Henshin: A Model Transformation Language and its Use for Search-Based Model Optimisation in MDEOptimiser

Part 2

Daniel Strüber[1], Alexandru Burdusel[2], Stefan John[3], Steffen Zschaler[2]

[1] Universität Koblenz-Landau,
[2] King's College London, [3] Philipps-Universität Marburg

Fachtagung Modellierung
February 21, 2018

Overview

- Part 1: Henshin: A Guided Tour
  - Language
  - In Action (interactive)
  - Features
  - Applications
- Part 2: Henshin in Search-Based Model Optimization
  - Background
  - MDEOptimiser
  - In Action
    - Case 1: Class Responsibility Assignment (interactive)
    - Case 2: SCRUM Planning (interactive)

Optimization problems in software engineering

Architecture refactoring

Sprint planning

Component deployment

Optimization problems in software engineering

Architecture refactoring

Sprint planning

Component deployment

Common task: find an optimal solution among a vast number of candidates

Optimization problems in software engineering

Architecture refactoring

Sprint planning

Component deployment

Solutions

Optimality

Optimization problems in software engineering

Architecture refactoring

Sprint planning

Component deployment

Solutions

Assignment
Classes→
Packages

Optimality

Optimization problems in software engineering

			
	Architecture refactoring	**Sprint planning**	**Component deployment**
Solutions	Assignment Classes→ Packages		
Optimality	max. Cohesion min. Coupling		

Optimization problems in software engineering



	Architecture refactoring	Sprint planning	Component deployment
Solutions	Assignment Classes→ Packages	Assignment Work Items→ Sprints	
Optimality	max. Cohesion min. Coupling		

Optimization problems in software engineering

	 **Architecture refactoring**	 **Sprint planning**	 **Component deployment**
Solutions	Assignment Classes→ Packages	Assignment Work Items→ Sprints	
Optimality	max. Cohesion min. Coupling	max. Items/Sprint max. Customer Satisfaction	

Optimization problems in software engineering

	Architecture refactoring	**Sprint planning**	**Component deployment**
Solutions	Assignment Classes→ Packages	Assignment Work Items→ Sprints	Assignment Components→ Hosts
Optimality	max. Cohesion min. Coupling	max. Items/Sprint max. Customer Satisfaction	

Optimization problems in software engineering

	Architecture refactoring	Sprint planning	Component deployment
Solutions	Assignment Classes→ Packages	Assignment Work Items→ Sprints	Assignment Components→ Hosts
Optimality	max. Cohesion min. Coupling	max. Items/Sprint max. Customer Satisfaction	min. Price min. Overhead

Example: Class Responsibility Assignment (CRA)

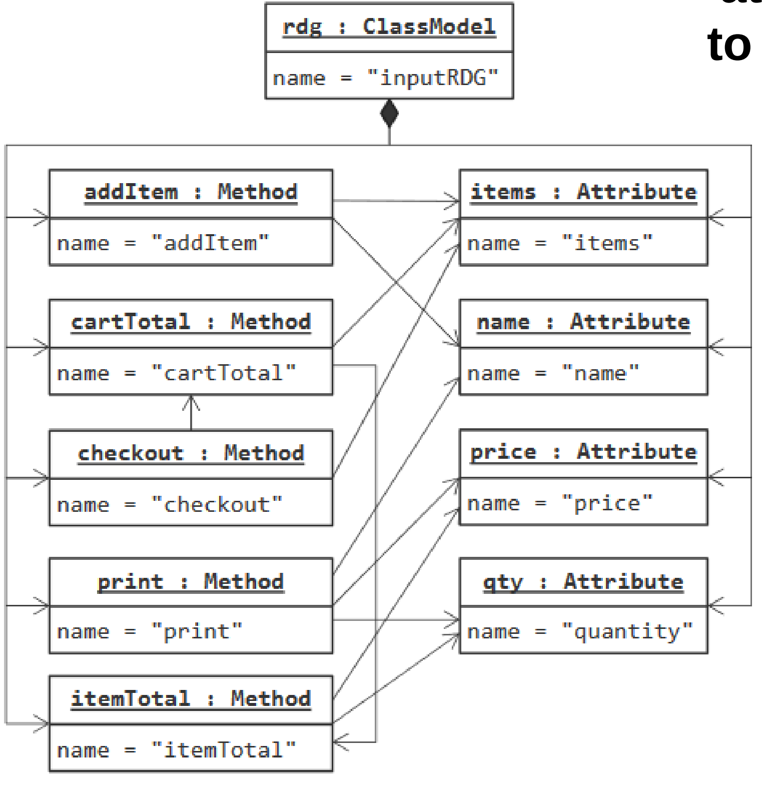
- **CRA** deals with the creation of **high-quality object-oriented models**
- For **solving** a particular CRA problem, one needs to decide **where responsibilities**, i.e., class operations and attributes, **belong**
- **When** do we have to deal with CRA problems?
  - **Generating class diagrams**: When migrating an application from a procedural language to an object-oriented language
  - **Optimizing class diagrams**: During the refactoring of an existing object-oriented model
- CRA is a **computationally challenging problem**
  - Huge search space!
  - Considered as an optimization problem

Courtesy of Fleck et al. [TTC 2016]

Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes

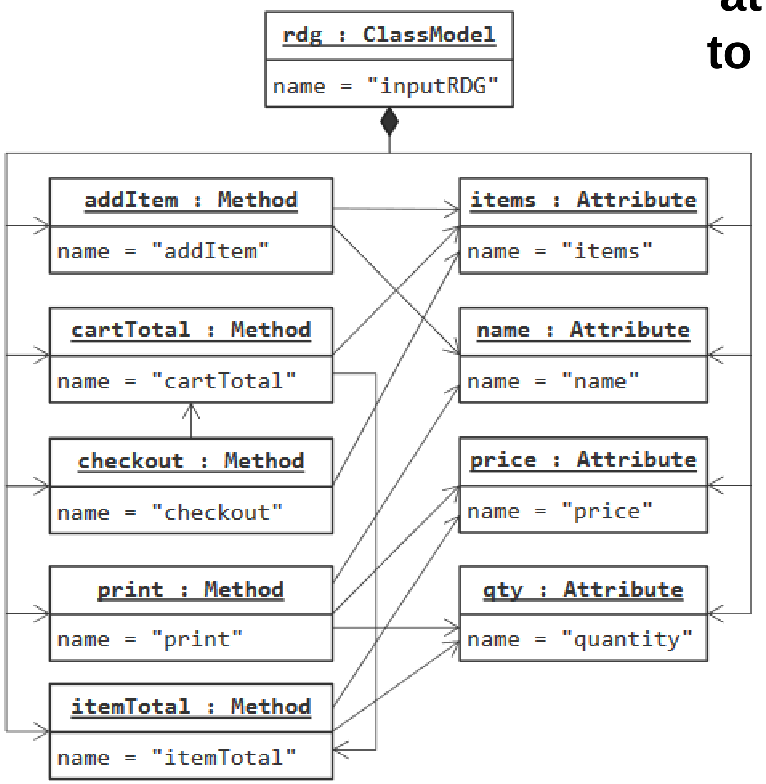


Courtesy of Fleck et al. [TTC 2016]

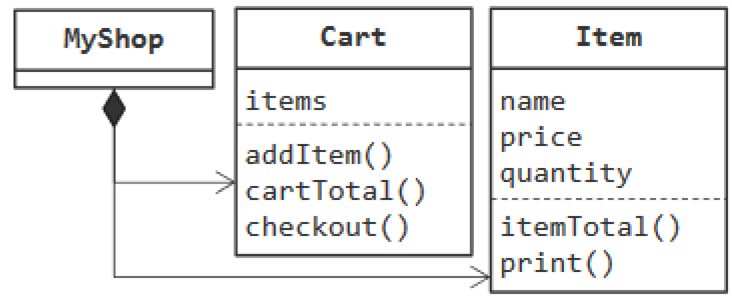
Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes



Example solution



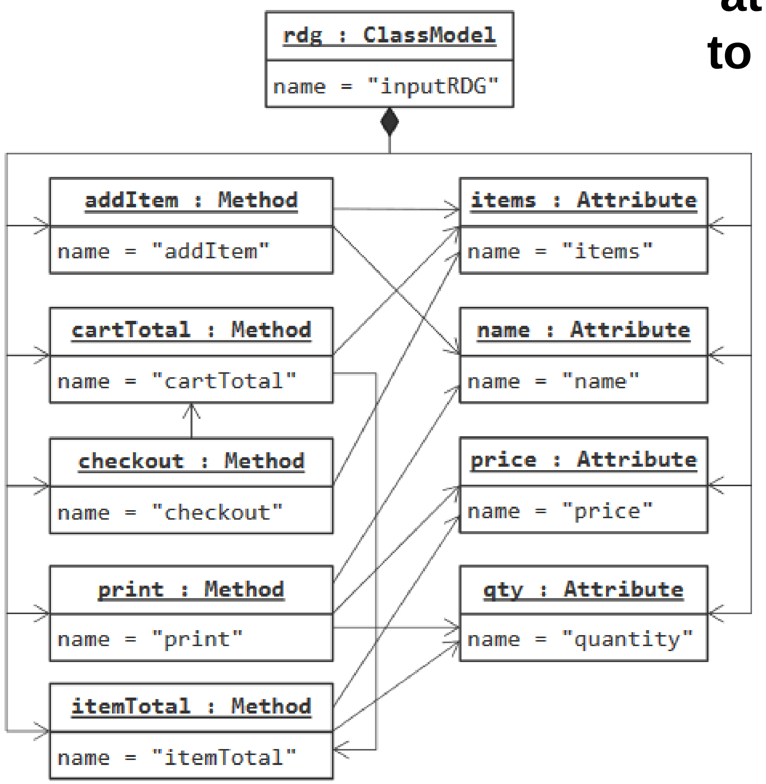
	Cart	Item
MAI(c_i, c_i)	3	5
MMI(c_i, c_i)	1	0
CohesionRatio	1.1667	0.8333
MAI(c_i, c_j)	1	0
MMI(c_i, c_j)	1	0
CouplingRatio	0.4444	0
∑ CohesionRatio	2	
∑ CouplingRatio	0.4444	
CRA-Index	1.5556	

Courtesy of Fleck et al. [TTC 2016]

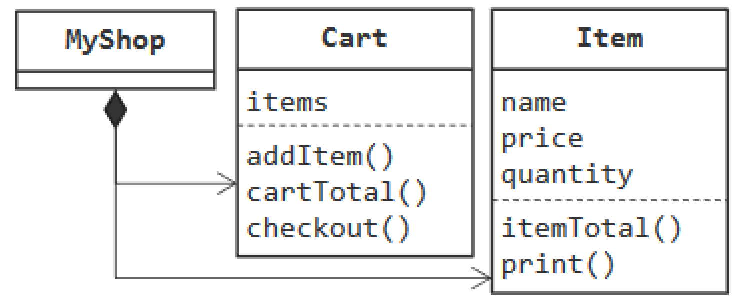
Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes



Example solution



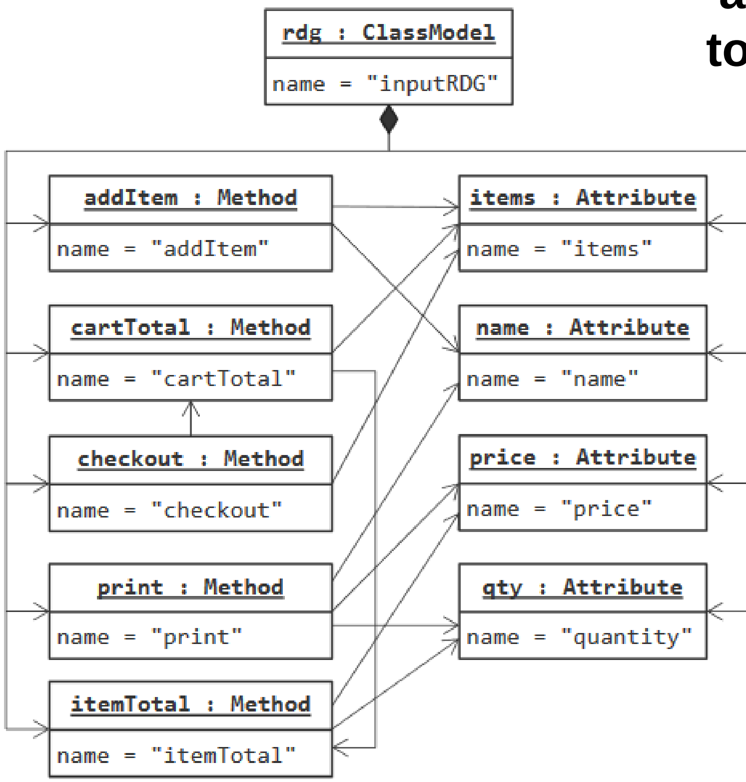
	Cart	Item
MAI(c_i, c_i)	3	5
MMI(c_i, c_i)	1	0
CohesionRatio	1.1667	0.8333
MAI(c_i, c_j)	1	0
MMI(c_i, c_j)	1	0
CouplingRatio	0.4444	0
∑ CohesionRatio	2	
∑ CouplingRatio	0.4444	
CRA-Index	1.5556	

Courtesy of Fleck et al. [TTC 2016]

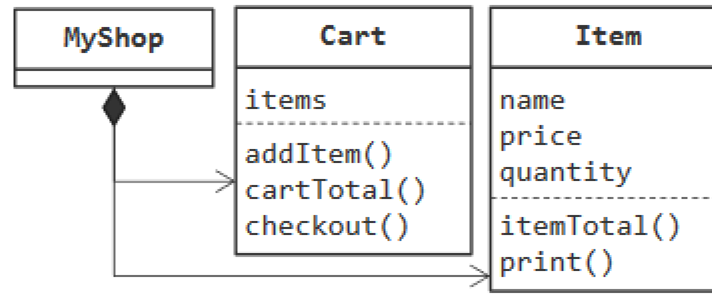
Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes



Example solution



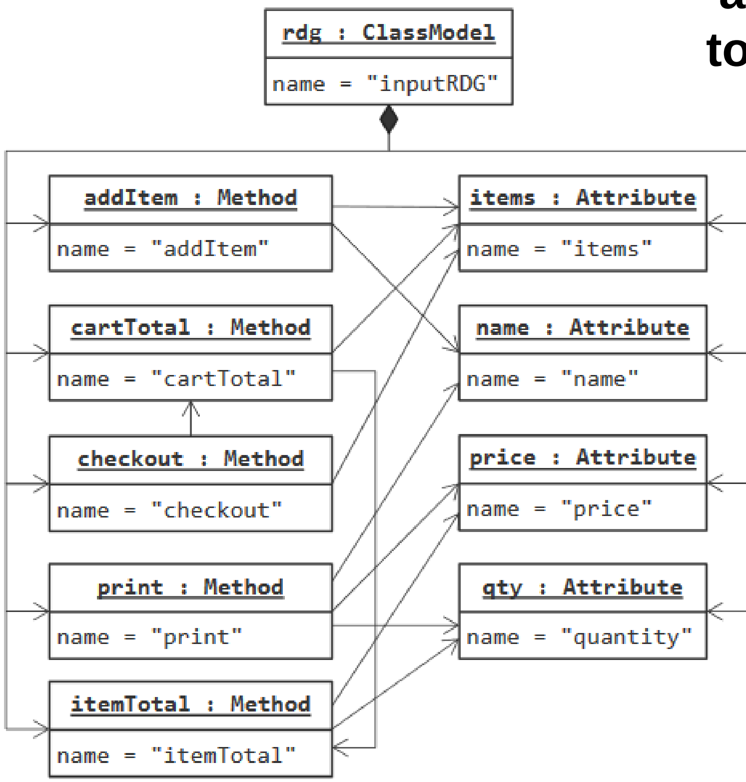
	Cart	Item
$MAI(c_i, c_i)$	3	5
$MMI(c_i, c_i)$	1	0
CohesionRatio	1.1667	0.8333
$MAI(c_i, c_j)$	1	0
$MMI(c_i, c_j)$	1	0
CouplingRatio	0.4444	0
$\sum$ CohesionRatio	2	
$\sum$ CouplingRatio	0.4444	
CRA-Index	1.5556	

Courtesy of Fleck et al. [TTC 2016]

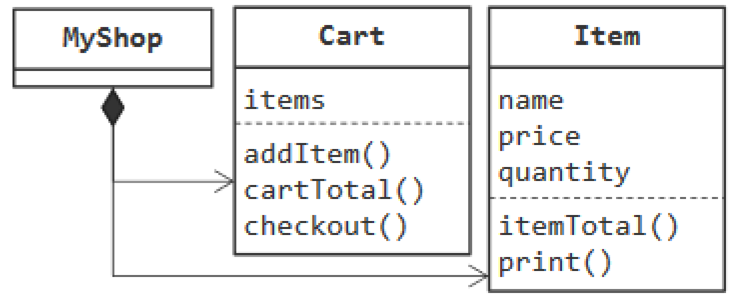
Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes



Example solution



Quality

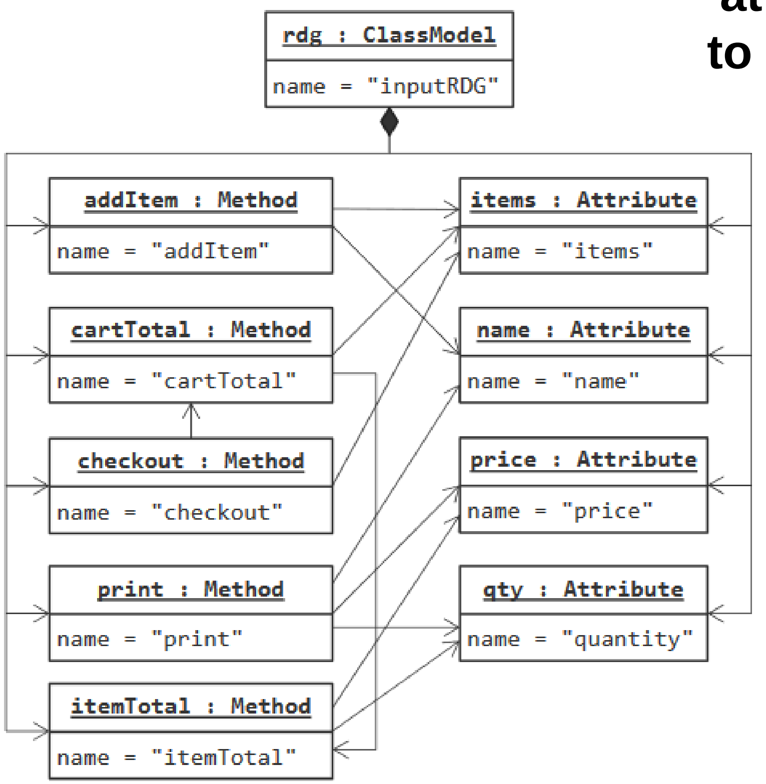
	Cart	Item
MAI(c_i, c_i)	3	5
MMI(c_i, c_i)	1	0
CohesionRatio	1.1667	0.8333
MAI(c_i, c_j)	1	0
MMI(c_i, c_j)	1	0
CouplingRatio	0.4444	0
$\sum$ CohesionRatio	2	
$\sum$ CouplingRatio	0.4444	
CRA-Index	1.5556	

Courtesy of Fleck et al. [TTC 2016]

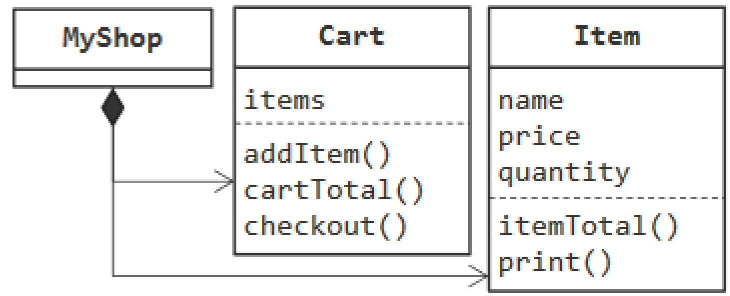
Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes



Example solution



Quality

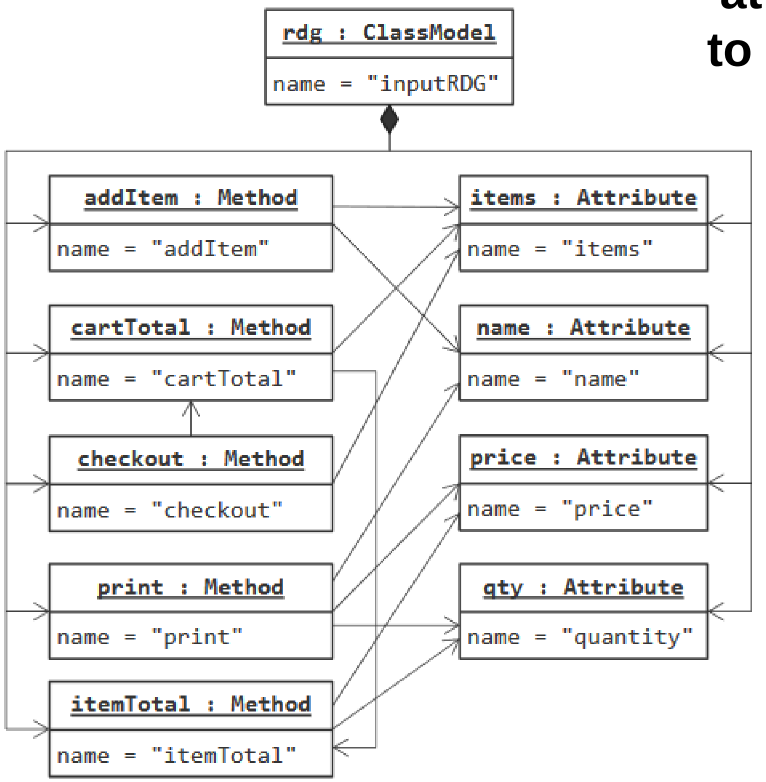
	Cart	Item
MAI(c_i, c_i)	3	5
MMI(c_i, c_i)	1	0
CohesionRatio	1.1667	0.8333
MAI(c_i, c_j)	1	0
MMI(c_i, c_j)	1	0
CouplingRatio	0.4444	0
$\sum$ CohesionRatio	2	
$\sum$ CouplingRatio	0.4444	
CRA-Index	1.5556	

Courtesy of Fleck et al. [TTC 2016]

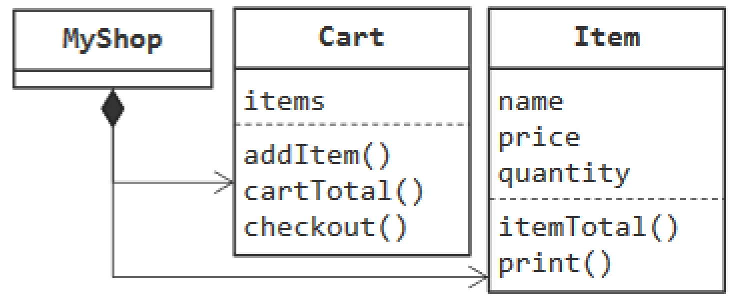
Example: Class Responsibility Assignment (CRA)

Input model

Task: Assign methods+ attributes to classes



Example solution



Quality

	Cart	Item
$MAI(c_i, c_i)$	3	5
$MMI(c_i, c_i)$	1	0
CohesionRatio	1.1667	0.8333
$MAI(c_i, c_j)$	1	0
$MMI(c_i, c_j)$	1	0
CouplingRatio	0.4444	0
$\sum$ CohesionRatio	2	
$\sum$ CouplingRatio	0.4444	
CRA-Index	1.5556	

Courtesy of Fleck et al. [TTC 2016]

Objective combining cohesion + coupling

Example: Class Responsibility Assignment (CRA)

Fitness function

$$CRA\text{-}Index = CohesionRatio - CouplingRatio$$

$$CohesionRatio = \sum_{c_i \in Classes} \frac{MAI(c_i, c_i)}{|M(c_i)| \times |A(c_i)|} + \frac{MMI(c_i, c_i)}{|M(c_i)| \times |M(c_i) - 1|}$$

$$CouplingRatio = \sum_{\substack{c_i, c_j \in Classes \\ c_i \neq c_j}} \frac{MAI(c_i, c_j)}{|M(c_i)| \times |A(c_j)|} + \frac{MMI(c_i, c_j)}{|M(c_i)| \times |M(c_j) - 1|}$$

$$MMI(c_i, c_j) = \sum_{\substack{m_i \in M(c_i) \\ m_j \in M(c_j)}} DMM(m_i, m_j)$$

$$MAI(c_i, c_j) = \sum_{\substack{m_i \in M(c_i) \\ a_j \in A(c_j)}} DMA(m_i, a_j)$$

$$DMA(m_i, a_j) = \begin{cases} 1 & \text{if there is a dependency between method } m_i \text{ and attribute } a_j \\ 0 & \text{otherwise} \end{cases}$$

$$DMM(m_i, m_j) = \begin{cases} 1 & \text{if there is a dependency between method } m_i \text{ and } m_j \\ 0 & \text{otherwise} \end{cases}$$

Courtesy of Fleck et al. [TTC 2016]

Hamid Masoud and Saeed Jalili. A clustering-based model for class responsibility assignment problem in object-oriented analysis. Journal of Systems and Software, 93(0):110 – 131, 2014.

Example: Class Responsibility Assignment (CRA)

Fitness function

$$CRA\text{-}Index = CohesionRatio - CouplingRatio$$

$$CohesionRatio = \sum_{c_i \in Classes} \frac{MAI(c_i, c_i)}{|M(c_i)| \times |A(c_i)|} + \frac{MMI(c_i, c_i)}{|M(c_i)| \times |M(c_i) - 1|}$$

$$CouplingRatio = \sum_{\substack{c_i, c_j \in Classes \\ c_i \neq c_j}} \frac{MAI(c_i, c_j)}{|M(c_i)| \times |A(c_j)|} + \frac{MMI(c_i, c_j)}{|M(c_i)| \times |M(c_j) - 1|}$$

$$MMI(c_i, c_j) = \sum_{\substack{m_i \in M(c_i) \\ m_j \in M(c_j)}} DMM(m_i, m_j)$$

$$MAI(c_i, c_j) = \sum_{\substack{m_i \in M(c_i) \\ a_j \in A(c_j)}} DMA(m_i, a_j)$$

$$DMA(m_i, a_j) = \begin{cases} 1 & \text{if there is a dependency between method } m_i \text{ and attribute } a_j \\ 0 & \text{otherwise} \end{cases}$$

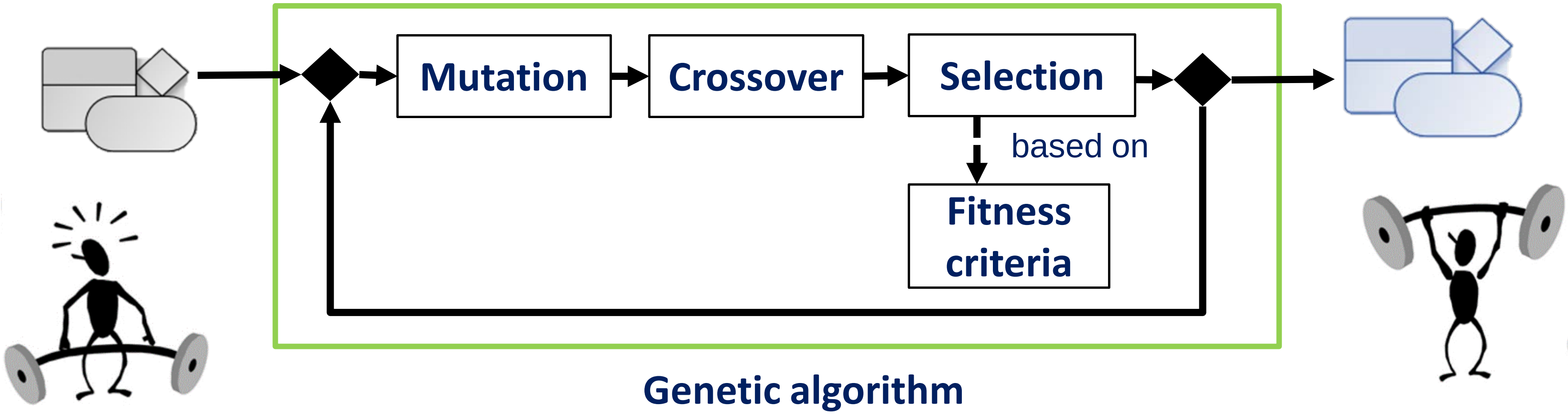
$$DMM(m_i, m_j) = \begin{cases} 1 & \text{if there is a dependency between method } m_i \text{ and } m_j \\ 0 & \text{otherwise} \end{cases}$$

Courtesy of Fleck et al. [TTC 2016]

Hamid Masoud and Saeed Jalili. A clustering-based model for class responsibility assignment problem in object-oriented analysis. Journal of Systems and Software, 93(0):110 – 131, 2014.

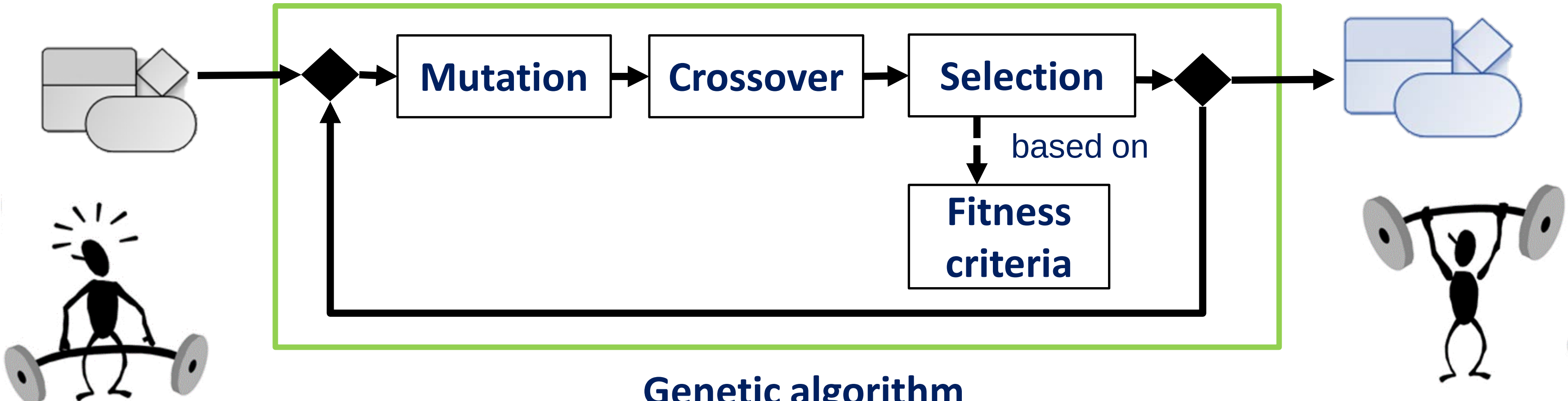
Search-based software engineering

- **Problem**: Search space usually too large to enumerate all solutions
- **Solution**: *Guided search* can explore space more efficiently than humans



Search-based software engineering

- **Problem**: Search space usually too large to enumerate all solutions
- **Solution**: *Guided search* can explore space more efficiently than humans

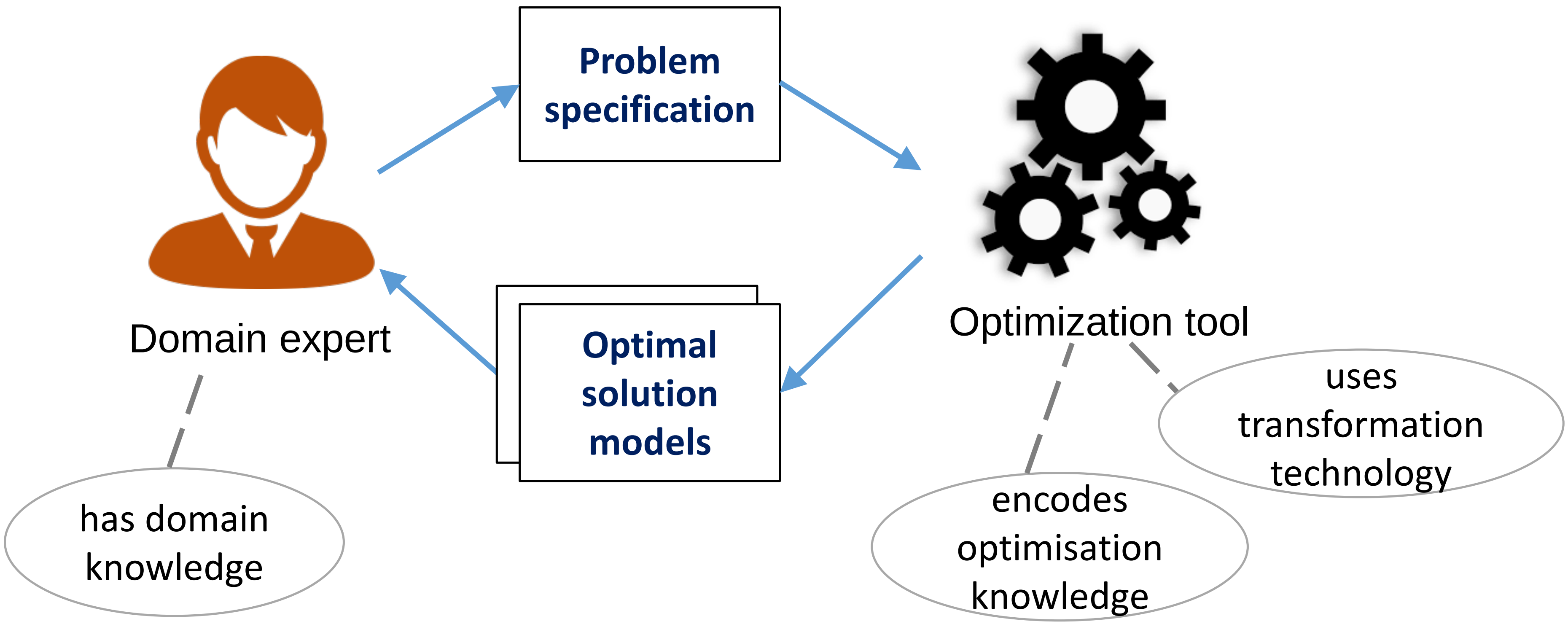


Genetic algorithm

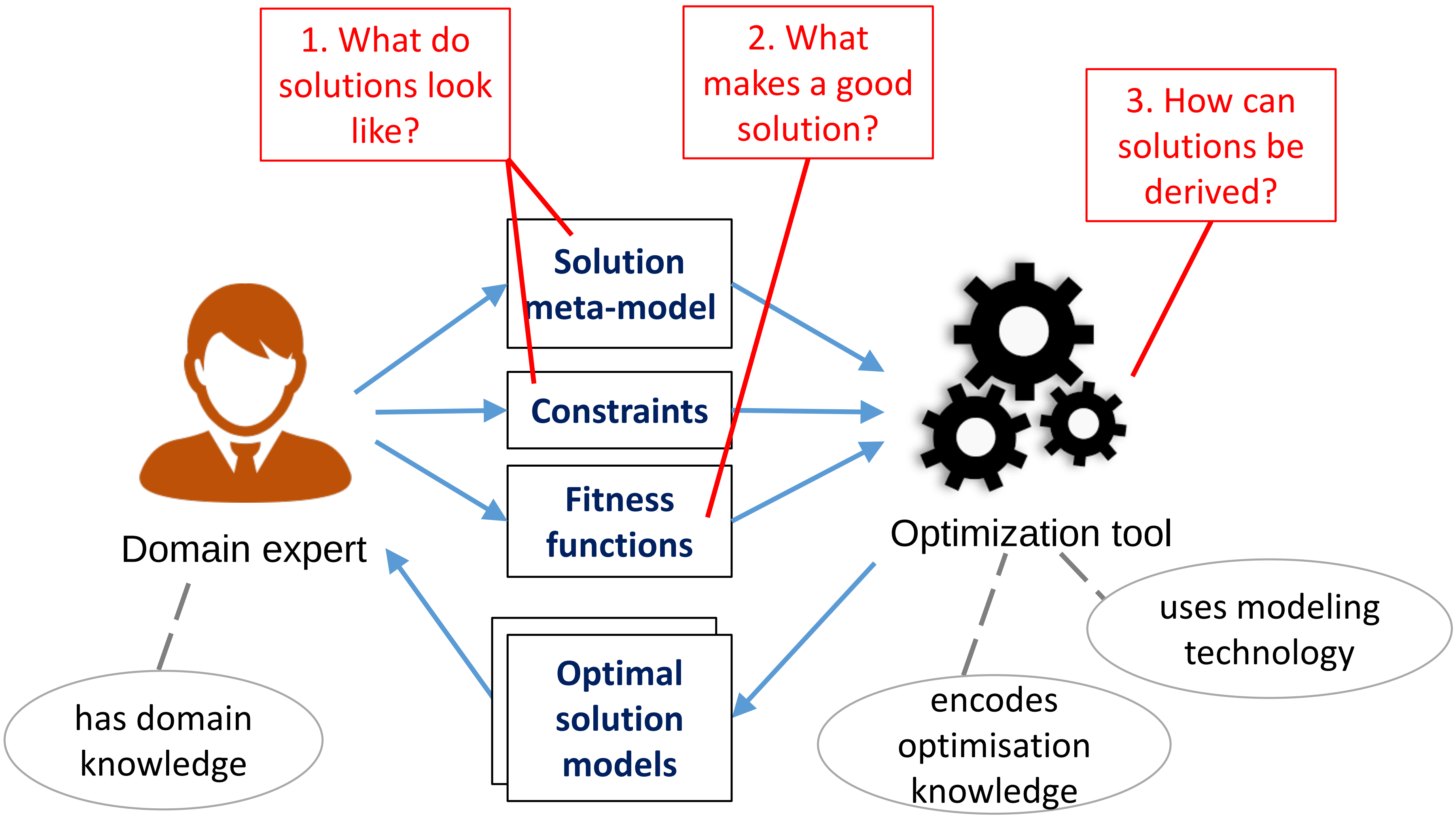
- **Cost**: Search algorithms need to be customized to problem at hand; substantial expertise required

Solution: Search-based model optimisation

- Use models to describe solutions
- Standard manipulations available (model transformations!)
- **Move optimisation knowledge from humans to tools**



Search-based model optimisation: what's needed?

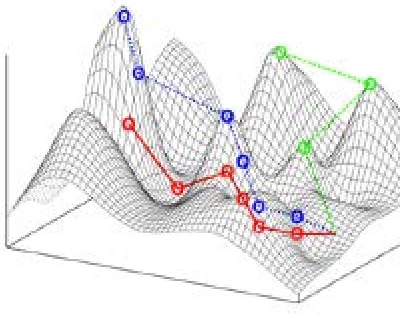


Overview

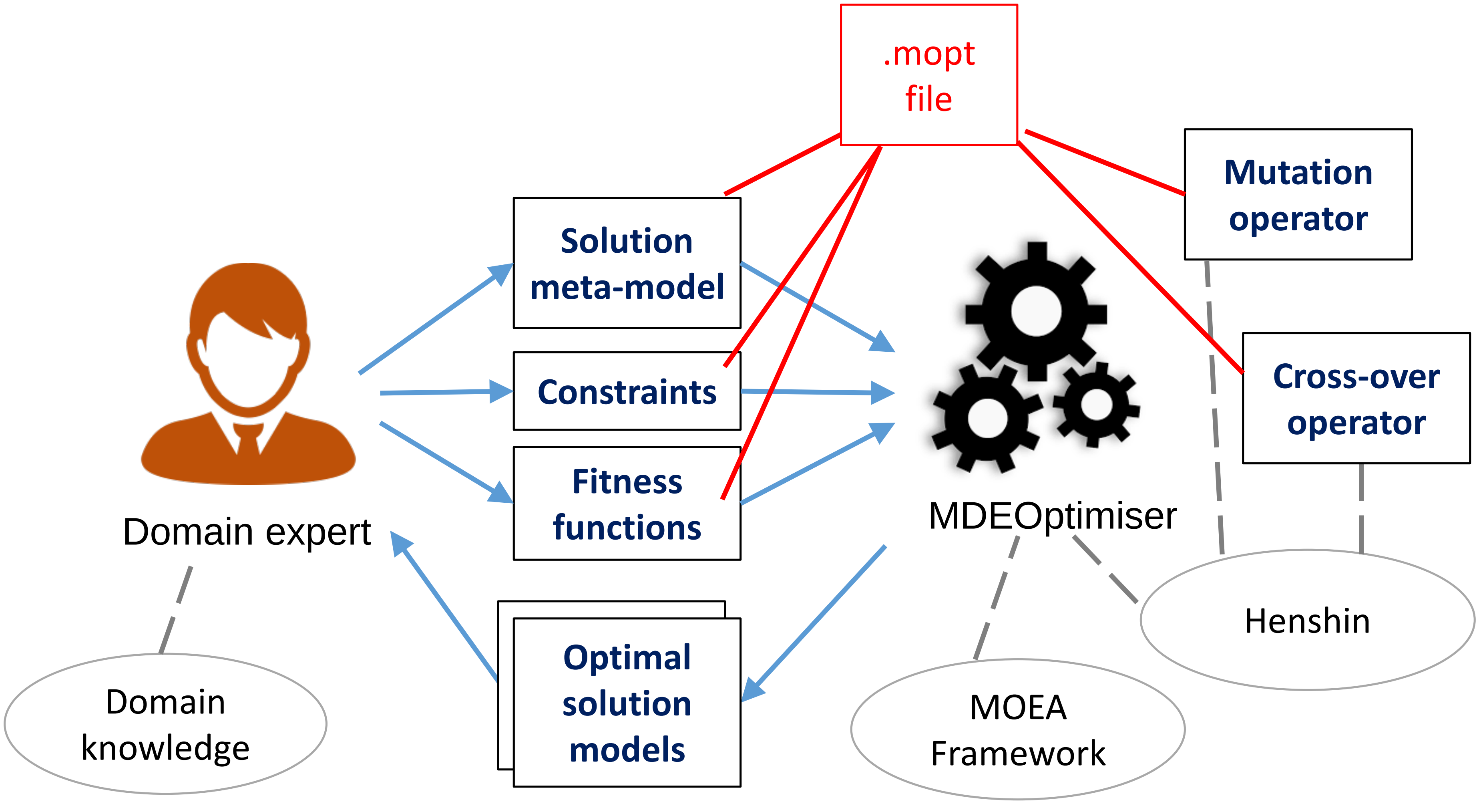
- Part 1: Henshin: A Guided Tour
  - Language
  - In Action (interactive)
  - Features
  - Applications
- Part 2: Henshin in Search-Based Model Optimization
  - Background
  - MDEOptimiser
  - In Action
    - Case 1: Class Responsibility Assignment (interactive)
    - Case 2: SCRUM Planning (interactive)

MDE Optimiser

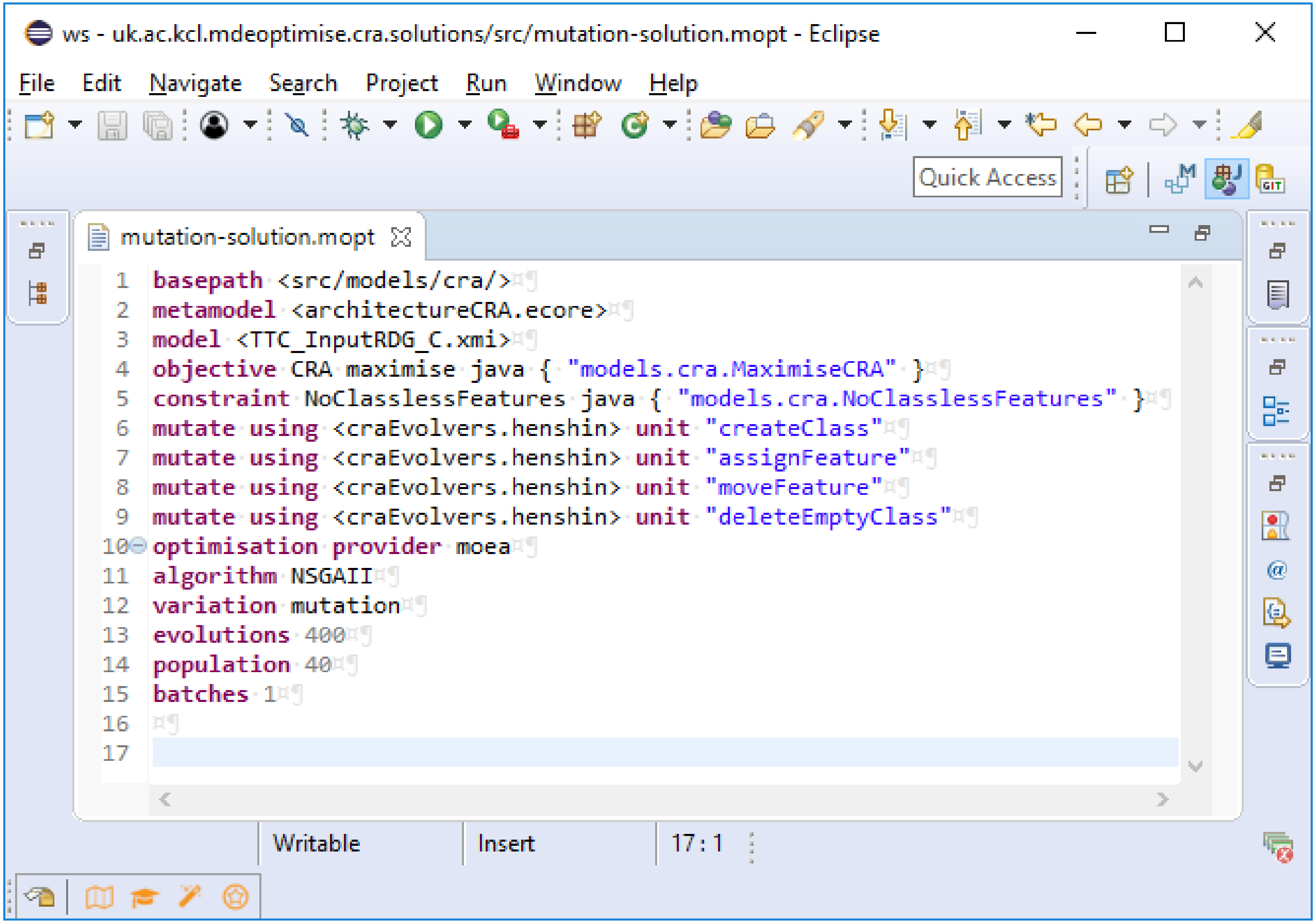
- Multi-objective optimization
- Directly over models (no separate solution encoding)
- Specification language + kernel
- Uses Henshin to specify evolutionary operators



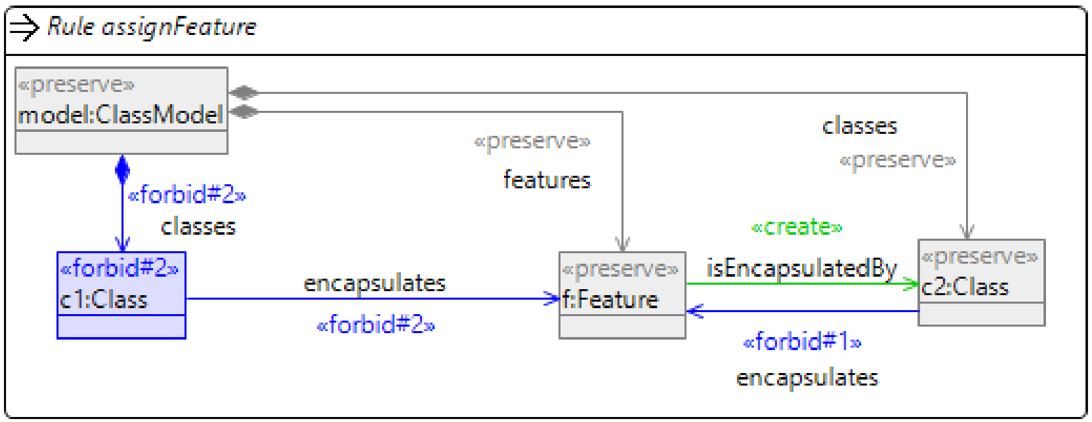
MDE Optimiser

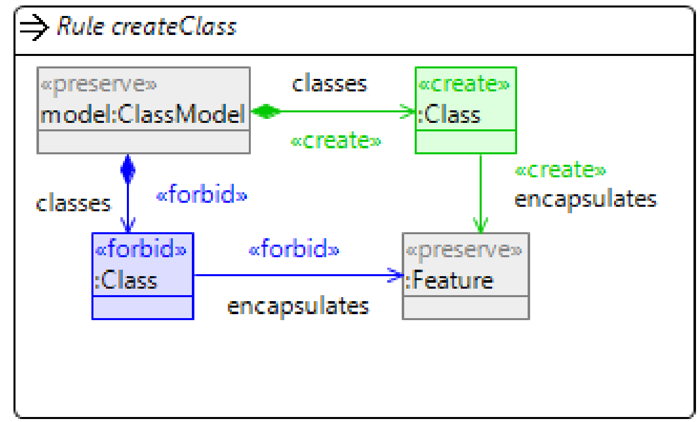


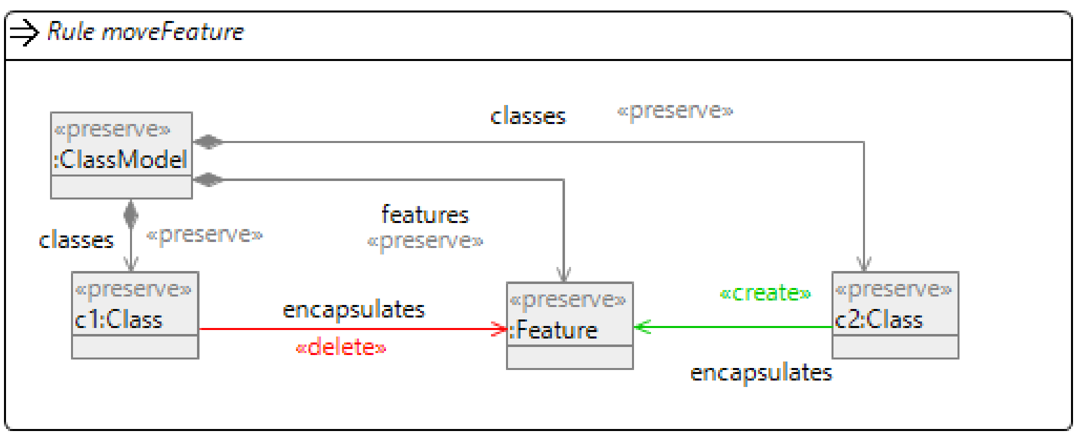
Problem specification in MDEOptimiser: CRA

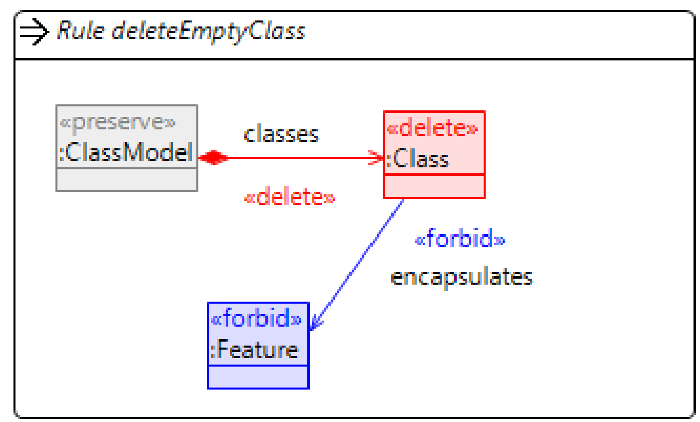


Pre-defined mutation rules for CRA









Pre-defined objective function for CRA

MaximiseCRA.xtend

```
package models.cra

import org.eclipse.emf.ecore.EObject

class MaximiseCRA extends AbstractModelQueryFitnessFunction {

	override double computeFitness(EObject model) {
		val cohesion = calculateCohesionRatio(model);
		val coupling = calculateCouplingRatio(model);

		return (cohesion - coupling) * -1
	}

	def double calculateCohesionRatio(EObject classModel) {
	def double calculateCouplingRatio(EObject classModel) {
	def double calculateCouplingRatio(EObject classSource, EObject classModel) {
	def mai(EObject classSource, EObject classTarget) {
	def mmi(EObject classSource, EObject classTarget) {
		override getName() {
		return "Maximise CRA"
	}

}
```

Pre-defined constraint for CRA

NoClasslessFeatures.xtend

```
package models.cra

import org.eclipse.emf.common.util.EList...

class NoClasslessFeatures extends AbstractModelQueryFitnessFunction {

	override computeFitness(EObject model) {
		var fitness = (model.getFeature("features") as EList<EObject>).
		filter[feature | feature.getFeature("isEncapsulatedBy") === null].size;
		println("Classless features: " + fitness)
		return fitness;
	}

	override getName() {
		return "No classless features"
	}

}

```

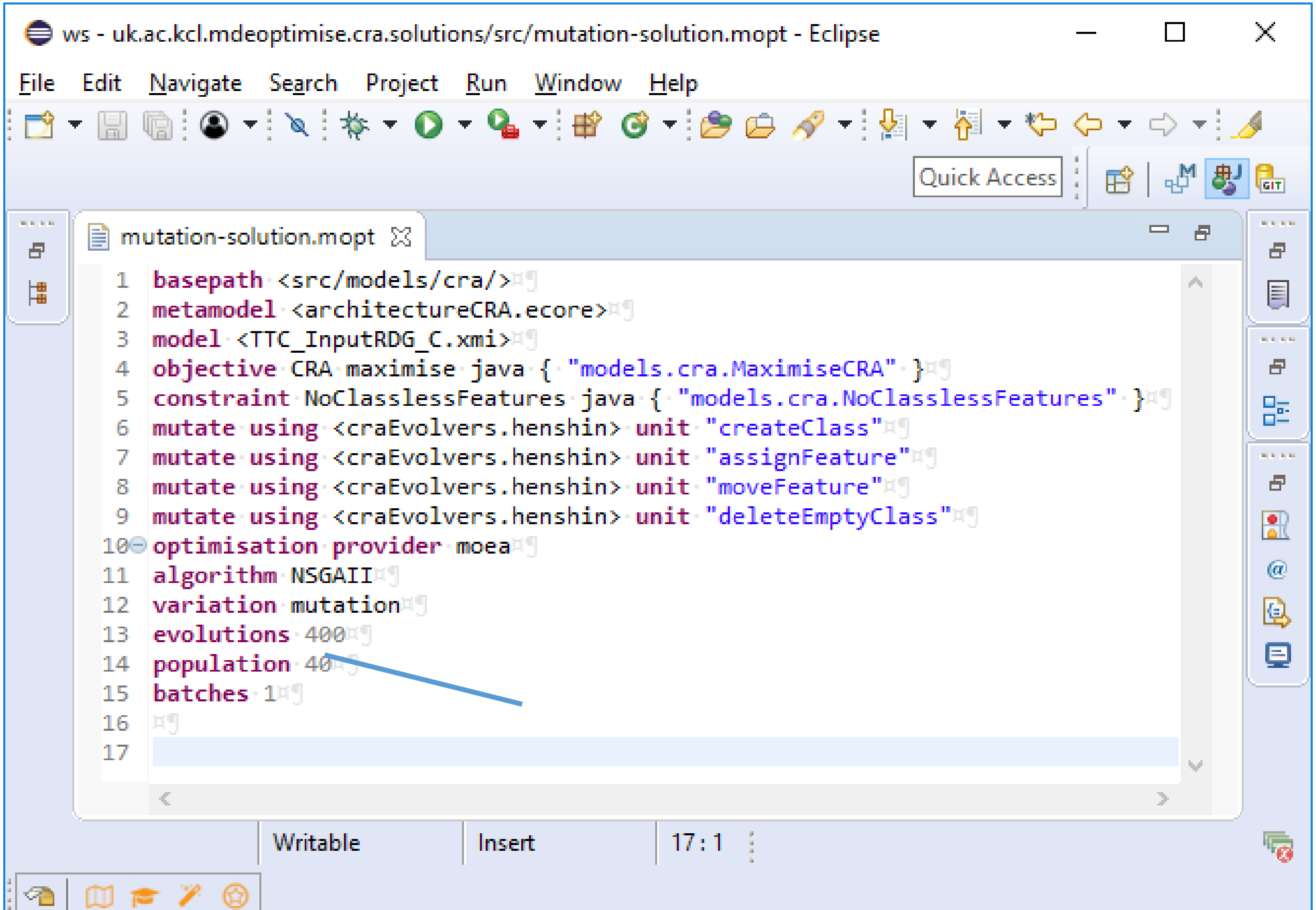
Pre-defined constraint for CRA

NoClasslessFeatures.xtend

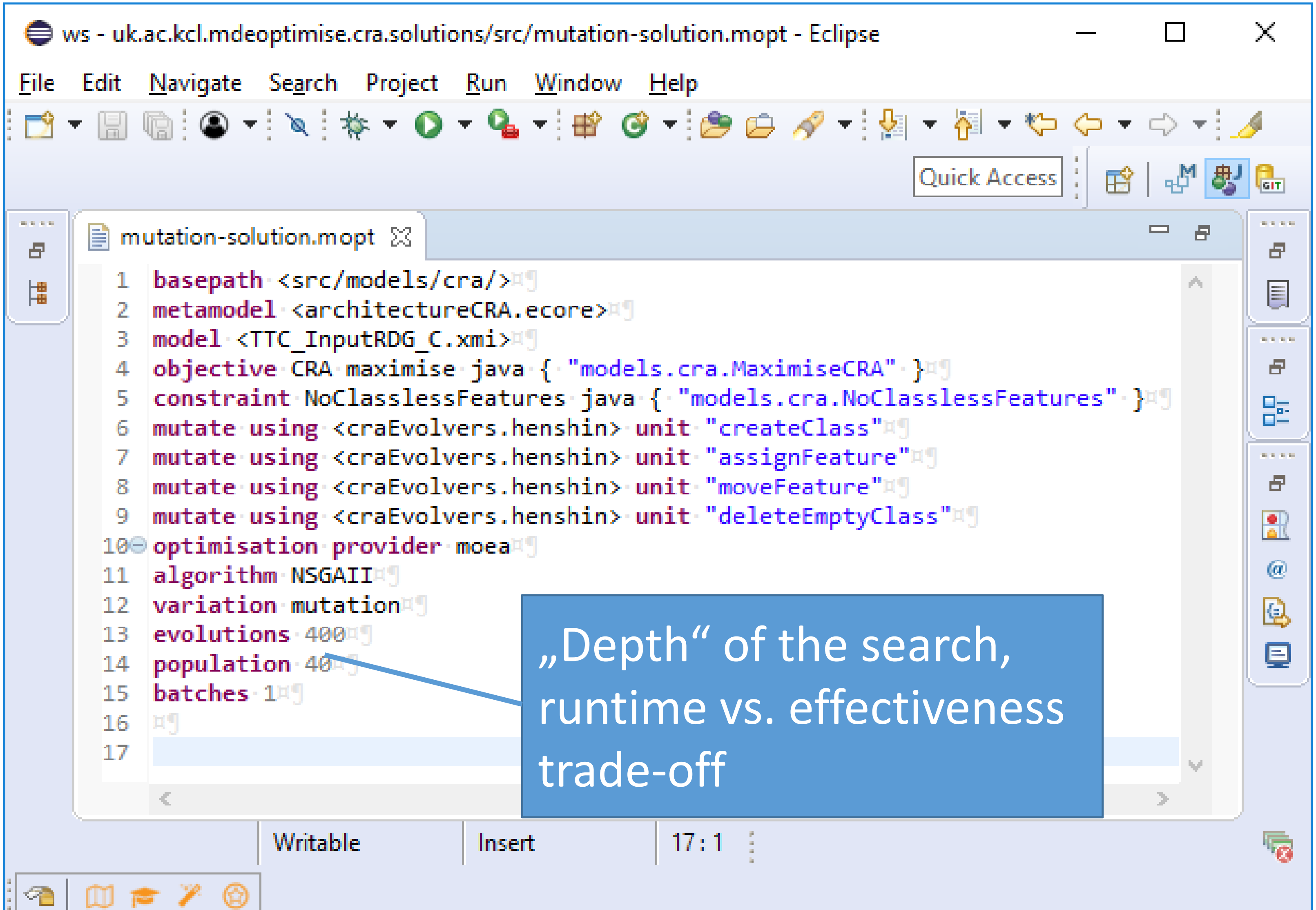
```
package models.cra

import org.eclipse.emf.common.util.EList

class NoClasslessFeatures extends AbstractModelQueryFitnessFunction {

    override computeFitness(EObject model) {
        var fitness = (model.getFeature("features") as EList<EObject>).
        filter[feature | feature.getFeature("isEncapsulatedBy") === null].size;
        println("Classless features: " + fitness)
        return fitness;
    }

    override getName() {
        return "No classless features"
    }

}
```

Solution‘s „fitness“ w.r.t. a constraint: How far is the solution away from fulfilling the constraint?

Problem specification in MDEOptimiser: CRA



Problem specification in MDEOptimiser: CRA



Overview

- Part 1: Henshin: A Guided Tour
  - Language
  - In Action (interactive)
  - Features
  - Applications
- Part 2: Henshin in Search-Based Model Optimization
  - Background
  - MDEOptimiser
  - In Action
    - Case 1: Class Responsibility Assignment (interactive)
    - Case 2: SCRUM Planning (interactive)

MDEOptimiser in action: CRA case

1. Import projects

2. View the specification of CRA case

3. Set up and apply run configuration

4. View created results

5. Design a good mutation operator

MDEOptimiser in action: CRA case

1. Import projects

2. View the specification of CRA case

3. Set up and apply run configuration

4. View created results

5. Design a good mutation operator

Import projects

- In Eclipse, do *File* → *Import…* → *General* → *Existing Projects Into Workspace* → *Next*
- Do *Select Archive File* → Choose **optimization-cases.zip**
- By now, you should be an expert on importing projects. :-)

MDEOptimiser in action: CRA case

1. Import projects

2. View the specification of CRA case

3. Set up and apply run configuration

4. View created results

5. Design a good mutation operator

View the specification of CRA case

- In the Package Explorer, navigate to project **uk.ac.kcl.mdeoptimise.cra.solutions**, folder **src/models.cra**

- Have a look at the files:

MDEOptimiser spec: **cra-solution.mopt**
meta-model: **architectureCRA.ecore**
five input models: **TTC_InputRDG_<A-E>.xmi**
objective function**: MaximiseCRA.xtend**
constraint**: NoClasslessFeatures.xtend**
mutation operators: **craEvolvers.henshin_diagram**

MDEOptimiser in action: CRA case

1. Import projects

2. View the specification of CRA case

3. **Set up and apply run configuration**

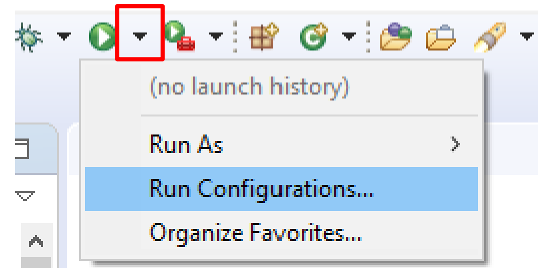
4. View created results

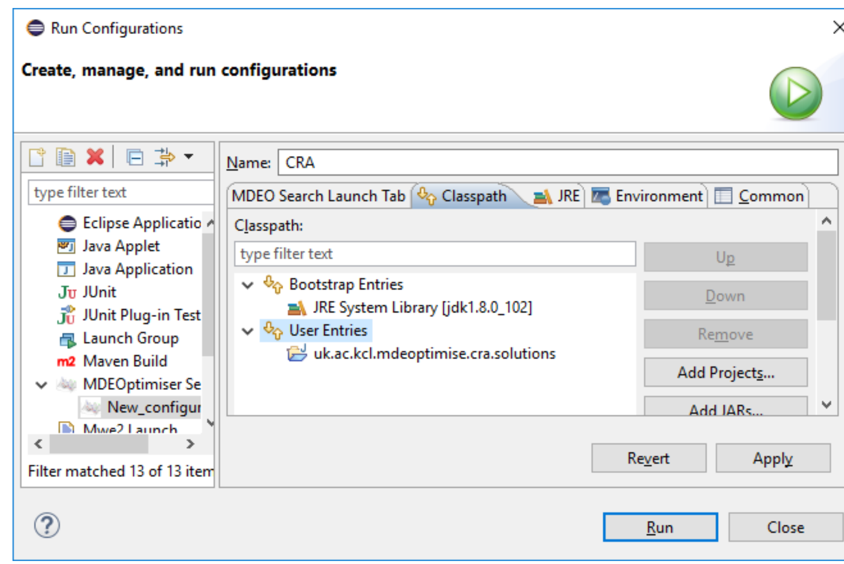
5. Design a good mutation operator

Set up and apply run configuration

To execute **cra-solution.mopt**, create a new run configuration:

- Click on the triangle next to the *Run* Icon, select *Run Configurations…*
- Right click on *MDEOptimiser Search*, select *New.* As *Name* for the configuration, enter: *CRA*
- Do *Browse Workspace ->* Select Source *cra-solution.mopt*
- In the *Classpath* tab, add the current project as a User Entry, using *Add Projects…*
- Hit *Apply and Run.* If everything works correctly, you will see console output like on the right.






```
Calculated CRA : 1.3707386363636362
Classless features: 0
Calculated CRA : 0.5790719696969693
Classless features: 0
```

MDEOptimiser in action: CRA case

1. Import projects

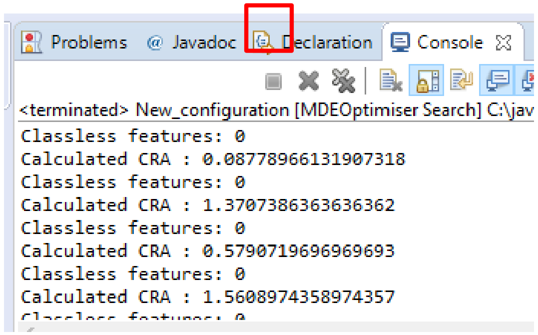
2. View the specification of CRA case

3. Set up and apply run configuration

4. View created results

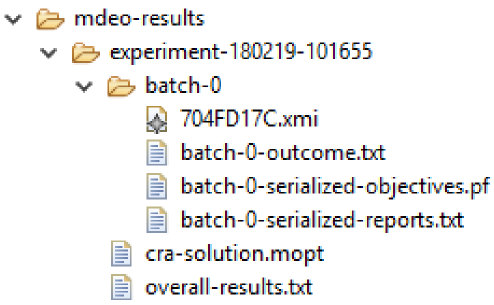
5. Design a good mutation operator

View created results



After approximately 15 seconds, the run is finished: No new console outputs, „Terminate“ switch has turned gray.

Results are in a newly created folder:
In the Package Explorer, click on *mde-results*, hit F5 (refresh) and find the folder.



Open **overall-results.txt**. This shows the execution time and an overview of the best solution with its CRA index.

MDEOptimiser in action: CRA case

1. Import projects
2. View the specification of CRA case
3. Set up and apply run configuration
4. View created results
5. **Design a good mutation operator**

Design a good mutation operator

Problem: Search is time-consuming.

Solution: Improve the search by designing good evolutionary operators.

> A mutation operator m1 is *better* than a mutation operator m2, if the solutions found using m1 are better than those found using m2 (assuming an otherwise equal configuration).

Task: Design a better mutation operator for the CRA case than the given one.

Reference values:

Model	**A**	**B**	**C**	**D**	**E**
CRA	1.6	2.2	1.8	2.4	-11.6

Hint: What are desirable structures from the perspective of the objective function, and how to create them?

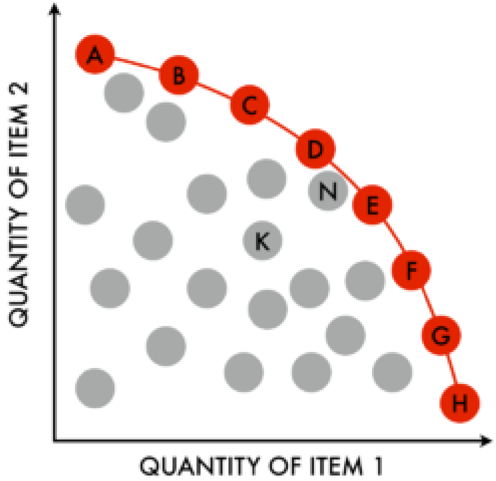
Overview

- Part 1: Henshin: A Guided Tour
  - Language
  - In Action (interactive)
  - Features
  - Applications
- Part 2: Henshin in Search-Based Model Optimization
  - Background
  - MDEOptimiser
  - In Action
    - Case 1: Class Responsibility Assignment (interactive)
    - Case 2: SCRUM Planning (interactive)

SCRUM Planning

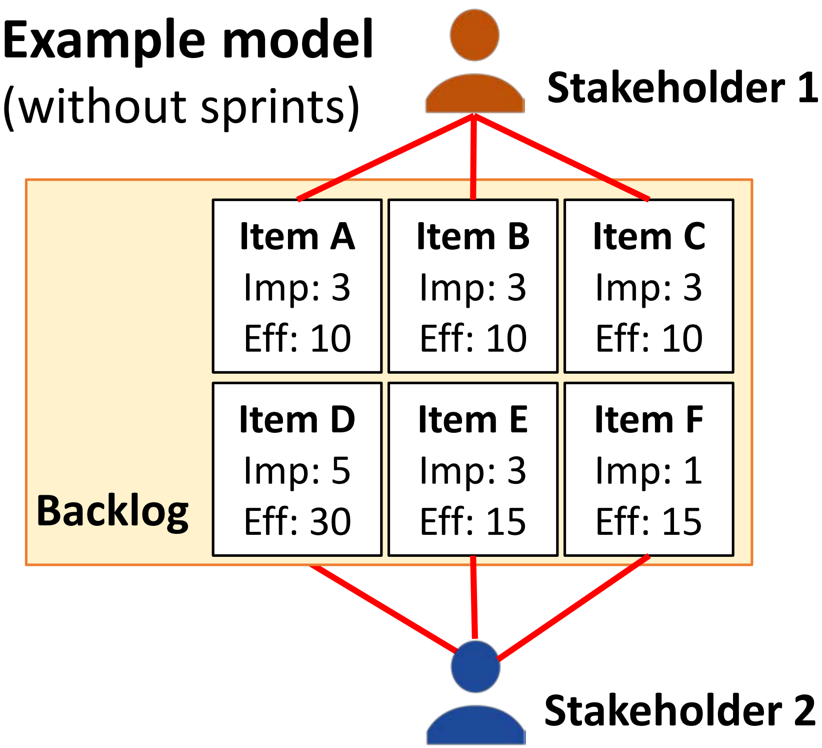
- Have a **backlog** of **work items** for different **customers.**
  Need to allocate work items to **sprints.**

- Goals: **Be fast** and **keep customers satisfied.**
- Multi-objective optimisation problem: pareto front instead of one best solution

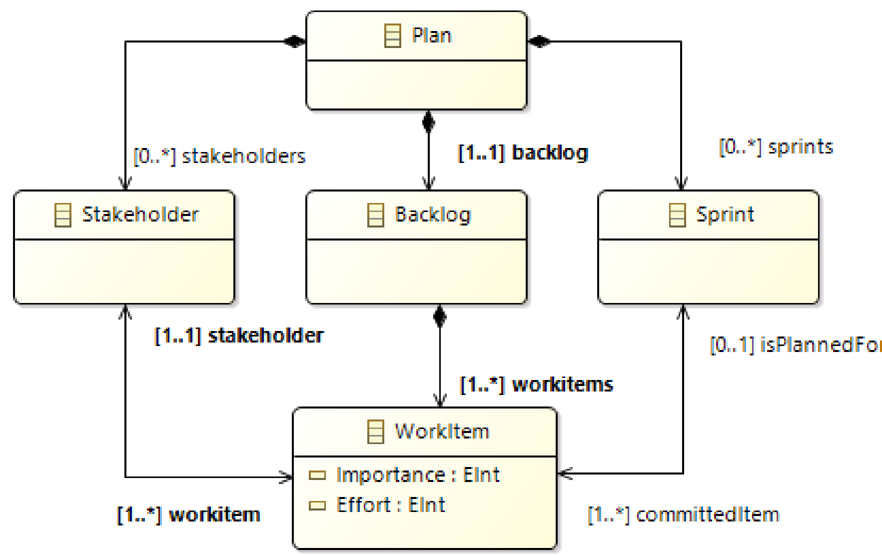


SCRUM Planning

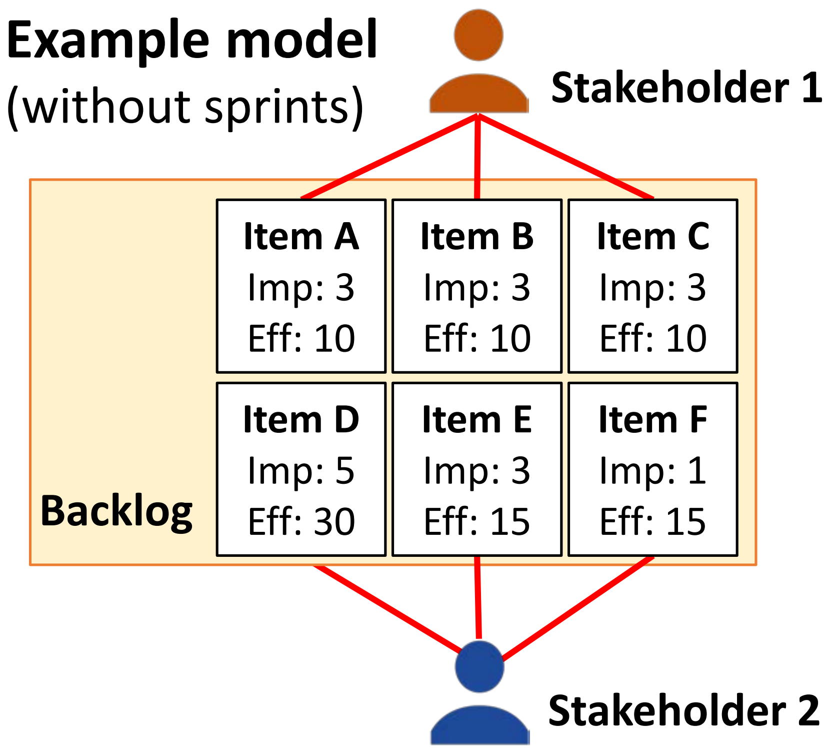
Example model
(without sprints)



Meta-model



SCRUM Planning: Objectives and constraints



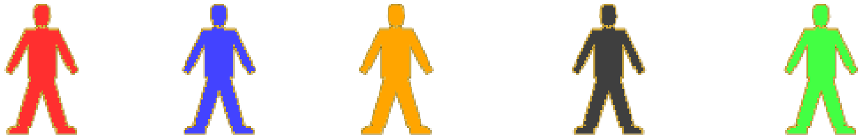
Objective 1: *Sprints are as balanced as possible.*

Objective 2: *Maximize customer satisfaction -> next slide*

Constraint 1: *Number of sprints is below a given maximum*

Constraint 2: *Every work item is assigned to a sprint.*

SCRUM Planning: Customer satisfaction index



Stakeholder 1 Stakeholder 2 Stakeholder 3 Stakeholder 4 Stakeholder 5

Sprint 1 Capacity: 28
- WorkItem 2
- WorkItem 4
- WorkItem 5
- WorkItem 7
- WorkItem 8
- WorkItem 10

Sprint 2 Capacity: 27
- WorkItem 1
- WorkItem 3
- WorkItem 6
- WorkItem 11
- WorkItem 9

Good solution

Customer Satisfaction Index

0.80

Sprint 1 Capacity: 28
- WorkItem 1
- WorkItem 2
- WorkItem 3
- WorkItem 4
- WorkItem 8
- WorkItem 10

Sprint 2 Capacity: 27
- WorkItem 7
- WorkItem 5
- WorkItem 6
- WorkItem 11
- WorkItem 9

Bad solution

Customer Satisfaction Index

5.39

Low means good:

Indicates low standard deviation

Backlog

WorkItem 1
Stakeholder: 1 Importance: 3 Effort: 8

WorkItem 2
Stakeholder: 1 Importance: 8 Effort: 5

WorkItem 3
Stakeholder: 1 Importance: 8 Effort: 8

WorkItem 4
Stakeholder: 2 Importance: 1 Effort: 1

WorkItem 5
Stakeholder: 2 Importance: 1 Effort: 8

WorkItem 6
Stakeholder: 3 Importance: 5 Effort: 5

WorkItem 7
Stakeholder: 3 Importance: 3 Effort: 8

WorkItem 8
Stakeholder: 3 Importance: 3 Effort: 1

WorkItem 9
Stakeholder: 4 Importance: 1 Effort: 3

WorkItem 10
Stakeholder: 5 Importance: 5 Effort: 5

WorkItem 11
Stakeholder: 5 Importance: 5 Effort: 3

SCRUM Planning: Customer satisfaction index

Sprint 1 Capacity: 28
- WorkItem 2
- WorkItem 4
- WorkItem 5
- WorkItem 7
- WorkItem 8
- WorkItem 10

Sprint 2 Capacity: 27
- WorkItem 1
- WorkItem 3
- WorkItem 6
- WorkItem 11
- WorkItem 9

Good solution

Customer Satisfaction Index

0.80

Backlog

WorkItem 1
Stakeholder: 1 Importance: 3 Effort: 8

WorkItem 2
Stakeholder: 1 Importance: 8 Effort: 5

WorkItem 3
Stakeholder: 1 Importance: 8 Effort: 8

WorkItem 4
Stakeholder: 2 Importance: 1 Effort: 1

WorkItem 5
Stakeholder: 2 Importance: 1 Effort: 8

WorkItem 6
Stakeholder: 3 Importance: 5 Effort: 5

WorkItem 7
Stakeholder: 3 Importance: 3 Effort: 8

WorkItem 8
Stakeholder: 3 Importance: 3 Effort: 1

WorkItem 9
Stakeholder: 4 Importance: 1 Effort: 3

WorkItem 10
Stakeholder: 5 Importance: 5 Effort: 5

WorkItem 11
Stakeholder: 5 Importance: 5 Effort: 3

Satisfaction of customer c in sprint s

$$sat(c,s) = \sum_{i \in s.items, i.cust=c} i.importance$$

Customer satisfaction index of plan p

$$csi(p) = std.dev_{c \in p.customers}(std.dev_{s \in p.sprints}\,(sat(c,s)))$$

Problem specification in MDEOptimiser: SCRUM

```
basepath <src/models/scrum/>
metamodel <planning.ecore>
model <sprint-planning-model-5-stakeholders-119-items.xmi>
objective MinimiseCustomerSatisfactionIndex minimise java
 { "models.scrum.MinimiseCustomerSatisfactionIndex" }
objective MinimiseSprintEffortDeviation minimise java
 { "models.scrum.MinimiseSprintEffortDeviation" }
constraint HasNoUnassignedWorkItems java
 { "models.scrum.HasNoUnassignedWorkItems" }
constraint HasTheAllowedMaximalNumberOfSprints java
 { "models.scrum.HasTheAllowedMaximalNumberOfSprints" }
mutate using <sprint-repair.henshin> unit "createSprint"
mutate using <sprint-repair.henshin> unit "addItemToSprint"
mutate using <sprint-repair.henshin> unit "createSprint_lb_repair"
mutate using <sprint-repair.henshin> unit "move_item_between_sprints"
mutate using <sprint-repair.henshin> unit "deleteSprint_lb_repair"
optimisation provider moea
algorithm NSGAII
variation mutation
evolutions 500
population 30
batches 1
```

MDEOptimiser in action: SCRUM case

1. View the specification of SCRUM case
2. Set up and apply run configuration
3. View created results

View the specification of the SCRUM case

- In the Package Explorer, navigate to project **uk.ac.kcl.mdeoptimise.scrum.planning.solutions**, folder **src/models.scrum**

- Have a look at the files:

  MDEOptimiser spec: **scrum-planning.mopt**
  meta-model: **planning.ecore**
  input models: **input** directory
  objective functions**: MinimiseSprintEffortDeviation.xtend**
  and **MinimiseCustomerSatisfactionIndex.xtend**
  constraints**: HasTheAllowedMaximalNumberOfSprints.xtend**
  and **HasNoUnassignedWorkItems.xtend**
  mutation operators: **sprint-repair.henshin_diagram**

MDEOptimiser in action: SCRUM case

1. View the specification of SCRUM case

2. Set up and apply run configuration

3. View created results

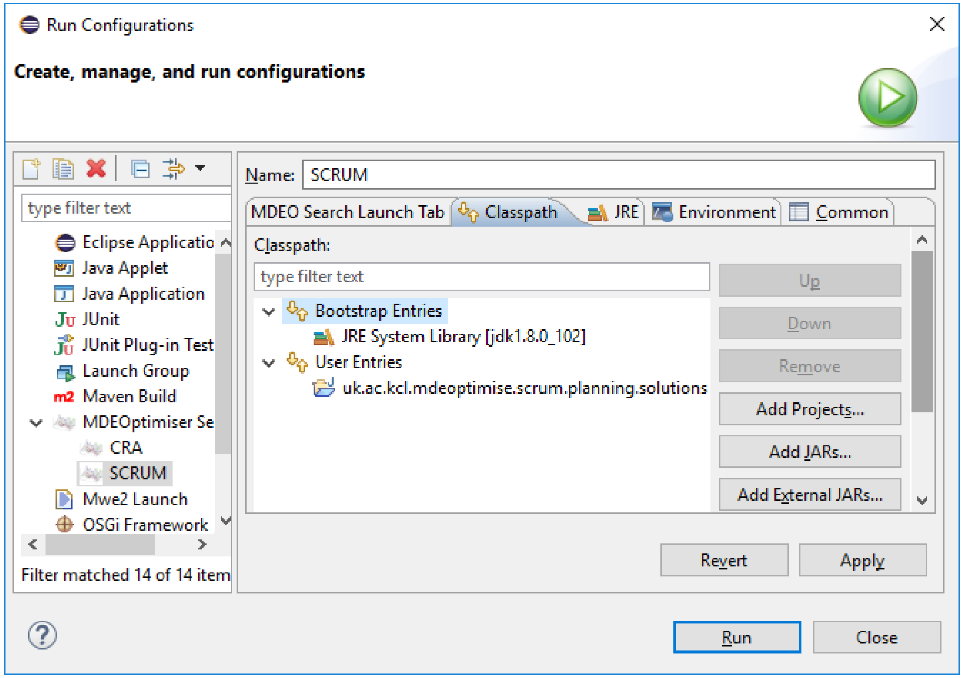
Set up and apply run configuration

Similar to the CRA case, create a new configuration

Name: *SCRUM*

Source: *scrum-planning.mopt*

Make sure to add the corresponding classpath entry:



MDEOptimiser in action: SCRUM case

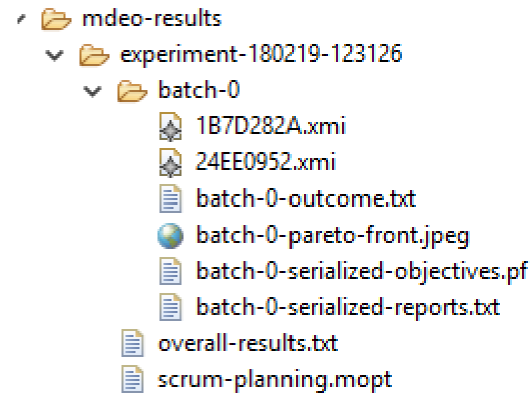
1. View the specification of SCRUM case

2. Set up and apply run configuration

3. View created results

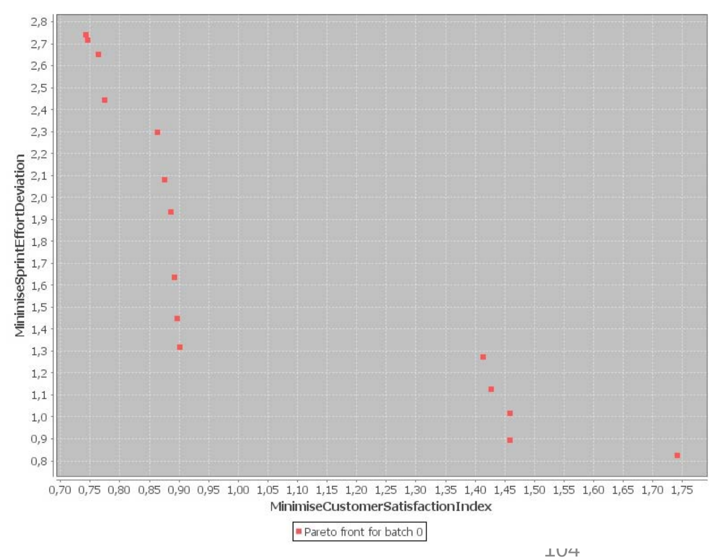
View created results

Results are, again, in a newly created folder:
In the Package Explorer, click on *mde-results*, hit F5 (refresh) and find the folder.

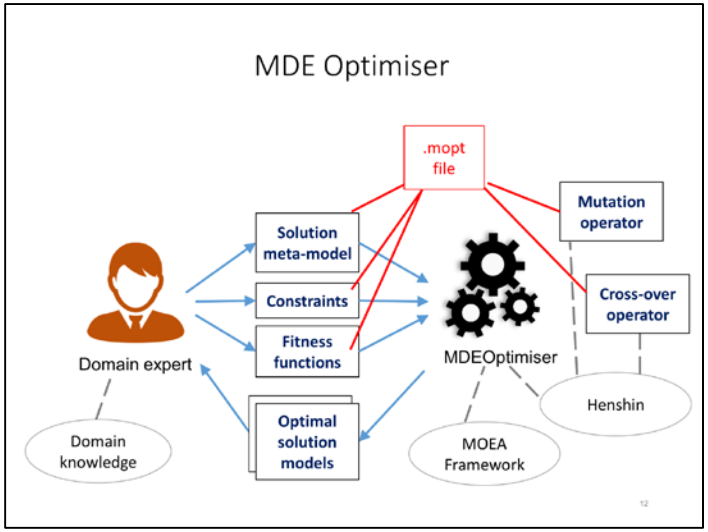


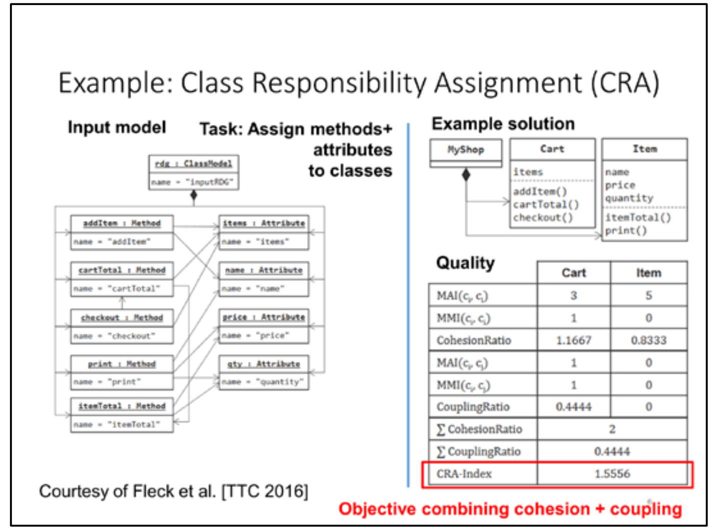
Open **overall-results.txt**. This shows the execution time and an overview of the best solutions with their objective values.

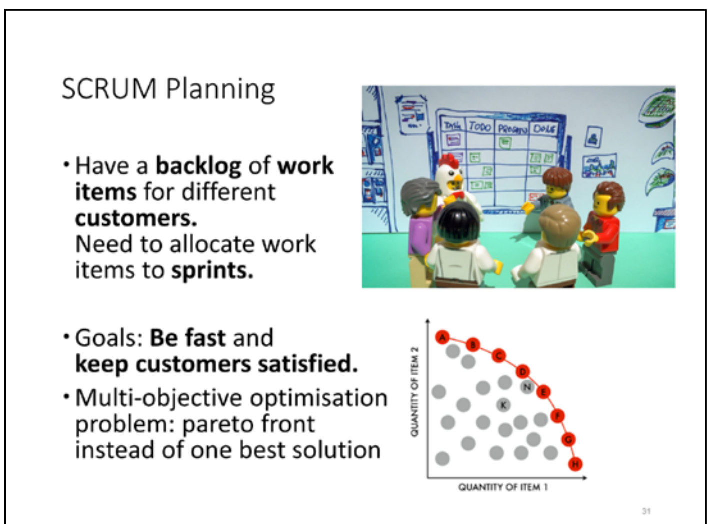
Since we have multiple solutions in general, an image file **batch-0-pareto-front.jpeg** is generated showing the pareto front.

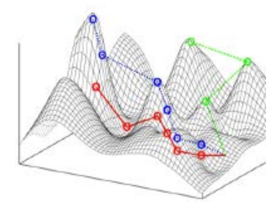


Summary of Part 2









Further information:
www.eclipse.org/henshin
mde-optimiser.github.io