Heuristic-based Recommendation for Metamodel–OCL Coevolution

Edouard Batot, Wael Kessentini, Houari Sahraoui, Michalis Famelis

DIRO, Université de Montréal
To exclude the bad ones, we need this OCL constraints:

**Context Person:**

- `inv inheritance_bares_no_loop_mum:
  self.mother->closure() ->excludes(self)`
- `inv inheritance_bares_no_loop_dad:
  self.father->closure() ->excludes(self)`

A person cannot be its own mother or father ... transitively.
Schematically:

Metamodel

Type graph

OCL
OCL Coevolution

What happens if the type model changes?

Name change in type graph impacts OCL constraints

Context Person:

inv inheritance_bares_no_loop_mum:
  self.mum->closure()->excludes(self)
inv inheritance_bares_no_loop_dad:
  self.father->closure()->excludes(self)

• More complicated example of evolution

self.nameCategory

self.category.nameCategory

Employee

| name: Estring |
| age: EInt |
| salary: Edouble |
| nameCategory: EString |

Indirection inserted

Employee

| name: Estring |
| age: EInt |
| salary: Edouble |

Category

| nameCategory: EString |
Problem: OCL Coevolution

Usually:
- Force determinism, or
- Trace rationale behind high-level changes
Our Approach

Heuristic search

Recommendation
Additional Benefits

• Does **not** depend on tracing of **high level** changes (tedious, error-prone)

• Search a large **solution space** (chance for innovation)

• Extensible **framework** for new coevolution strategies (as new mutation ops)
Non-sorting Genetic Algorithm – II (NSGA-II)
Heuristic Search: Genetic programming

• Representation: a solution is a set of OCL constraints

• Each constraint: Ecore Abstract Syntax Tree

• In each generation, OCL evolved by crossovers and mutations

• End condition: Stabilization, at 300 iterations
Heuristic Objectives

1. Minimize change
   • # of mutations/crossover applied from original

2. Minimize syntax error
   • # of rules fired (Not strict zero to loosen the search)

3. Minimize information loss
   • # of metamodel elements removed during OCL evolution
Genetic operators

• Solution representation and creation
  • **Initial generation = original set of OCL constraints**

• Point-cut crossover

![Diagram showing point-cut crossover between Parent 1 and Parent 2 to produce Child 1 and Child 2 with a point-cut](image)

• Mutation patterns
  • Renaming
  • Context change
  • Pruning
  • Indirection insertion
  • Change typing method

![Diagram showing mutation from Parent to Child with an insertion of constraint C1-4](image)

- Constraint: `C`
Output Set: Solutions

- NSGA-II: Multi-objective non-dominant search

- Output: **Pareto front of solutions**
  - No total ordering
  - All solutions in the front are “equally good”
  - No solution dominates

- Output can be too large to present to user

- Instead: generate recommendations
Outline

Heuristic search

Recommendation
Recommending solutions

- NSGA-II gives many solutions (Pareto front)
  Which one is of user’s interest?

Two strategies:
- **Ranking solutions** using fitness objectives
- **Clustering solutions** with syntactic comparison Levenstein distance

![Diagram showing Pareto front, clusters, centroid, and recommendations](image)
Evaluation

- **RQ0**: Are the results of our approach attributable to the search strategy or to the number of explored solutions?
- **RQ1**: To which extent our approach finds the expected solution?
- **RQ2**: To which extent our approach recommends the expected solution?

**Setup**
- **3 metamodels** of different sizes (*Family, State Machine, Project Management*)
- **30 executions** for each metamodel/OCL couple
- **300 iterations** with a population of **100 set of constraints**
RQ0: Sanity check

- Better than random search
- Same number of solutions explored, way better results

<table>
<thead>
<tr>
<th></th>
<th>Average Random search</th>
<th>Average Our Approach</th>
<th>Mann Witney p-value</th>
<th>Effect size Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
<td>2,25</td>
<td>3,97</td>
<td>&lt;0,001</td>
<td>4,35</td>
</tr>
<tr>
<td>State Machine</td>
<td>1,17</td>
<td>4,53</td>
<td>&lt;0,001</td>
<td>4,87</td>
</tr>
<tr>
<td>Project Management</td>
<td>1,83</td>
<td>4</td>
<td>&lt;0,001</td>
<td>3,11</td>
</tr>
</tbody>
</table>
RQ1: Algorithm recall

# of expected solutions found

- Family and State Machine cases: solutions found
- Project Management case contains two missing constraints that require particularly complex changes
- To address such changes we aim at expanding the mutation operator store
RQ2: Recommendation system precision

<table>
<thead>
<tr>
<th>Family</th>
<th>State Machine</th>
<th>Project Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters</td>
<td>Clusters</td>
<td>Clusters</td>
</tr>
<tr>
<td>Ranking</td>
<td>Ranking</td>
<td>Ranking</td>
</tr>
<tr>
<td>Pareto</td>
<td>Pareto</td>
<td>Pareto</td>
</tr>
</tbody>
</table>

# of expected solutions recommended

- Accuracy of the recommender grows with the number of recommendations
- No dramatic increase beyond 7
- Simple ranking is a better recommendation strategy
  - Clustering is computationally costlier
Conclusion

• Metamodel-OCL Coevolution is crucial to DSML design and maintenance

• Our approach
  1. Multi-objective optimization problem
  2. Recommendation of a subset of generated solutions

• Benefits
  • Does not depend on tracing high-level changes
  • Does not assume single solution
  • Explores a large solution space
  • Extensible with new coevolution strategies
  • High recall; Efficient ranking-based recommendation