A Pattern-Matching Based Approach for Problem Solving in Model Transformations

Youness Laghouaouta, Pierre Laforcade, Esteban loiseau

Computer Science Laboratory of Le Mans University (LIUM - France)

28 July, 2018

ICSOFT 2018

Porto, Portugal

Escape'it project

- Develop a mobile serious game to train visual performance skills for children with Autism Spectrum Disorder.
  - Matching object to an identical object, sorting similar objects...
- Mechanics from "espace-room" games (opening a locked door to escape the room).

Escape'it project - Problematic

- **Adaptativity** : every new game session has to be adapted to the child's profile.
- **Generalization** : the game has to propose a large variety of scenes settings in order to support the generalization of skills.

Escape'it project - Problematic

- **Adaptativity** : every new game session has to be adapted to the child's profile.
- **Generalization** : the game has to propose a large variety of scenes settings in order to support the generalization of skills.

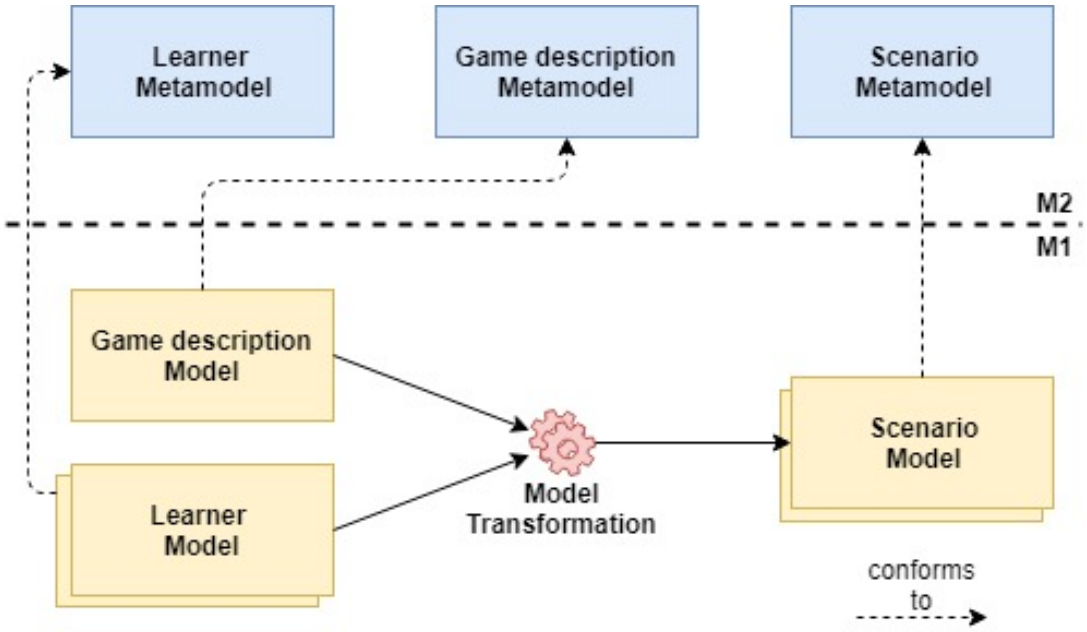
Cost to design and implement all combinations

How to automatically generate learning scenarios that are adapted to children' profiles ?

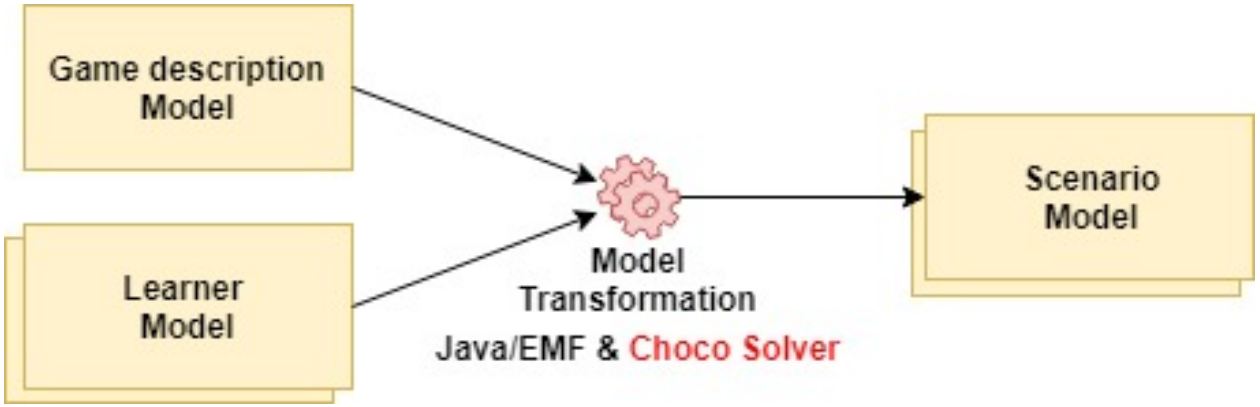
Escape'it project - Proposed solution

A MDE solution to support the adaptative generation of learning scenarios :

- **Metamodeling** : define the static domain elements related to the game :
  - what the game can do ?
  - what a learner knows and did ?
  - what the game session should look like ?
- **Transformation** : express the generation of adapted scenarios as a model transformation.



Escape'it project - Issues



- **Expression of satisfaction problem** : use of numeric values instead of model elements.
- **Change of domain rules** : identification of the fragments impacted by the expressed change.

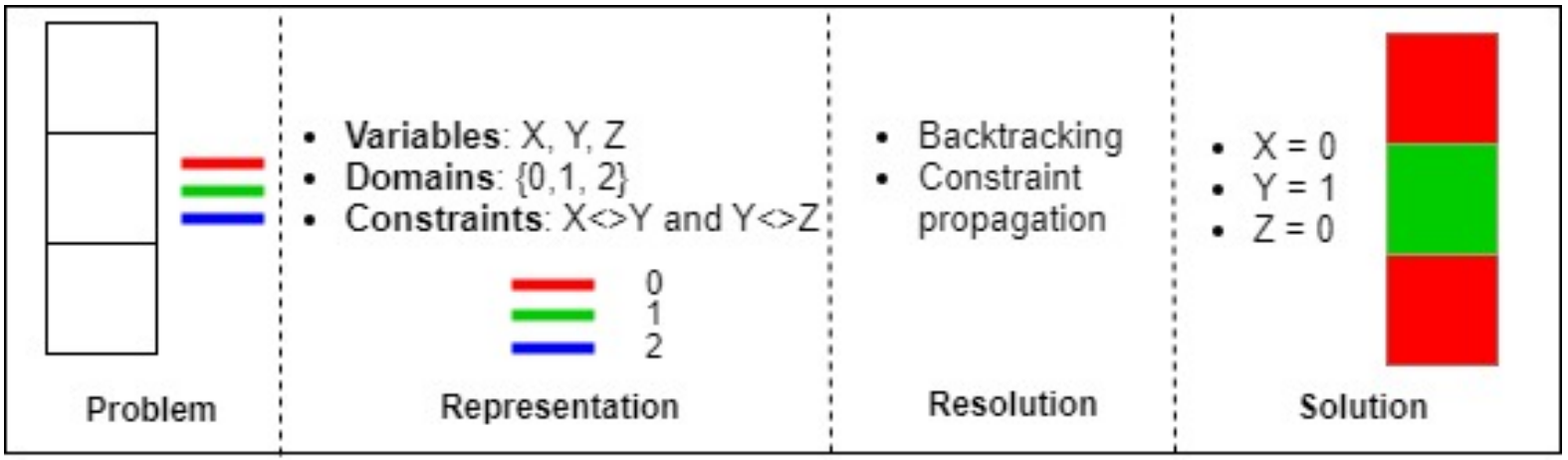
Objectives

- Generation of target models by transforming source ones (**Model transformation**).
- Specification of constraints to enforce on target models with a practical manner and constraint solving (**Constraints solving**).
- Structuring the transformation with a view to supporting changes (**Modularization**).

Outline

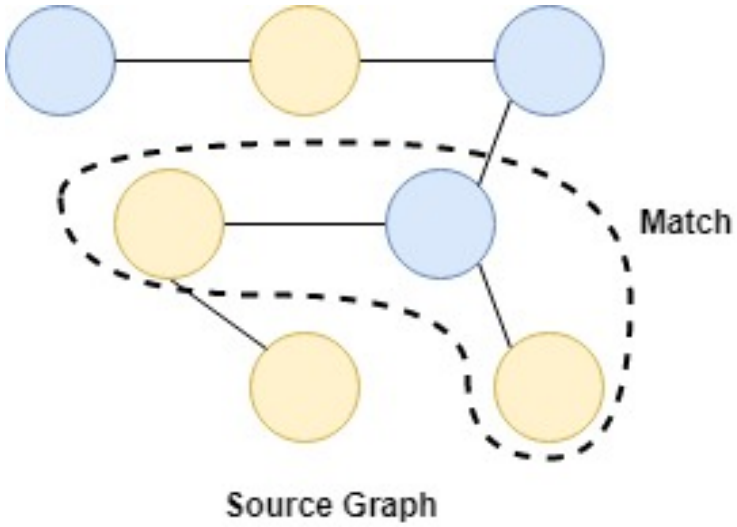
Constraint Satisfaction Problem (CSP)

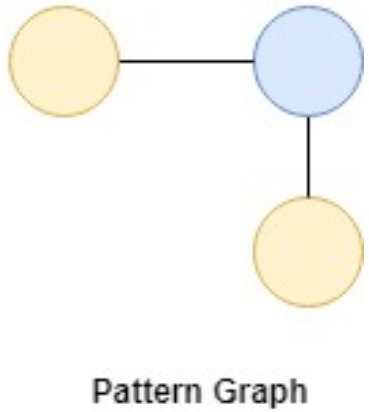
- A **CSP** is defined by : {**Variables**, **Domains**, **Constraints**}
- A solution is an assignment of a value to each variable that satisfies every constraint.



Graph Pattern Matching

- **Graph pattern matching** : it is based on (sub)graph isomorphism and requires finding an image (**match**) of a given graph (**pattern graph**) in another graph (**source graph**).





CSP - Graph Pattern Matching

Graph pattern matching is expressed and resolved as a CSP :

- CSP variables correspond to the objects of the pattern graph.
- Variable domains correspond to the source graph objects to be matched into.
- Constraints correspond to the restrictions that apply to a graph morphism.

CSP - Graph Pattern Matching

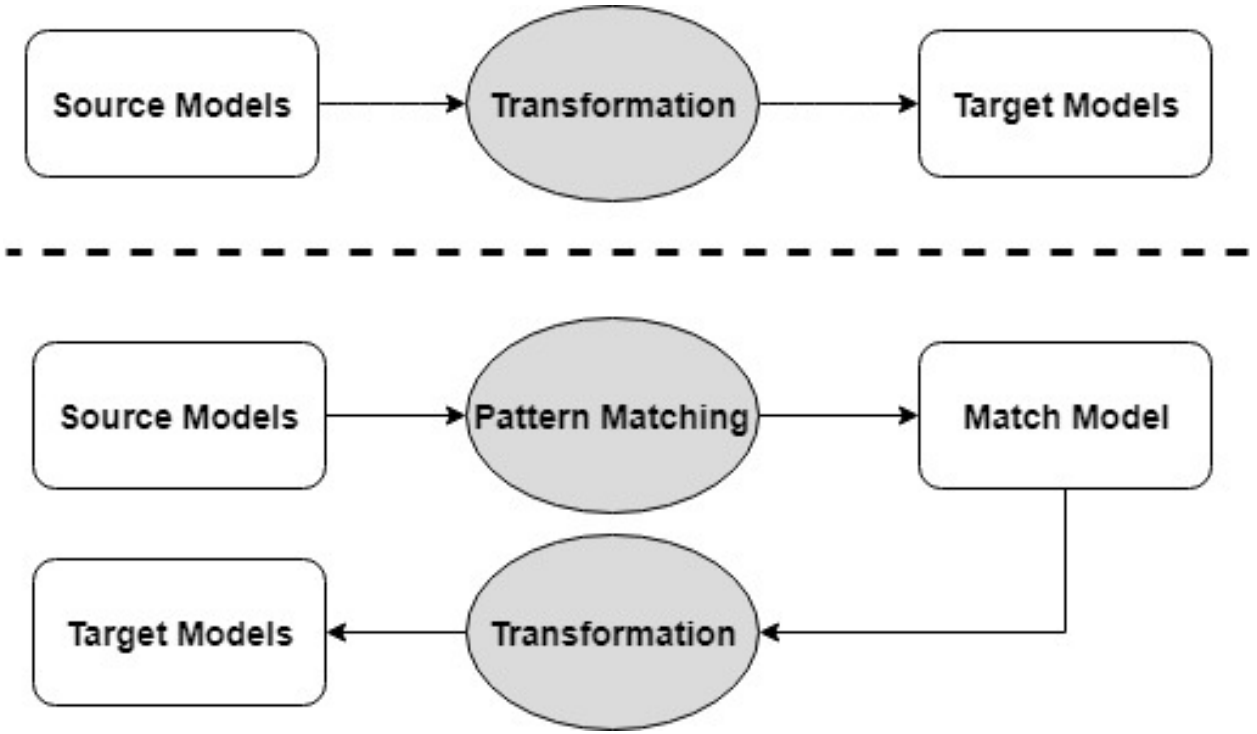
Graph pattern matching is expressed and resolved as a CSP :

- CSP variables correspond to the objects of the pattern graph.
- Variable domains correspond to the source graph objects to be matched into.
- Constraints correspond to the restrictions that apply to a graph morphism.

Pattern matching problem as a high level specification of a CSP

CSP problem over a model can be expressed by means of model elements rather than establishing non evident correspondences.

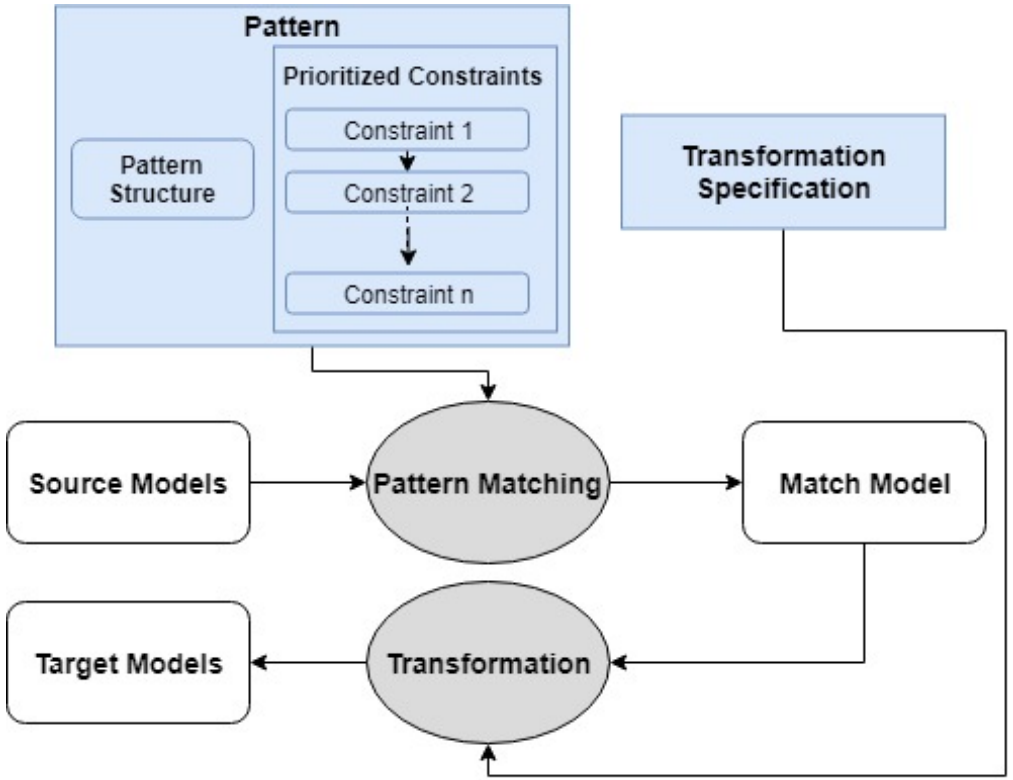
Transformation process



Pattern requirements

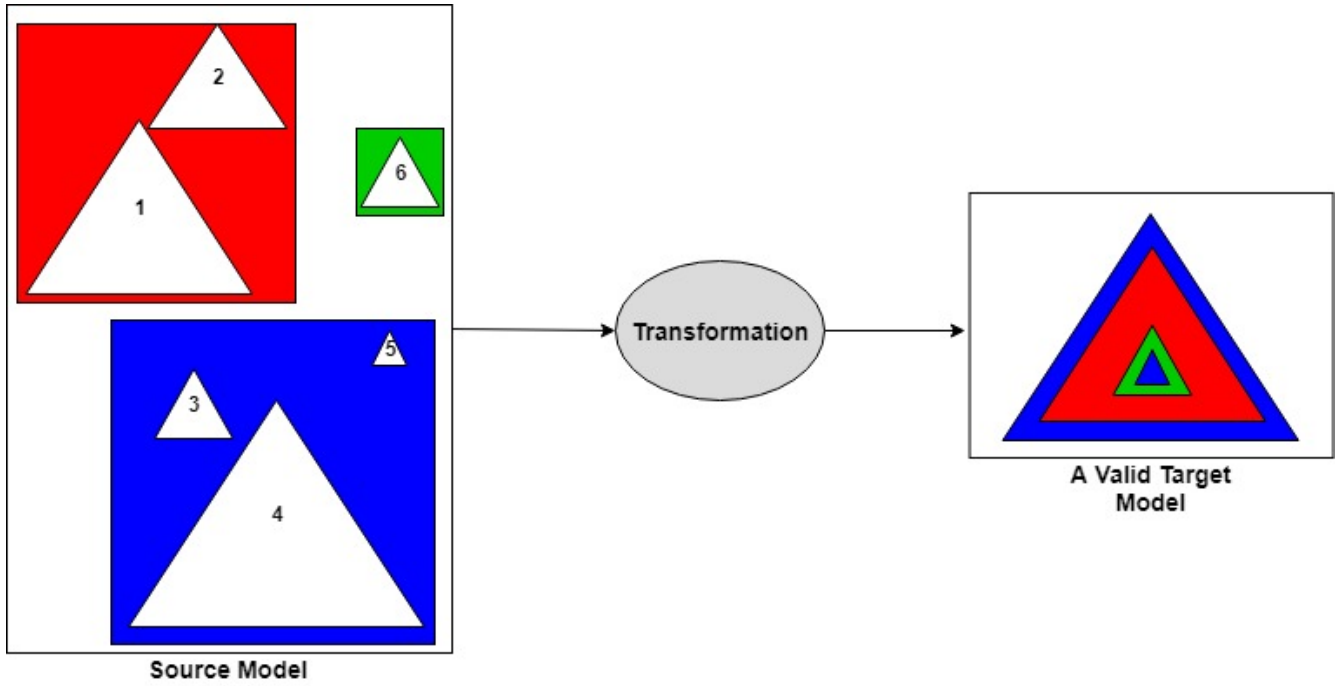
- Satisfaction of all constraint to enforce on target models.
- Sufficient for a complete generation of target models.

Pattern definition



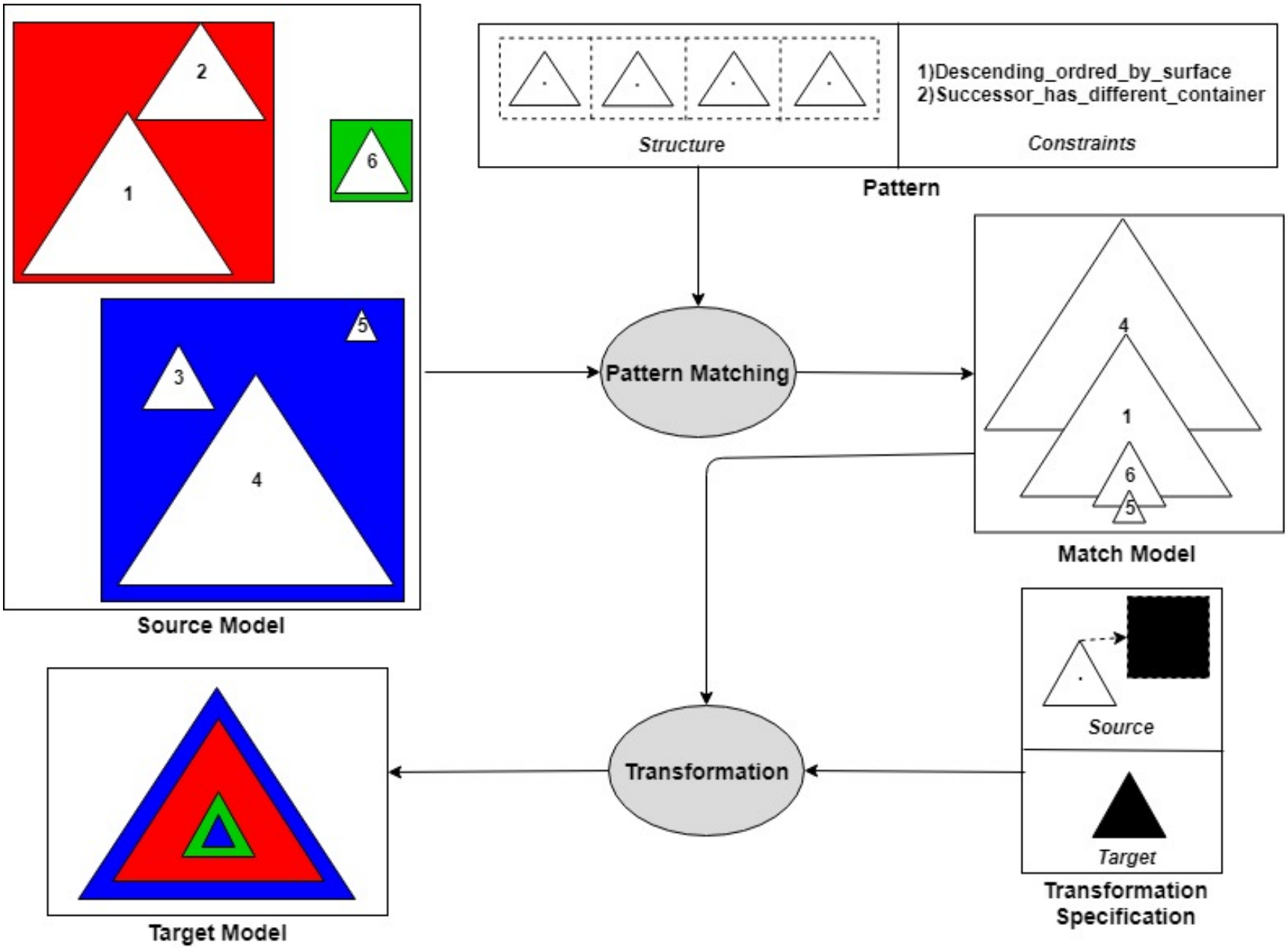
- Different constraints can be associated to the same pattern.
- Transformation is decoupled from constraints.

Example



- **Transformation scenario** : Generate a piling of 4 triangles.
- **Constraints** :
  - The produced piling has to be coherent.
  - Two adjacent triangles must have different colors.

Example


2
1
6
5
3
4
Source Model
Structure
1)Descending_ordred_by_surface
2)Successor_has_different_container
Constraints
Pattern
Pattern Matching
4
1
6
5
Match Model
Source
Target
Transformation Specification
Transformation
Target Model

Pattern specification

- Implied metamodels and managed models
- Pattern structure :
  - Number of elements to match.
  - Types.
  - Restrictions.
- Validation constraints.

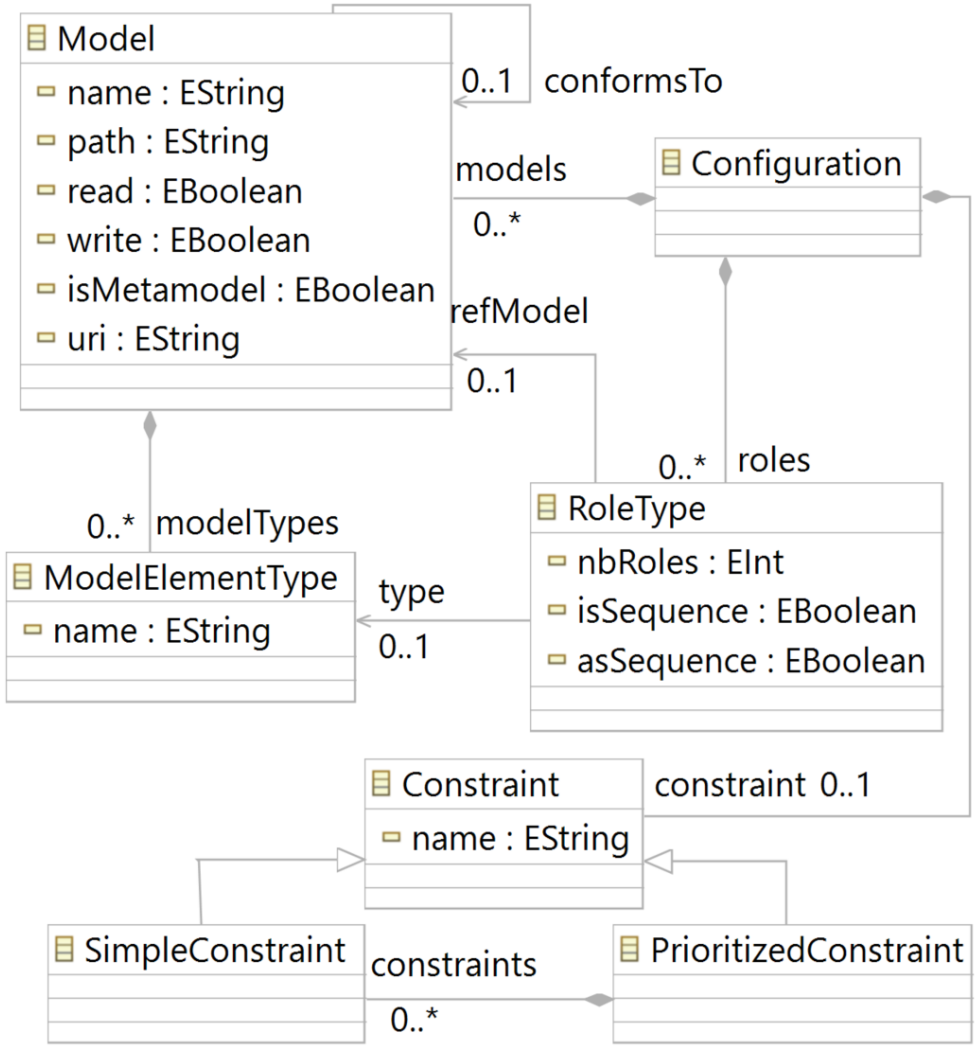
Pattern specification

- Implied metamodels and managed models
- Pattern structure :
  - Number of elements to match.
  - Types.
  - Restrictions.
- Validation constraints.

Note

The relevant information is considered as a configuration for the transformation process.

Configuration metamodel



Generation of target models ?!

A configuration model is not sufficient to generate target models.

- Constraints specification, domain restrictions.
- Transformation specification.
- Execution workflow (search for a valid match, transformation).

Generation of target models ?!

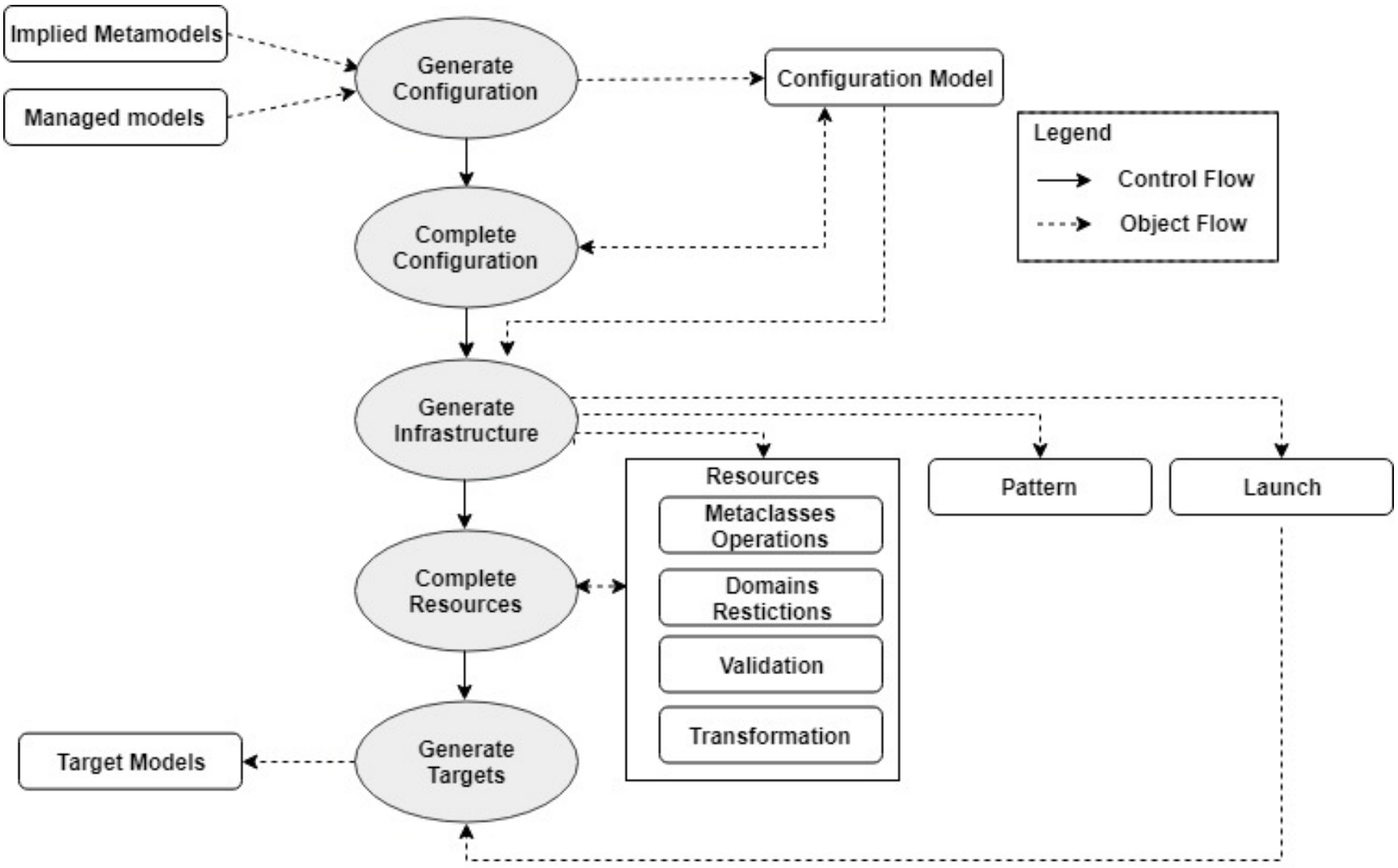
A configuration model is not sufficient to generate target models.

- Constraints specification, domain restrictions.
- Transformation specification.
- Execution workflow (search for a valid match, transformation).

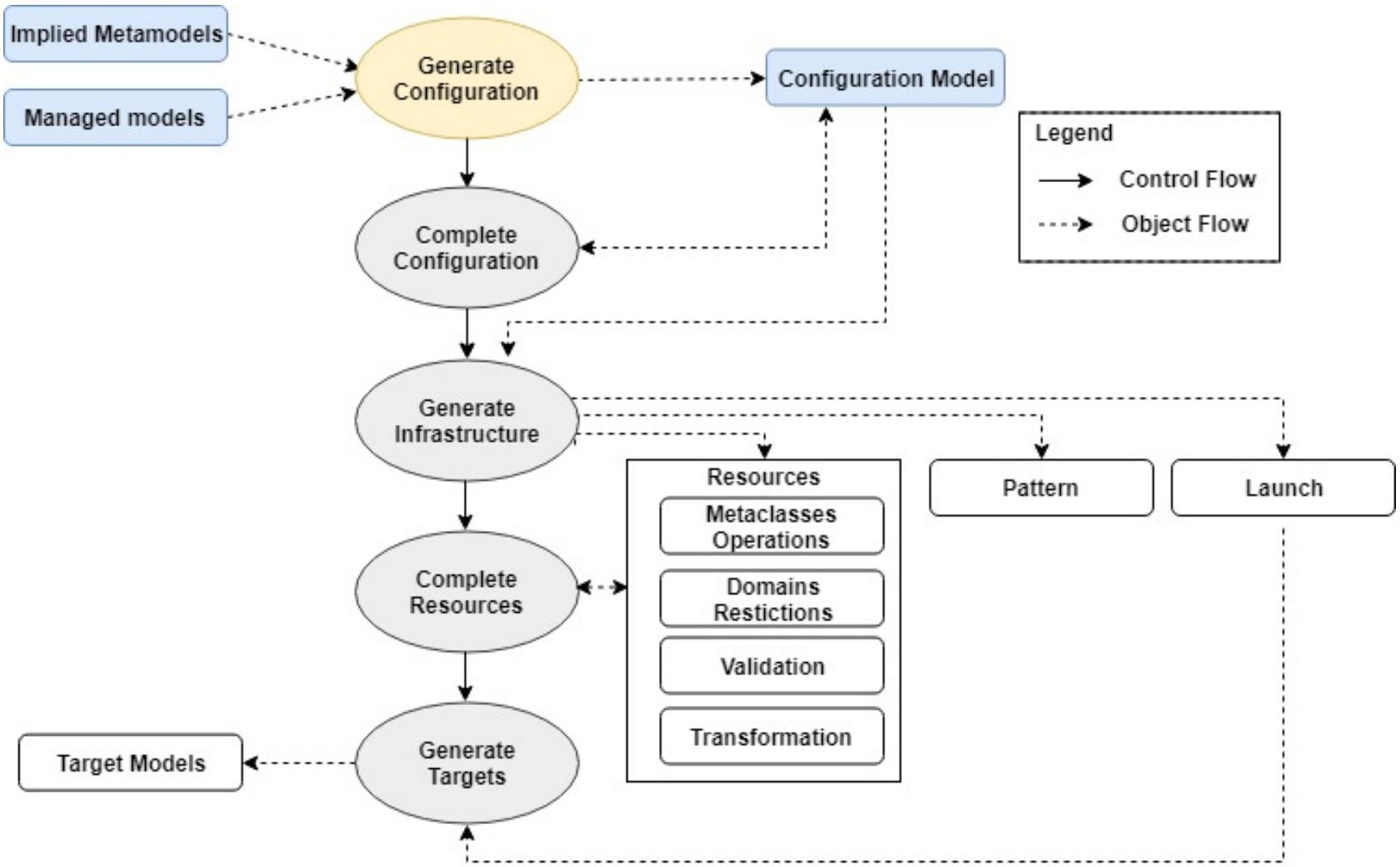
Generate a transformation infrastructure that supports the proposed approach

- Derived from the configuration model.
- References resources that have to be manually completed.

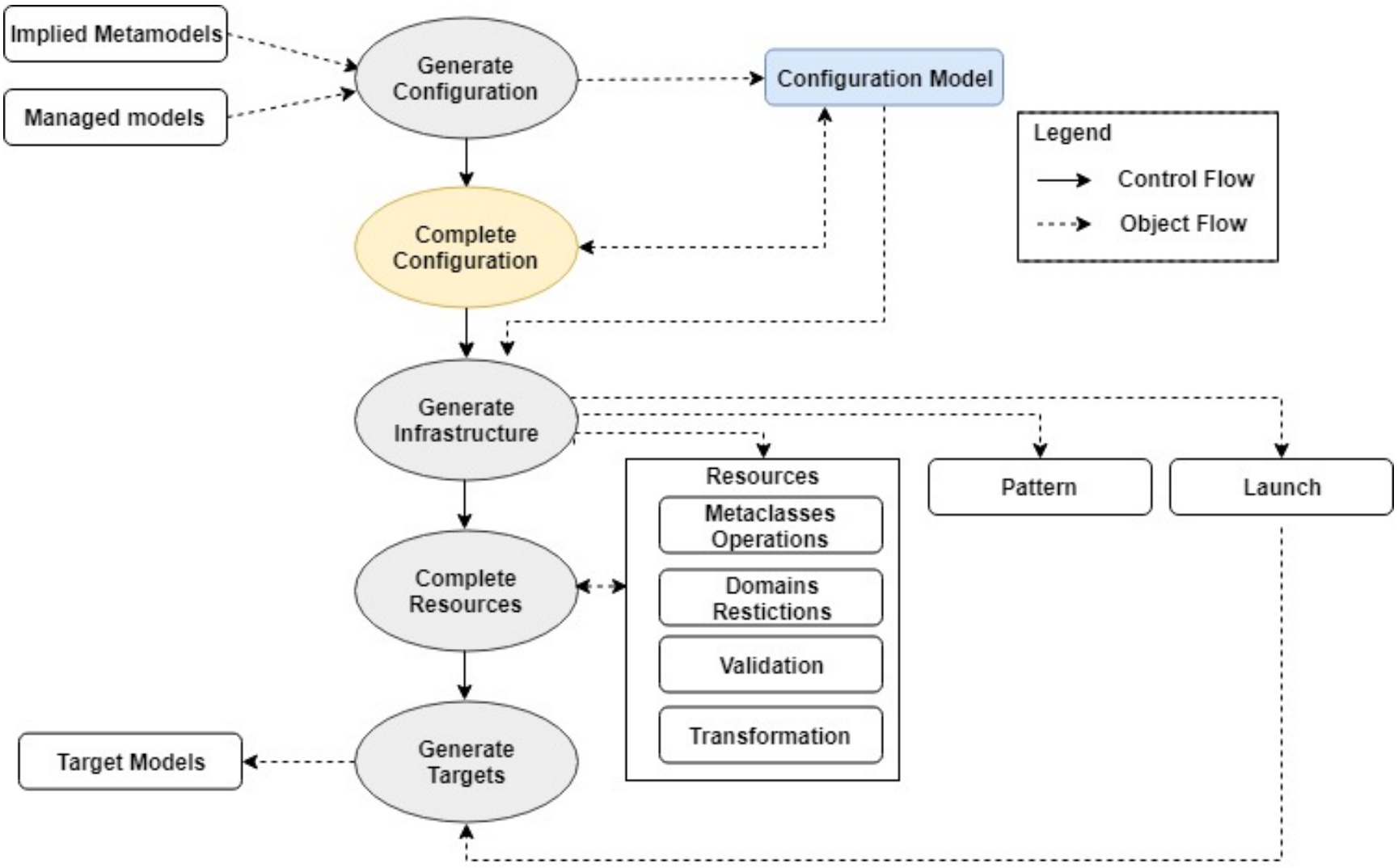
Generation of transformation infrastructure



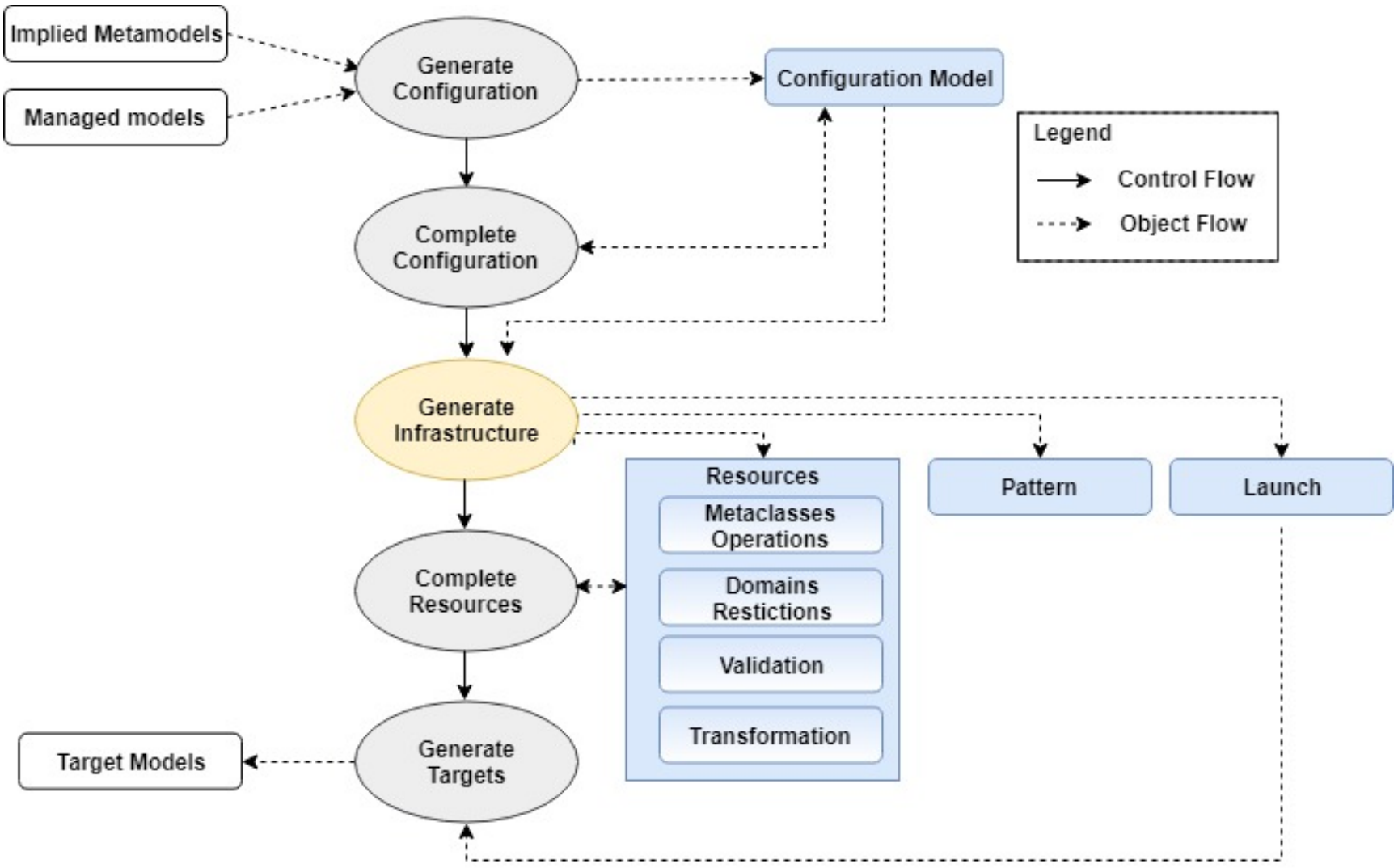
Generation of transformation infrastructure



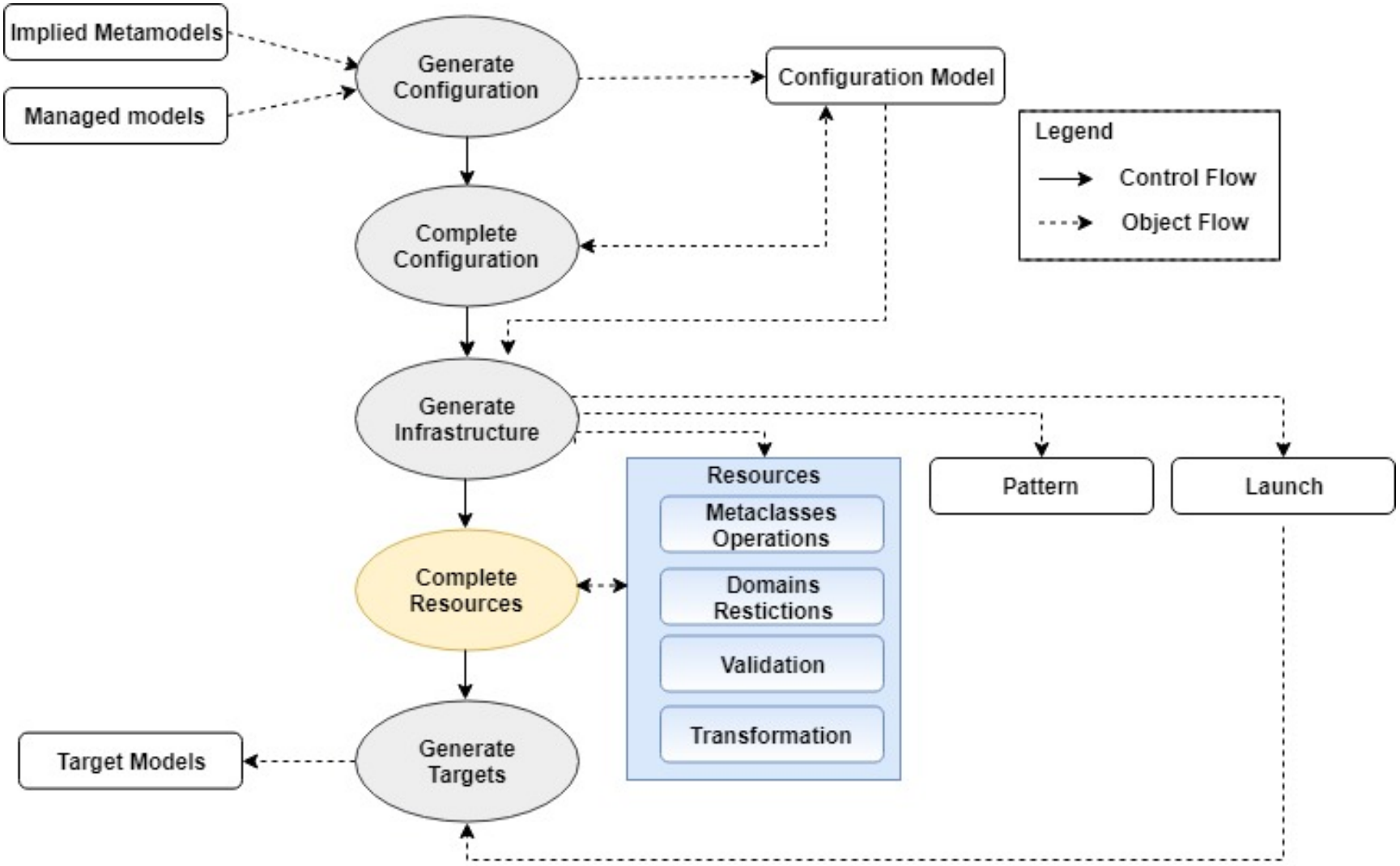
Generation of transformation infrastructure



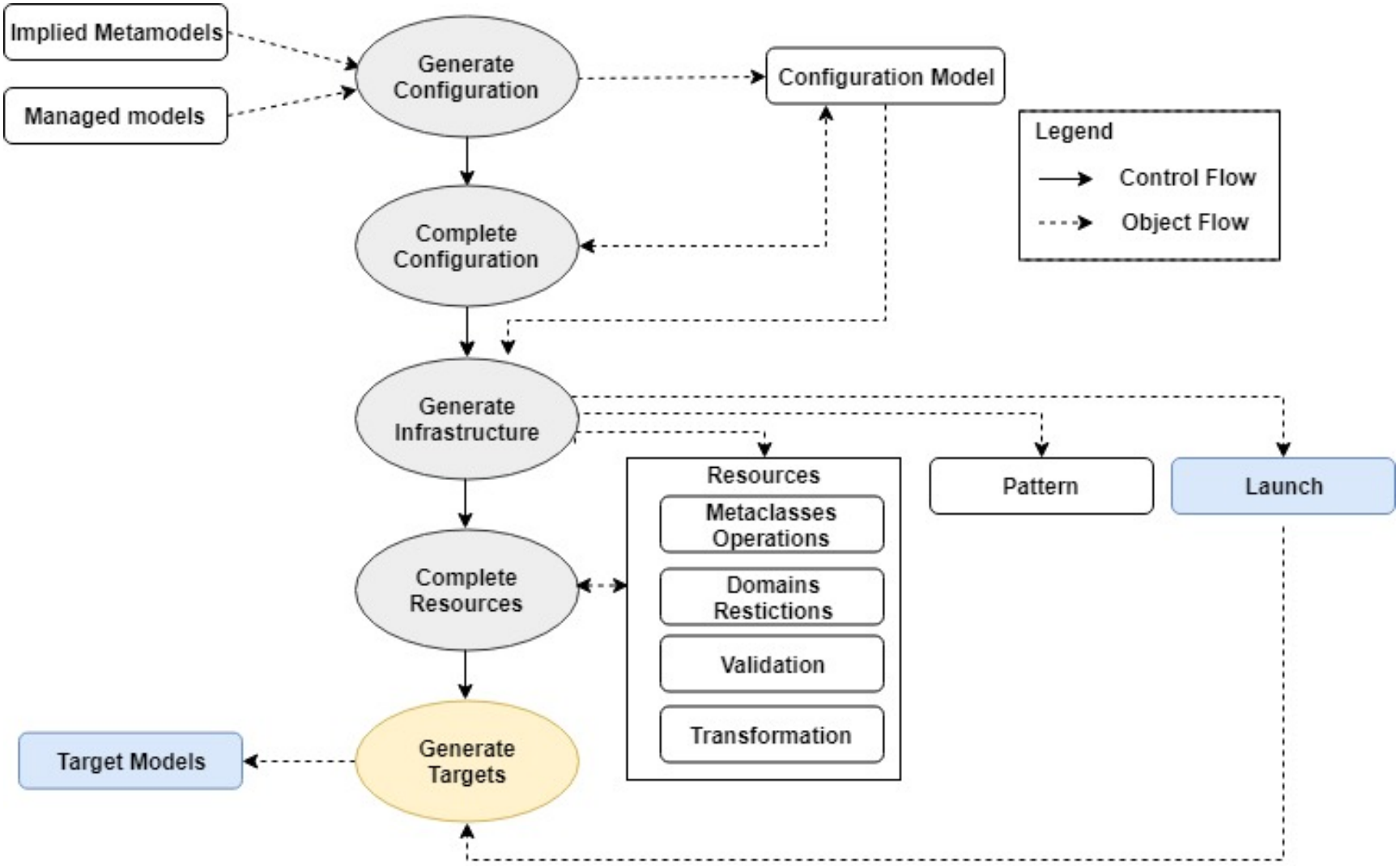
Generation of transformation infrastructure



Generation of transformation infrastructure



Generation of transformation infrastructure



Technical details

- For each constraint level, an EPL pattern is generated.
- Orchestration of the generated EPL patterns :
  - An ANT workflow is automatically generated and it specifies conditions to employ a less prioritized pattern.
- All these details are hidden from the developer :
  - the generation of a valid target model is based on one pattern with different constraints.

Outline

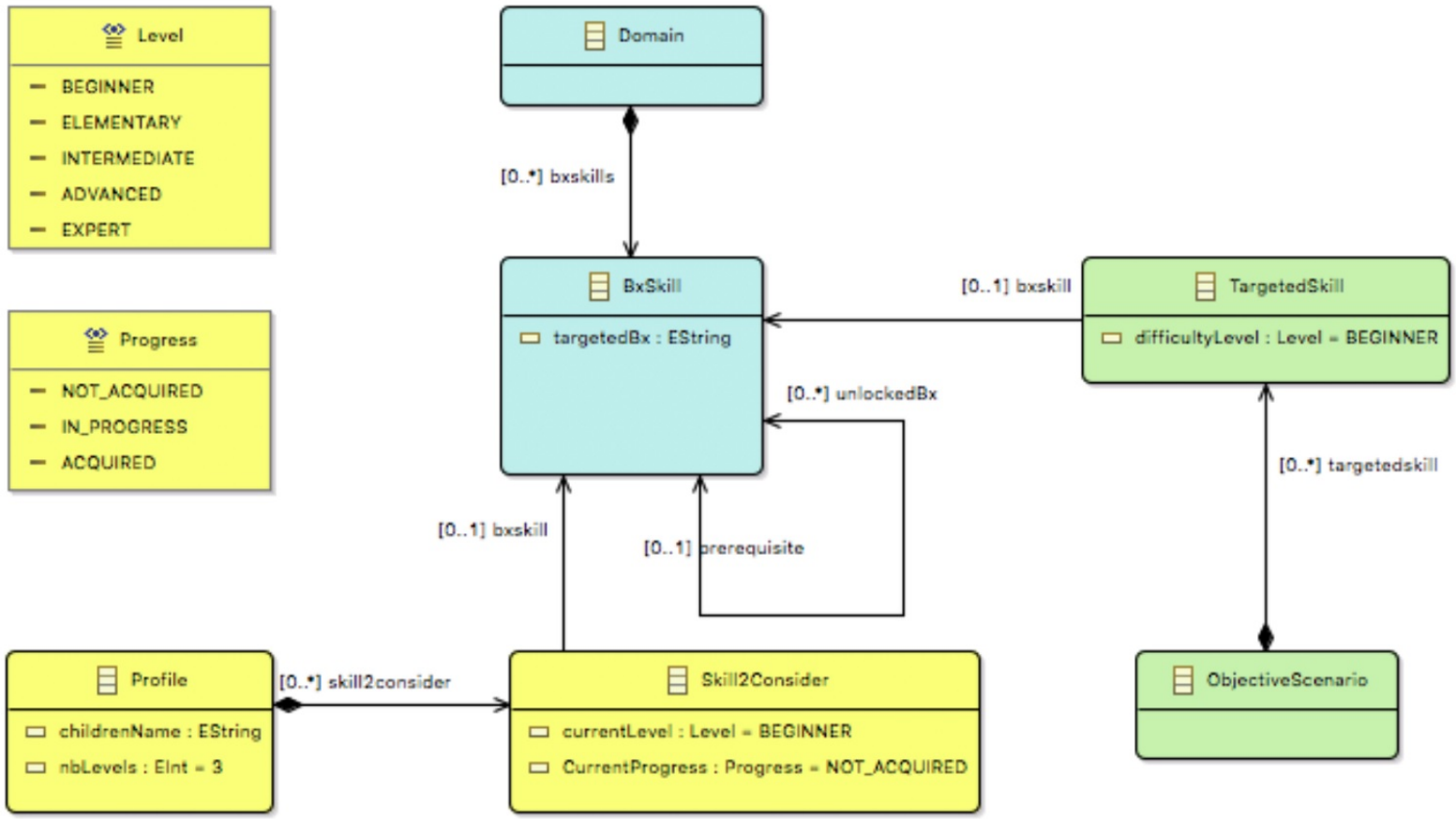
Transformation scenario

- Generation of the objective scenario :
  - Elicitation of the visual performance skills in accordance with the number of levels to generate, the considered skills and the child's progression.
  - Source models :
    - Game description.
    - Child profile.

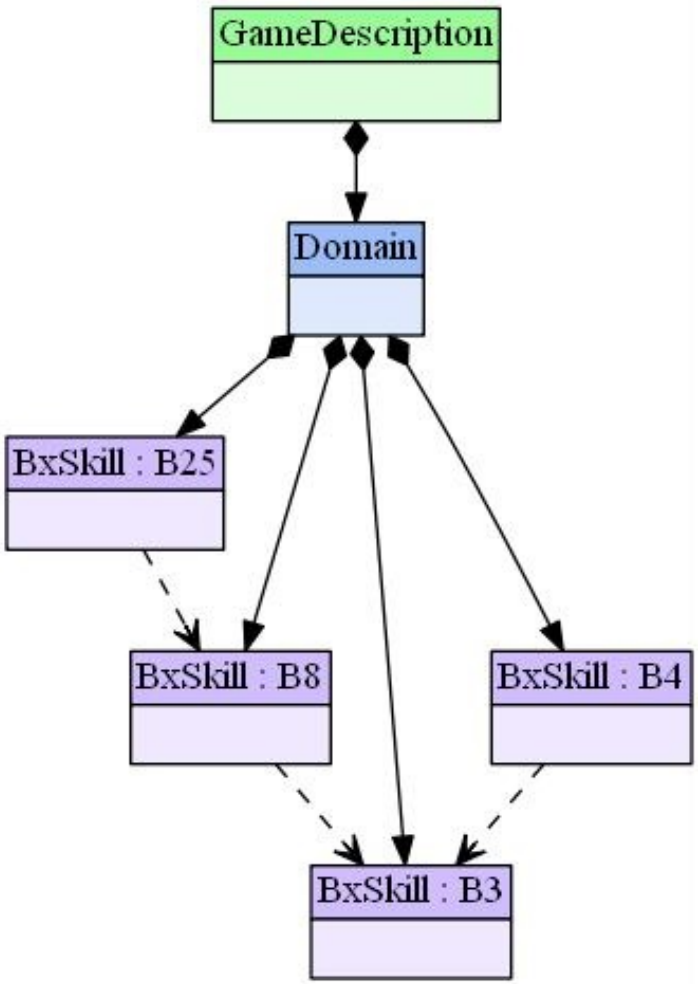
Transformation scenario

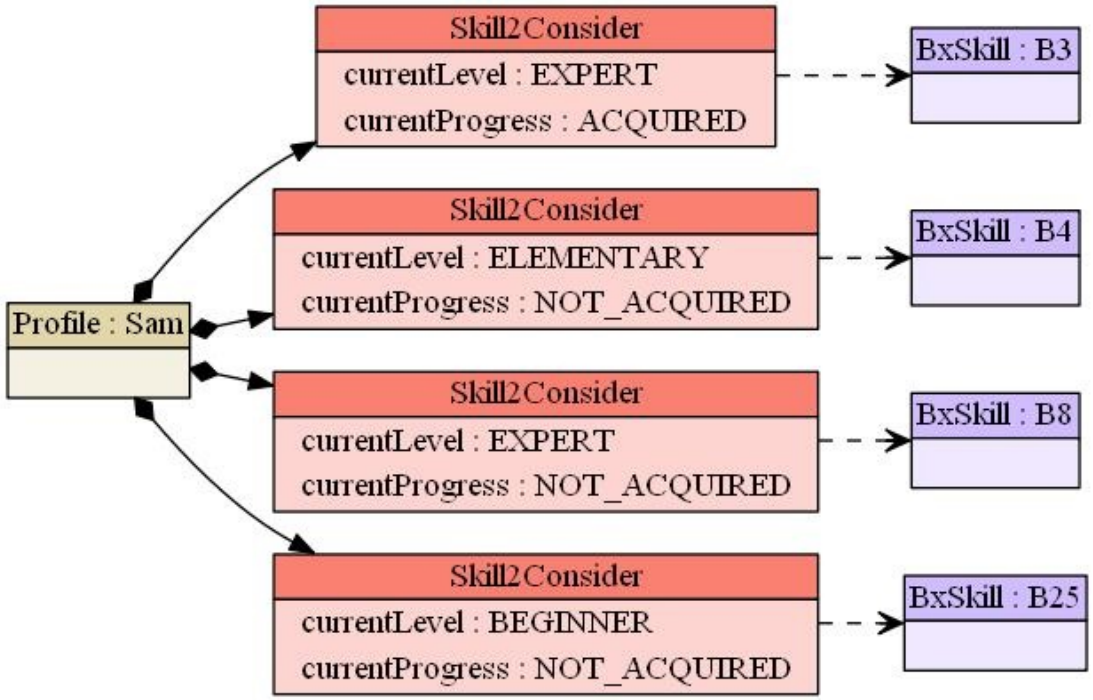
- Generation of the objective scenario :
  - Elicitation of the visual performance skills in accordance with the number of levels to generate, the considered skills and the child's progression.
  - Source models :
    - Game description.
    - Child profile.
- Constraints :
  1. The objective scenario has to include exactly four targeted skills.
  2. Targeted skills are proposed among the profile skills for which the child has acquired the corresponding prerequisites and did not reach a maximum level.
  3. It would be best to propose different targeted skills.
  4. The objective scenario must be arranged in view of excluding successor elements referencing the same skill.

Implied metamodel

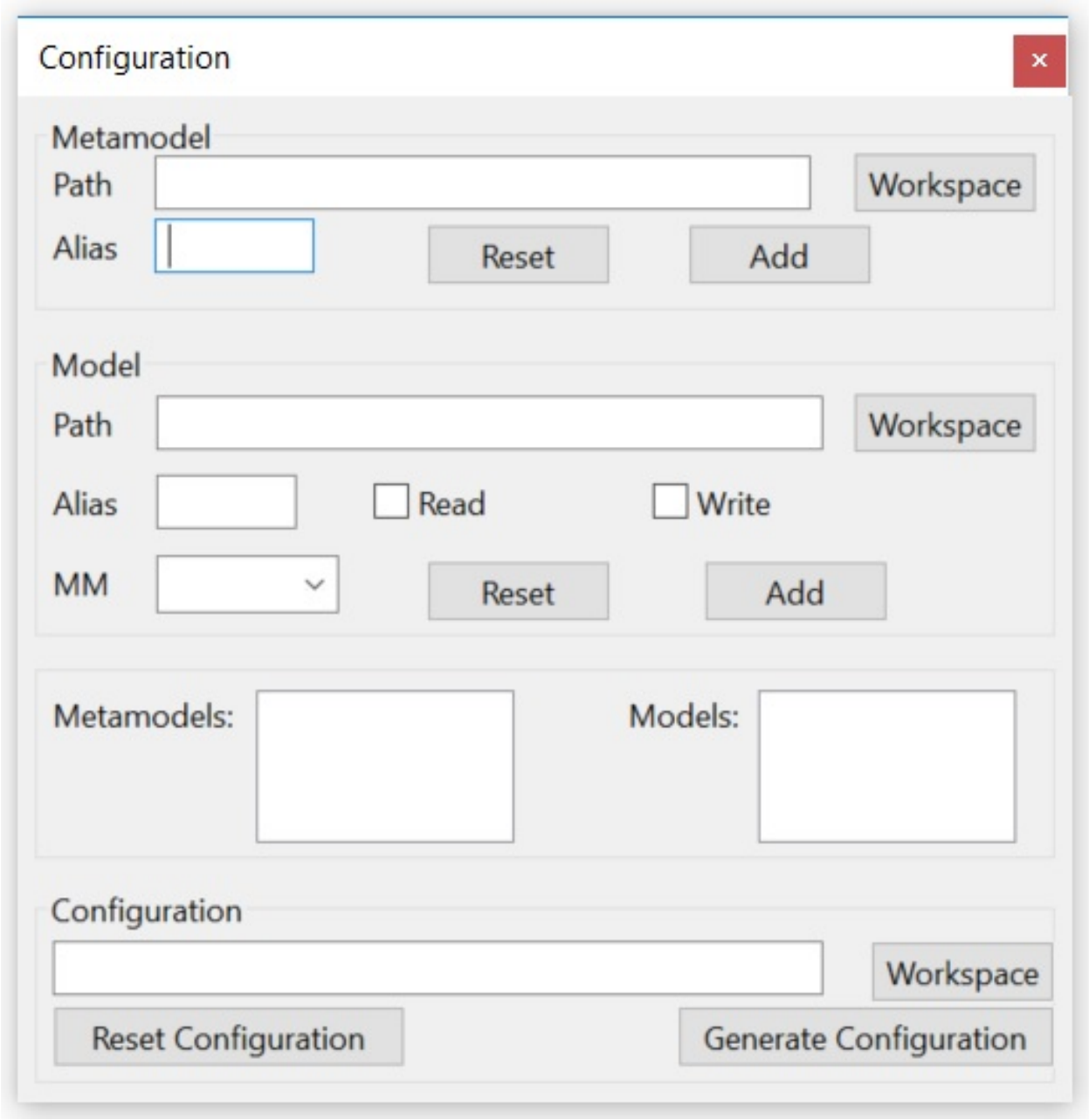


Source models





Generate configuration - UI



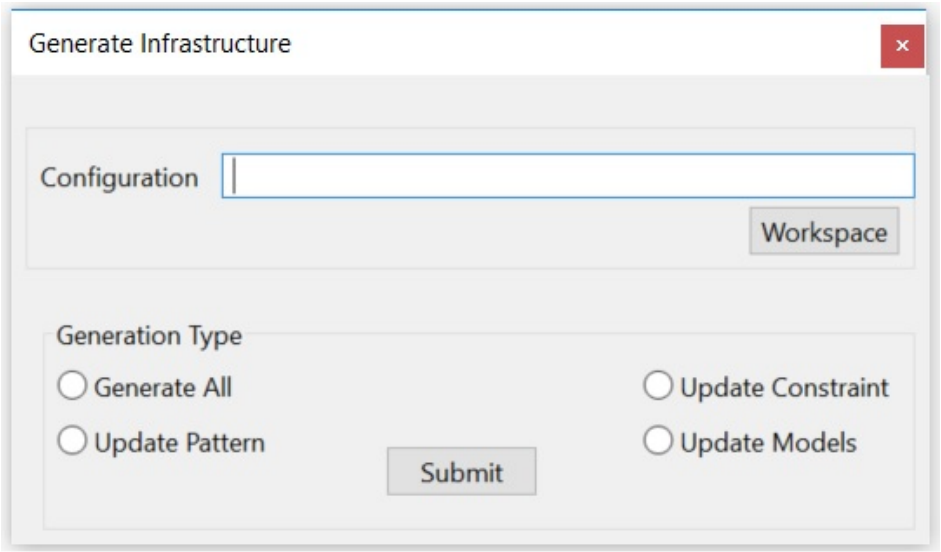
Generate configuration

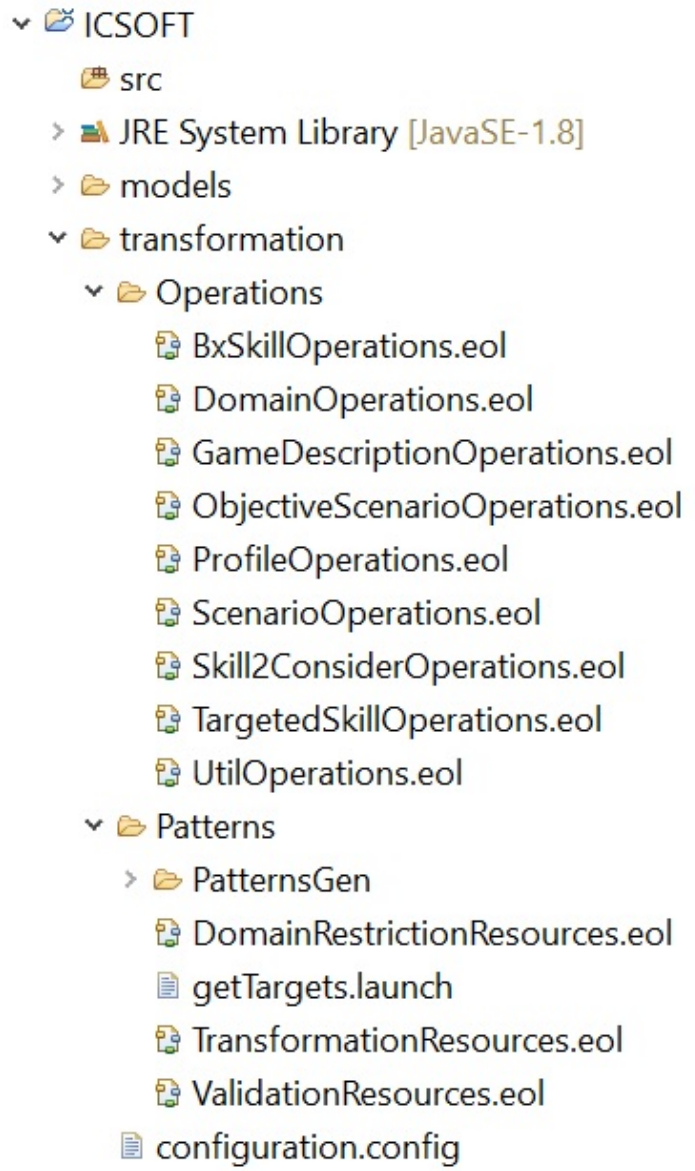
```
Configuration
{
Models:
    metamodel "mmcs" "C:/Eclipse/workspace/ICSOFT/models/MMCS.ecore"
        uri: "http://mmcs/1.0"
    Types:
        "GameDescription"
        "Domain"
        "BxSkill"
        "Skill2Consider"
        "Profile"
        "TargetedSkill"
        "ObjectiveScenario"
        "Scenario"
    model "m1" "C:/Eclipse/workspace/ICSOFT/models/Profile.xmi" read mmcs
    model "m2" "C:/Eclipse/workspace/ICSOFT/models/GameDescription.xmi" read mmcs
    model "m3" "C:/Eclipse/workspace/ICSOFT/models/Scenario.xmi" write mmcs
Pattern:

Constraint:

}
```

Complete configuration

```
Configuration
{
Models:
    metamodel "mmcs" "C:/Eclipse/workspace/ICSOFT/models/MMCS.ecore"
        uri: "http://mmcs/1.0"
    Types:
        "GameDescription"
        "Domain"
        "BxSkill"
        "Skill2Consider"
        "Profile"
        "TargetedSkill"
        "ObjectiveScenario"
        "Scenario"
    model "m1" "C:/Eclipse/workspace/ICSOFT/models/Profile.xmi" read mmcs
    model "m2" "C:/Eclipse/workspace/ICSOFT/models/GameDescription.xmi" read mmcs
    model "m3" "C:/Eclipse/workspace/ICSOFT/models/Scenario.xmi" write mmcs
Pattern:
    roleType m1!mmcs.Profile [1]
    roleType m1!mmcs.Skill2Consider [4]
Constraint:
    SimpleConstraint "allDifferent"
    SimpleConstraint "followingNotMatch"
}
```

Generate infrastructure





Complete resources - Domain restriction

```
/*
pattern Pattern
    r0 : m1!Profile          guard : r0.ProfileDomainRestriction(),
    r1 : m1!Skill2Consider   guard : r1.Skill2ConsiderDomainRestriction(),
    r2 : m1!Skill2Consider   guard : r2.Skill2ConsiderDomainRestriction(),
    r3 : m1!Skill2Consider   guard : r3.Skill2ConsiderDomainRestriction(),
    r4 : m1!Skill2Consider   guard : r4.Skill2ConsiderDomainRestriction()
{
match: validatePatternConstraint(r0,r1,r2,r3,r4)
onmatch
    {
    transformPattern(r0,r1,r2,r3,r4);
    }
}
*/

operation m1!Profile ProfileDomainRestriction(): Boolean{
    return true;
}
operation m1!Skill2Consider Skill2ConsiderDomainRestriction(): Boolean{
    return not self.maxLevel() and self.hasPrerequiste();
}
```

Complete resources - Validation

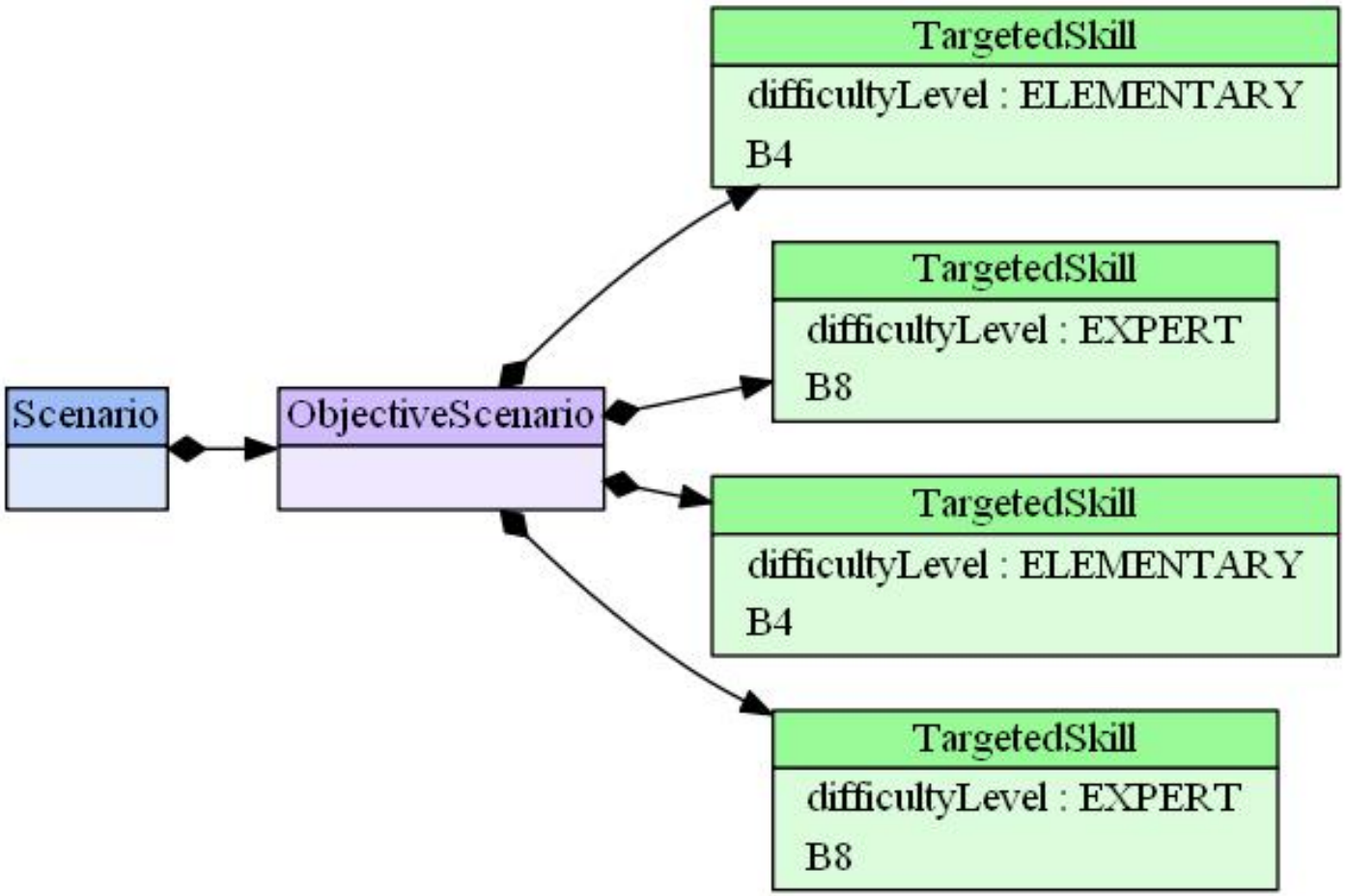
```
pattern Pattern
    r0 : m1!Profile          guard : r0.ProfileDomainRestriction(),
    r1 : m1!Skill2Consider   guard : r1.Skill2ConsiderDomainRestriction(),
    r2 : m1!Skill2Consider   guard : r2.Skill2ConsiderDomainRestriction(),
    r3 : m1!Skill2Consider   guard : r3.Skill2ConsiderDomainRestriction(),
    r4 : m1!Skill2Consider   guard : r4.Skill2ConsiderDomainRestriction()
{
match:x=true and validatePatternConstraint(r0,r1,r2,r3,r4)
onmatch
    {
    transformPattern(r0,r1,r2,r3,r4);
    x=false;
    }
}
*/

operation validatePatternallDifferent(r0 : m1!Profile,r1 : m1!Skill2Consider,
    r2 : m1!Skill2Consider,r3 : m1!Skill2Consider,r4 : m1!Skill2Consider) :Boolean{
        return allDifferent(Sequence{r1,r2,r3,r4});
}
operation validatePatternfollowingNotMatch(r0 : m1!Profile,r1 : m1!Skill2Consider,
    r2 : m1!Skill2Consider,r3 : m1!Skill2Consider,r4 : m1!Skill2Consider) :Boolean{
        return followingNotMatch(Sequence{r1,r2,r3,r4});
}
```

Complete resources - Transformation

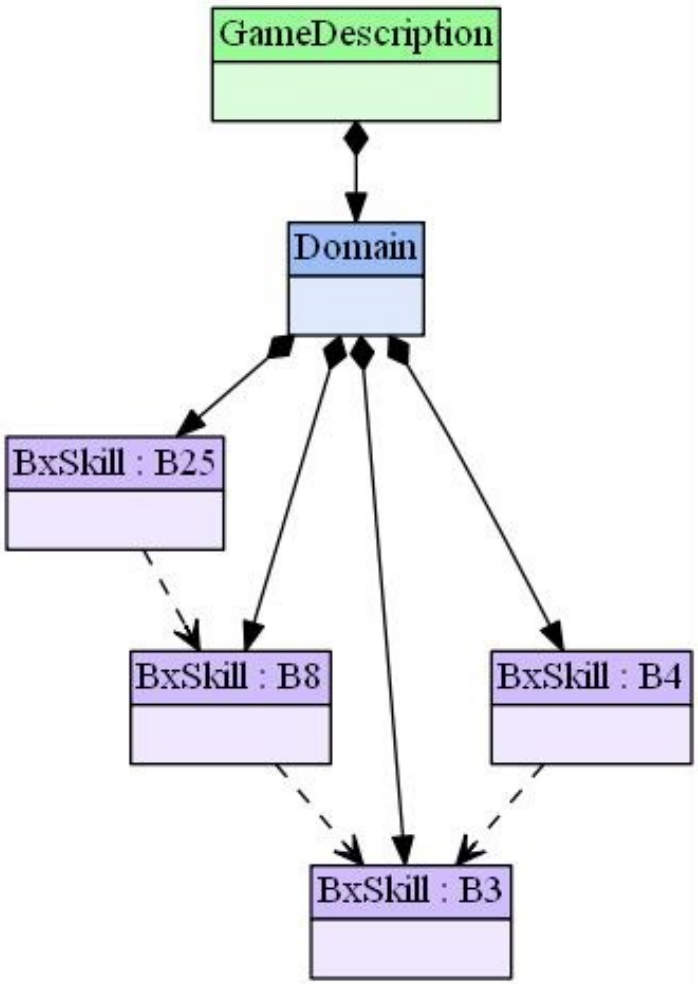
```
operation transformPattern(r0 : m1!Profile,r1 : m1!Skill2Consider,
    r2 : m1!Skill2Consider,r3 : m1!Skill2Consider,r4 : m1!Skill2Consider){
        var s=createS();
        var os=createOS();
        s.objectiveScenario=os;
        for(s in Sequence{r1,r2,r3,r4}){
            os.targetedSkills.add(createTS(s));
        }
}
```

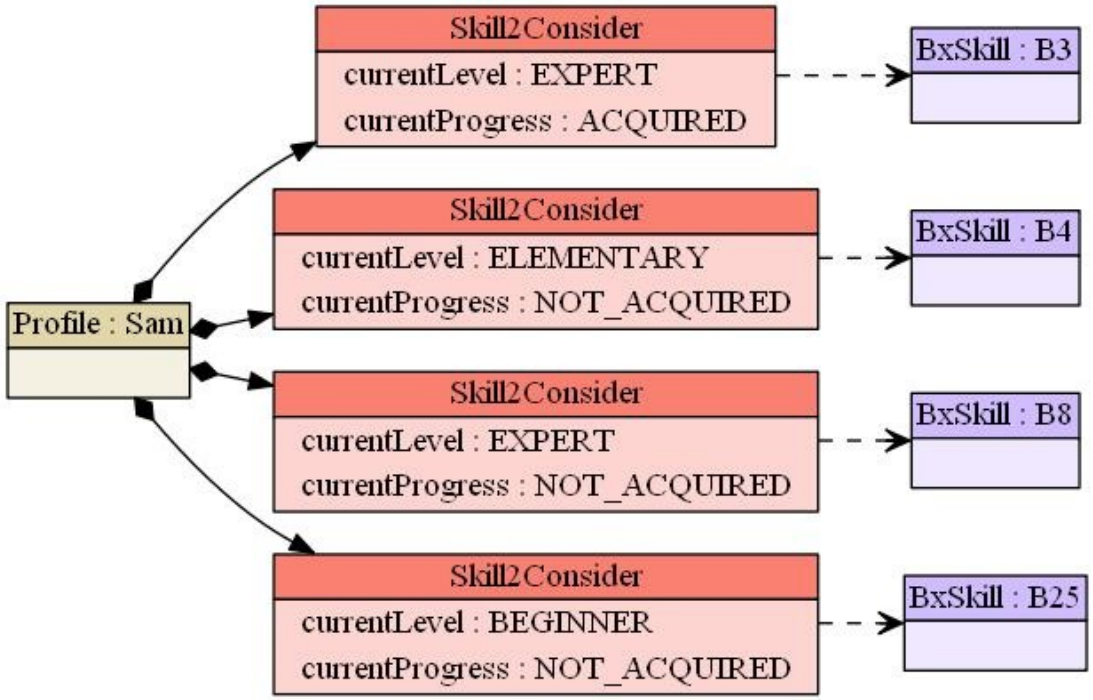
```
operation createS() : m3!Scenario{
    return new m3!Scenario;
}
operation createOS() : m3!ObjectiveScenario{
    return new m3!ObjectiveScenario;
}
operation createTS(s2c : m1!Skill2Consider) : m3!TargetedSkill{
    var ts= new m3!TargetedSkill;
    ts.difficultyLevel=s2c.currentLevel;
    ts.bxSkill=s2c.bxSkill;
    return ts;
}
```

Generate targets



Generate targets - Source models





Outline

Conclusion

- A practical approach for constraint solving in model transformation :
  - Pattern matching problem as a high level specification of CSP.
  - Specification of the satisfaction problem using model elements.
  - Pattern structure is decoupled from the validation constraints.
- Semi-automatic generation of a transformation infrastructure that supports the proposed approach :
  - Pattern specification.
  - Match model search.
  - Transformation of the match into valid target models.

Future work

- Support parameterized patterns.
- Configure the matching process :
  - First match.
  - Random match.
  - All matches.
- Naturally express the satisfaction problem by means of target model elements :
  - Define the related pattern using target model elements.
  - Automatically generate the equivalent source pattern based on relevant inputs :
    - Source-target correspondences.
    - One time or multiple matching mechanism.
    - ...