Optimisation Methods for Model-Driven Engineering

Alexandru Burdusel Steffen Zschaler

Department of Informatics
King's College London

alexandru.burdusel@kcl.ac.uk
szschaler@acm.org

October 5, 2018

Who am I?

PhD Student at King's College London

Research Interests

- **Model-Driven Engineering**
- **Domain-Specific Modelling Languages**
- **Search-Based Model Engineering**

Model-Driven Engineering

MDE promotes the use of **models** as abstract representations of complex systems.

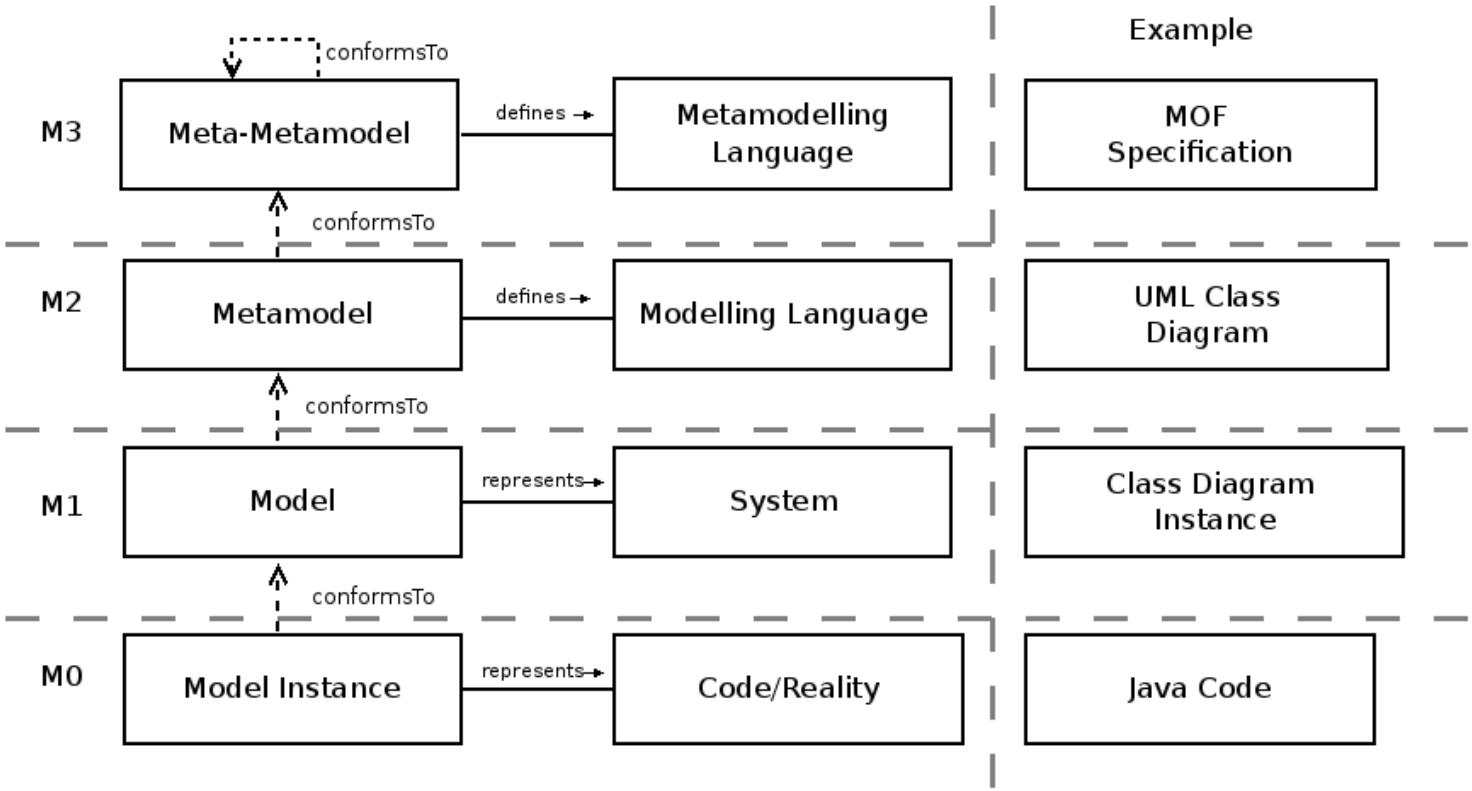


Figure: MDE Layers Stack

Optimisation Problems

Optimisation problems are generally difficult to solve.

Common signature:

- A set of constraints
- A set of control variables
- A (set of) objective function(s) to evaluate the quality of a given solution

Examples in **Software Engineering**:

- Architecture refactoring
- Sprint planning
- Component deployment

Example in **Pattern Databases**:

- Pattern selection
- Other unexplored uses?

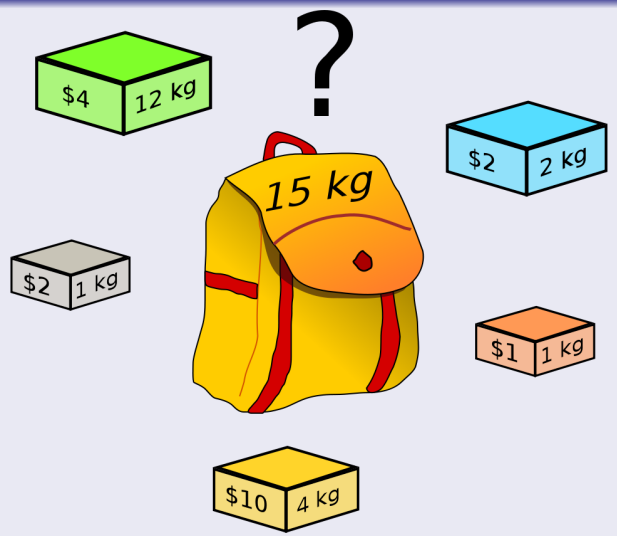
Focus on Combinatorial Optimisation Problems

A distinctive classification of optimisation problems are **Combinatorial Optimisation** (CO) problems.

Combinatorial Otimisation Problem

A combinatorial optimisation problem is a minimisation or maximisation problem consisting of (Garey 1979):

- A domain of instantiations;
- A set of possible candiate solutions for each domain instance;
- A fitness function for solution quality evaluation.



Knapsack Problem

Running example - Class-Responsibility Assignment (CRA)

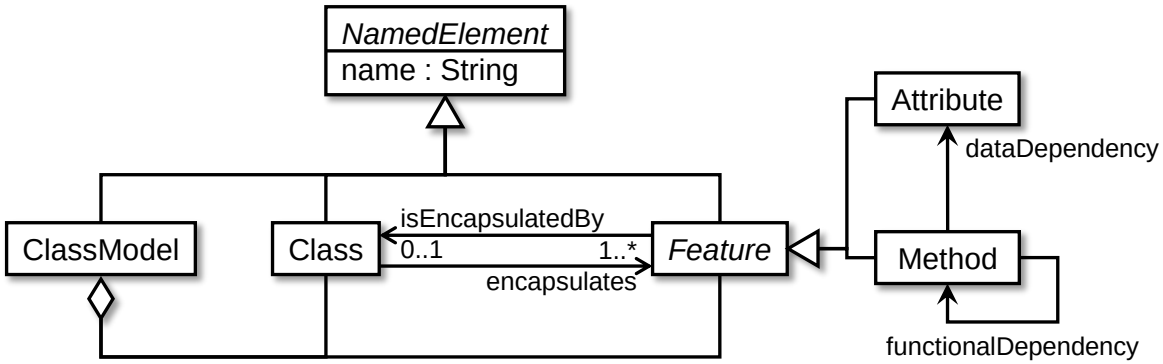
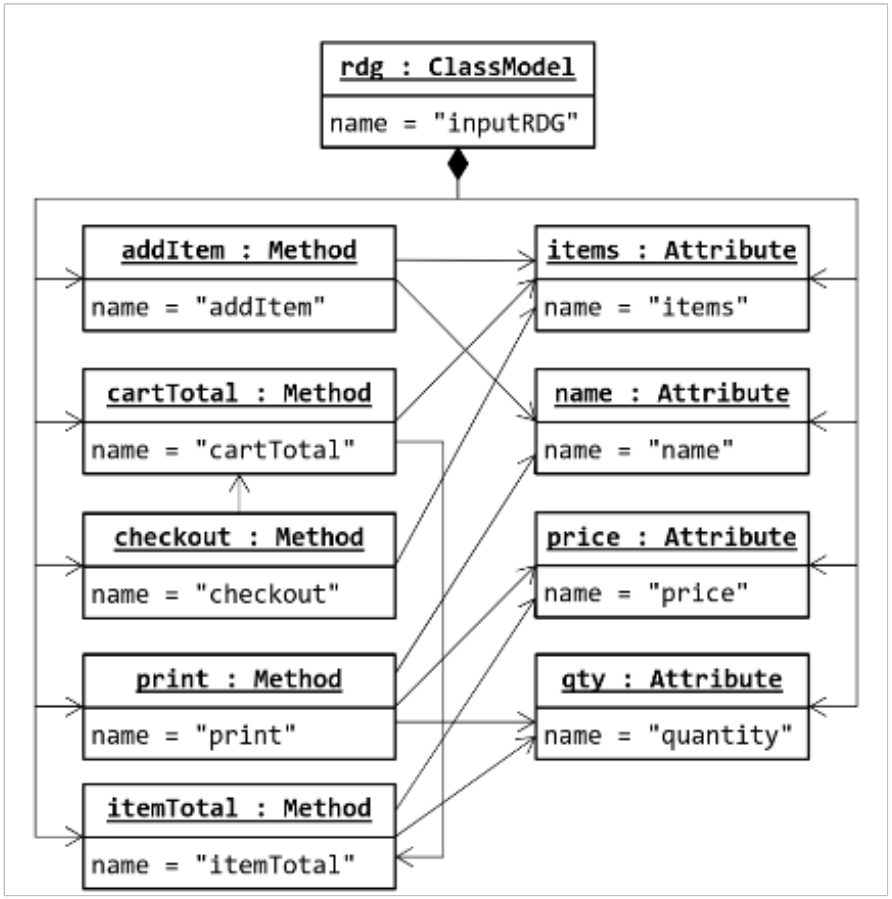
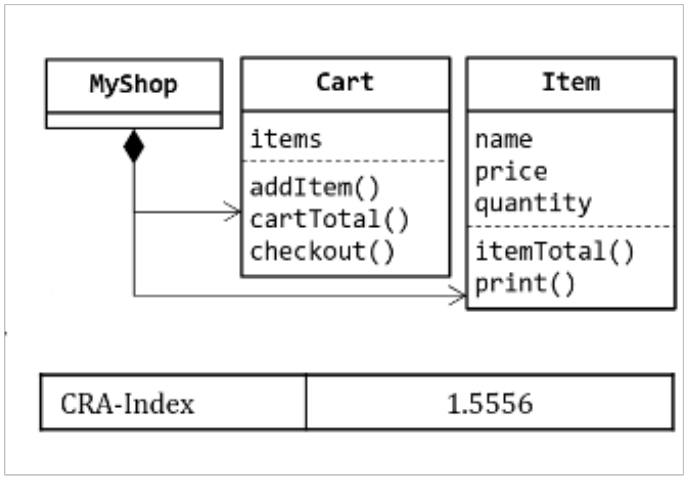


Figure: CRA Problem Metamodel

- The goal of this problem is to create optimal object-oriented designs.
- To solve the CRA problem, we need to decide to which **Class** responsibilities(**Features**), i.e., functions and attributes belong.
- Example CRA problems:
  - Migrate from procedural code (e.g. COBOL) to object-oriented code (e.g. Java)
  - Refactor existing "spaghetti code"
- Huge search space. This is a partitioning problem.

An Example CRA Problem





Search Based Optimisation (SBO)

In our approach we currently focus on solving optimisation problems using **Evolutionary Search** (ES).

Not the only/best way to solve all optimisation problems, but a promising start for CO problems.

To specify an ES problem, the following *key* ingredients are needed:

- A **search space description**, which includes all the possible solutions to the problem being solved;
- A method for **encoding** individual solution candidates;
- A set of **search operators**;
- A method for **evaluating the quality** of the individual solutions based on the optimisation goals specified in the problem.

SBO Solution Encoding Problem

A challenge in ES is the encoding (representation) of the solution candidates such that search operators can generate new solutions from existing ones.

Common challenges:

- Search operators applied to binary vector representations can sometimes **generate invalid solutions**, requiring an additional repair step after application;
- The **Genotype-Phenotype** translation can be a computationally expensive step;
- Encoded problems become incomprehensible to the domain expert.

Search-Based Model Engineering (SBME) (1)

MDE already has an encoding that works for domain experts. Why not use it for running search directly on models?

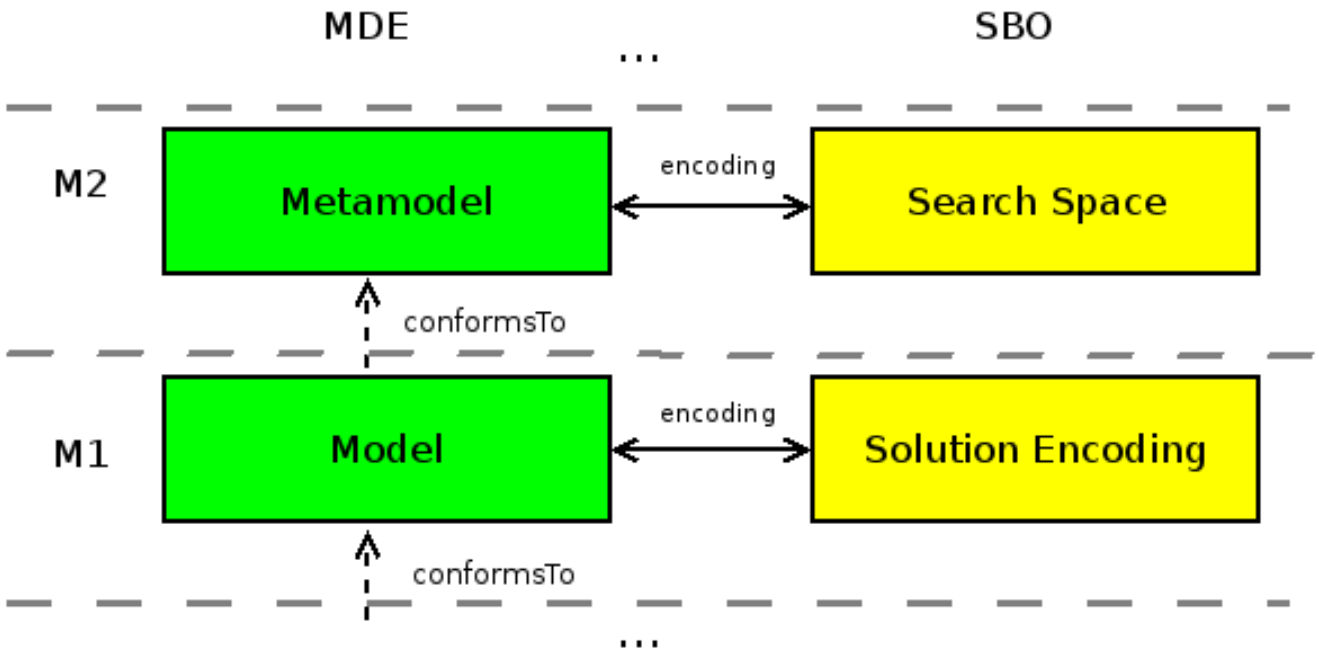


Figure: Use models as solution encodings

SBME (2)

Model Transformations as Search Operators

In-place model transformations can be used as consistency preserving search operators that run directly on the phenotype. Model transformations can be used to specify both mutation and breeding operators.

Model Querying as Fitness Functions

The fitness of solution candidates encoded as MDE models can be evaluated using model querying languages such as OCL or using Java. Additional problem constraints can be specified as model refinements or Java classes.

Transformation Rules as Search Operators

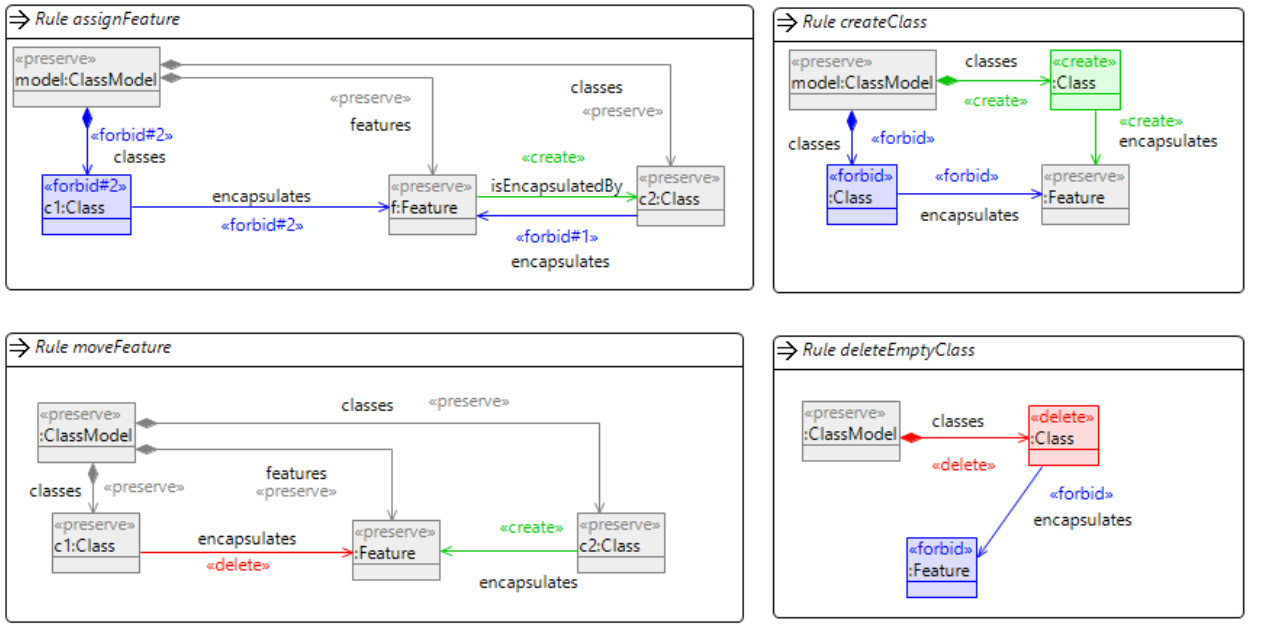


Figure: Manually specified mutation operators for the CRA case

- MDE models as **typed attributed graphs**
- **Henshin** model transformation language
- A graph transformation rule is formed of a Left Hand Side(LHS), a Right Hand Side(RHS) and a set of Application Conditions

SBME Key Idea

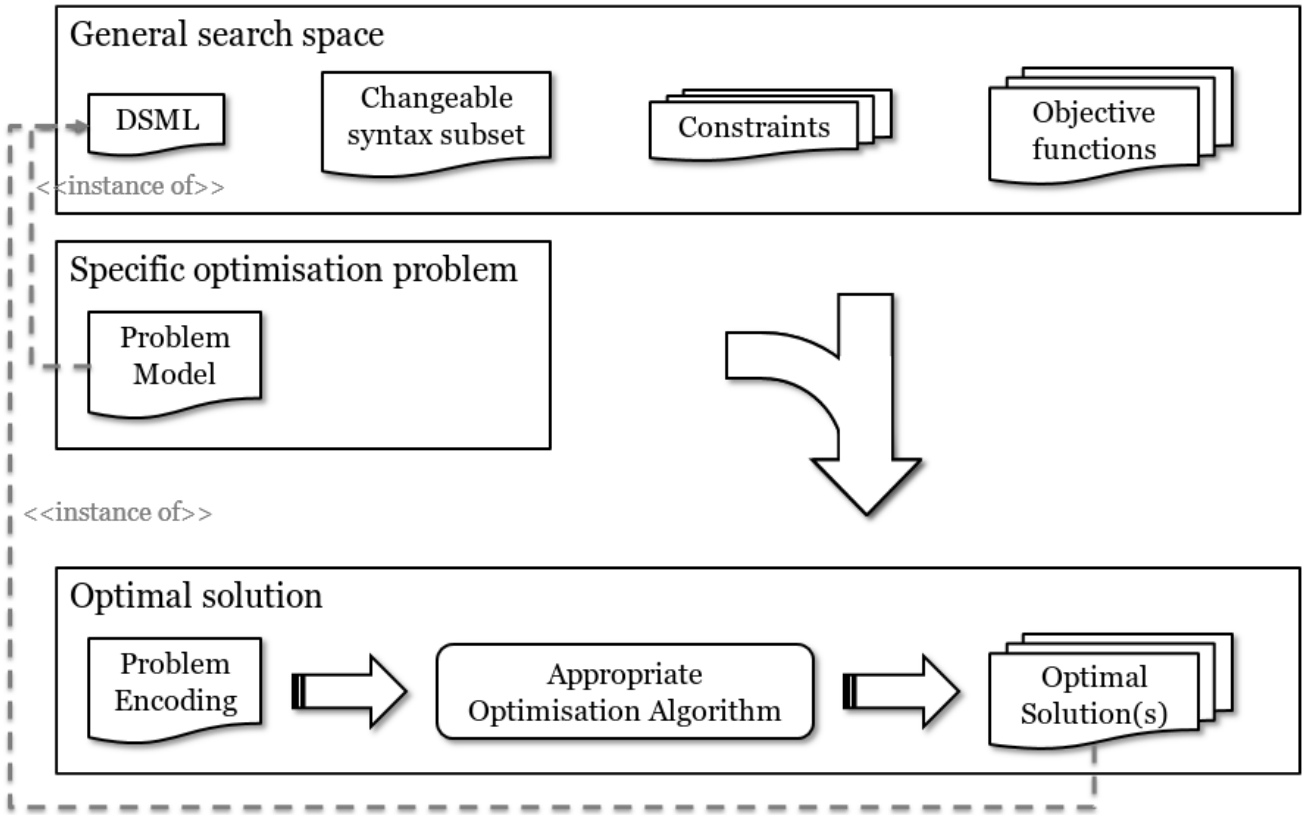


Figure: SBME Workflow

SBME Using Evolutionary Search

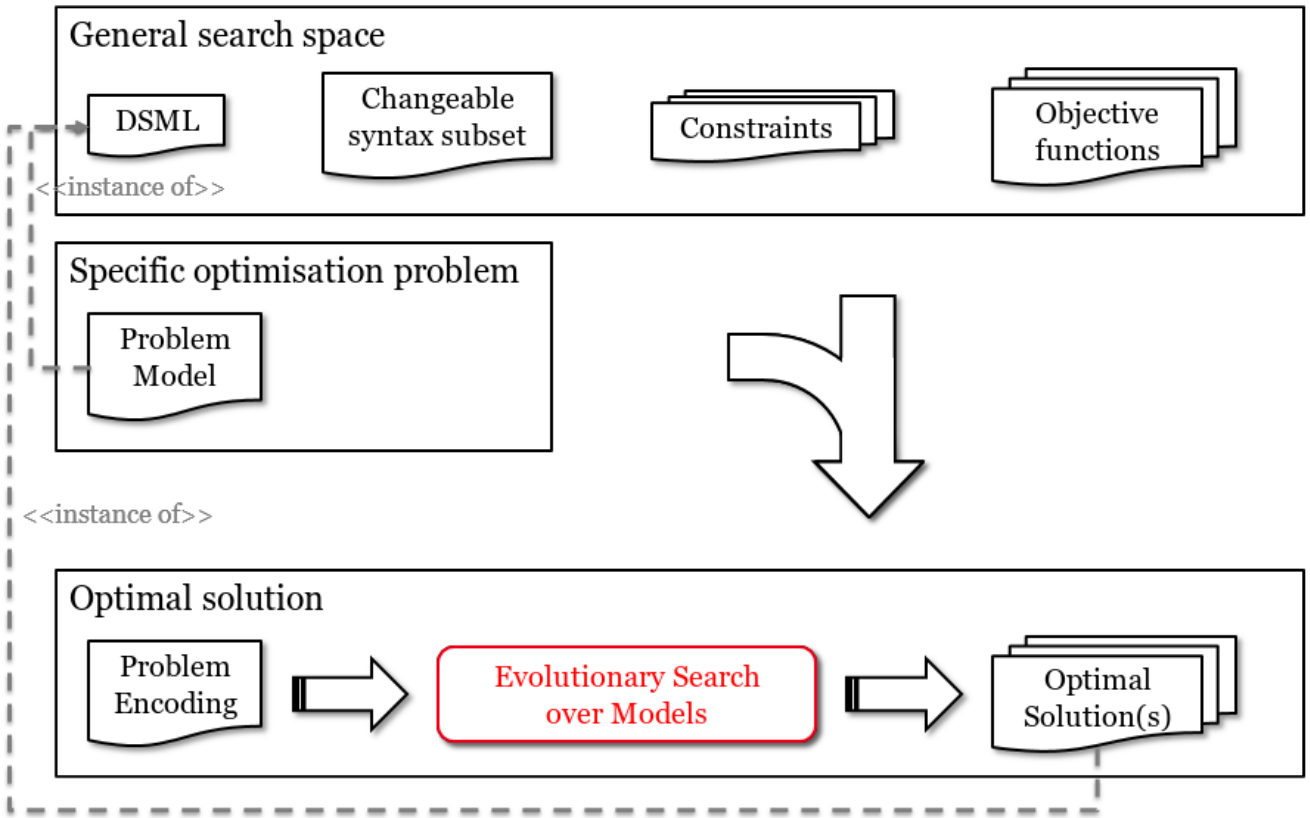


Figure: SBME Workflow

SBME Search Over Models

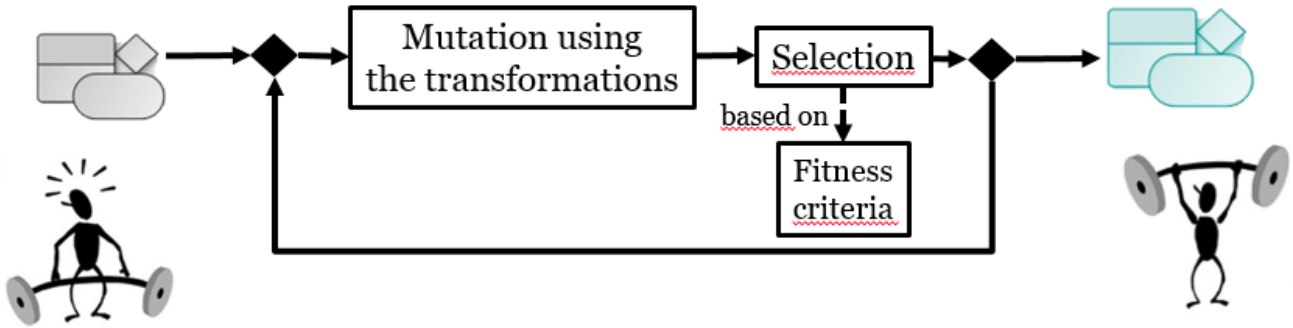


Figure: MDEOptimiser Evolutionary Search Workflow

Current Research

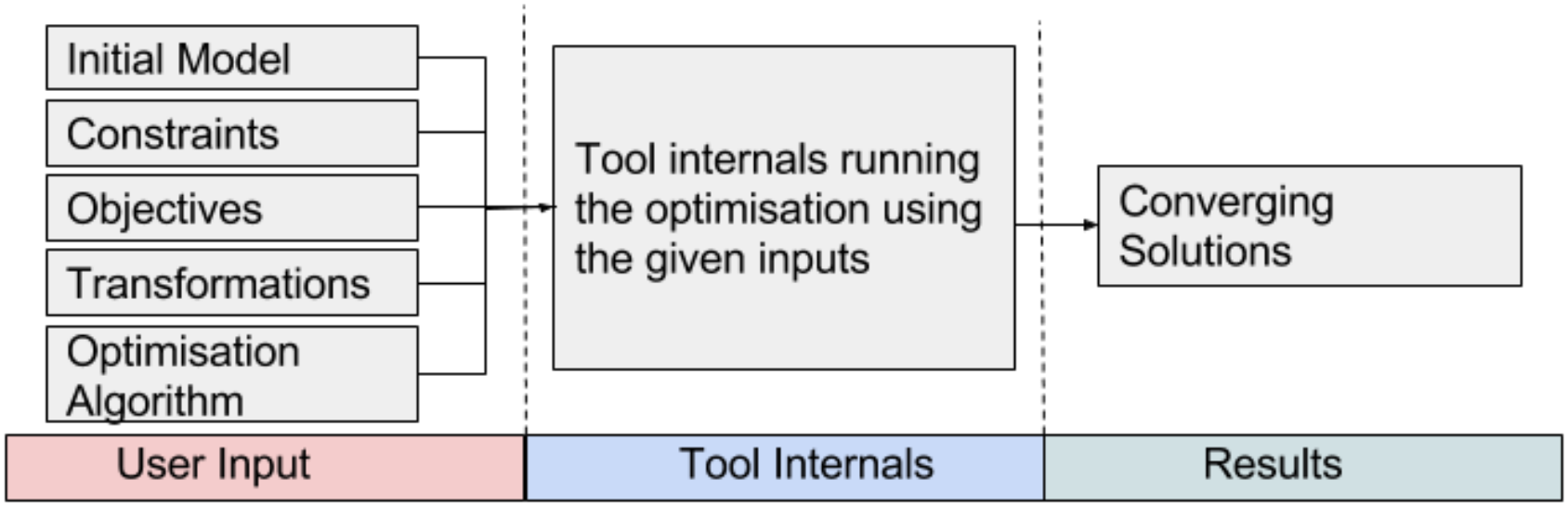


Figure: Generic MDE-SBO tools architecture

Problem: Manually specifiying mutation operators is not trivial.

RQ: Can we run optimisation on models without explicitly specifying mutation operators?

Running example - MDEOptimiser DSL

```
1  basepath <src/main/resources/models/cra/>
2  metamodel <architectureCRA.ecore>
3  model <TTC_InputRDG_D.xmi>
4  objective CRA maximise java { "models.cra.fitness.MaximiseCRA" }
5  constraint MinimiseClasslessFeatures java { "models.cra.fitness.MinimiseClasslessFeatures" }
6  mutate using <craEvolvers.henshin> unit "createClass"
7  mutate using <craEvolvers.henshin> unit "assignFeature"
8  mutate using <craEvolvers.henshin> unit "moveFeature"
9  mutate using <craEvolvers.henshin> unit "deleteEmptyClass"
10 optimisation provider moea algorithm NSGAII variation mutation parameters {
11     population: 40
12 }
13 termination {
14     time: 60
15 }
16 batches 1
```

Figure: MDEOptimiser - CRA Problem Specification

- Running Evolutionary Search with mutation only
- The mutation operators specified using **mutate** keyword in the DSL

Current Research - Automatic mutation generation

```
1  basepath <src/main/resources/models/cra/>
2  metamodel <architectureCRA.ecore>
3  model <TTC_InputRDG_A.xmi>
4  refine metamodel {"Feature", "isEncapsulatedBy", 1, 1}
5  objective CRA maximise java { "models.cra.fitness.MaximiseCRA" }
6  constraint MinimiseClasslessFeatures java { "models.cra.fitness.MinimiseClasslessFeatures" }
7  mutate {"Class"}
8  optimisation provider moea algorithm NSGAII variation mutation parameters {
9      population: 40
10 }
11 termination {
12     time: 60
13 }
14 batches 1
```

Figure: MDEOptimiser DSL for automatic mutation generation

Automatic Mutation Generation - CRA Results

Table: Summary of MDEO TTC '16 input models

	A	**B**	**C**	**D**	**E**
Attributes	5	10	20	40	80
Methods	4	8	15	40	80
Data Dep.	8	15	50	150	300
Functional Dep.	6	15	50	150	300

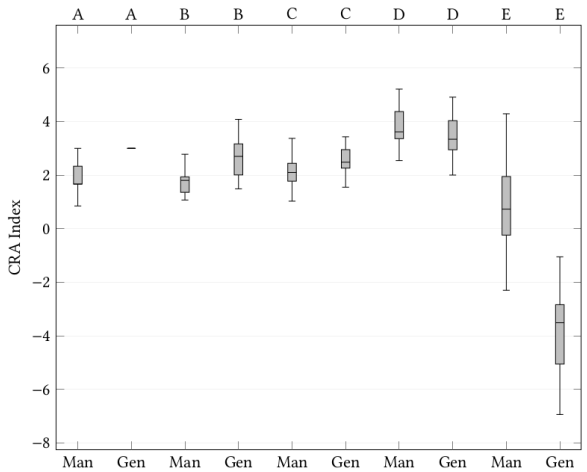


Figure: CRA values comparison for Manual and Generated mutations

Future work

- **Model Evolutions** - Generate more powerful evolution operators.
- **Algoritm Selection** - Problem based algorithm selection
- **Case Studies** - Implement more case studies to validate the proposed solutions.
- **Tool Improvement** - Improve the functionality of the tool by making it easy for the users to use the DSL for the specified use cases.

Questions?

Email: alexandru.burdusel@kcl.ac.uk
KCL Site: https://nms.kcl.ac.uk/alex.burdusel/
MDEO Website: http://mde-optimiser.github.io