AI*IA 2007 & WOA 2007

Curricula Modeling and Checking

M. Baldoni, C. Baroglio, G. Berio, and E. Marengo

Dipartimento di Informatica – Università degli Studi di Torino
c.so Svizzera, 185, Torino (Italy)

Automatic composition of learning resources

- Greater and greater availability of learning resources through the web
- Learning resources must be combined for allowing users to acquire some knowledge

- Compositions should be verified to check:
  - Achievement of the user's learning goal
  - Compliance to the course-design goals
  - Competency gaps

Verifications

- **Competency gaps**: whenever I use a learning resource, all the knowledge that is necessary to understand its content is available (either from the beginning or because supplied by a previous resource)
- **Learning goal achievement**: a curriculum is designed by a student with the aim of gaining some knowledge
- **Course-design goal achievement**: the curriculum satisfies the specification made at design time

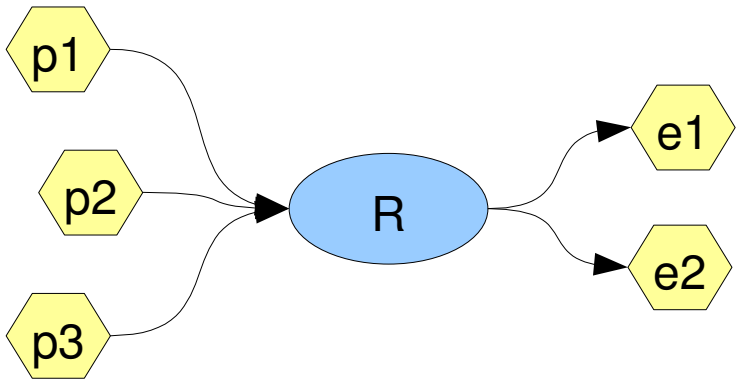
Aim of the work: *to support the automatic verification of resource organizations, by checking competency gaps, the achievement of the user's learning goal, and the satisfaction of the course design goals*

Competences and Resources

- In this perspective we distinguish between **competencies** and **resources**
  - **Competency**: *a piece of knowledge*, e.g. knowledge that is supplied or that is required to attend a curriculum
  - **Resource**: *an item that either requires or supplies competencies*, e.g. learning materials or courses

Resources/courses as actions

- Learning resources are modeled in an *action-based* way:
  - a resource has a *precondition* and an *effect*
  - **precondition**: a set of competences that are considered as necessary to understand the topics taught by means of the resource
  - **effect**: a set of competences that are supplied by the resource

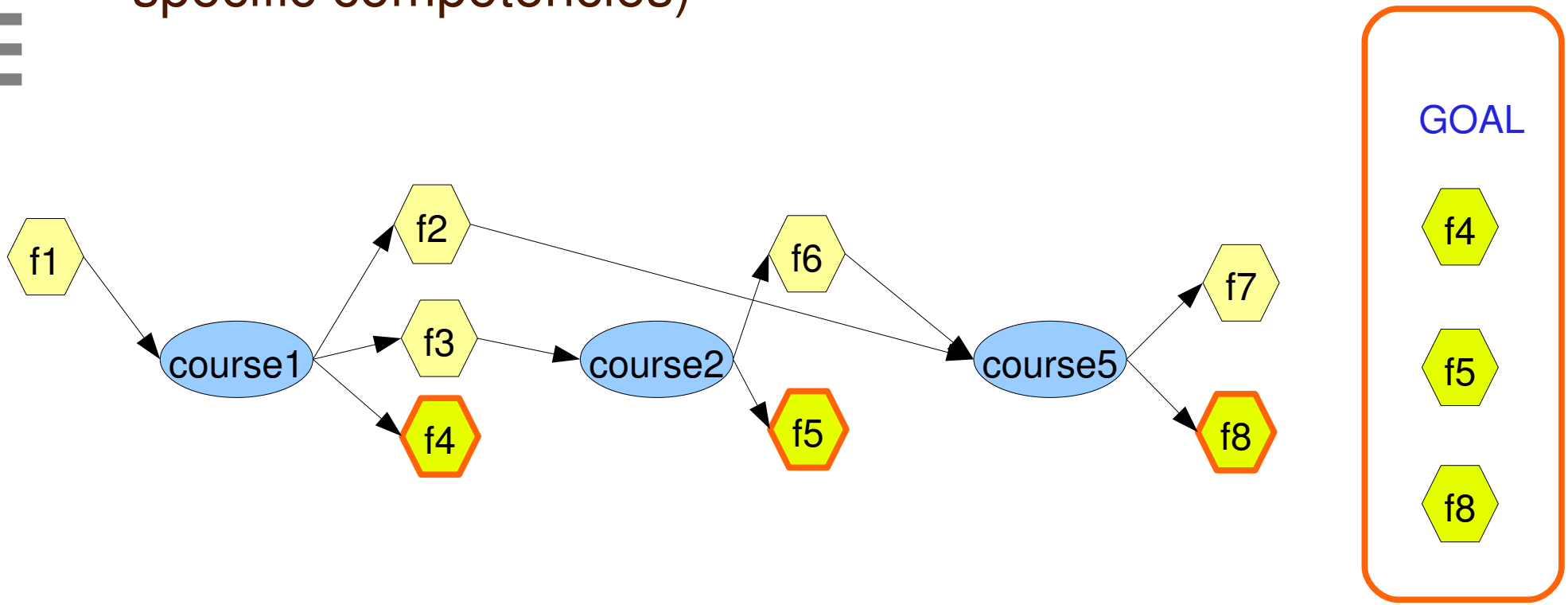


PRECONDITION

EFFECT

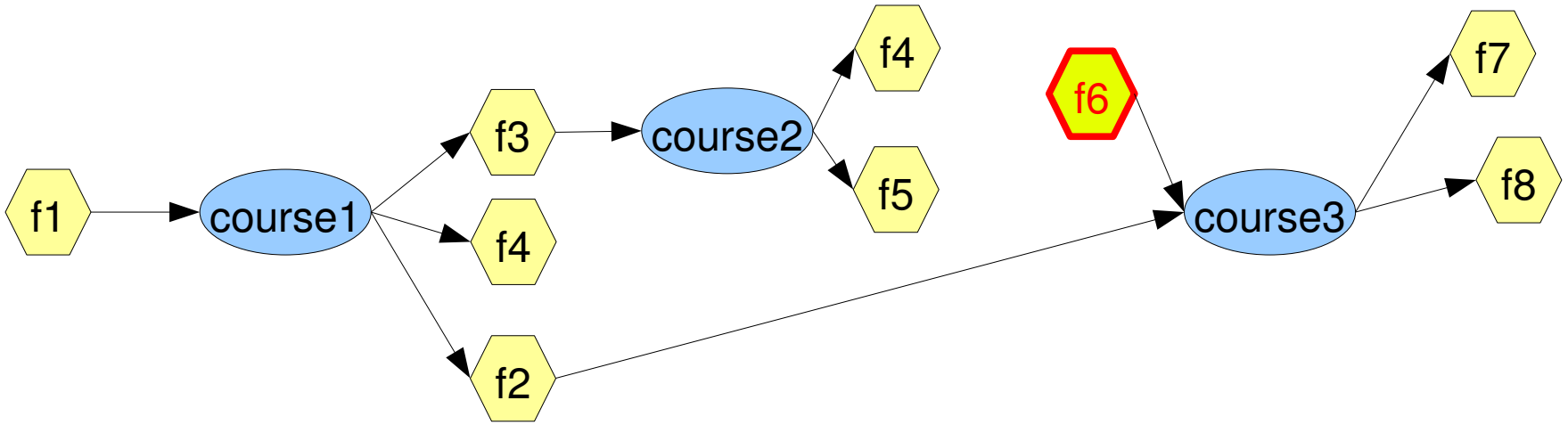
Curricula

- Intuitively, a **curriculum** is a sequence of resources aimed at pursueing a learning goal (i.e. to acquire specific competencies)



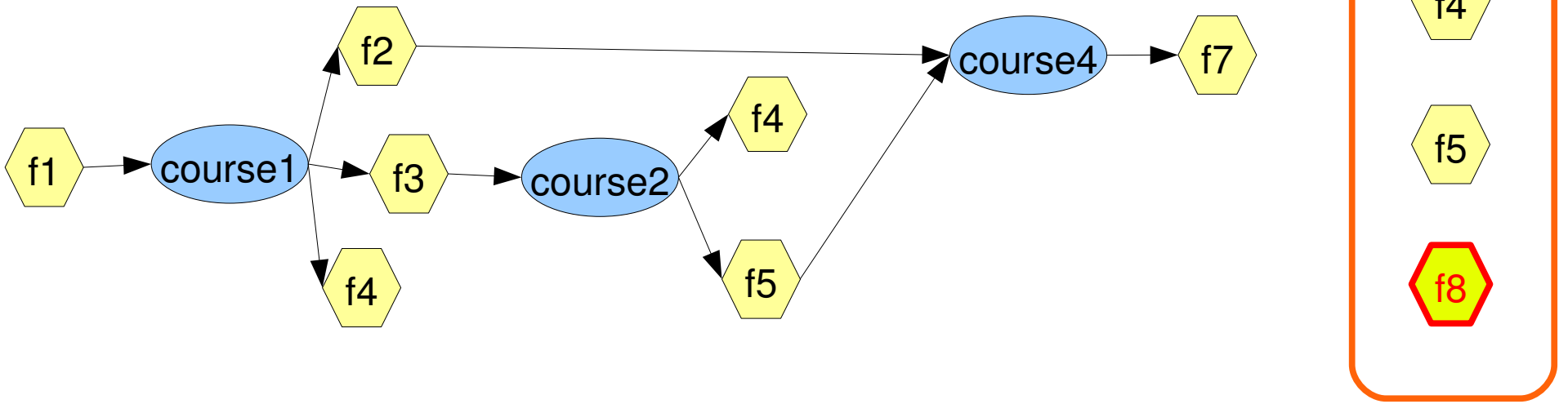
Soundness of a curriculum

- A curriculum is sound if it shows no ***competency gap*** and it allows the achievement of the *user's learning goal*
- By reasoning on resources as actions, it is easy to perform these verifications on a given curriculum
- Distinguishing between competencies and resources supports the openness of systems



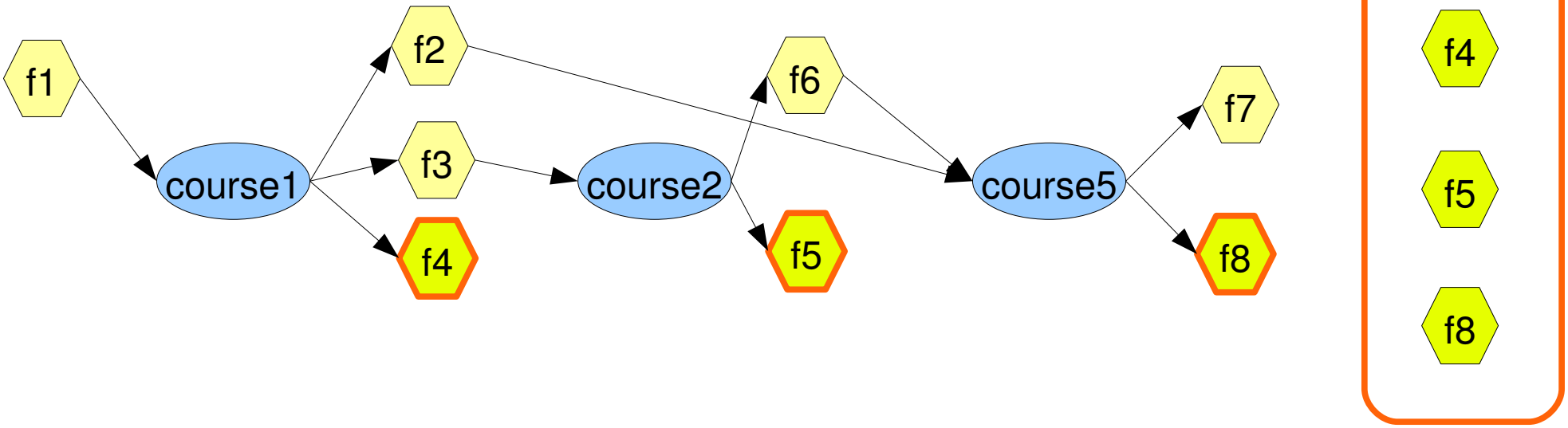
Soundness of a curriculum

- A curriculum is sound if it shows no *competency gap* and it allows the achievement of the ***user's learning goal***
- By reasoning on resources as actions, it is easy to perform these verifications on a given curriculum
- Distinguishing between competencies and resources supports the openness of systems



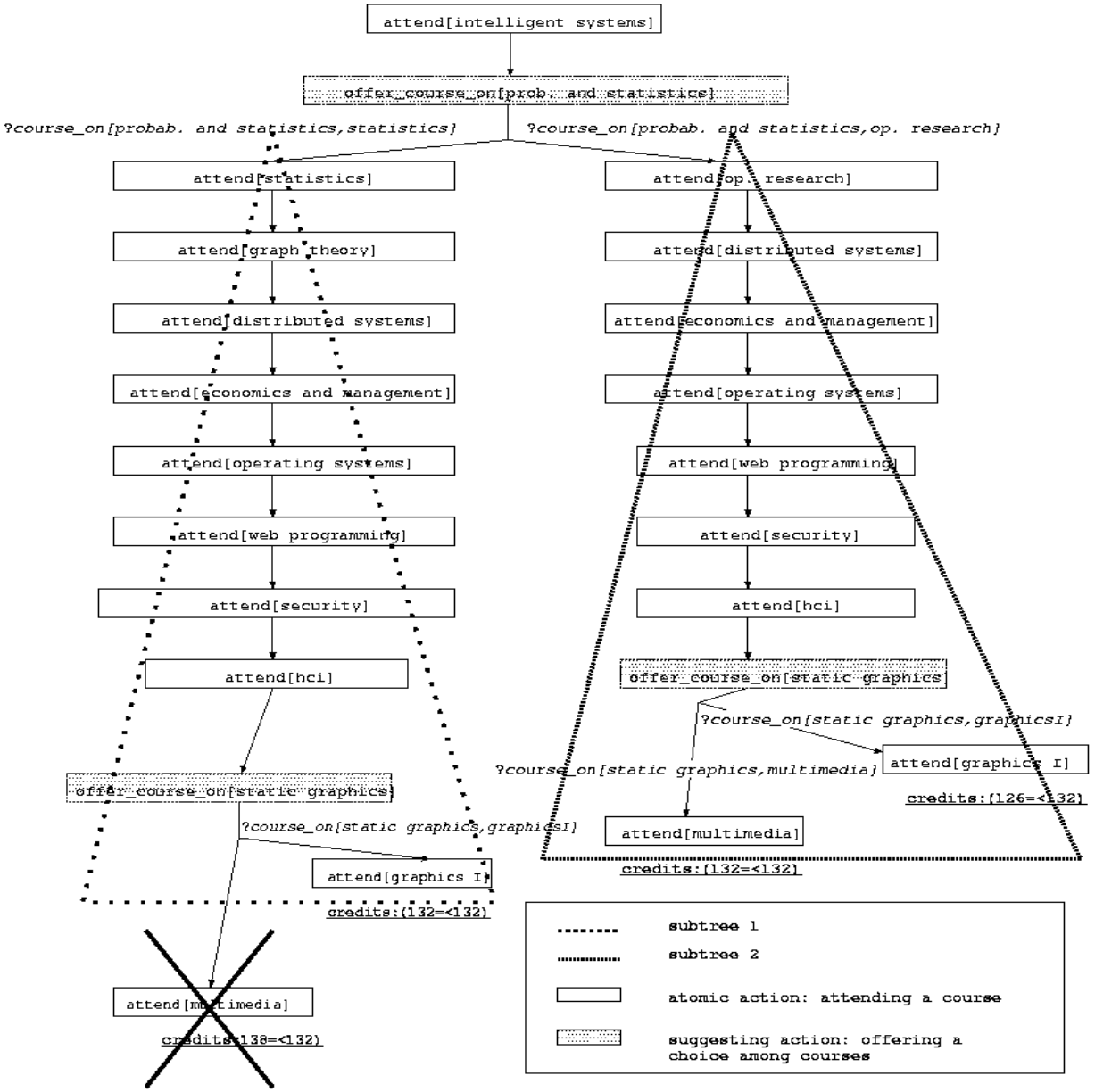
Soundness of a curriculum

- A curriculum is sound if it shows no *competency gap* and it allows the achievement of the *user's learning goal*
- By reasoning on resources as actions, it is easy to perform these verifications on a given curriculum
- Distinguishing between competencies and resources supports the openness of systems



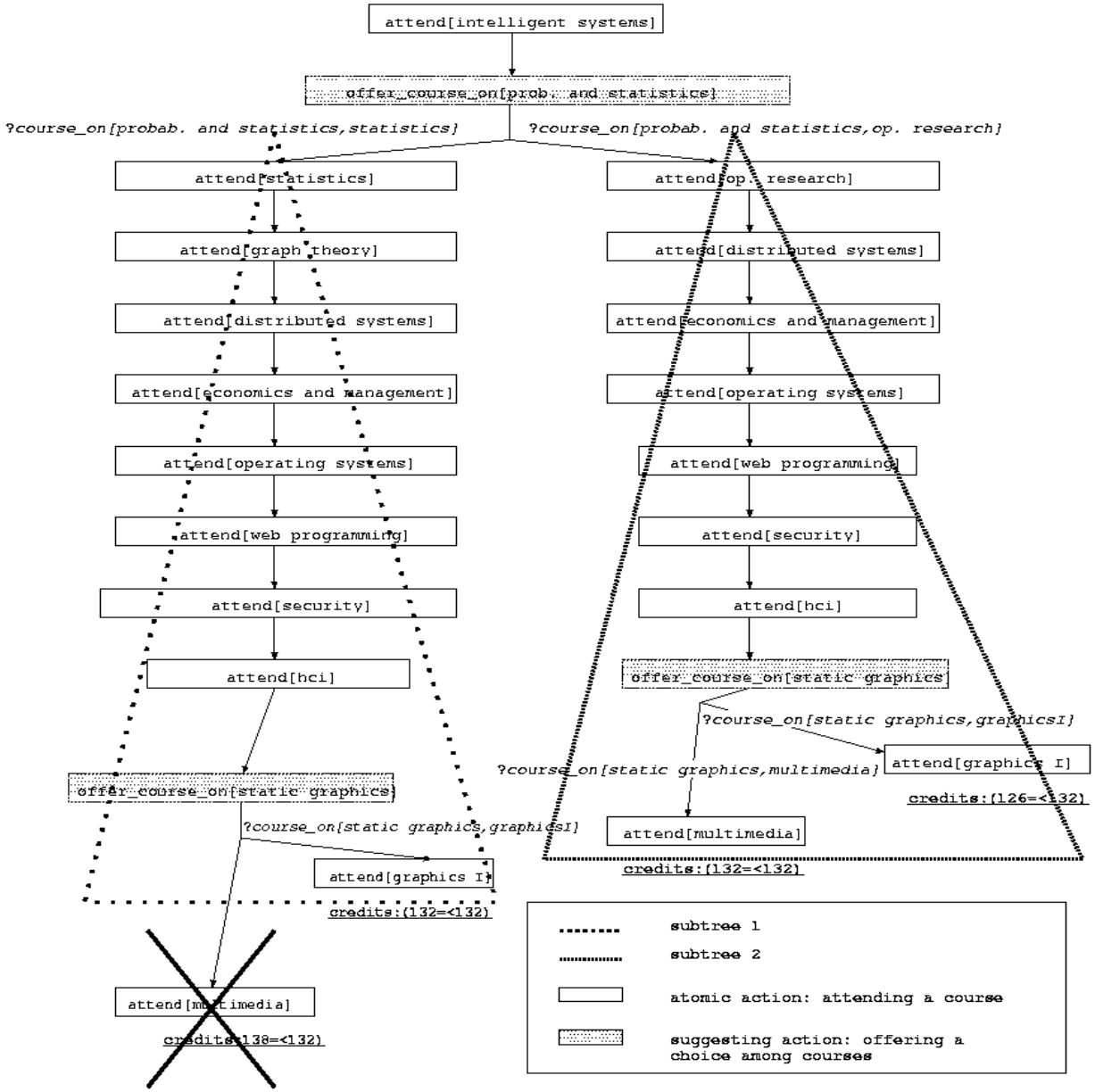
Compliance to course design

- Curricula should match the “design document” specified by the institution which offers the courses/resources
- E.g. in previous work prolog-like clauses used to capture the structure of a curriculum, based on competencies



Compliance to course design

- In open domains, resources are added, modified, ...
- Specification of the design document: *generative definition* of allowed curricula
- *Prescriptive solution!* Not adequate: whatever is not specifically foreseen is illegal



Compliance to course design

- The idea explored in this work is to define only the *necessary constraints*
- Inspired by Singh's *social approach* for representing interaction protocols
- **Curricula model**: it specifies the properties that the curricula (proposed either by the students of some organization or by the organization itself) should satisfy

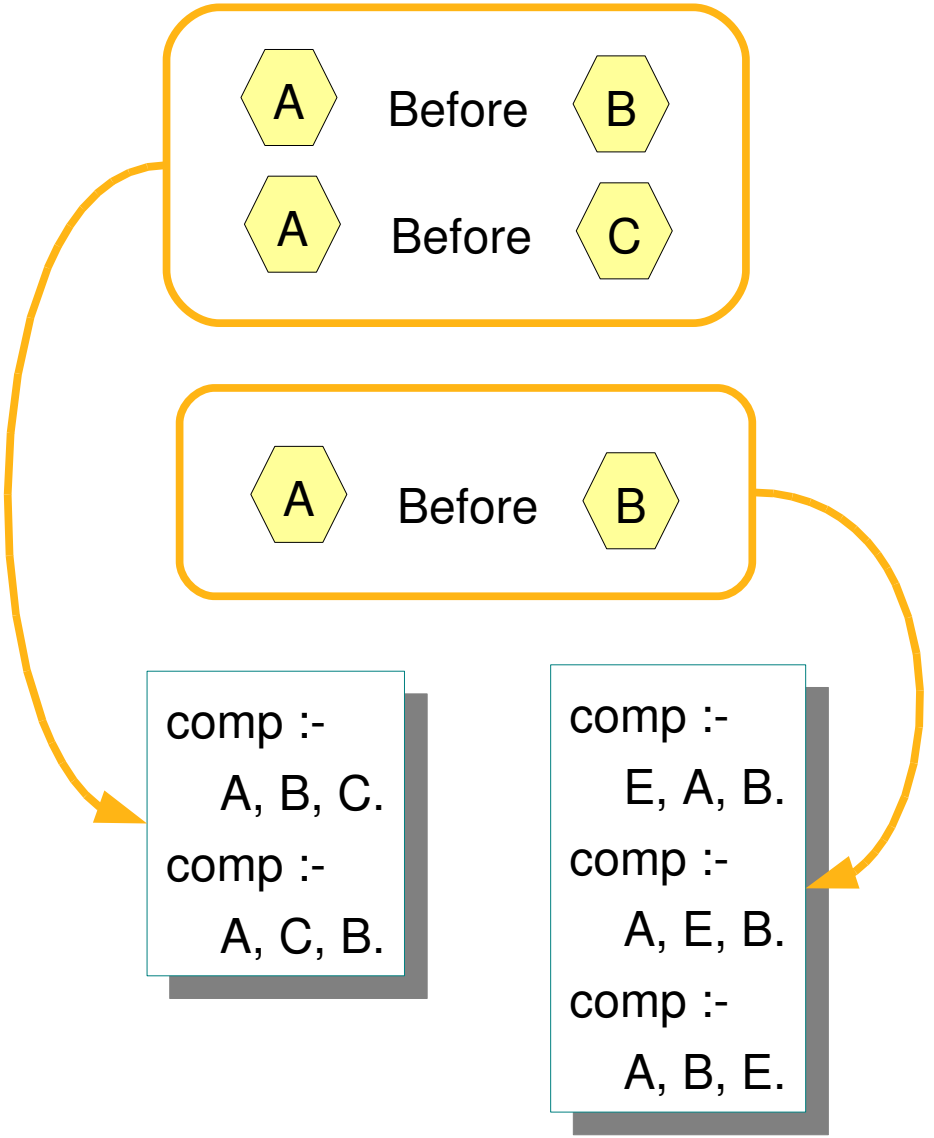
- **Advantage**: it is not necessary to force the specification of all the legal sequences of courses, avoiding overspecification

Constraint vs. procedural representation

- Goal = {A, B, C}
- Possible solutions:
  A B C, A C B

- Goal = {A, B, E}
- Possible solutions:
  E A B, A E B, A B E

- A procedural representation should include all the sequences with a non-deterministic choice

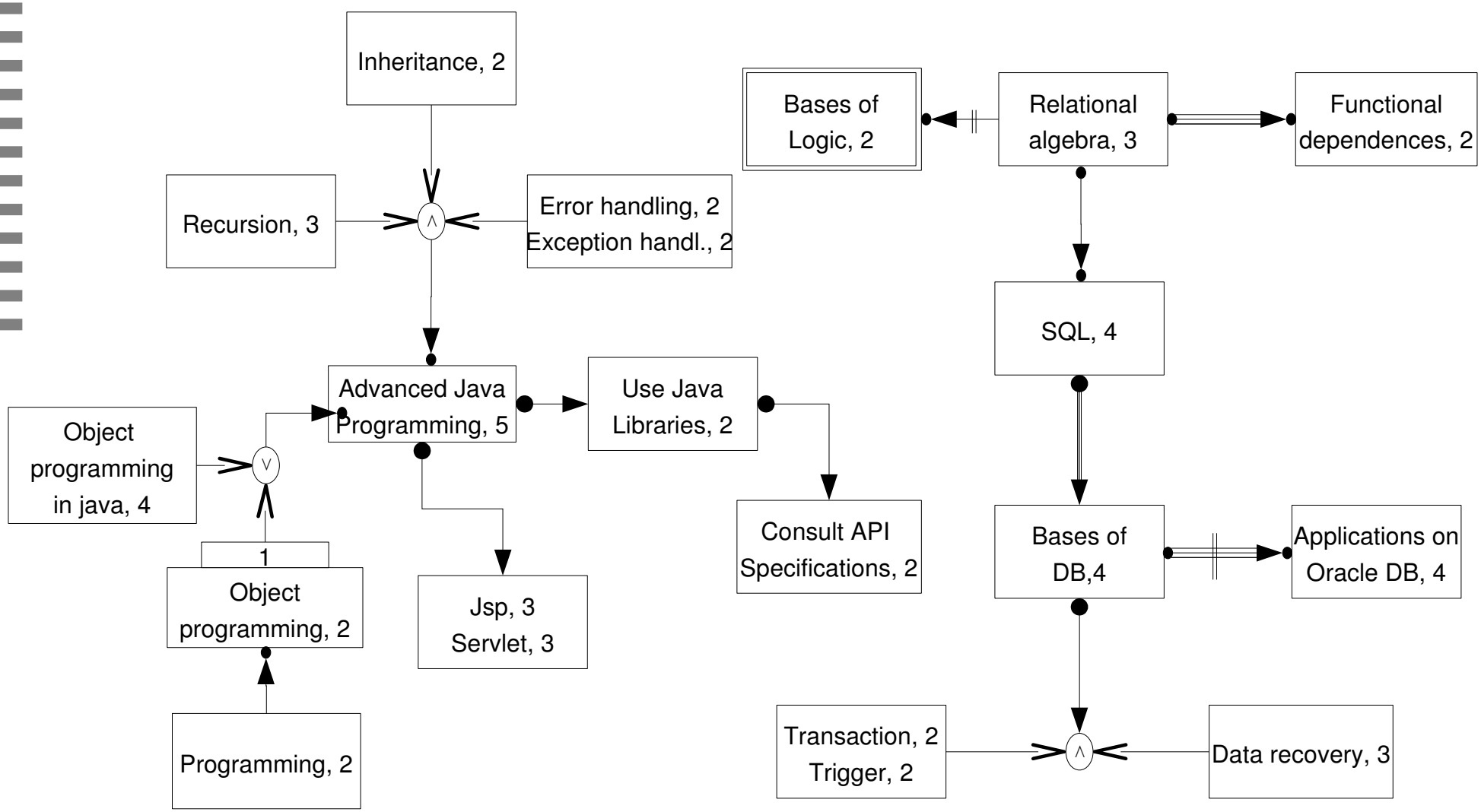


DCML

- **DCML** (Declarative Curricula Model Language) is a graphical language grounded in *linear temporal logic* (LTL)
- It supplies primitives for representing competences (competencies plus proficiency level) and various kinds of constraints

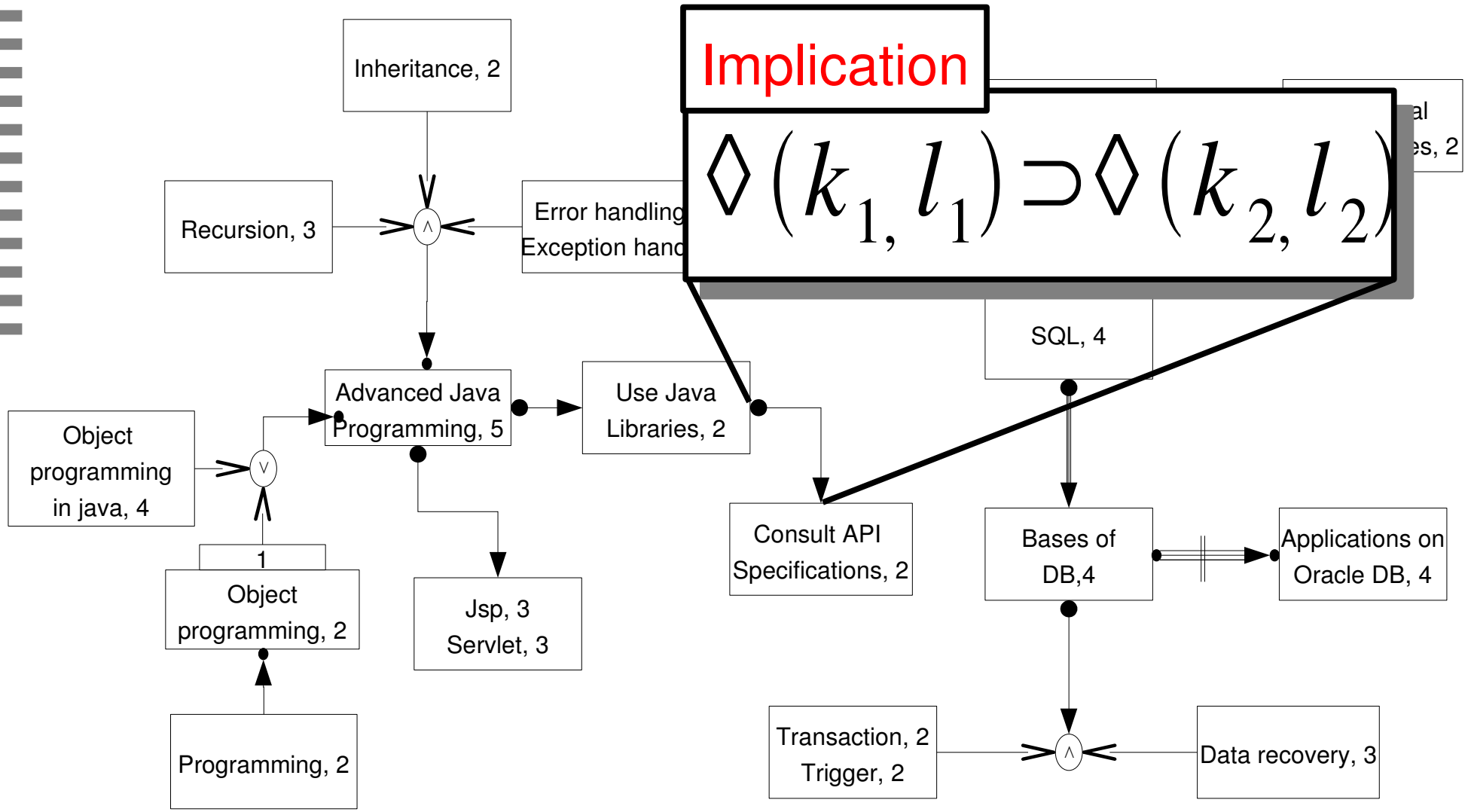
- *The graphical notation facilitates the designer, who does not need to be an expert of temporal logic notations*
- *The logical grounding enables automatic forms of verification*

DCML at a glance

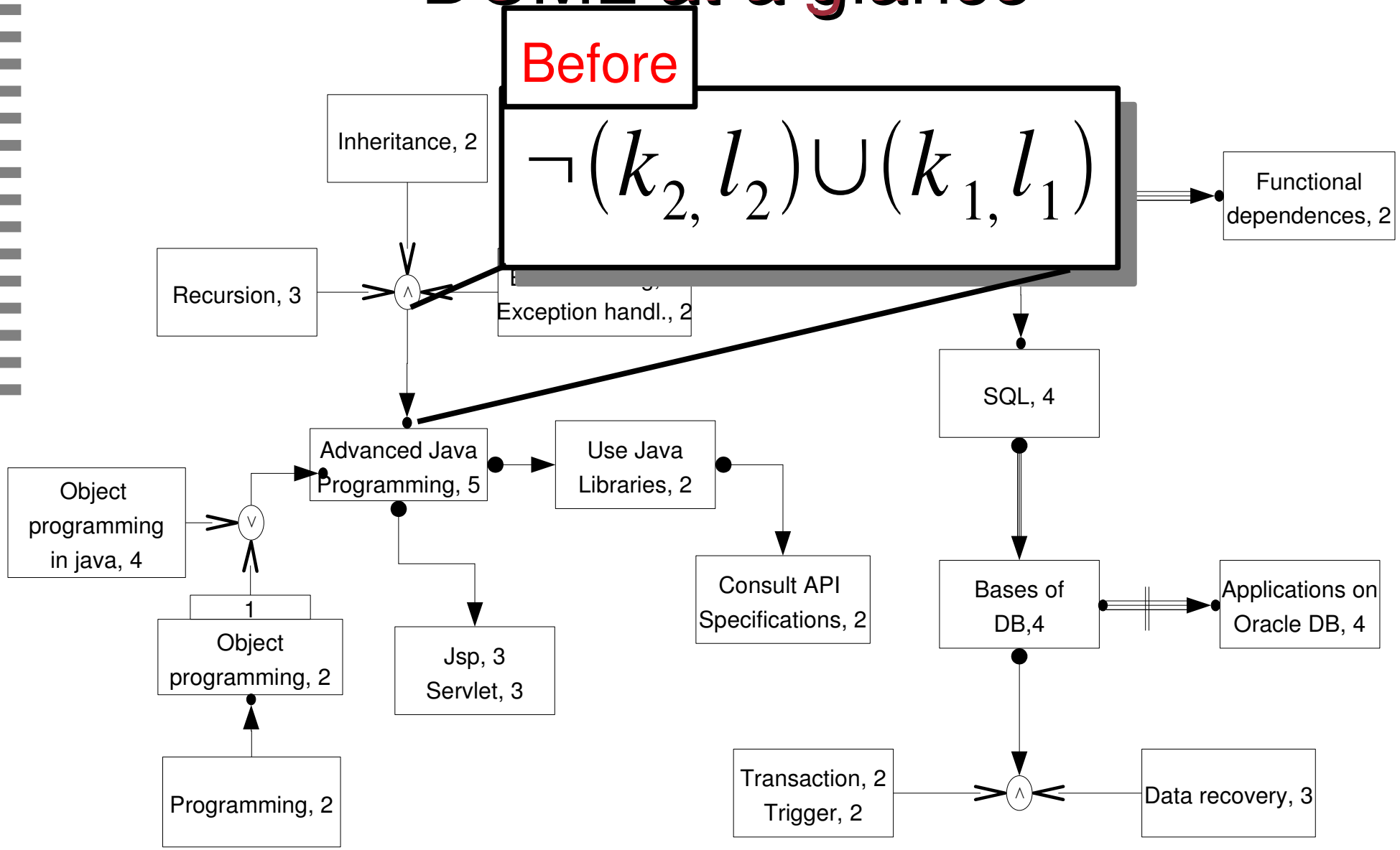


Competences for University courses of a Computer Science curriculum and their dependencies

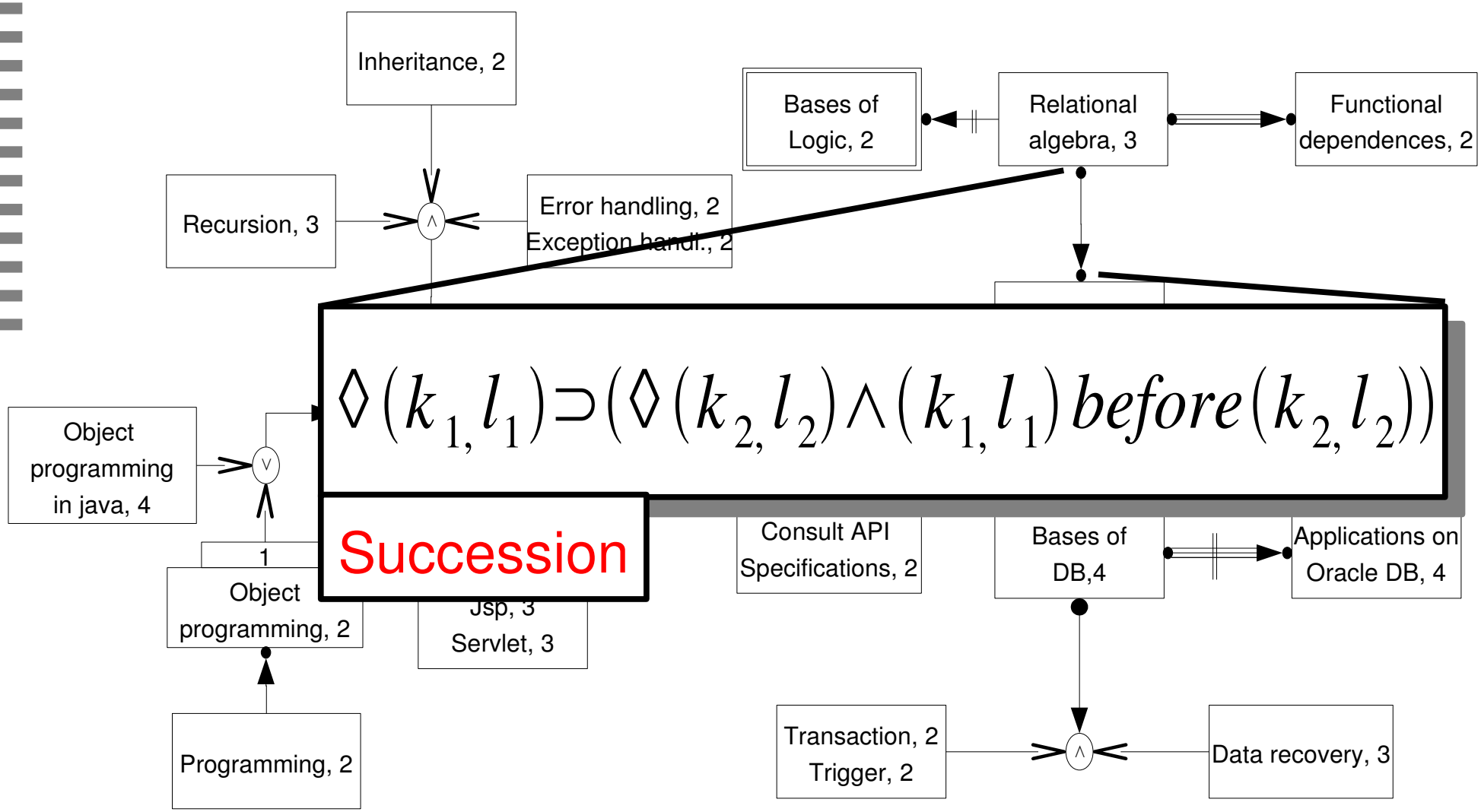
DCML at a glance



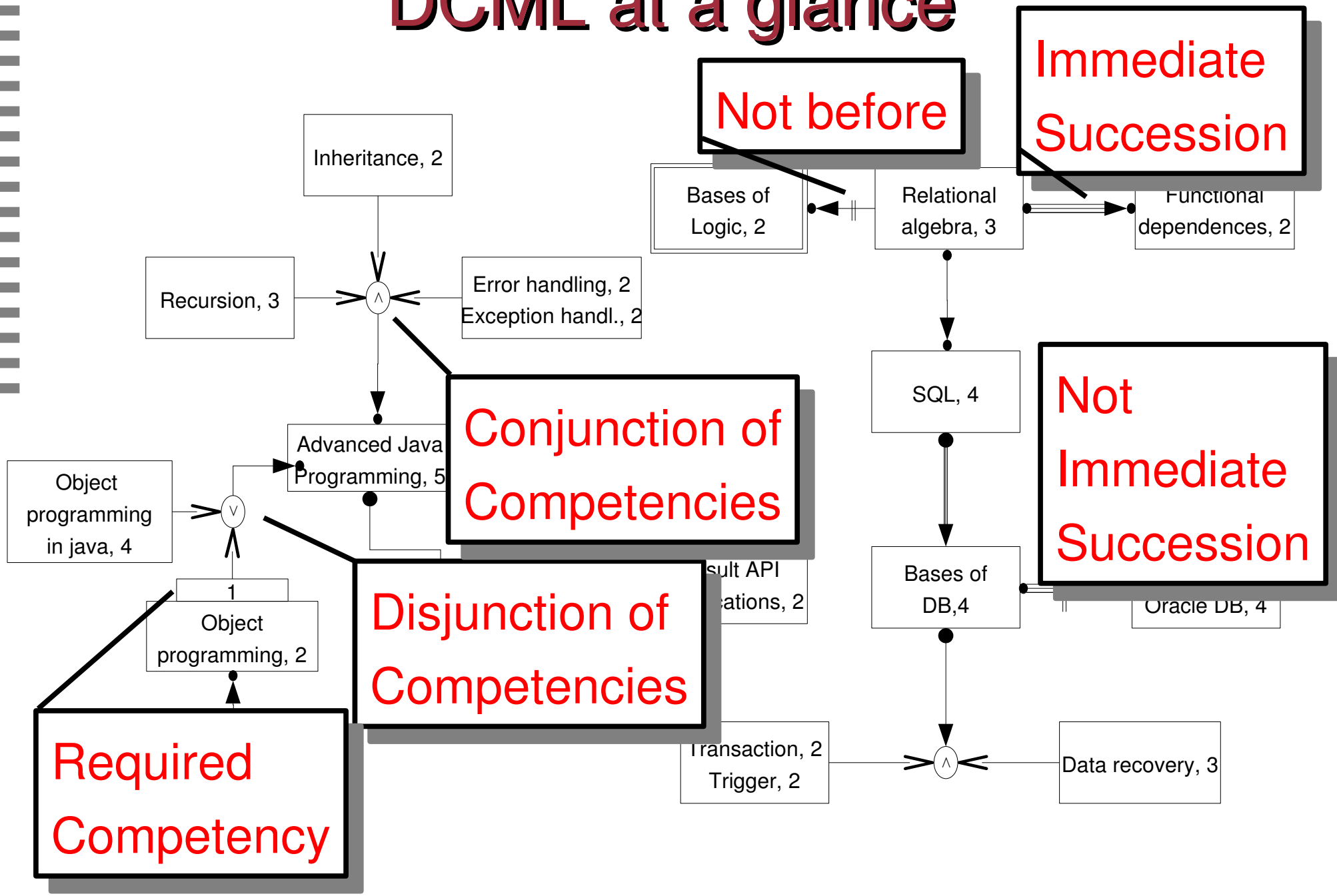
DCML at a glance



DCML at a glance



DCML at a glance

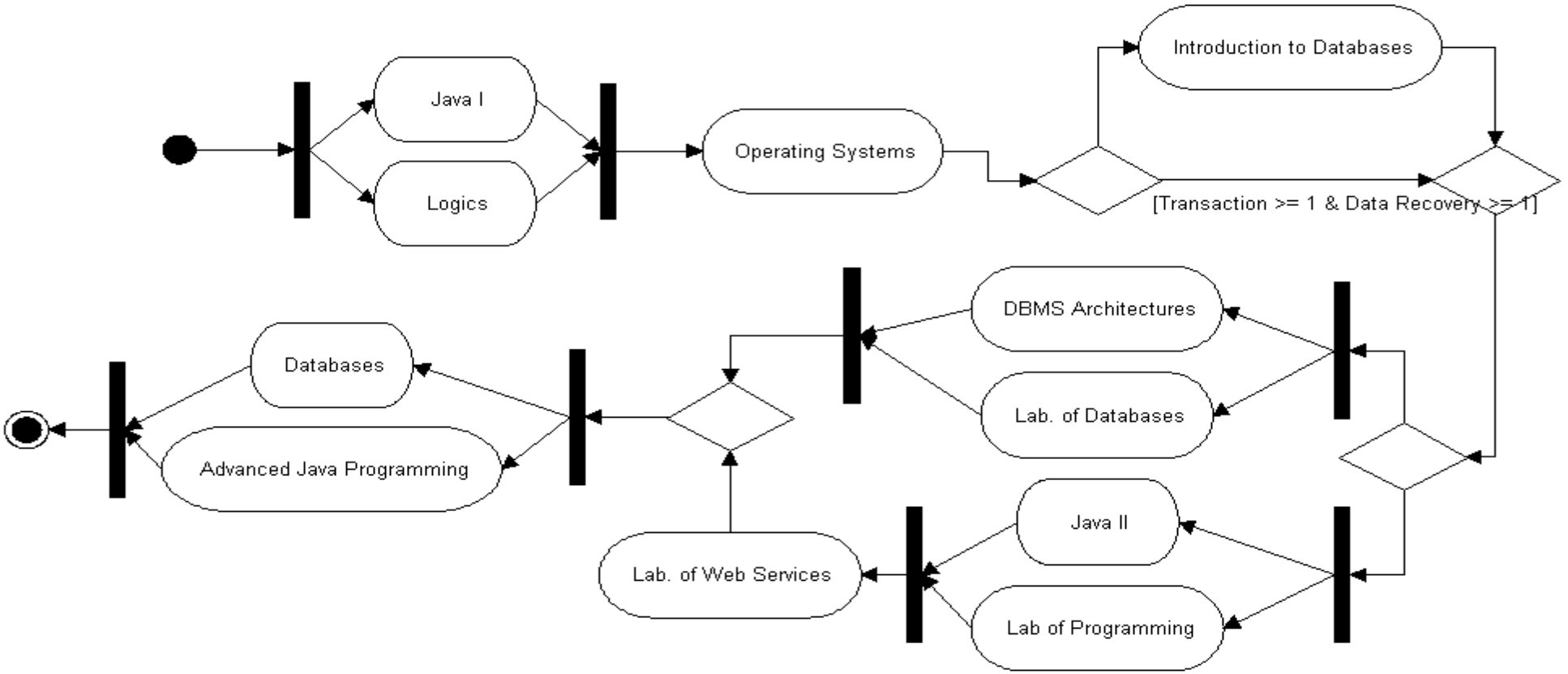


Curricula representation

- How to represent curricula?
- We need a representation that includes:
  - Resources
  - Decision points
  - Alternatives
  - Mandatory and non-mandatory resources
- We can represent curricula by means of **UML activity diagrams**

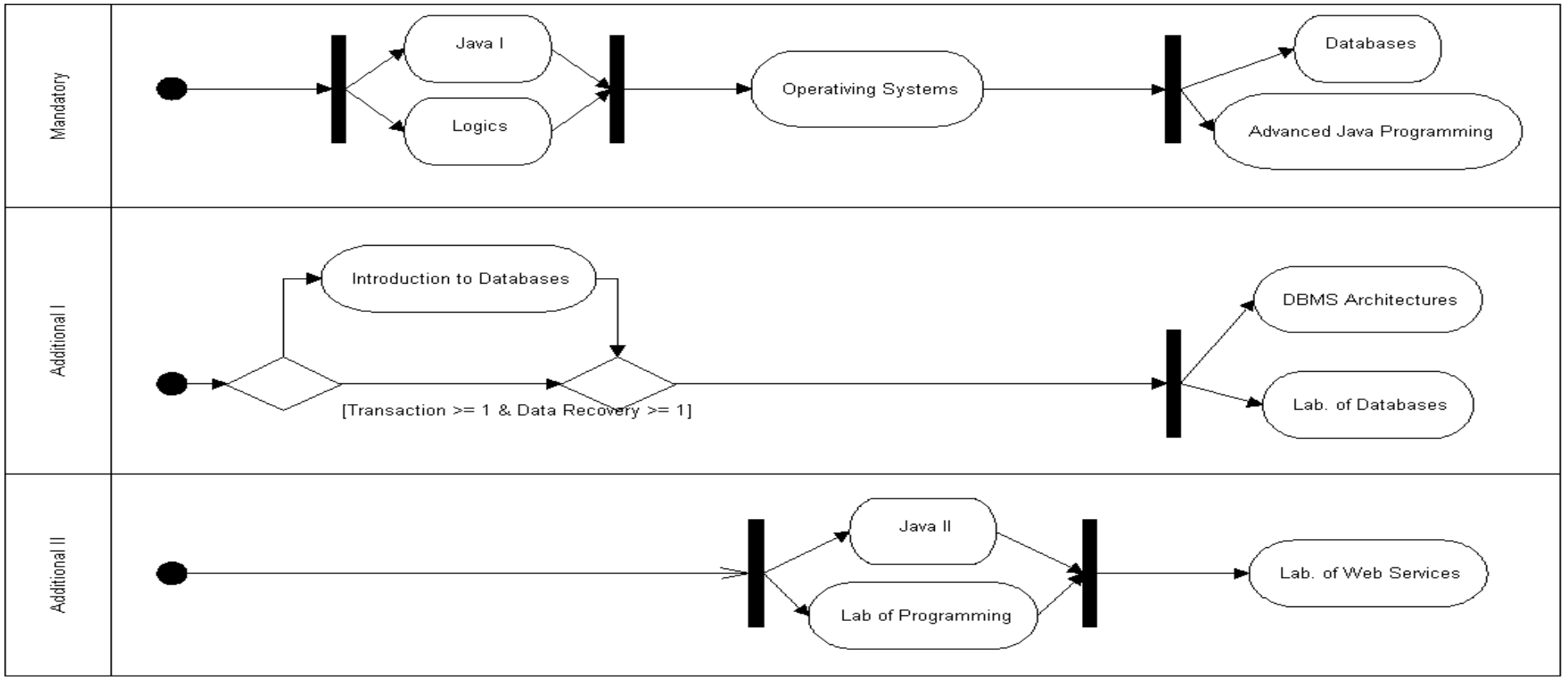
Curricula representation

- How to clearly distinguish between mandatory and optional parts of the curricula?
- Introducing and reorganizing curses in **swim-lines**



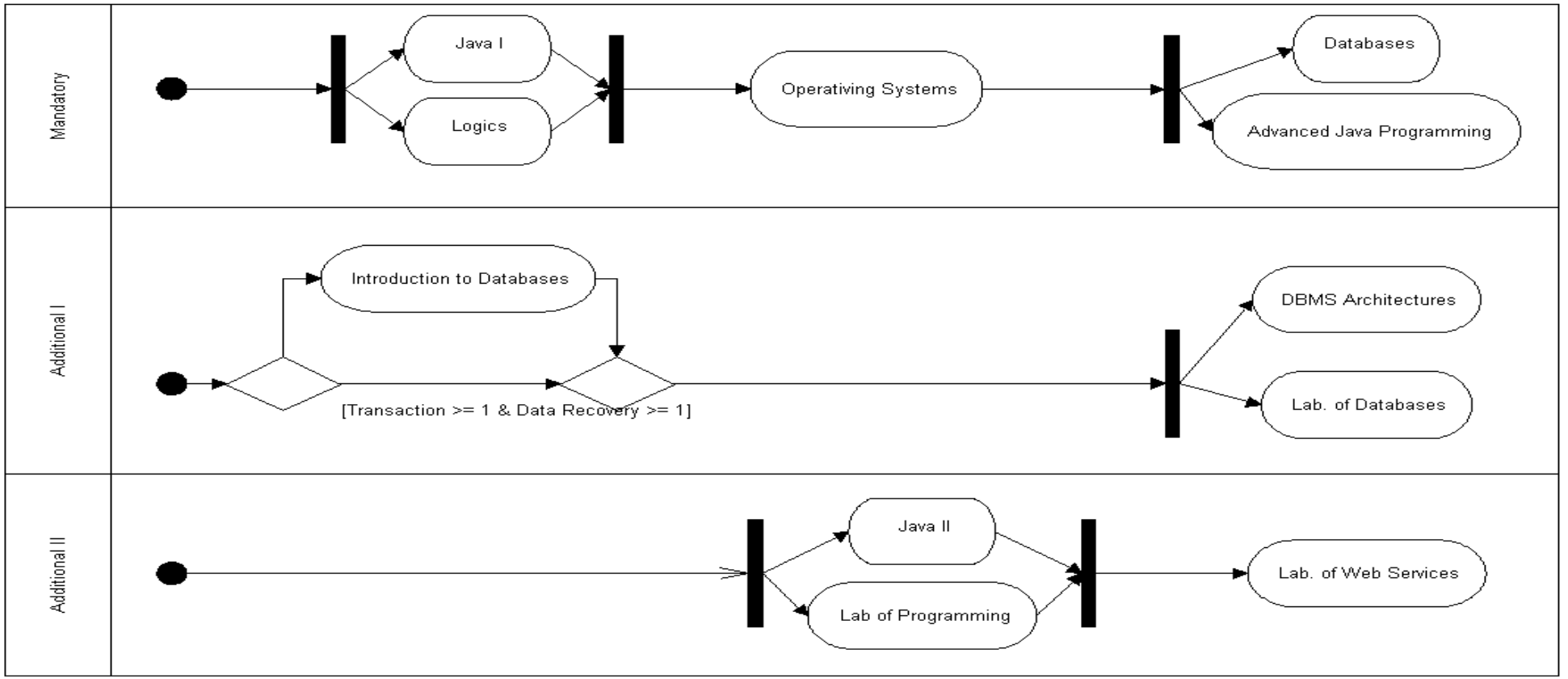
Curricula representation

- How to clearly distinguish between mandatory and optional parts of the curricula?
- Introducing and reorganizing curses in **swim-lines**



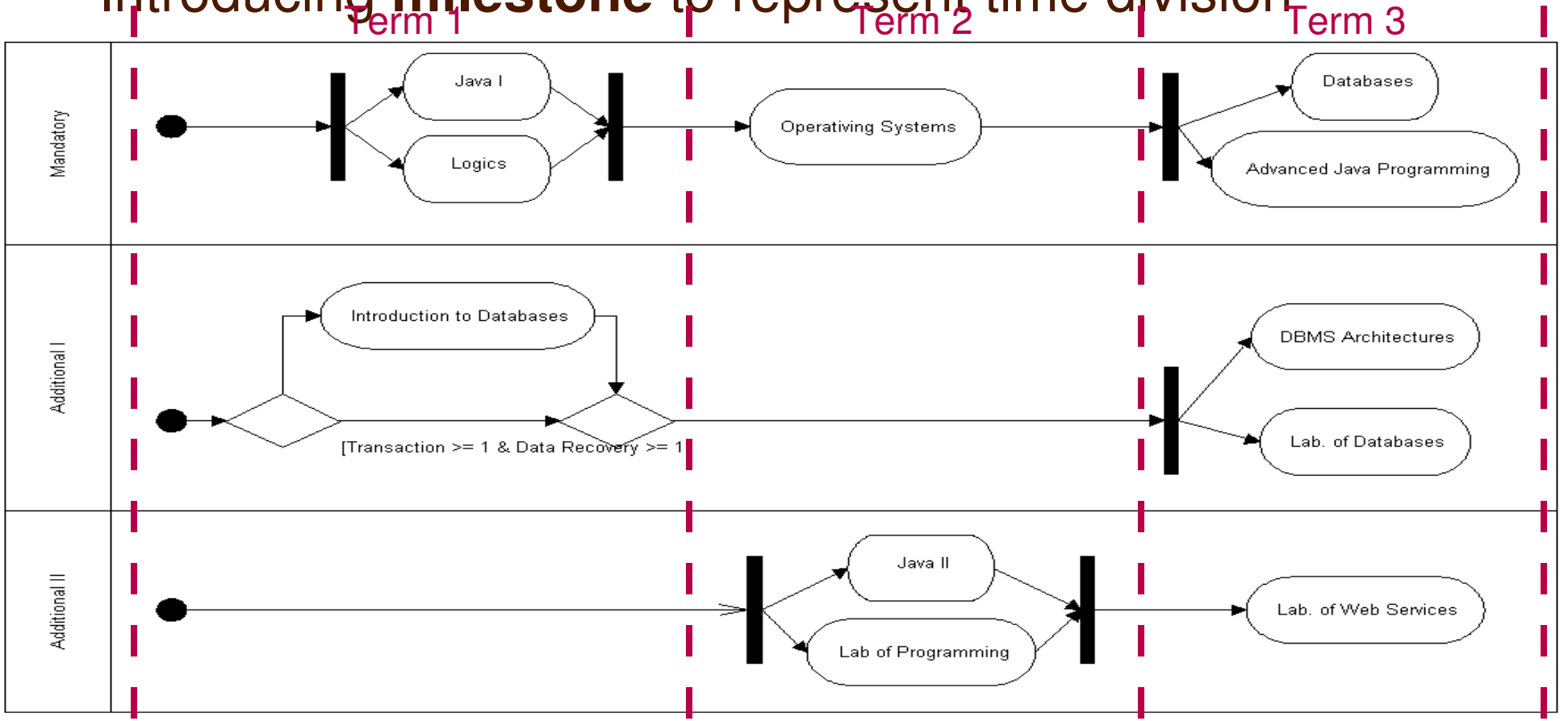
Curricula representation

- How to represent how courses are distributed along the academic year?
- Introducing **milestone** to represent the time division

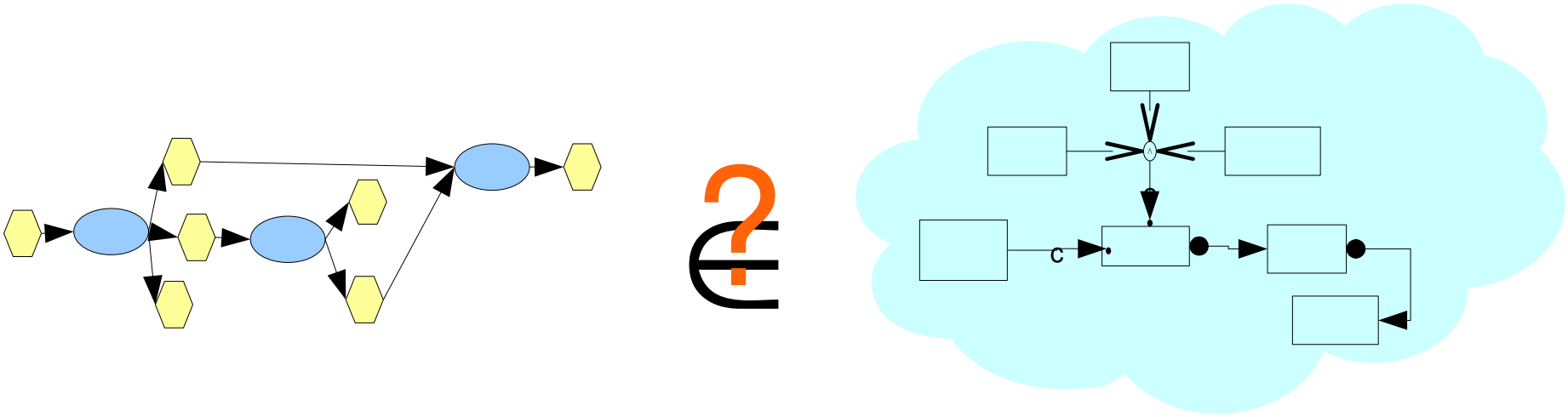


Curricula representation

- How to represent how courses are distributed along the academic year?
- Introducing **milestone** to represent time division



Verification of compliance



- Checking if a curriculum satisfies the course design goals
- In other words, does it satisfy the constraints imposed by the curricula model?

Using SPIN to verify compliance

- Verifications are performed by means of the model checker SPIN
  - it allows to verify if an execution path satisfies an LTL formula: curriculum => execution path, curricula model => LTL formula
  - it returns a *counterexample* in case the formula is not satisfied: very important for the designer!

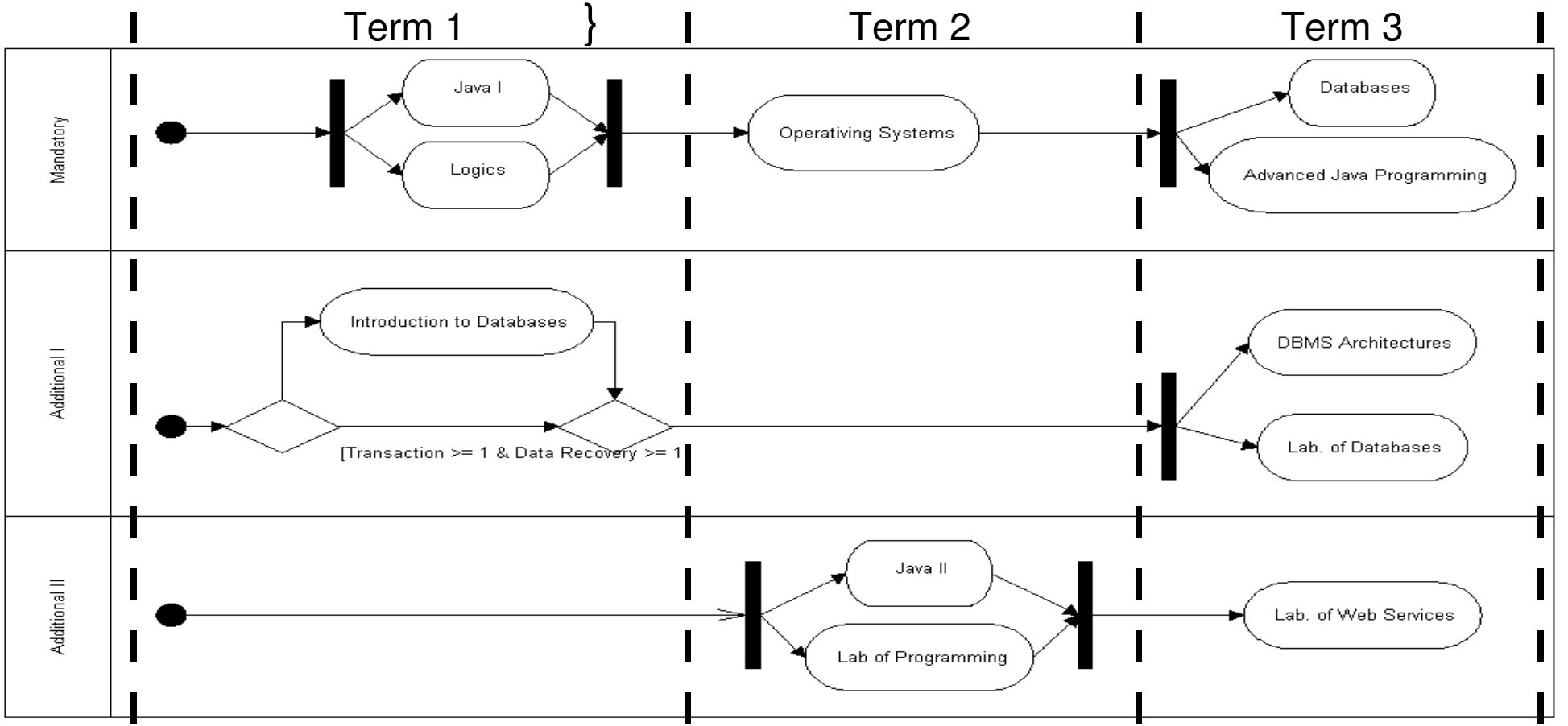
- DCML -> LTL formula
- Activity diagram -> Promela program

UML activity diagrams -> Promela

- The literature offers methods of translation whose aim is debugging. We cannot use them directly
- Our translation:
  - competence -> integer variable
  - CurriculumVerification: list of all the periods in which are divided the curricula
  - For each milestone:
    - First all the preconditions are checked
    - Then the effects are added
  - At the end the Learning Goal is checked

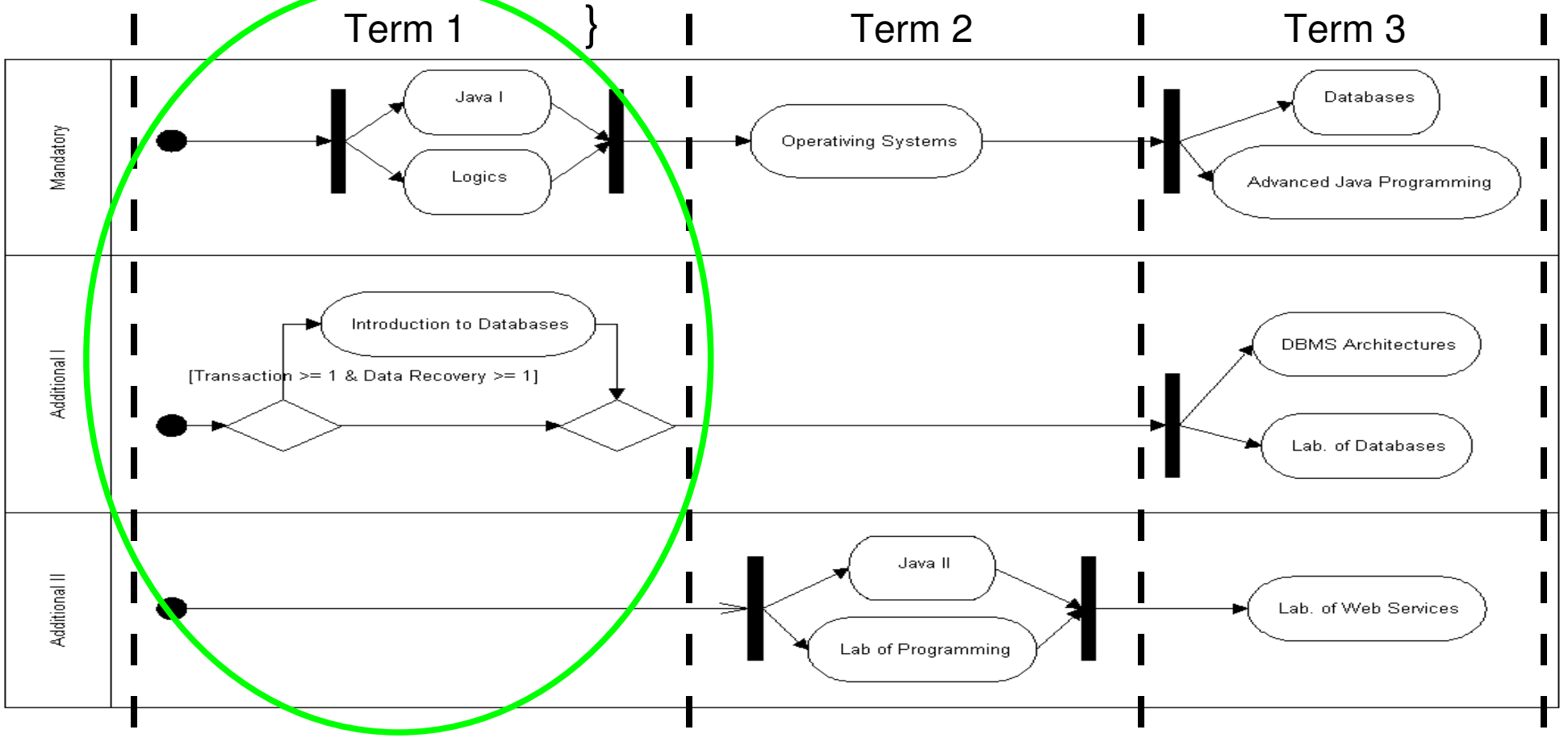
Example

```
proctype CurriculumVerification()
{
    milestone_1();
    milestone_2();
    milestone_3();
    LearningGoal();
}
```



Example

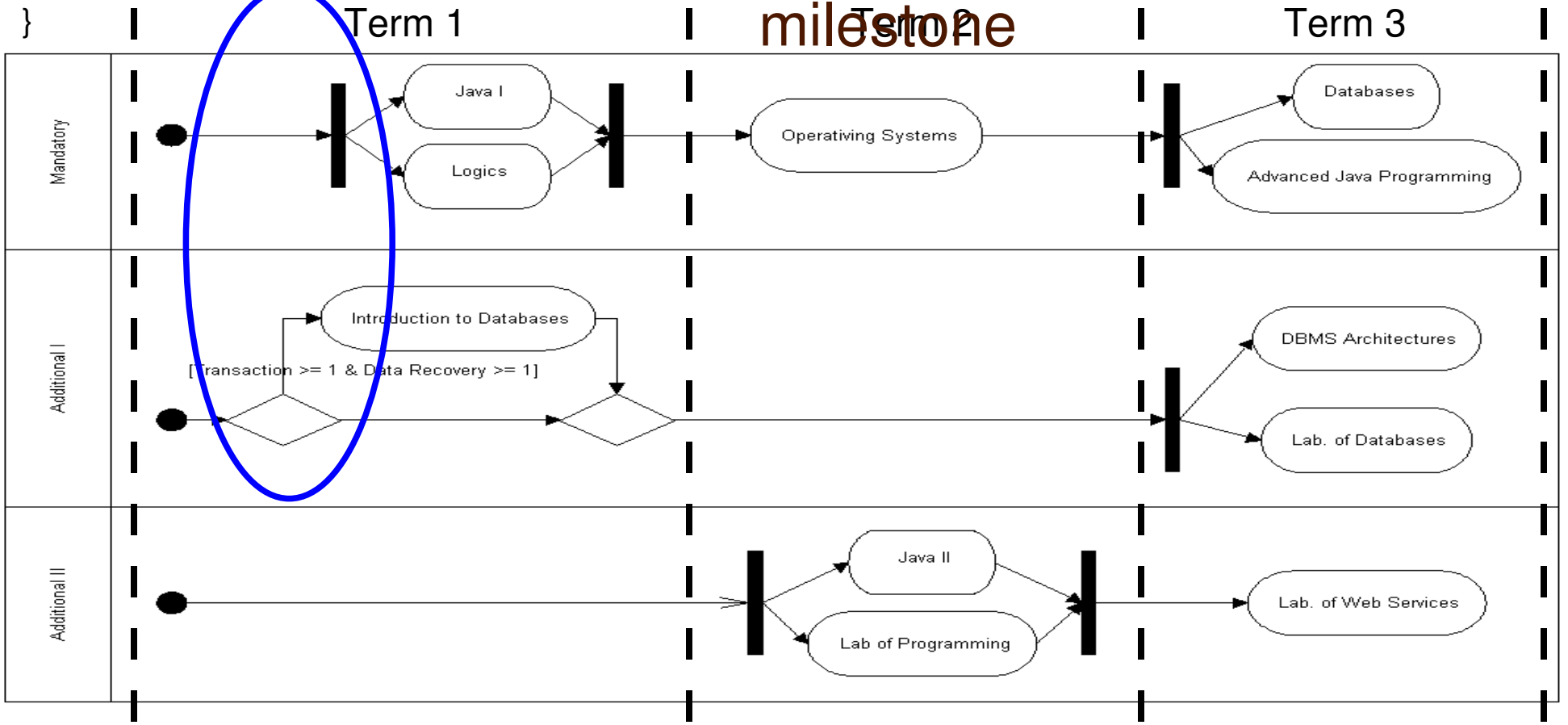
```
proctype CurriculumVerification()
{
    milestone_1();
    milestone_2();
    milestone_3();
    LearningGoal();
}
```



Example

```
proctype CurriculumVerification()
{
    milestone_1();
    milestone_2();
    milestone_3();
    LearningGoal();
}
```

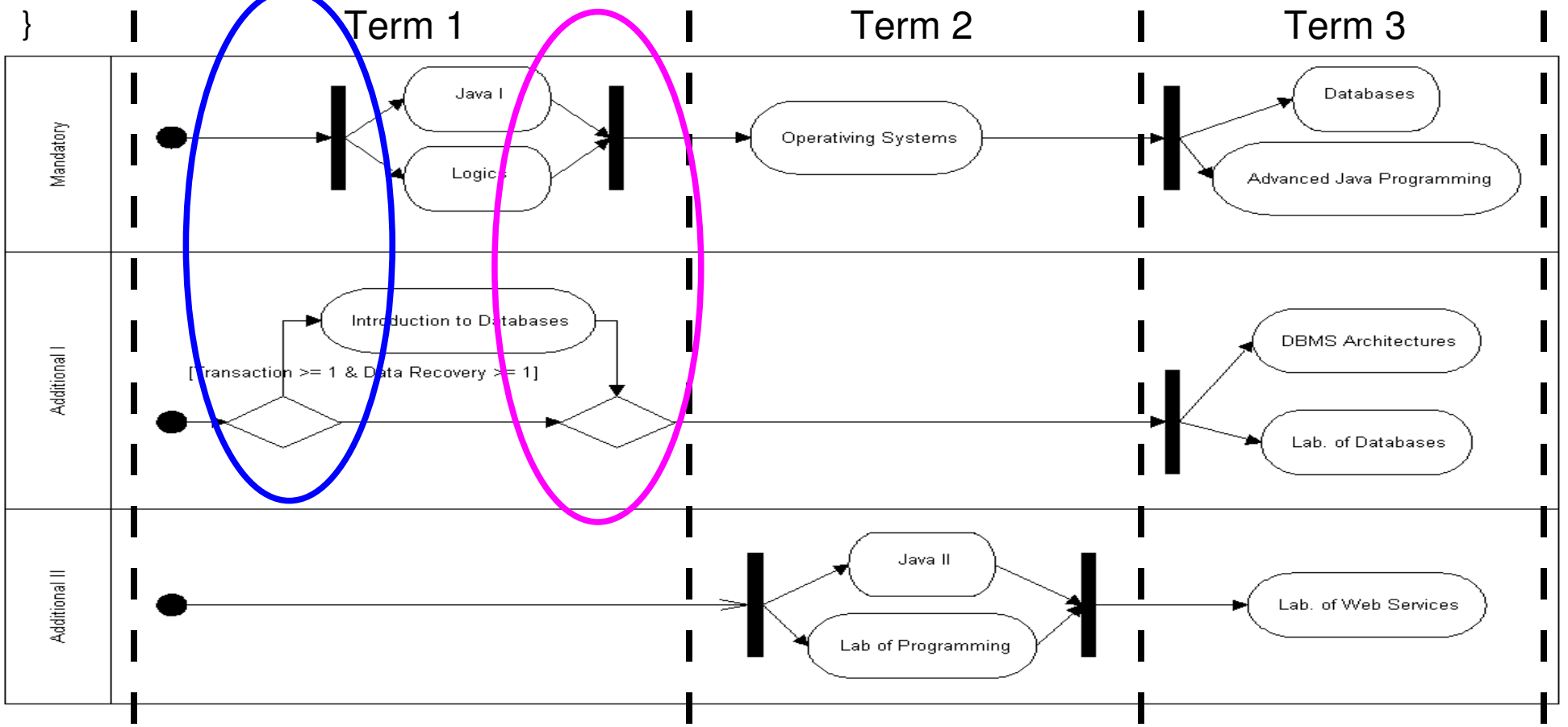
- Check preconditions of all courses that **start** in this milestone
- Add the effects of all the courses that **end** in this milestone



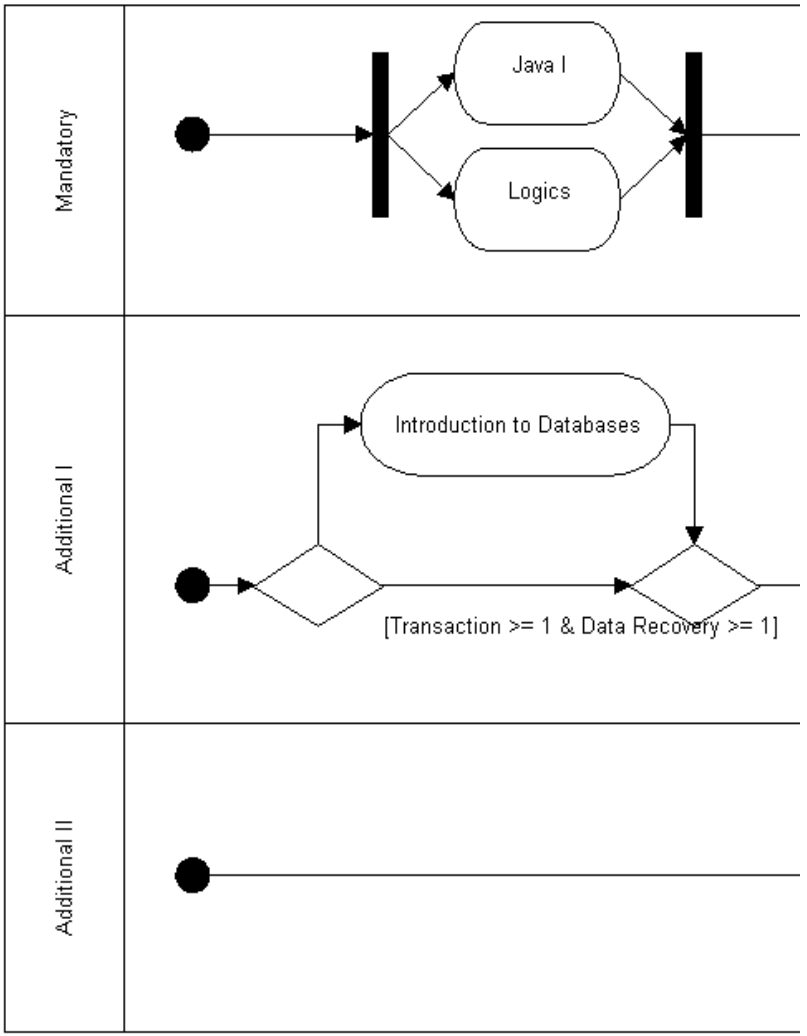
Example

```
proctype CurriculumVerification()
{
    milestone_1();
    milestone_2();
    milestone_3();
    LearningGoal();
}
```

- Check preconditions of all courses that **start** in this milestone
- Add the effects of all the courses that **end** in this milestone



Example



Preconditions

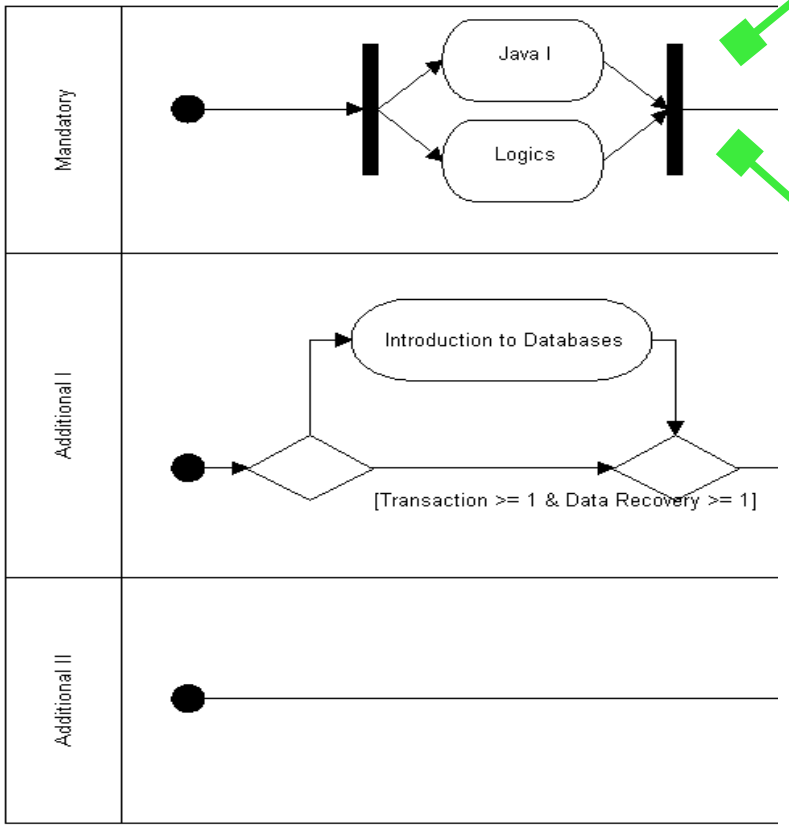
```
Inline milestone_1() {
preconditions_course_Java_I();
preconditions_course_Logics();
if
::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
     preconditions_course_Introduction_to_DB();
::(path == 1 && Transaction >=1
  && Data recovery >= 1) -> skip;
::else -> skip;
fi
```

Effects

```
effects_course_Java_I();
effects_course_Logics();
if
::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
              effects_course_Introduction_to_DB();
::(path == 1 && Transaction >=1
   && Data recovery >= 1) -> skip;
::else -> skip;
fi
}
```

Example

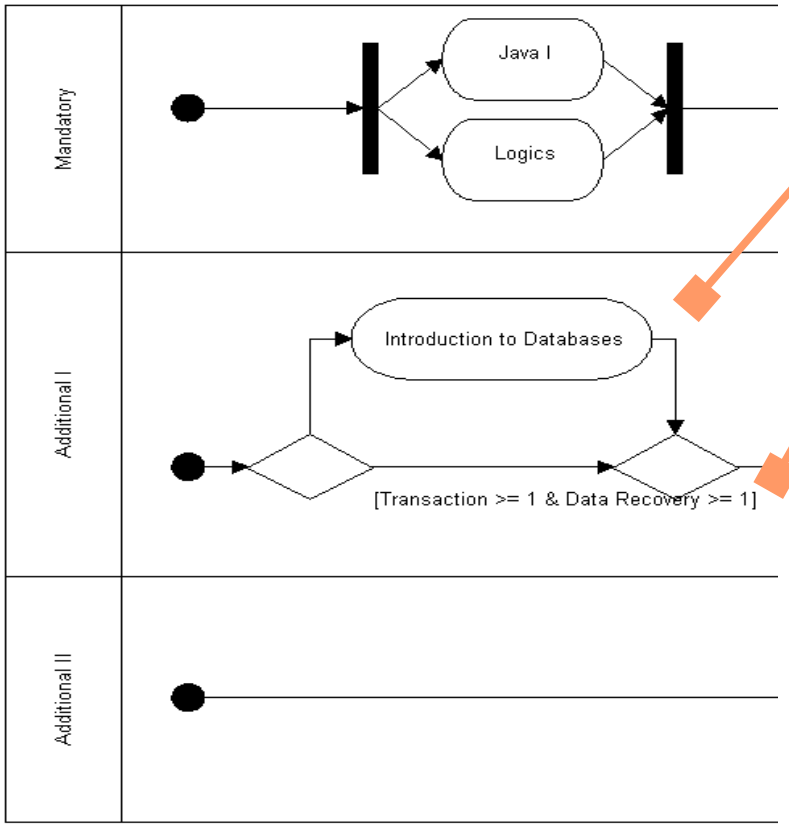
- Mandatory courses




```
Inline milestone_1() {
  preconditions_course_Java_I();
  preconditions_course_Logics();
  if
  ::(path == 1 && (Transaction <1
          || Data recovery < 1)) ->
       preconditions_course_Introduction_to_DB();
   ::(path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi

  effects_course_Java_I();
  effects_course_Logics();
  if
  ::(path == 1 && (Transaction <1
          || Data recovery < 1)) ->
                effects_course_Introduction_to_DB();
  ::(path == 1 && Transaction >=1
     && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi
}
```

Example

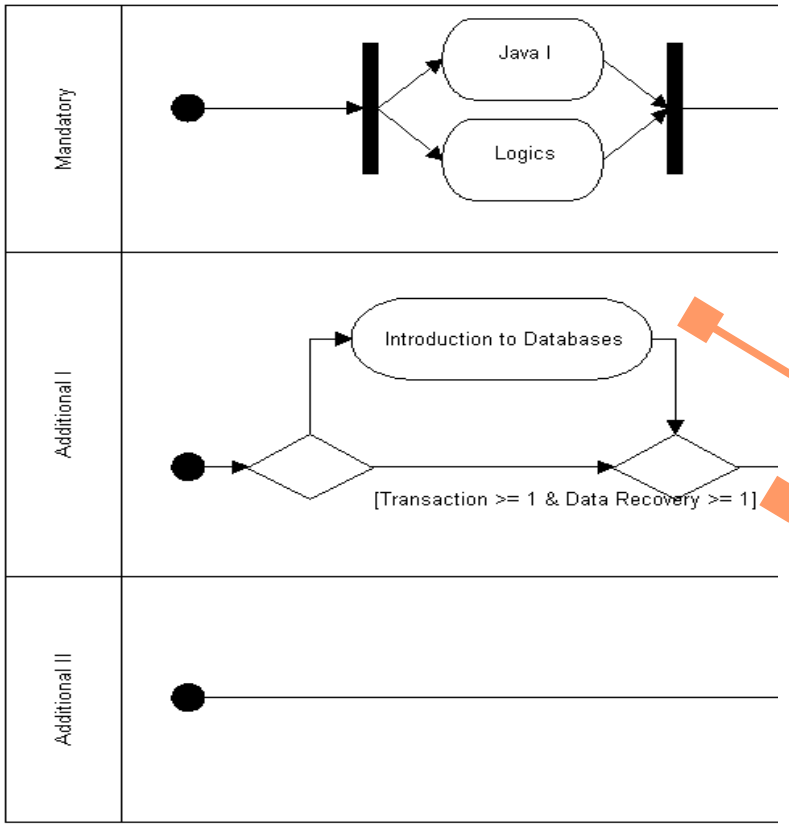
- Additional courses




```
Inline milestone_1() {
  preconditions_course_Java_I();
  preconditions_course_Logics();
  if
  ::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
  preconditions_course_Introduction_to_DB();
  ::(path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi

  effects_course_Java_I();
  effects_course_Logics();
  if
  ::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
    effects_course_Introduction_to_DB();
  ::(path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi
}
```

Example

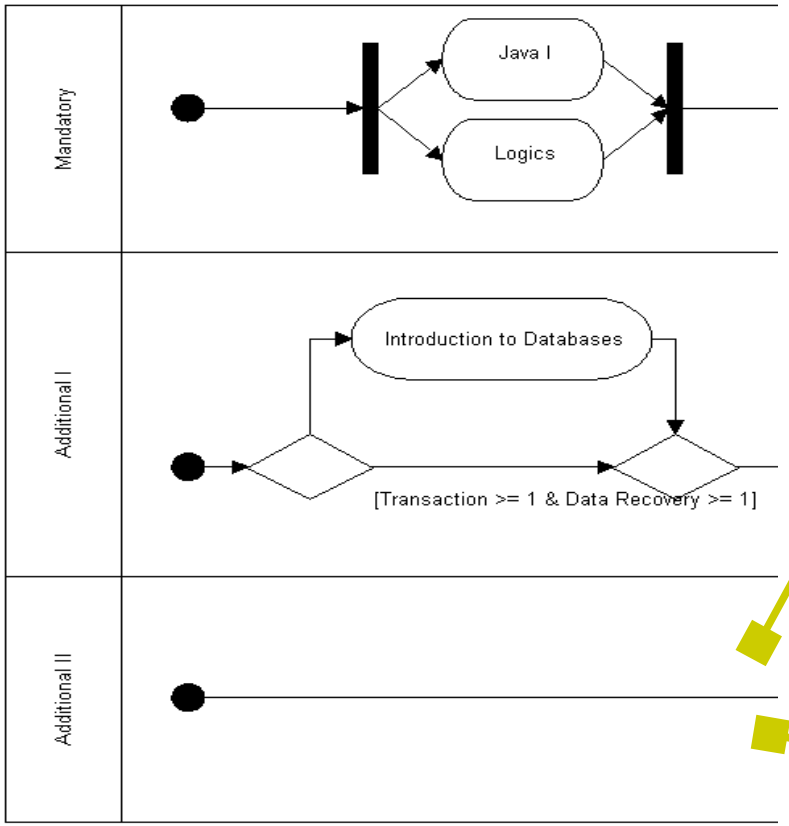
- Additional courses




```
Inline milestone_1() {
  preconditions_course_Java_I();
  preconditions_course_Logics();
  if
  ::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
      preconditions_course_Introduction_to_DB();
  ::(path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi

  effects_course_Java_I();
  effects_course_Logics();
  if
  ::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
    effects_course_Introduction_to_DB();
  ::(path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi
}
```

Example

- Additional courses




```
Inline milestone_1() {
  preconditions_course_Java_I();
  preconditions_course_Logics();
  if
  ::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
      preconditions_course_Introduction_to_DB();
  :: (path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;
  ::else -> skip;
  fi

  effects_course_Java_I();
  effects_course_Logics();
  if
  ::(path == 1 && (Transaction <1
        || Data recovery < 1)) ->
            effects_course_Introduction_to_DB();
  ::(path == 1 && Transaction >=1
    && Data recovery >= 1) -> skip;

  ::else -> skip;
  fi
}
```

Soundness 1

- **Competence gap**: verification by SPIN logical assertions, if no assertion is violated, the curricula given by the translation of an activity diagram show no competence gap

Soundness 1

- **Competence gap**: verification by SPIN logical assertions, if no assertion is violated, the curricula given by the translation of an activity diagram show no competence gap

PRECONDITIONS

```
inline preconditions_course_lab_of_web_services()
{ assert(N_tier_architectures >= 4 && sql >= 2); }
```

EFFECTS

```
inline effects_course_lab_of_web_services()
{ SetCompetenceState(jsp, 4); [...]
  SetCompetenceState(markup_language, 5); }
```

Soundness 2

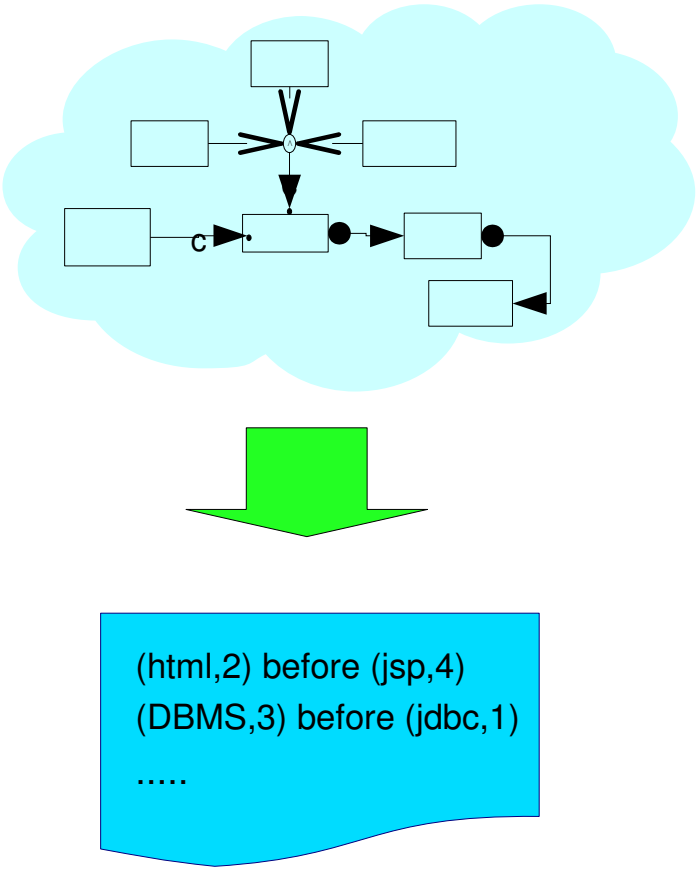
- **Goal achievement**: the learning goal is represented as an *assertion* that must be verified at the end of every execution

Soundness 2

- **Goal achievement**: the learning goal is represented as an *assertion* that must be verified at the end of every execution

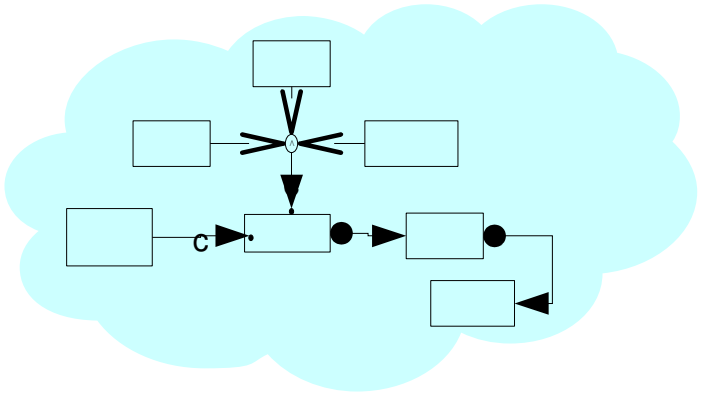
```
 proctype CurriculumVerification()
{
   milestone_1();
   milestone_2();
   milestone_3();
   LearningGoal();
}

 inline LearningGoal()
 { assert(advanced_java_programming>=5 && N_tier_architectures
        >= 4 && relational_algebra>=2 && ER_language>=2); }
```

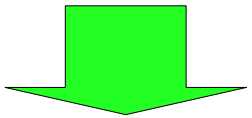
Future Works



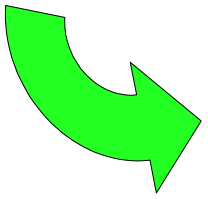
- Automatic translation of the DCML diagram into an intermediate language

Future Works





(html,2) before (jsp,4)
(DBMS,3) before (jdbc,1)
.....



$\neg (k_1,l_1)\ U\ (k_2,l_2)$

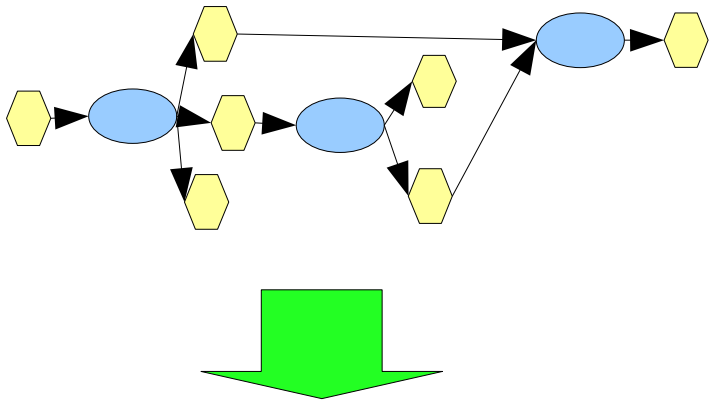
$\Diamond (k_2,l_2) \supset (k_3,l_3)$

$\Diamond (k_2,l_2)$

$\neg (k_5,l_5)$

- Automatic translation of the DCML diagram into an intermediate language
- Automatic translation from the intermediate language into LTL formula

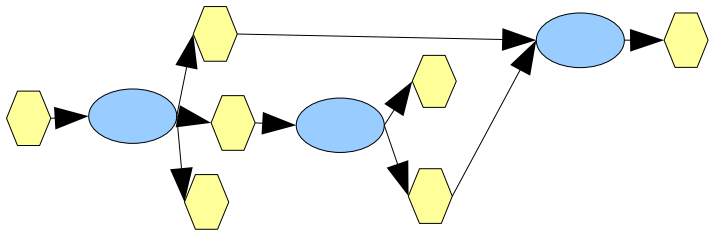
Future Works



course1 then course2
decision_point
parallel(course3, course4)
.....

- Automatic translation of the UML activity diagram into an intermediate language

Future Works



course1 then course2
decision_point
parallel(course3, course4)
.....

```
proctype CurriculumVerification()
{
    milestone_1();
    milestone_2();
    milestone_3();
    LearningGoal();
}
```

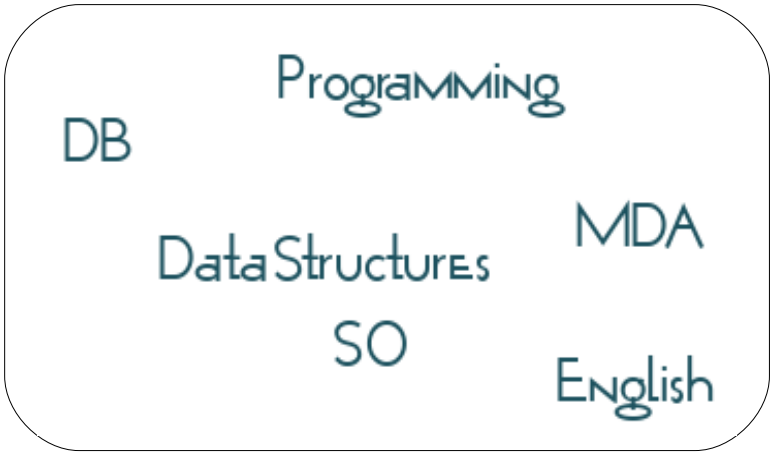
- Automatic translation of the UML activity diagram into an intermediate language
- Automatic translation of the curricula from the intermediate language into a Promela program

Conclusions

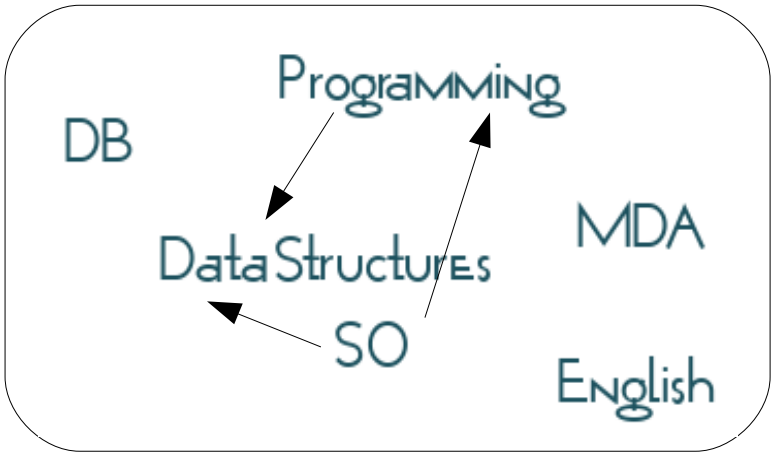
- *Inspired by work on medical guidelines (GLARE)*
- The importance of the concept of “competence” and the need to regulate the process of acquisition of competences is witnessed by a growing attention not only in the area of E-learning
- For example, in *corporations*, **competence management** concerns the way in which the competences of a group of individuals are organized and controlled
- We are currently working on an integration of this proposal with the CRAI competence model (*Harzallah*, *Berio & al.*)

Thanks!

Curricula model and temporal constraints



Competences



Dependencies

- Temporal Constraints:
  - Data Structures before Programming
  - Programming before Operating Systems

Competences and Resources

- The importance of the concept of “competence” and the need to regulate the process of acquisition of competences is witnessed by a growing attention not only in the area of E-learning
- For example, in *corporations*, *competence management* concerns the way in which the competences of a group of individuals are organized and controlled

Goal: to organize resources so as to supply competences in the best possible way

Two levels of representation

- Hence, two levels of representation are required:
  1) Dependencies between competences have a temporal nature; they define *model* for the desired solution
  2) Constraints on resources can be various, they depend on the specific kind of resource and of application domain

DCML

- Representation:
  1) What to use ?
  2) Workflows, activity diagrams

Competences and constraints