# BPMN and domain modeling

Complaint process example

# BPMN and data

- **In general, BPMN has weak support for data:**
  - data objects indicate that activities require and/or provide data
  - data objects can be associated with connections
  - the data types must be defined in external XML schemas

**Table 7.2 - BPMN Extended Modeling Elements**

| | | |
|---|---|---|
| Data Object | Data Objects provide information about what Activities require to be performed and/or what they produce (see page 205), Data Objects can represent a singular object or a collection of objects. Data Input and Data Output provide the same information for Processes. | **Data Object**<br>**Data Objec (Collection)**<br>Data Input Data Output |
| Data Store | Label | |

# We need a method for systematically annotating BPMN diagram with information about data (model) usage

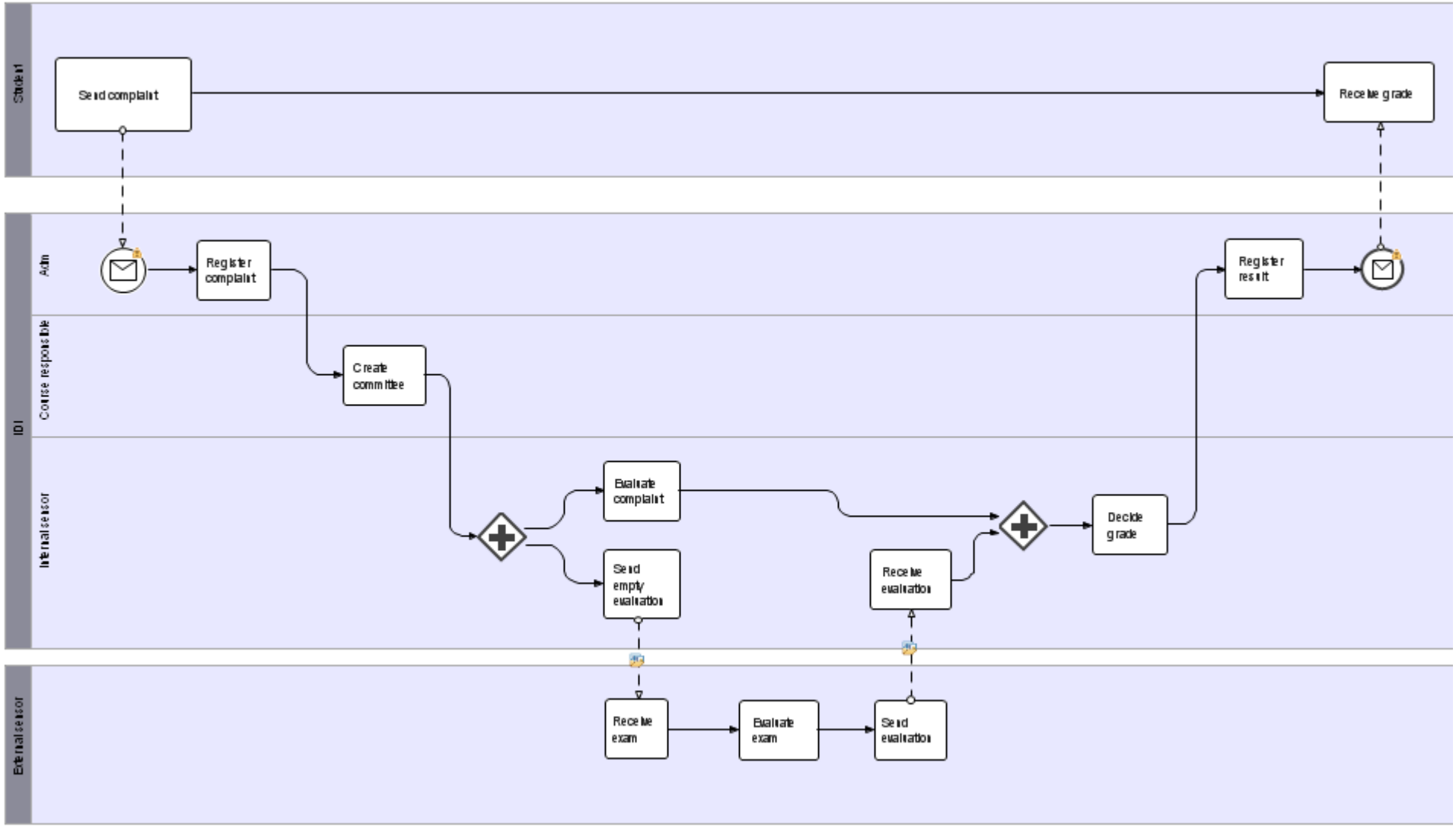
# Overview of method – BPMN and domain data

- **Create BPMN model of process**
  - identify roles, tasks, sequence constraints and messages
- **Create a domain model (is provided in the assignment)**
  - identify important concepts, associations and attributes
  - cardinalities are important: 0..1 and 0..*
  - ecore extras:
    - in an ecore model, every object must be (in)directly contained by a single root object, through aggregations. This may require the introduction of a root class, which I often call UoD for Universe of Discourse
    - add extra opposite associations to ensure the model may be navigated in the relevant directions
- **Annotate the BPMN model with domain elements**
  - describe what domain objects that a task requires (pre-condition)
  - describe how tasks create and change the state of domain objects (post-condition)
  - identify what information each message contains
  - the goal is to be explicit about the contextual information each task assumes (pre-condition) and provides (post-condition)

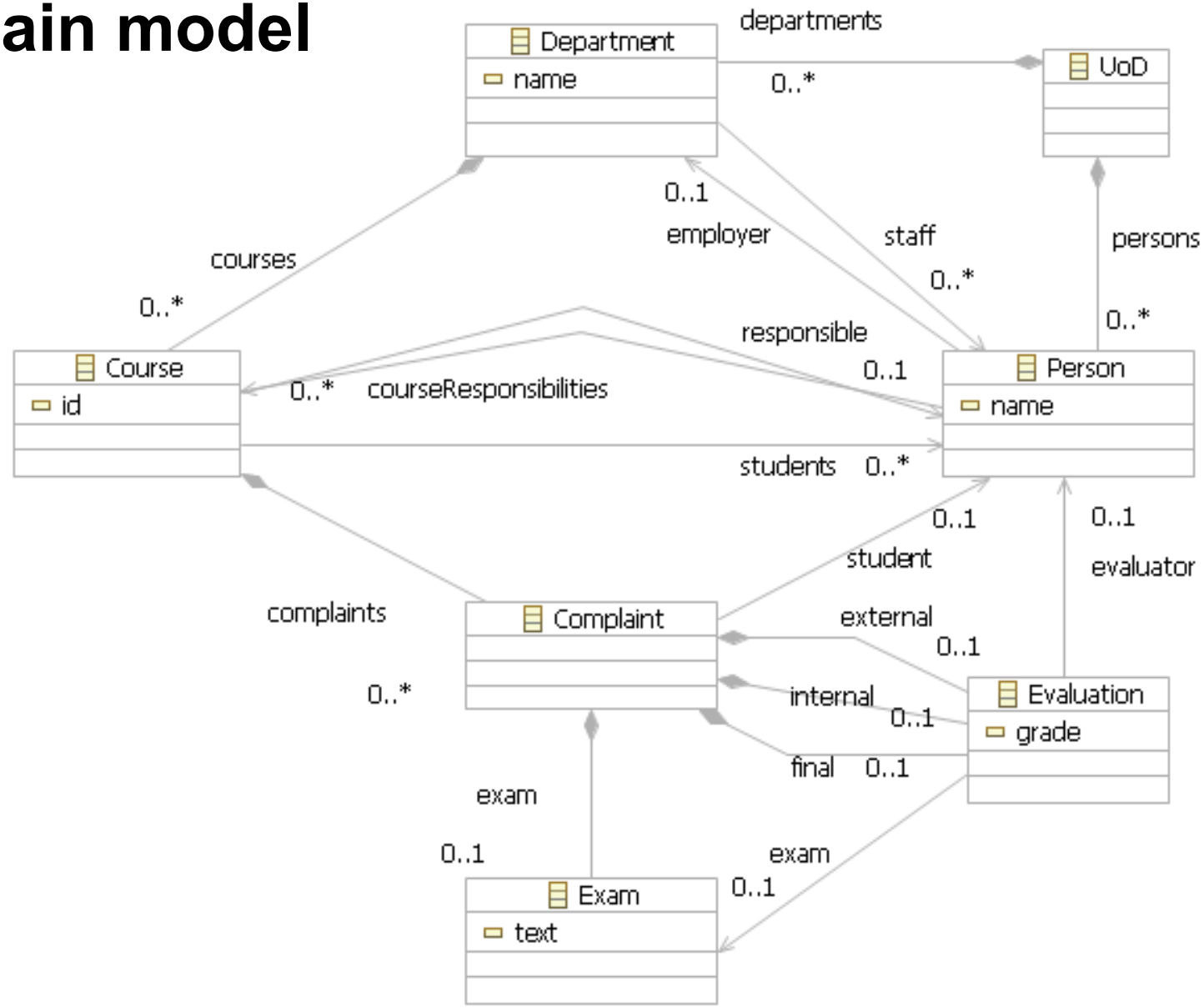
# Example: the exam complaint process

- Students take exams for each course they attend
- They may complain on the grade, by contacting the administration.
- The course responsible appoints a committee, consisting of an internal and an external evaluator.
- The administration sends (a copy of) the student's exam to the external evaluator, who evalutes the exam and sends back the result.
- The internal evaluator evaluates the exam and, based on the evaluation received from the external evaluator, decides upon the final grade.
- Finally, the administration registers the results and sends it to the student.

# Example: the exam complaint process



# The domain model

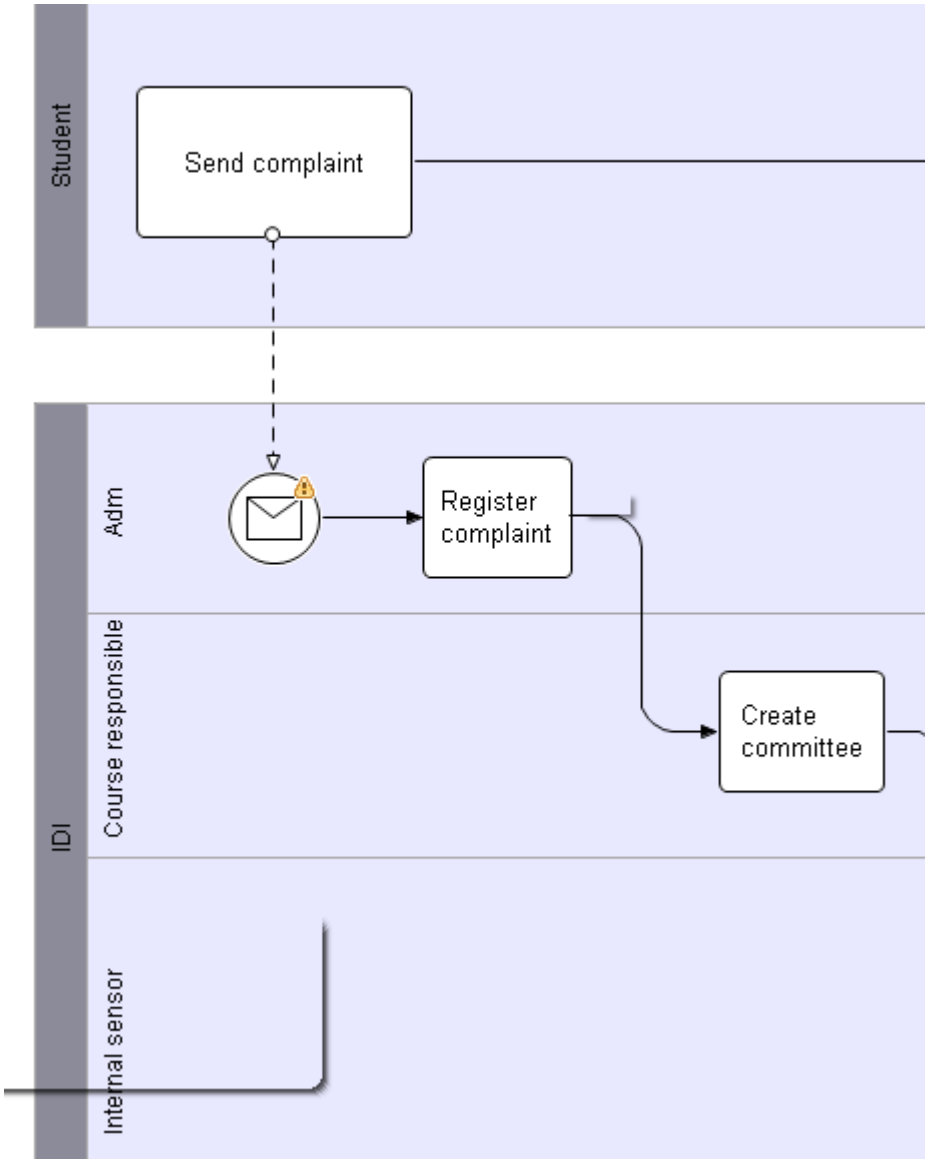


# Reflections about the domain model

- UoD is added as the root object and many assocations are aggregations, to ensure that every object is (in)directly contained in the UoD
- Several of the roles in the process are also present as classes and/or associations in the domain model
- Opposite associations have been added, to support navigating in the model in all relevant directions
- The lower limit of all associations are set to 0 and not 1, to allow intermediate, illogical states

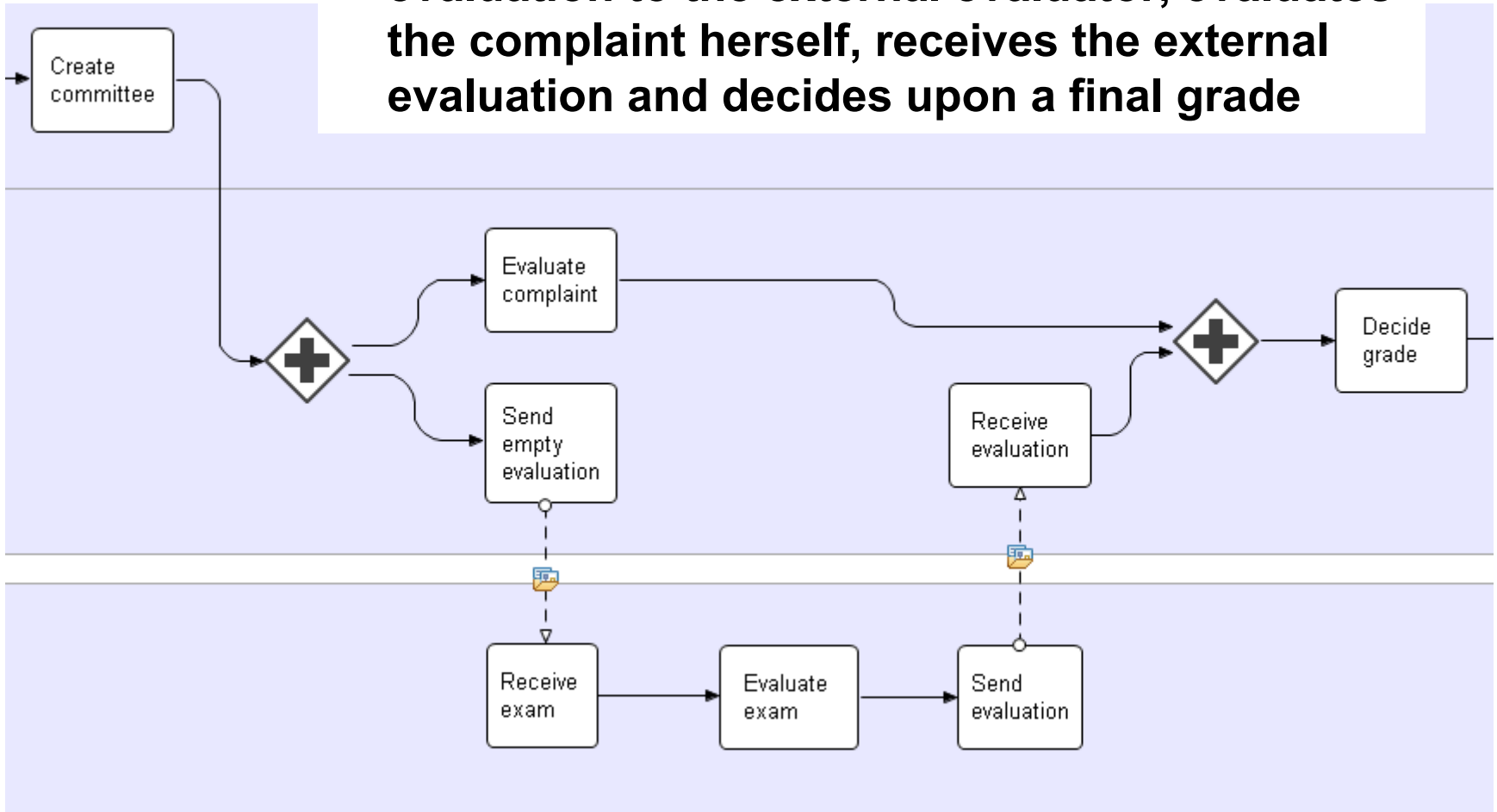
# The complaint process

- **A student sends a complaint,**
  - creates a Complaint object
- **which is registered by the adm.**
  - relates an Exam to the Complaint
- **The course responsible creates the committee,**
  - Persons are related to the Complaint through Evaluation objects and evaluator links
- **and hands the complaint over to the internal sensor**



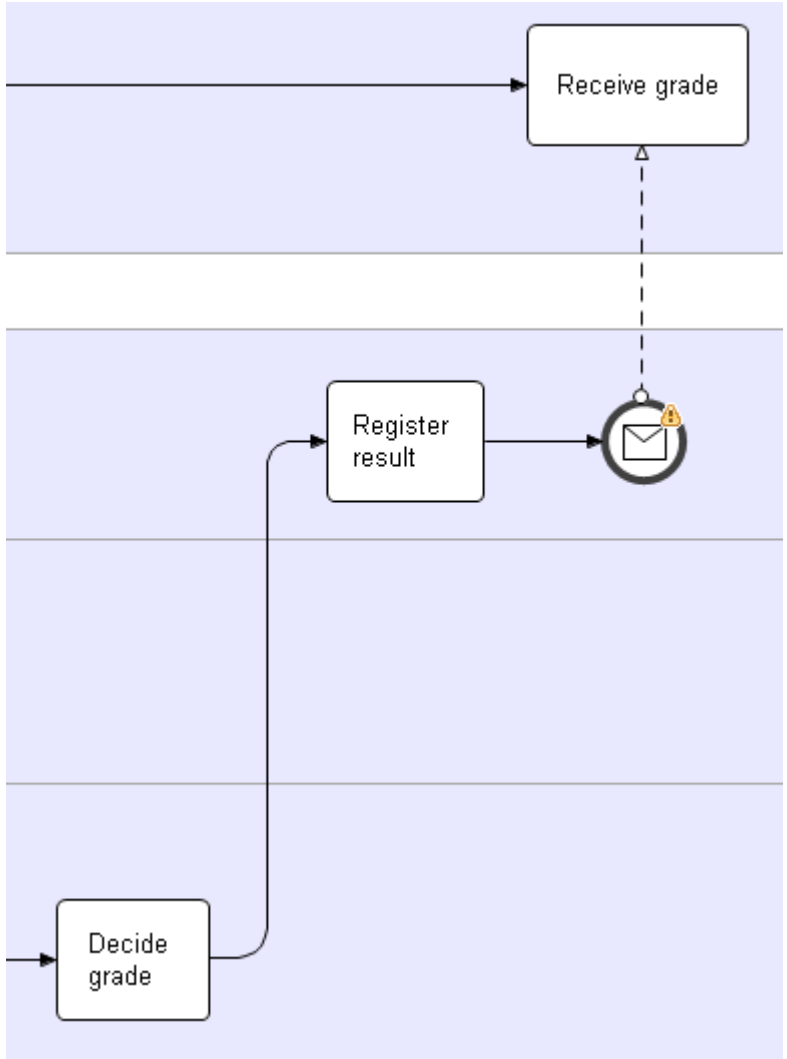
# The complaint process

- **The internal evaluator sends the empty evaluation to the external evaluator, evaluates the complaint herself, receives the external evaluation and decides upon a final grade**



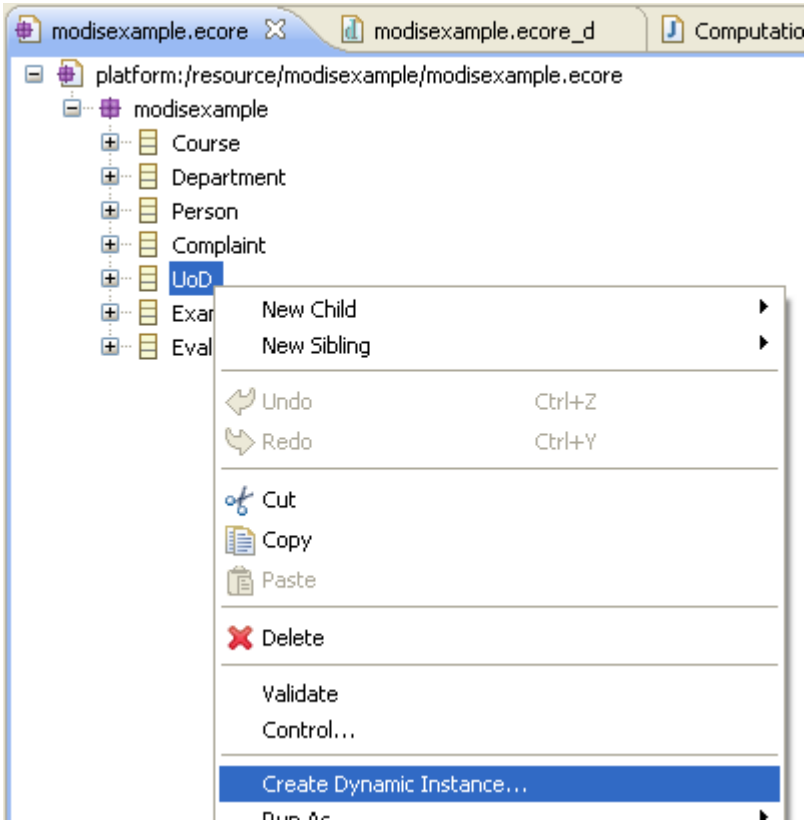
# The complaint process

- **The final evaluation is registered by the administration and sent to the complaining student**



# Test data

- **create xmi data file (an empty one will be provided for the assignment)**
  - open ecore file
  - open action menu on root class
  - Create Dynamic Instance... creates the data file with a root object of the desired class
  - name it <modelname>.xmi
- **build an object structure with relevant objects**



# Test data

- use the New child action to create new objects under a parent
- create relevant test data
- make sure to create links and enter property values