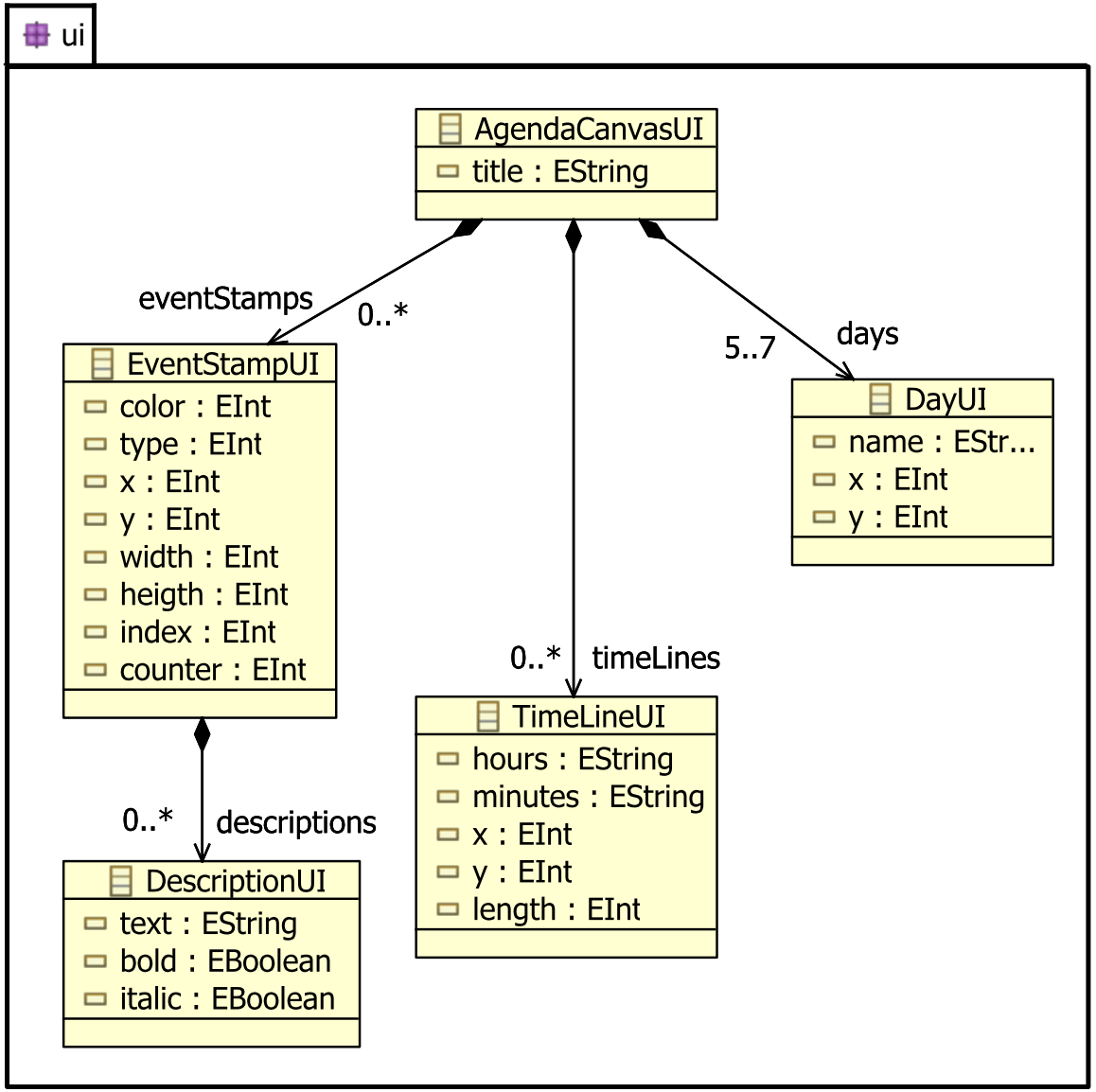
# Example (1)


planning
Week
schoolYear : EString
division : EChar
promotion : EString
weekNumber : EInt
days
1..5
Day
name : EString
Course
Tutorial
Practical
0..*
teachings
Teaching
group : EInt
topic
1..1
Topic
title : EString
timeSlots
1..2
teacher
1..*
teachers
Amphi
Lab
1..1
room
Room
name : EString
TimeSlot
start : EDate
end : EDate
0..1
Teacher
firstName : EString
lastName : EString

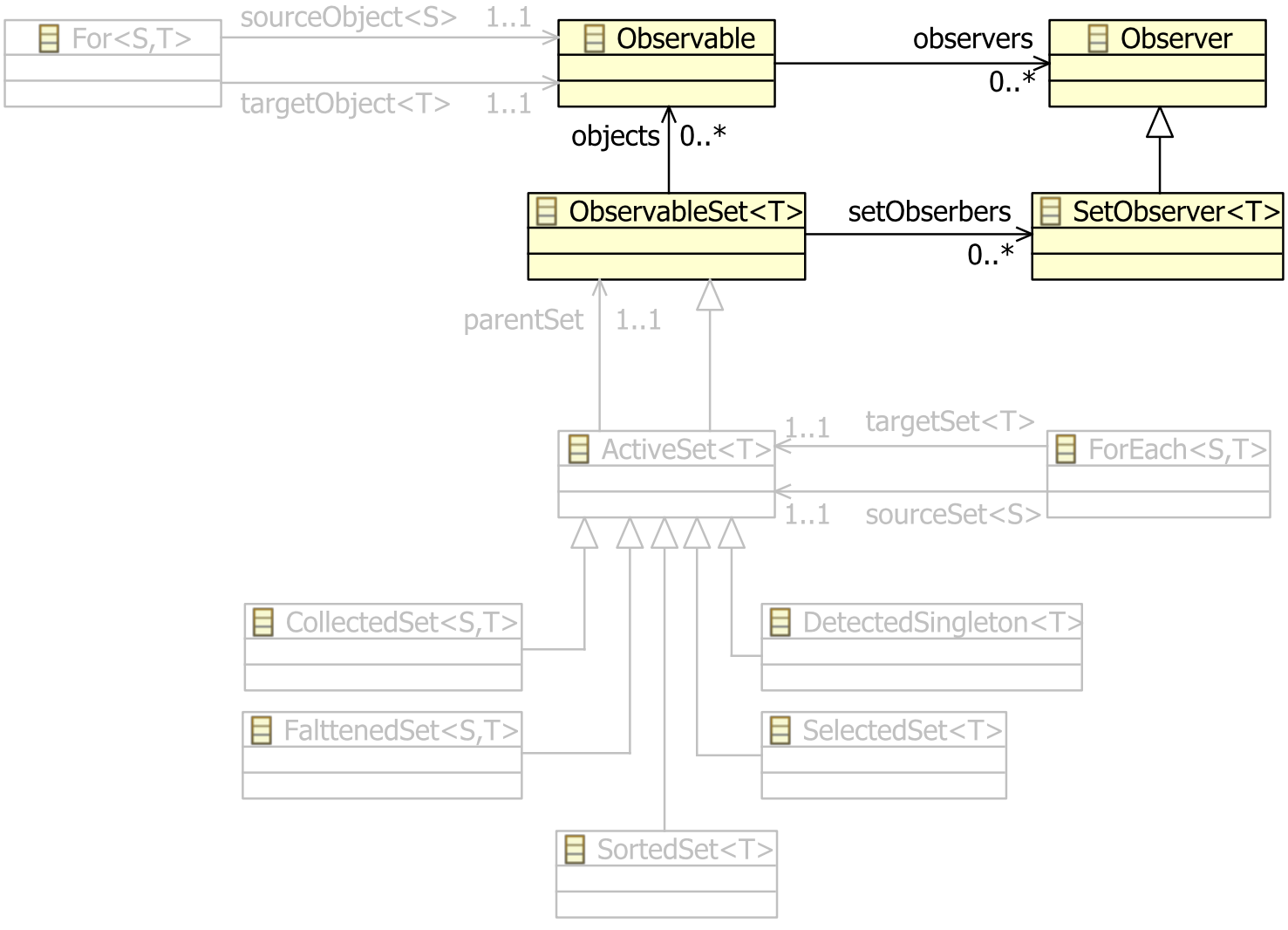
# Example (2)


agenda
AgendaCanvas
title : EString
eventStamps
0..*
EventStamp
type : EInt
day : EInt
time : EDate
duration : EInt
descriptions
0..*
timeLines
0..*
Description
text : EString
TimeLine
time : EDate

# Example (3)


ui
AgendaCanvasUI
title : EString
eventStamps
0..*
5..7
days
EventStampUI
color : EInt
type : EInt
x : EInt
y : EInt
width : EInt
heigth : EInt
index : EInt
counter : EInt
DayUI
name : EStr...
x : EInt
y : EInt
0..*
timeLines
TimeLineUI
hours : EString
minutes : EString
x : EInt
y : EInt
length : EInt
0..*
descriptions
DescriptionUI
text : EString
bold : EBoolean
italic : EBoolean

# Observables and Observable Sets (1)



# Observables and Observable Sets (2)

Base code:

```
class Week {
    attribute number: Integer
    attribute days: OrderedSet<Day>
}
```

# Observables and Observable Sets (2)

Base code:

```
class Week {
    attribute number: Integer
    attribute days: OrderedSet<Day>
}
```

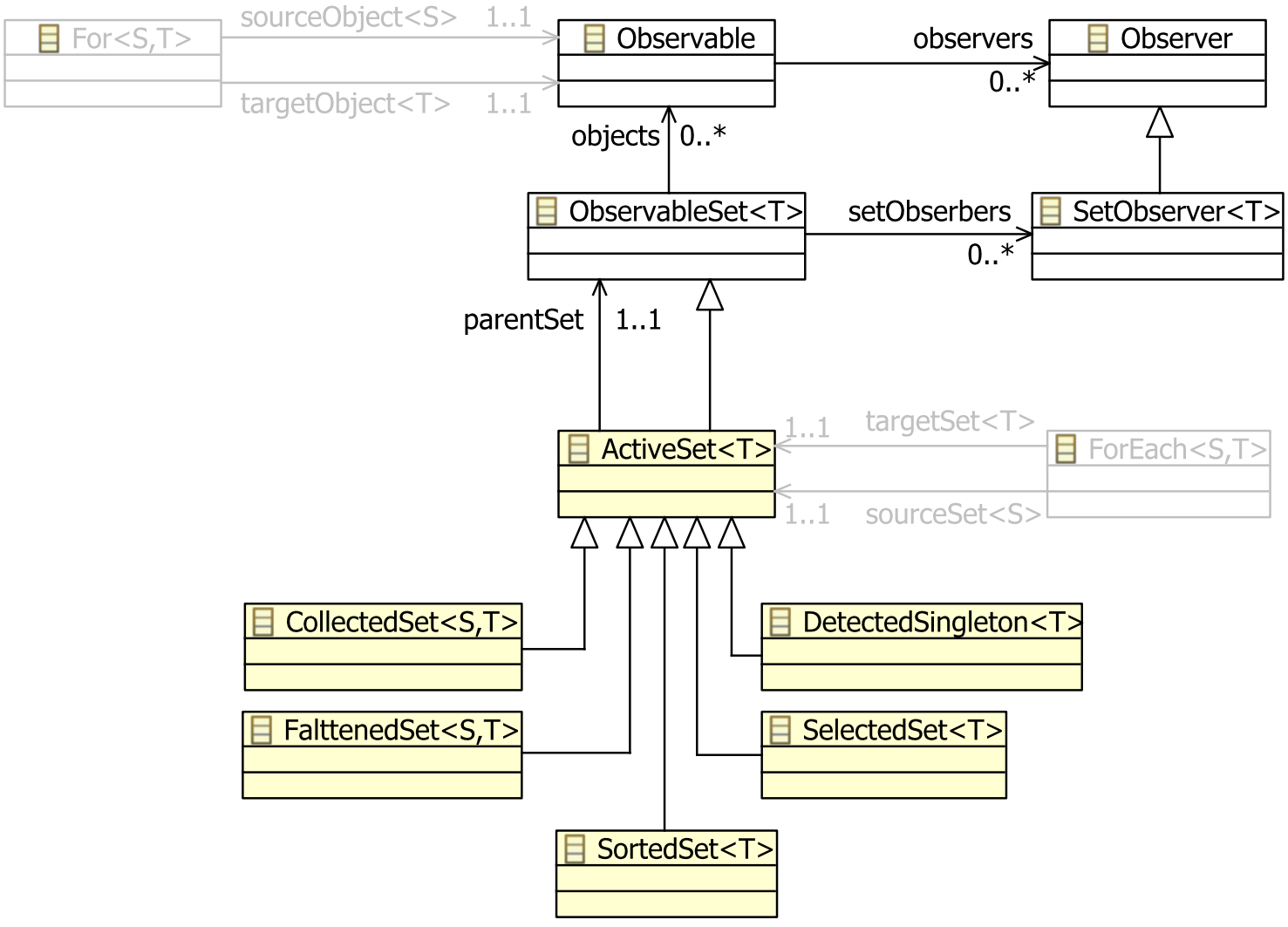
Generated code:

```
class Week inherits Observable {
    attribute _number: Integer

    property number: Integer
        getter is do result := _number end
      setter is do _number := value objectChanged() end

    attribute days: ObservableSet<Day>
}
```

# Active sets (1)



# Active sets (2)

- Select and collect:

```
var teachings: ObservableSet<Teaching>
var tutorials: CollectedSet<Tutorial>
tutorials := teachings
    .select{teaching | teaching.isKindOf(Tutorial)}
    .collect{tutorial | tutorial.asType(Tutorial)}
tutorials.eachAdded{index, tutorial |
    stdio.writeln(tutorial.topic.title) )
}
// populate 'teachings' afterward
...
```

# Active sets (3)

- Detect:

```
var weeks: ObservableSet<Week>
var week37: DetectedSingleton<Week>
week37 := weeks.detect{week | week.number == 37 }
...
```

# Active sets (3)

- Detect:

```
var weeks: ObservableSet<Week>
var week37: DetectedSingleton<Week>
week37 := weeks.detect{week | week.number == 37 }
...
```

- Flatten:

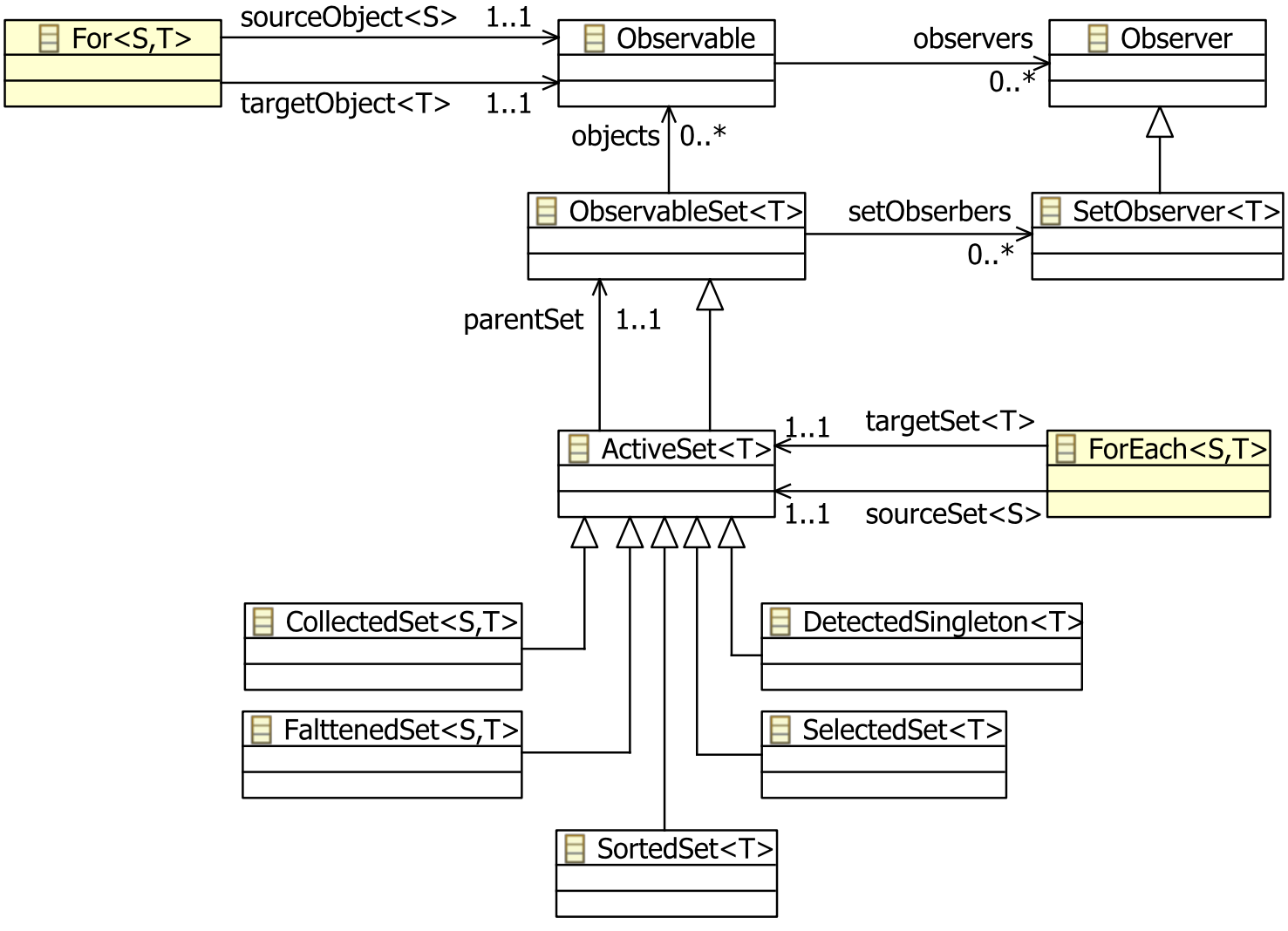
```
var week: Week
...
var courses: FlattenedSet<Course>
courses := week.days.flatten{day | day.courses}
// 'courses' represents 'week.days.courses'
...
```

# Active sets (4)

- Sort:

```
var teachers: ObservableSet<Teacher>
var sortedTeachers: SortedSet<Teacher>
sortedTeachers := teachers.sort{t1, t2 |
    t1.lastName < t2.lastName or
    (t1.lastName == t2.lastName and
     t1.firstName < t2.firstName)
}
...
```

# Mappings (1)


For<S,T>
sourceObject<S>
1..1
targetObject<T>
1..1
Observable
observers
0..*
Observer
objects
0..*
ObservableSet<T>
setObserbers
0..*
SetObserver<T>
parentSet
1..1
1..1
targetSet<T>
ActiveSet<T>
ForEach<S,T>
1..1
sourceSet<S>
CollectedSet<S,T>
DetectedSingleton<T>
FalttenedSet<S,T>
SelectedSet<T>
SortedSet<T>

# Mappings (2)

```
Week.new()
    .for(AgendaCanvas.new())
    .map{week, agenda |
        agenda.title := "Week " + week.weekNumber.toString()
        week.days
            .flatten{day | day.teachings}
            .forEach(agenda.eventStamps)
            .map{teaching, eventStamp |
                eventStamp.type :=
                    if teaching.isKindOf(Course) then 1
                    else if teaching.isKindOf(Tutorial) then 2
                        else 3
                        end
                    end
  eventStamp.descriptions.setAt(0, teaching.topic.title)
...
```

# Current Limitations

## Limites

- AcT.NET (IDM09) and AcT.Kermeta: only attributes and relations of *self* are observed.
- Example: *teaching.topic.title* is reevaluated only if the topic *instance* changes.

# Current Limitations

## Limites

- AcT.NET (IDM09) and AcT.Kermeta: only attributes and relations of *self* are observed.
- Example: *teaching.topic.title* is reevaluated only if the topic *instance* changes.

## Solution

- Mapping codes must be parsed.
- AcT.Kermeta: by using the Kermeta interpreter

# Contribution

- AcT.Kermeta: Active OCL sets $\Rightarrow$ active transformations
- Kermeta is both the MM platform and the running platform for active transformations
- Currently limited to "instance-to-instance" and "relation-to-relation" mappings
- Web site (alpha version): *http://gri.eseo.fr/software/act/kermeta*

# Perspectives

- A DSL dedicated to mappings (textual and graphical)
- A meta-model of active transformations + its implementation (will use active sets)
- Interaction model (based on instruments)
- Complex mappings (*e.g.* information visualization)
- Groupware (based on «event points»)
- Other application domains than GUI

# Questions?