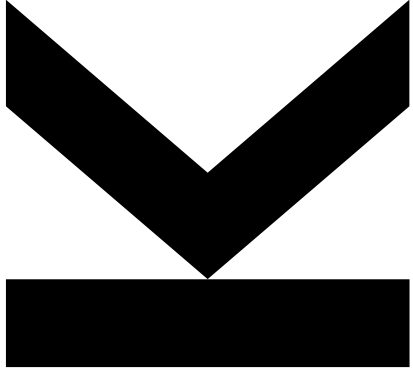


Model-Driven Optimization Hands-on with MOMoT



Stefan Klikovits, Institute of Business Informatics – Software Engineering, JKU Linz
<https://klikovits.net>

Get ready

Scan, clone, launch — before we start



Workspace repository

[github.com/hadiDHD/
fame-26-mdo-workspace](https://github.com/hadiDHD/fame-26-mdo-workspace)

- ☐ Eclipse at localhost:6080
- ☐ Docker running
- ☐ Repository cloned
- ☐ `./run.sh` or `run.ps1`



Problems? Raise your hand now — the first hands-on starts in ~20 minutes.



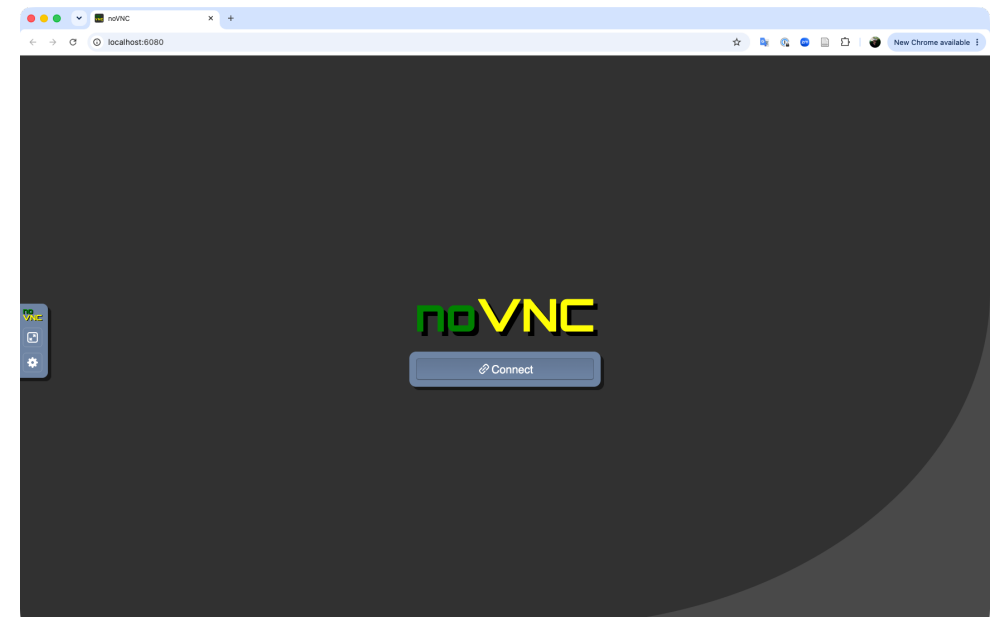
Instructions (PDF)

[klikovits.net/assets/pdf/2026-FAME/
2026-FAME-QuickStart.pdf](https://klikovits.net/assets/pdf/2026-FAME/2026-FAME-QuickStart.pdf)

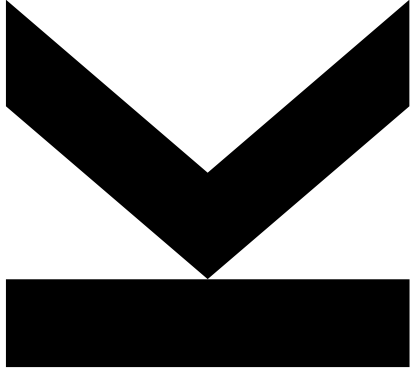


Slide deck

[klikovits.net/assets/pdf/
2026-FAME/2026-FAME-MDO.pdf](https://klikovits.net/assets/pdf/2026-FAME/2026-FAME-MDO.pdf)



Model-Driven Optimization Hands-on with MOMoT



Stefan Klikovits, Institute of Business Informatics – Software Engineering, JKU Linz
<https://klikovits.net>



**Stefan
KLIKOVITS**

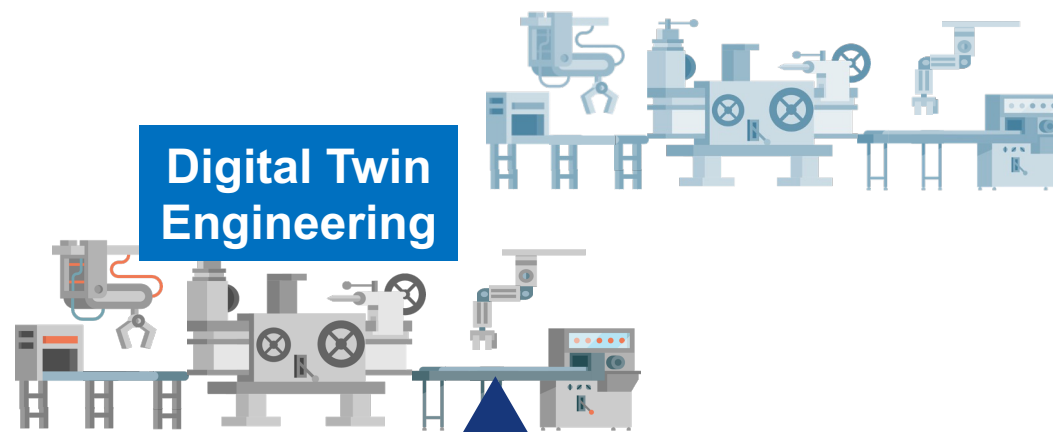
<https://klikovits.net>



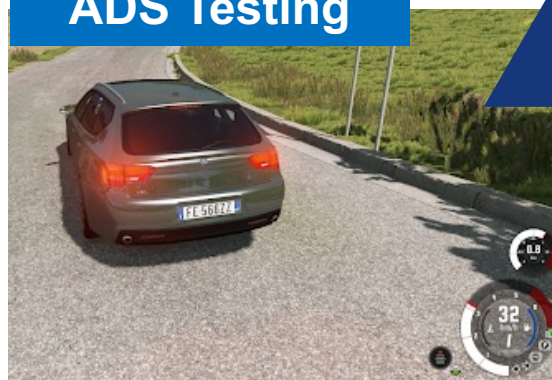
MANCHESTER
1824



**UNIVERSITÉ
DE GENÈVE**

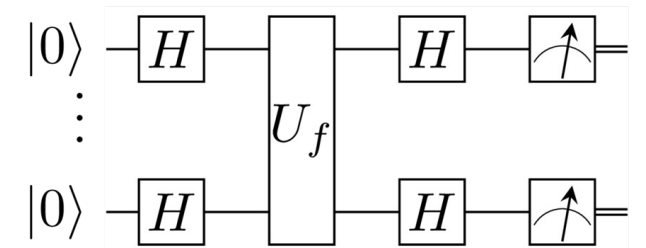


**Scenario-based
ADS Testing**

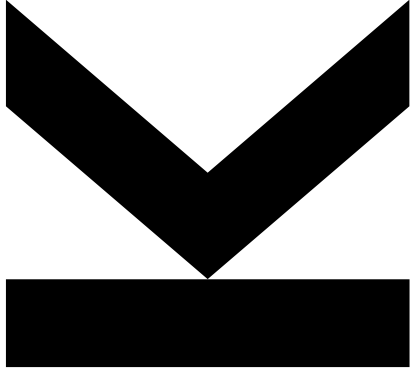


MDE

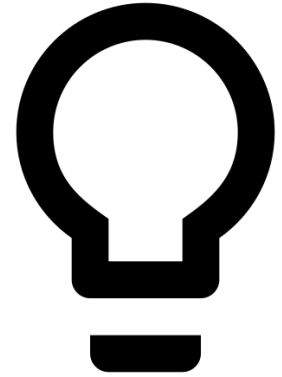
**Quantum
Software Eng.**



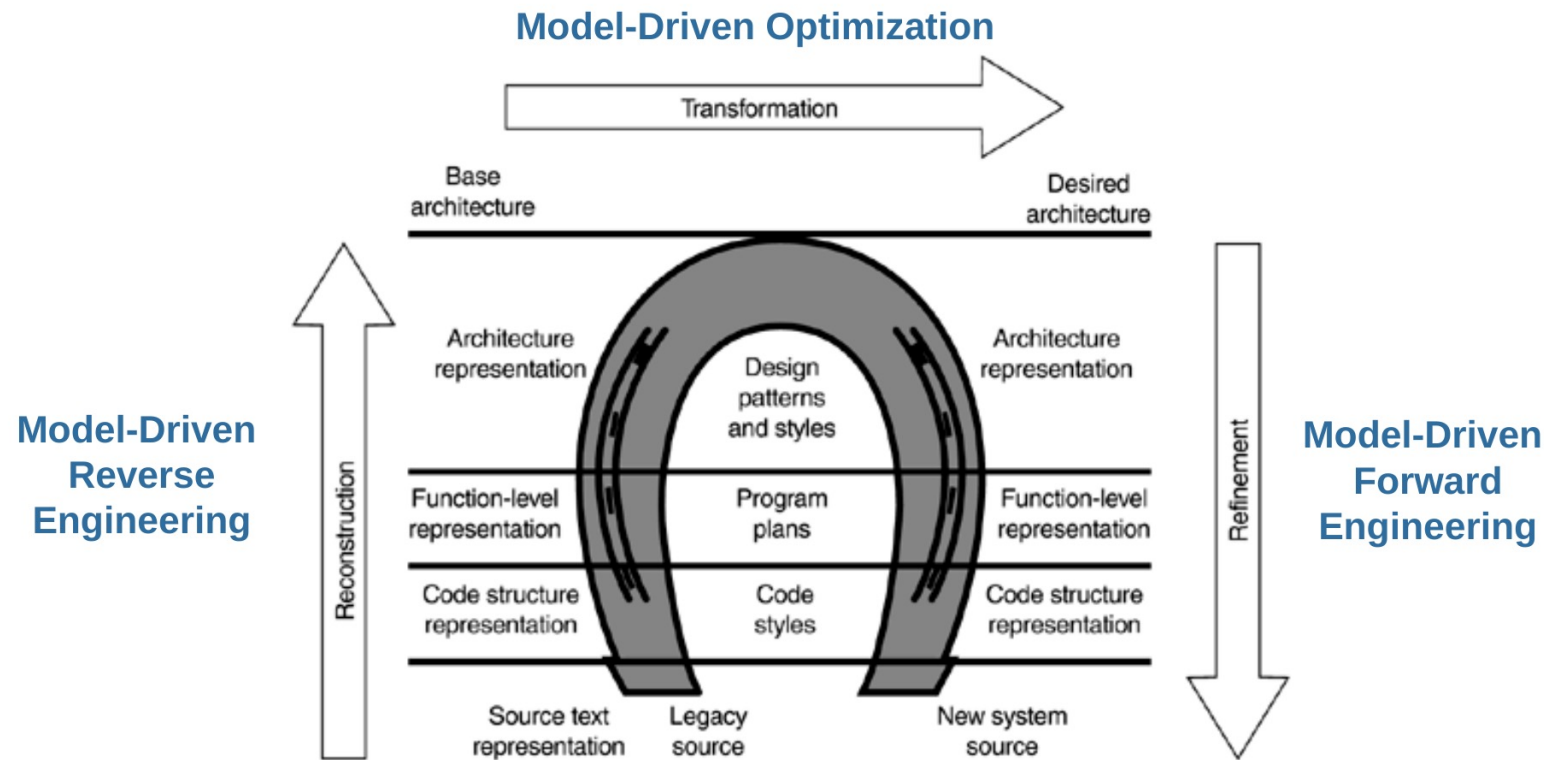
1 What is Model-Driven Optimization?



...and why would we use it?



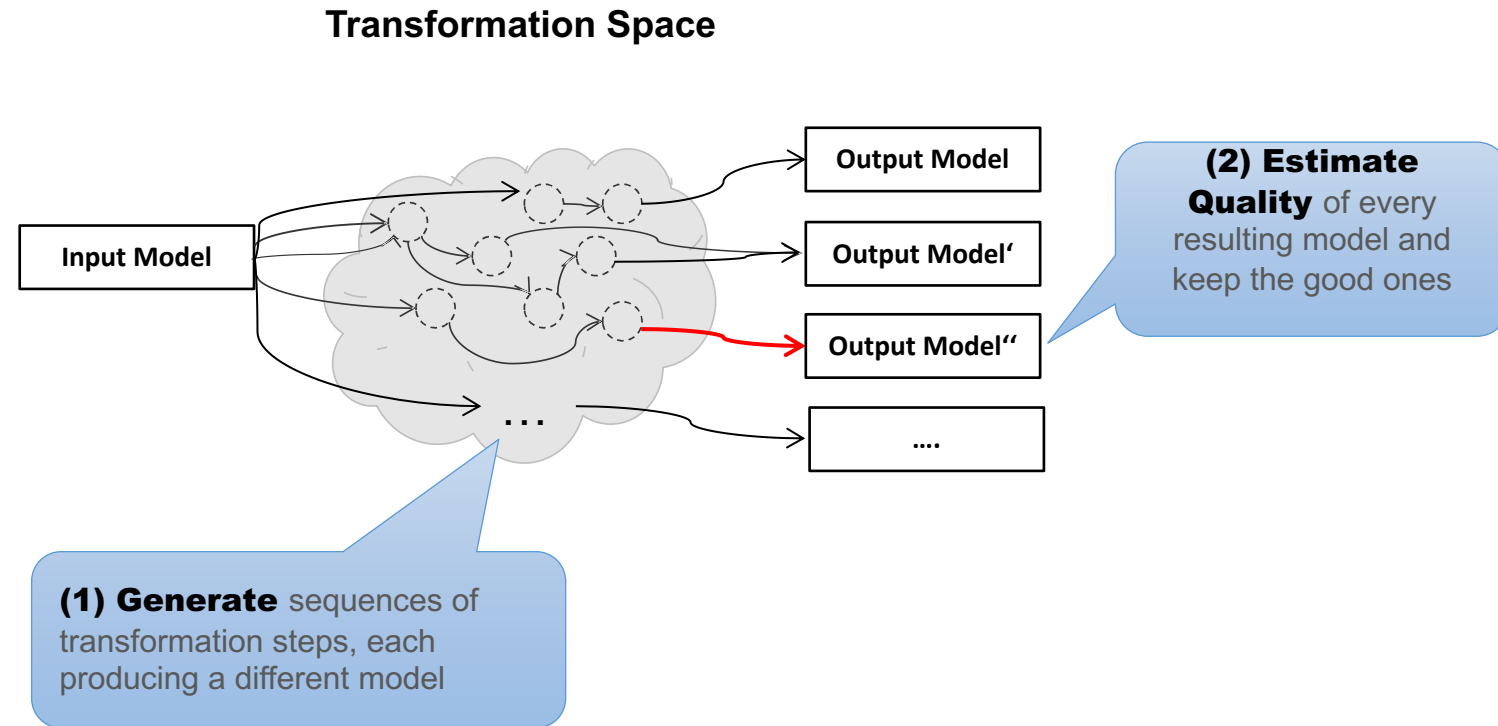
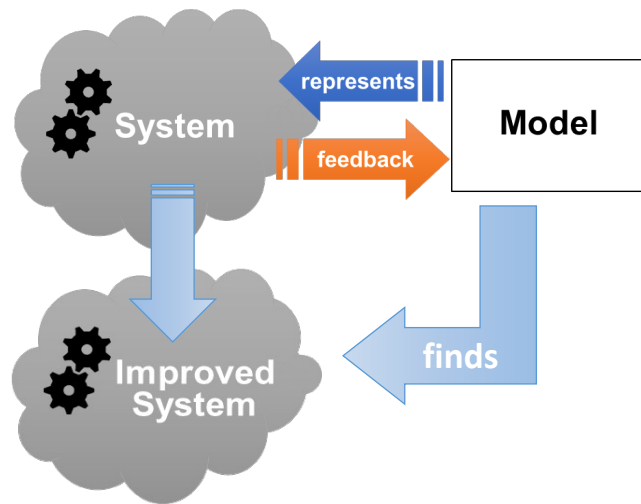
Engineering problems arrive as models



- **Software:** architectures, class diagrams, modules
- **Systems:** production lines, deployments, digital twins
- We already have **metamodels**, editors, constraints and **transformations** for them
- Question: which **changes** to a model make it **better**?

R. Kazman, S. G. Woods, S. J. Carrière: Requirements for integrating software architecture and reengineering models: CORUM II. Proc. WCRE 1998, pp. 154–163. Figure: M. Wimmer

Reasoning about possibilities

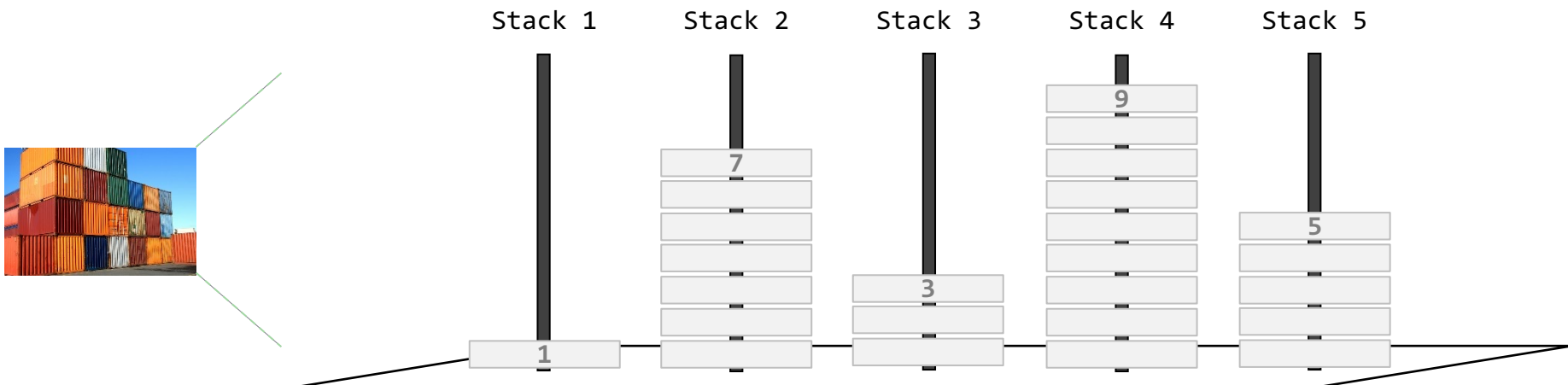


M. Fleck, J. Troya, M. Wimmer: Search-based model transformations. J. Softw. Evol. Process 28(12), pp. 1081–1117, 2016. Figure: M. Wimmer.

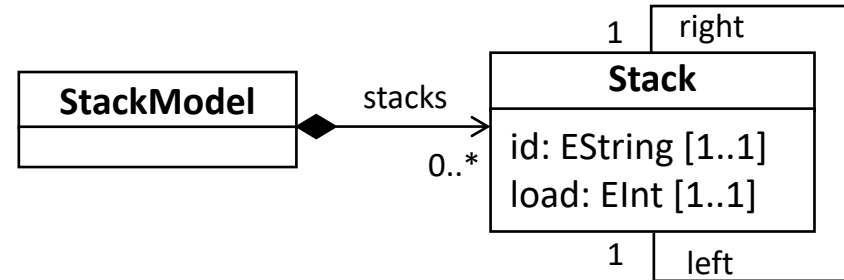
Introductory Example

Stack Load Balancing Problem

- Set of stacks with different loads
- Stacks are arranged in a circle
- Load should be distributed as equally as possible

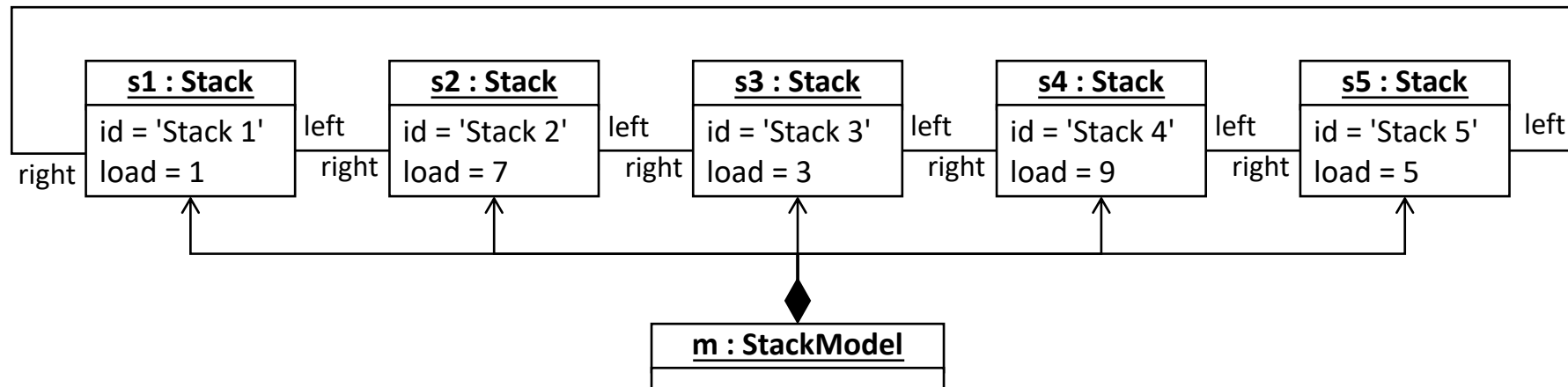


Load Balancing example



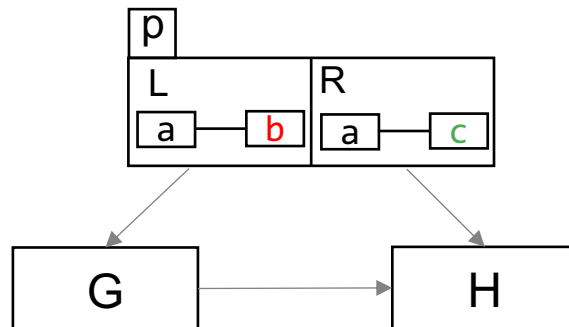
```

-- Stack ids are unique
context StackModel -- subsequent model elements are defined in the StackModel context
Stack.allInstances()->forall(s1, s2 | s1 <> s2 implies s1.id <> s2.id)
  
```

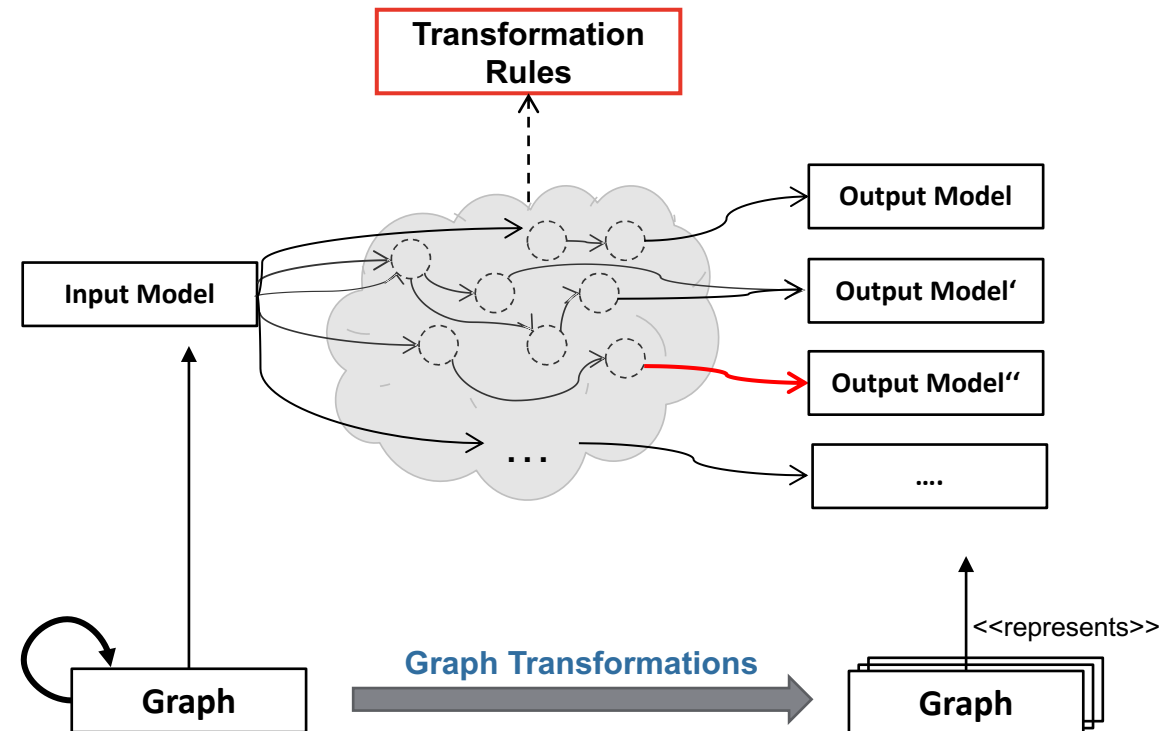


Operations and Executions (Formal Grounding)

A **graph transformation rule** $p: L \rightarrow R$ is a structure preserving, partial mapping between two graphs.

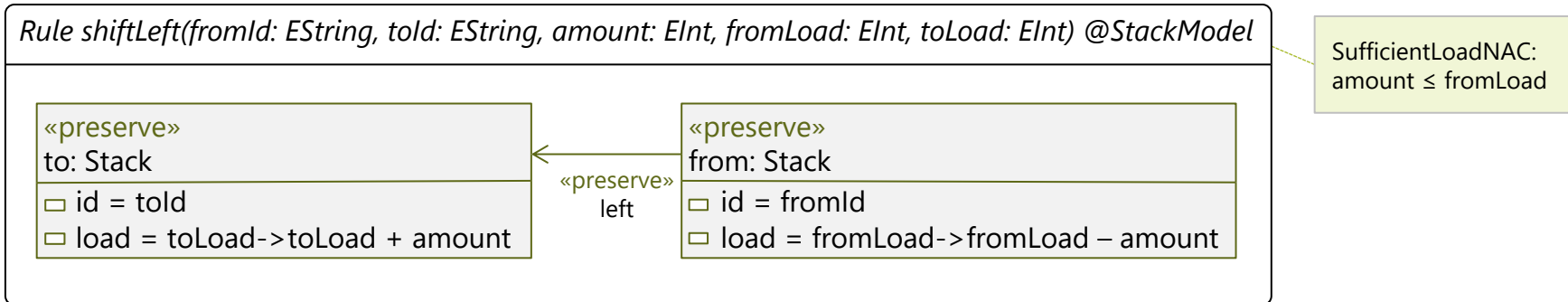


A **graph transformation** $t: G \rightarrow H$ is the result of the execution of a graph transformation rule $p: L \rightarrow R$ in the context of G .

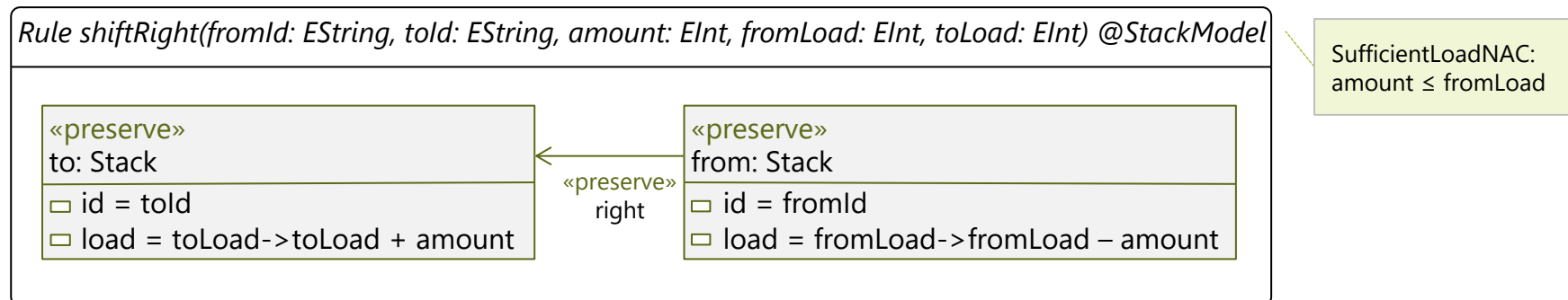


Manipulate models (problem instances) through model transformations

Rule:



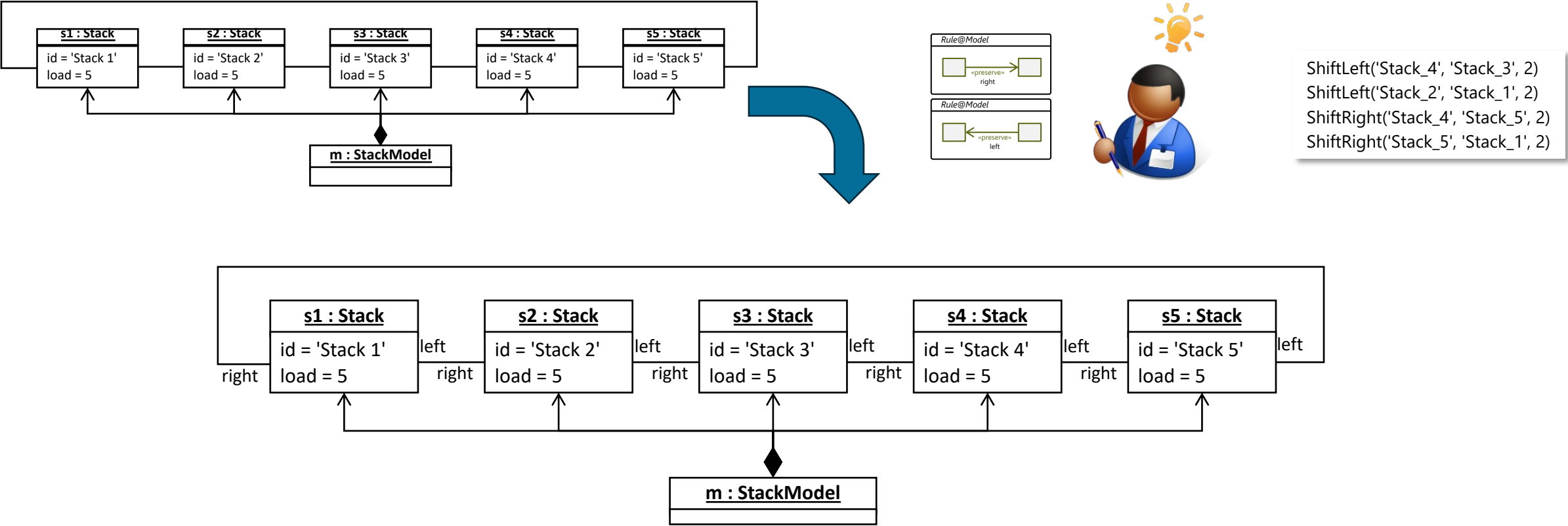
Rule:



Model-Driven Optimisation

Goal: Stacks should have equally distributed loads

Task: Find a good orchestration of transformation rules

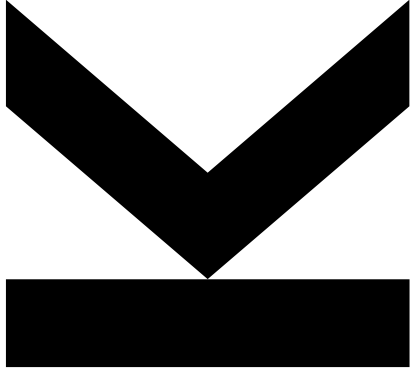


Why not by hand?

- A good rule order is not obvious
- Rule parameters may have very large value ranges
- Evaluating each resulting model manually is cumbersome
- A rule's effect on the goal is only implicit

→ **Use search-based optimisation (SBSE)**

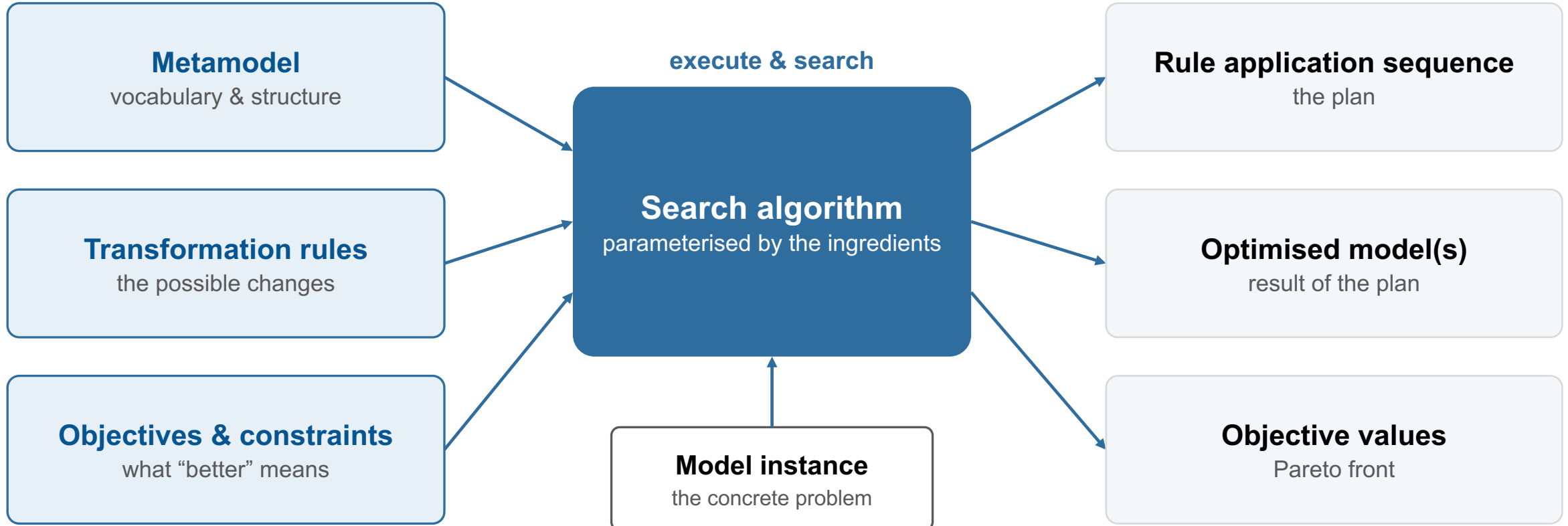
2 The ingredients of MDO and MOMoT



What do we have to provide?

The ingredients of MDO

Goal: find a **sequence of model transformations** that optimises the objectives.



Classic encoding vs. Model-Driven Optimization

Classic optimisation

- Hand-crafted encoding per problem (bit string, permutation, ...)
- Operators produce invalid solutions → repair or penalties
- Result: an end state
- Experts must understand the encoding

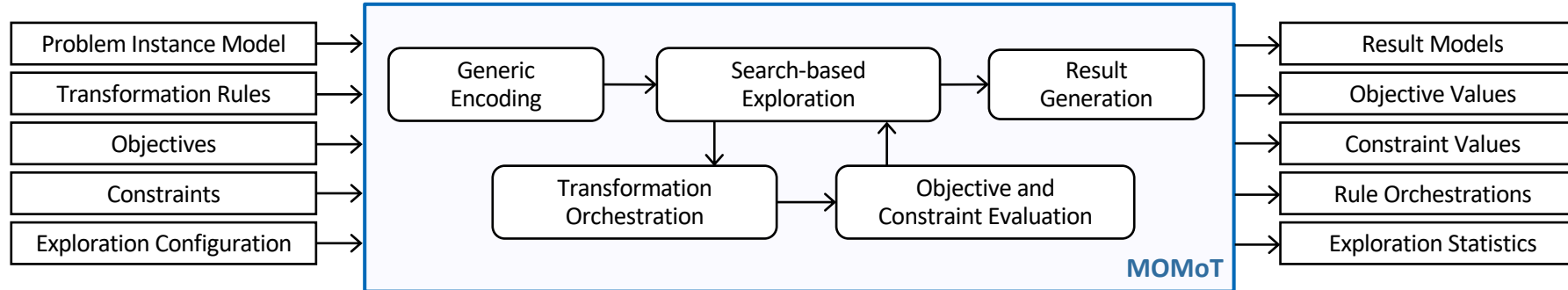
Model-Driven Optimization

- Model stays a model; search over **rule applications**
- Rules only fire where they match → **valid by construction**
- Result: an **explainable plan** (the rule sequence)
- Metamodel + rules are readable by domain experts

Costs:

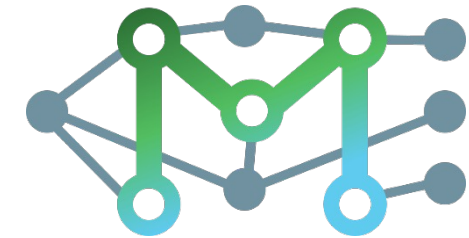
- rule application is more expensive than flipping bits
- the rule set and sequence length define what is reachable
- different sequences may yield the same model

MOMoT: Marrying Optimization and Model Transformations



MOMoT

- Based on [Eclipse](#) platform
- Eclipse [EMF](#) as modeling environment
- [Henshin](#) as model transformation engine
- [MOEA](#) Framework for SBO algorithms



M. Fleck, J. Troya, M. Wimmer: Search-based model transformations. J. Softw. Evol. Process. 28(12), pp. 1081-1117, 2016.

MOMoT Project: <http://martin-fleck.github.io/momot/>
Eclipse Modeling Framework: <https://eclipse.org/modeling/emf/>
Henshin Project: <https://www.eclipse.org/henshin/>
MOEA Framework: <http://moeaframework.org/>

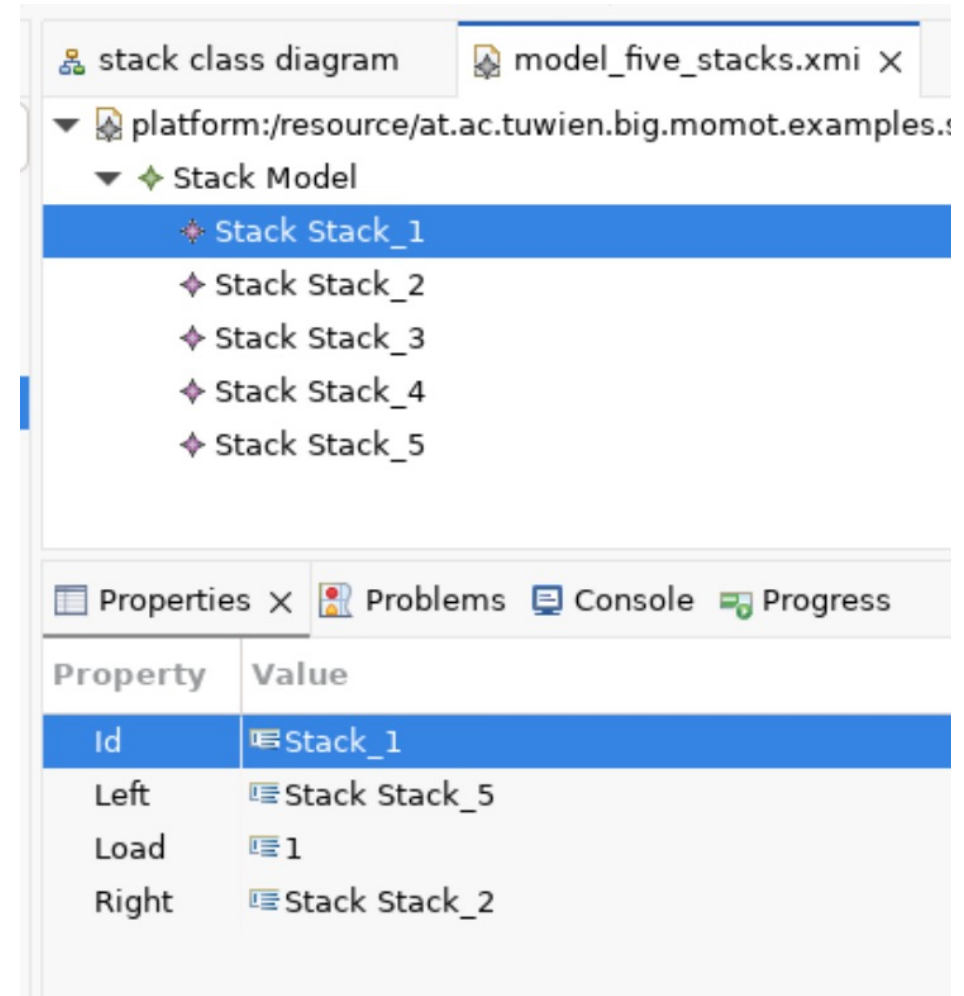
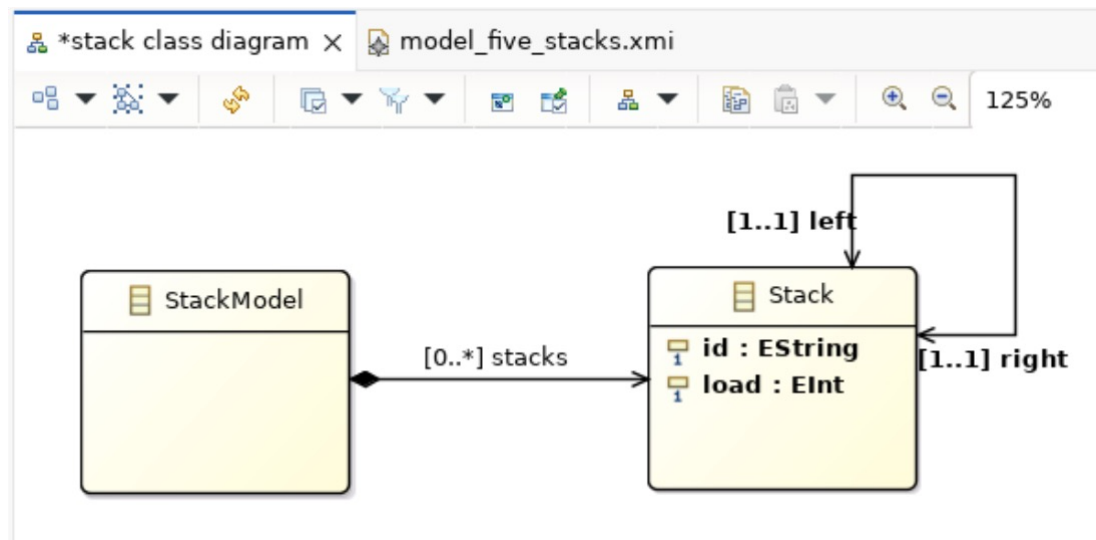
Ingredient 1: metamodel & model instance

Metamodel `.ecore`: concepts, attributes, references

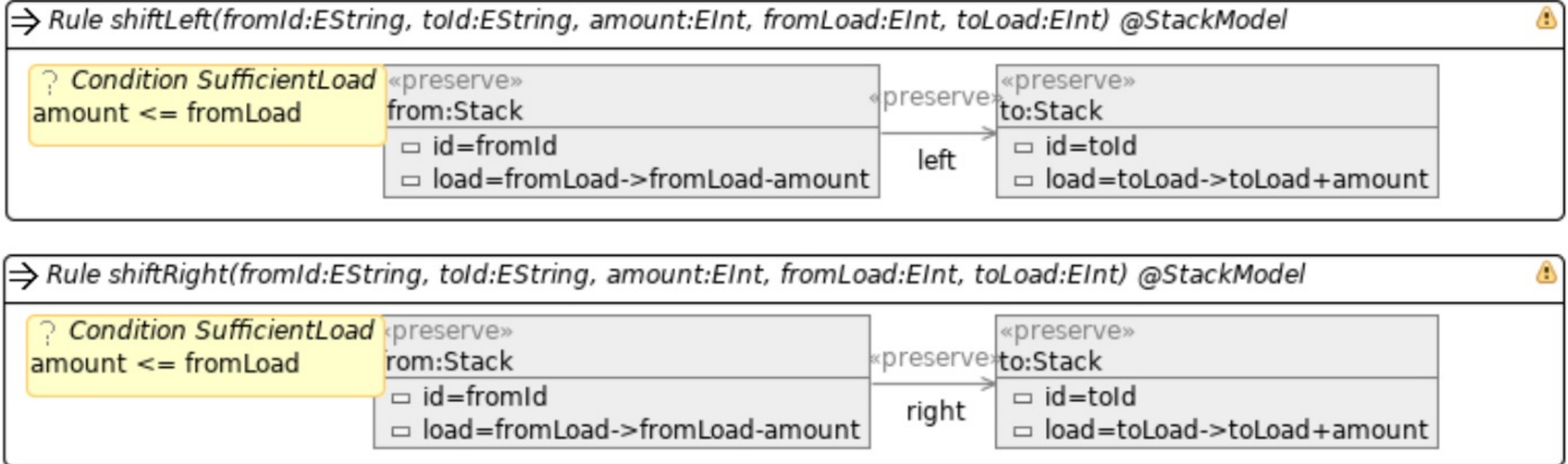
- Example: `StackModel` contains `Stacks` with `id`, `load`, `left/right`

Model instance `.xmi`: the concrete problem

- Example: five stacks in a ring, loads 1, 7, 3, 9, 5
- Multiplicities and OCL invariants act as **constraints for free**

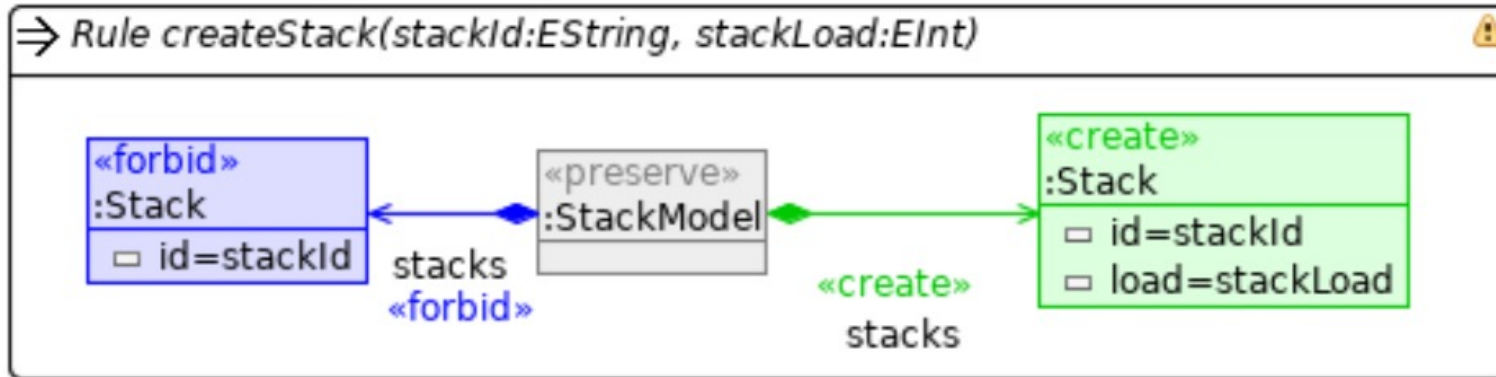


Ingredient 2: transformation rules



T. Arendt et al.: Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations. MODELS 2010, LNCS 6394, pp. 121–135. · R. Heckel, G. Taentzer: Graph Transformation for Software Engineers. Springer, 2020.

Ingredient 2: transformation rules



T. Arendt et al.: Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations. MODELS 2010, LNCS 6394, pp. 121–135. · R. Heckel, G. Taentzer: Graph Transformation for Software Engineers. Springer, 2020.

Ingredient 3: objectives & constraints

Domain-specific objective

Quality of the resulting model

`minimise` standard deviation of the loads

Solution-specific objective

Cost of the plan itself

`minimise` number of rule applications

Constraints

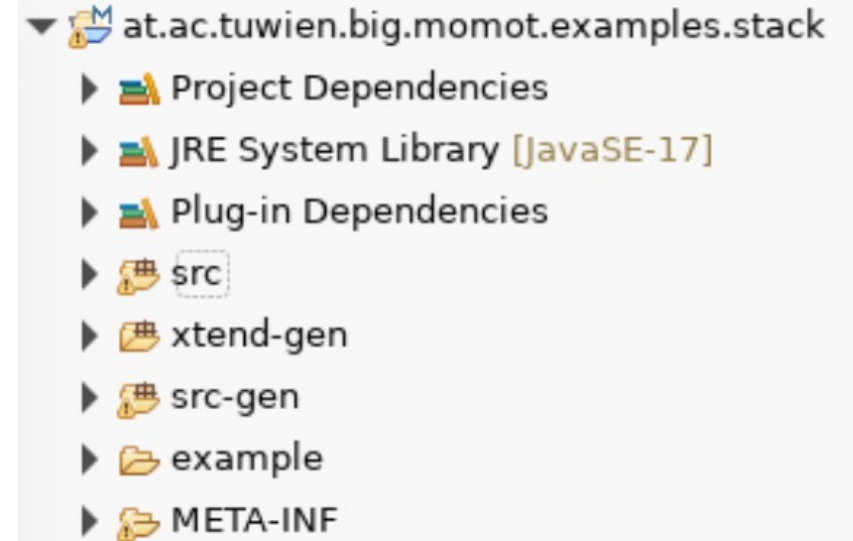
Mark solutions as invalid

e.g. no negative loads, no unassigned classes

```
objectives = { // could also use OCL as: minimize "ocl-expression"
  StandardDeviation : minimize { MathUtil.getStandardDeviation((root as StackModel).stacks.map[load]) }
  SolutionLength    : minimize new TransformationLengthDimension
}
```

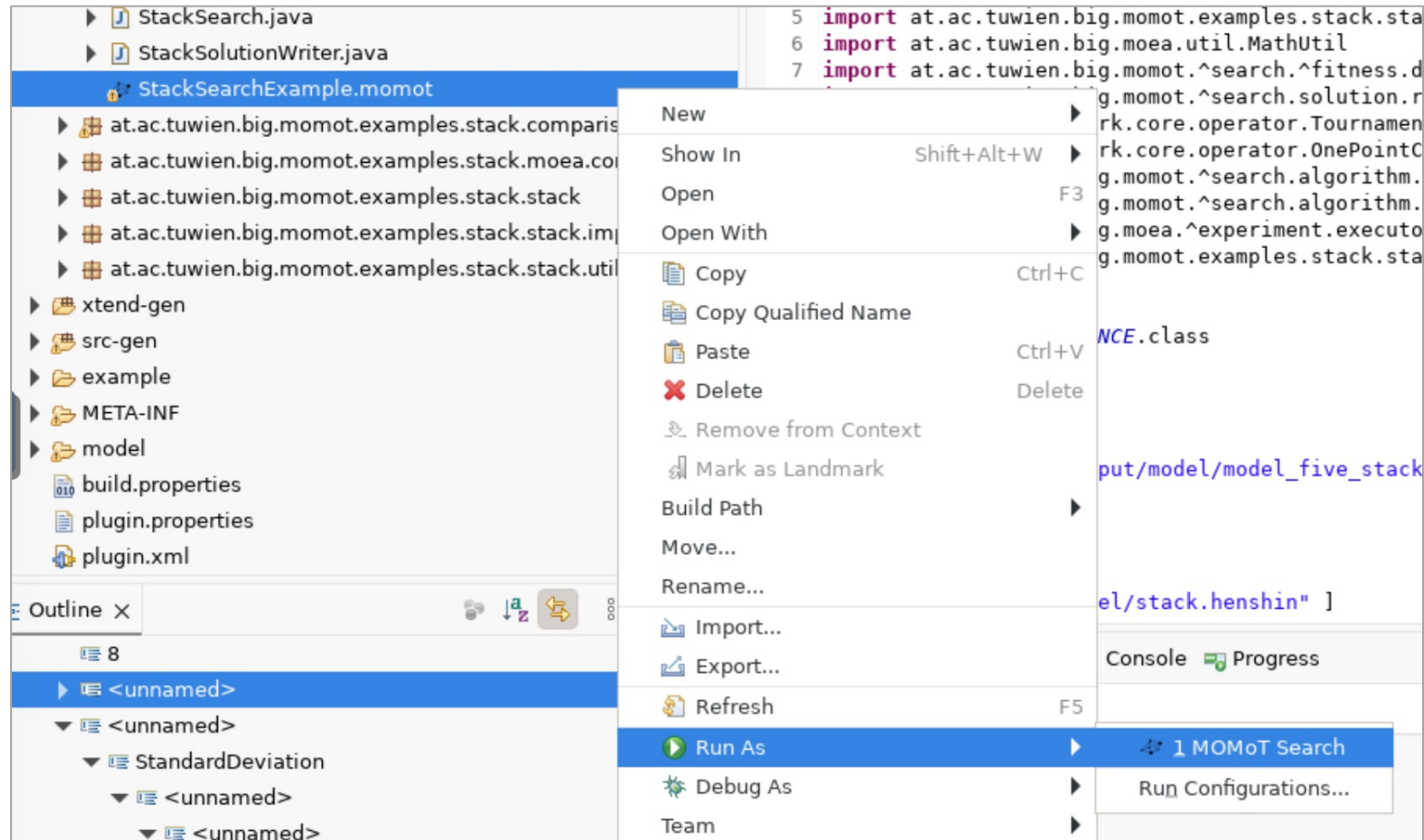
Your turn: find the ingredients

Project `at.ac.tuwien.big.momot.examples.stack`



Ingredient	File	Find ...
Metamodel	<code>model/stack.ecore</code>	the left / right references
Model instance	<code>model/input/model/model_five_stacks.xml</code>	the load of Stack_4
Transformation rules	<code>model/stack.henshin_diagram</code>	the condition of shiftLeft
Search specification	<code>src/.../stack/StackSearchExample.momot</code>	the two objectives

Hands-on: run the search



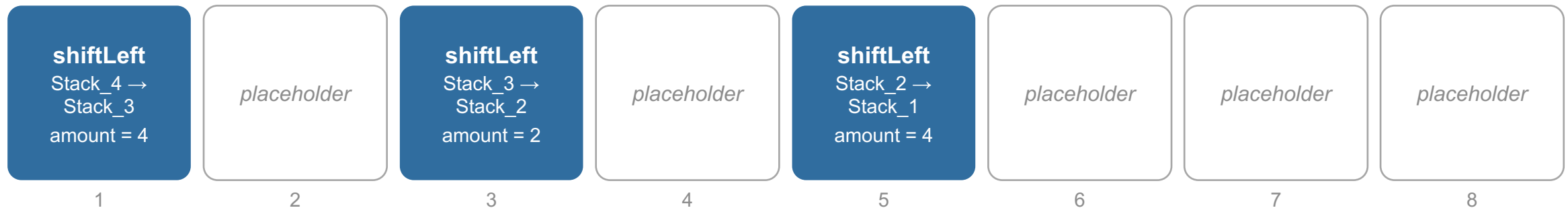
- 1 Right-click `StackSearchExample.momot`
- 2 Run As → **MOMoT Search**
- 3 Watch the Console
- 4 Refresh and open `example/output/`

While it runs: hover the warning on `nrRuns = 5` — what does the editor tell you?

**Explore the solutions, transformation chains,
results, etc.**

The search variable: a sequence of rule applications

Genotype = **transformation sequence** · Phenotype = **model obtained by executing it**



solutionLength = 8 slots; placeholders allow shorter plans



objectives
 $\sigma = 0.0$ · length = 3

The reveal: $\sigma = 0.0$ in 3 moves

	Stack_1	Stack_2	Stack_3	Stack_4	Stack_5
start	1	7	3	9	5
shiftLeft Stack_4 → Stack_3 : 4	1	7	7	5	5
shiftLeft Stack_3 → Stack_2 : 2	1	9	5	5	5
shiftLeft Stack_2 → Stack_1 : 4	5	5	5	5	5

- Net changes needed: +4, -2, +2, -4, 0
- Every step satisfies `amount <= fromLoad`
- Placeholders are not counted → length **3**

Who found (0.0, 3)? Who only (0.0, 4) or worse?

Metaheuristics give no guarantee → multiple runs, a random baseline, quality indicators and significance tests.

Hands-on: Let's alter the search. (Change the .momot ...)

```
search = {  
  model = { file = "model/input/model/model_five_stacks.xml" }  
  solutionLength = 8  
  transformations = {  
    modules = [ "model/stack.henshin" ]  
    ignoreUnits = [ CreateStack, ConnectStacks ]  
    parameterValues = { ...AMOUNT : new RandomIntegerValue(1, 5) }  
  }  
  fitness = { objectives = {  
    StandardDeviation : minimize { MathUtil.getStandardDeviation(...) }  
    SolutionLength : minimize new TransformationLengthDimension } }  
  algorithms = { Random : ... NSGA_II : ... NSGA_III : ... }  
}  
experiment = { populationSize = 100 maxEvaluations = 2000 nrRuns = 5 }  
analysis = { indicators = [ hypervolume ... ] ... }  
results = { objectives = { ... } solutions = { ... } models = { ... } }
```

model

model instance

transformations

rules (+ what the search may choose)

fitness

objectives & constraints

algorithms

search strategies

experiment / analysis

budget, runs, statistics

results

fronts, plans, models on disk

Hands-on: change the .momot file

Pick a task — everyone does A

A Turn the knobs

- `solutionLength`: 8 → 3 → 20
- `nrRuns`: 5 → 30 (warning disappears)
- `populationSize`, `maxEvaluations`

B Harder instance

- Use the `ten_stacks` model
- Is $\sigma = 0$ still reachable?
- Widen `amount` to 1–10

C New objective

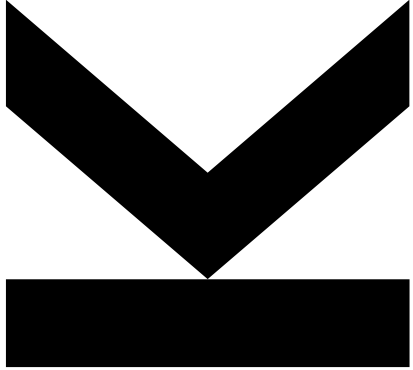
- Add `MaxLoad`
- minimise the largest load
- Hint: `.map[load].max`

D Other algorithms

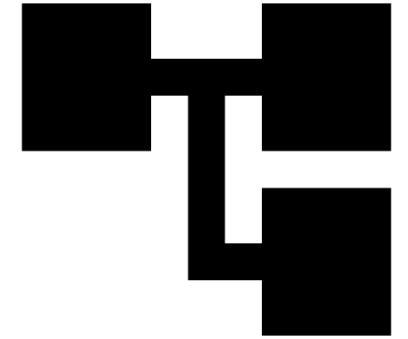
- Remove `NSGA_III`
- Add `createEpsilonMOEA`
- Change mutation rates

Remember: save before running · wait until the previous run has finished · compare the Pareto fronts

3 Search algorithms



Who consumes the ingredients?



Algorithm landscape

Same ingredients — different search strategies

Exact / classic

- Branch & bound
- Constraint programming
- Dynamic programming, A*

Guarantees, but only for small spaces

Local search

- Random descent
- Hill climbing
- Simulated annealing, tabu search

MOMoT: `local.createHillClimbing`

Evolutionary

- NSGA-II, NSGA-III
- ϵ -MOEA, SPEA2, IBEA
- Population of sequences

MOMoT: `moea.createNSGAI`

Learning-based

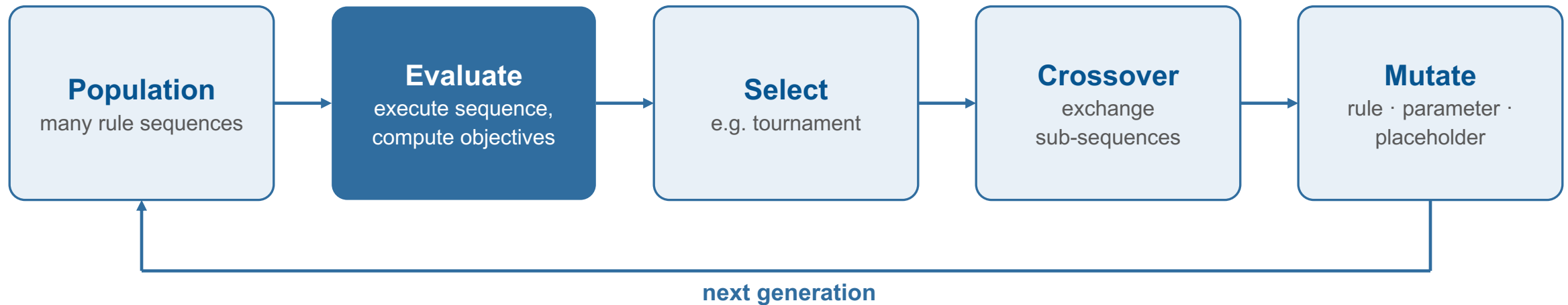
- Reinforcement learning
- Deep RL
- LLM-assisted search

MOMoT RL extension (Eisenberg & Wimmer 2025)

Always include a baseline: `moea.createRandomSearch()` — every result must beat random search.

Taxonomy after E.-G. Talbi: Metaheuristics: From Design to Implementation. Wiley, 2009.

Evolutionary search on transformation sequences

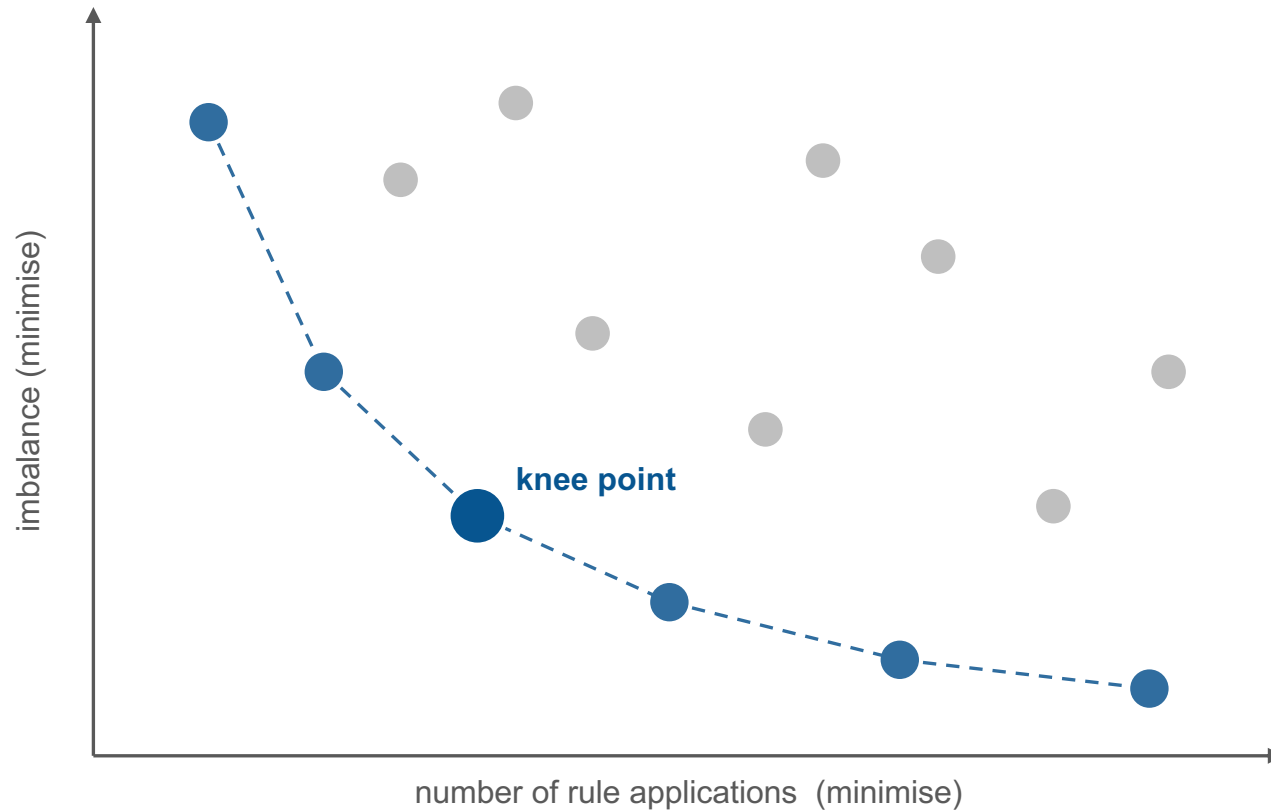


```
NSGA_II : moea.createNSGAII(  
    new TournamentSelection(2), new OnePointCrossover(1.0),  
    new TransformationPlaceholderMutation(0.15), new TransformationParameterMutation(0.1,  
    orchestration.moduleManager))
```

K. Deb, A. Pratap, S. Agarwal, T. Meyarivan: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), pp. 182–197, 2002.

Multi-objective optimisation

Objectives conflict — the result is a set, not a point



- **Pareto dominance:** better in one objective, not worse in any other
- **Pareto front:** all non-dominated solutions (blue)
- **Knee point:** best compromise — small gain in one objective costs a lot in the other
- The engineer chooses from the front

J. Branke, K. Deb, H. Dierolf, M. Osswald: Finding Knees in Multi-objective Optimization. PPSN VIII, LNCS 3242, pp. 722–731, 2004.

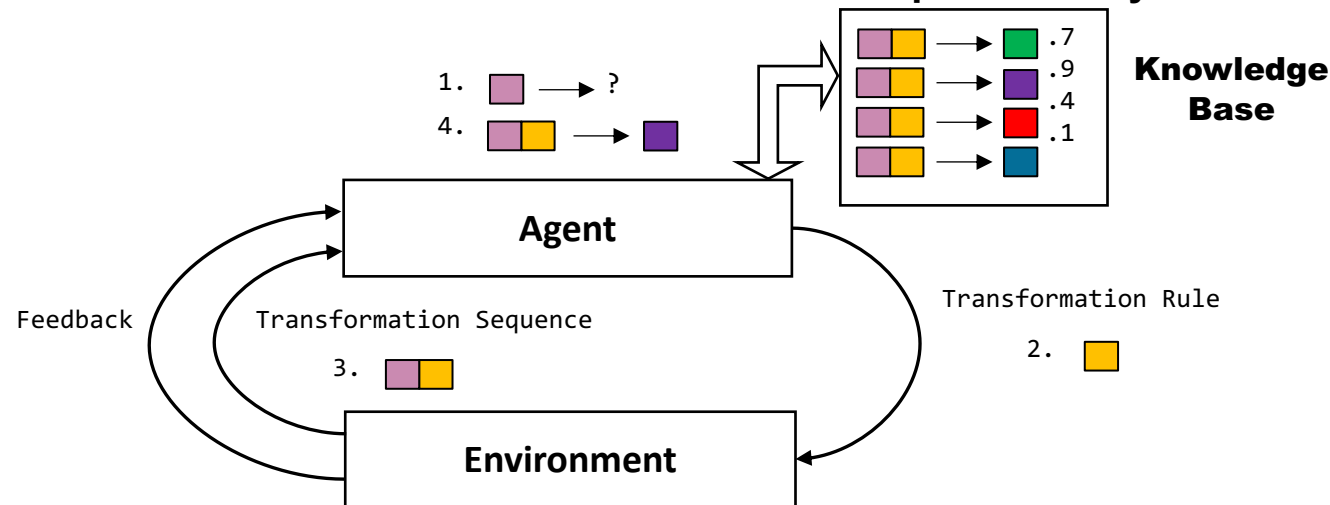
Beyond genetic search: RL and LLMs

Reinforcement learning

- State = current model, action = **rule application**
- Reward = improvement of the objectives
- Agent learns which rule to apply next
- Reported mostly on par with or better than the genetic encoding

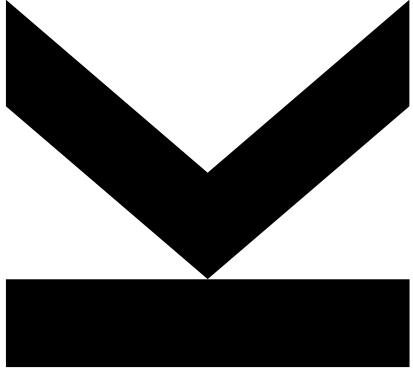
Large language models

- Ongoing work at JKU: class responsibility assignment by LLMs
- Toy model (9 features): correct solution, sensible class names
- Open question: **scale, correctness of fitness claims, reproducibility**

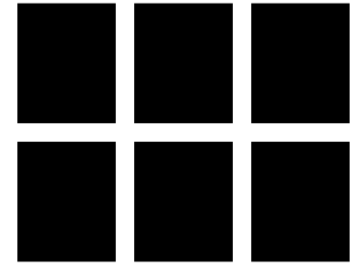


M. Eisenberg, M. Wimmer: From single-objective to multi-objective reinforcement learning-based model transformation. SoSyM 24(3), pp. 891–921, 2025.

4 Beyond toys: modularization

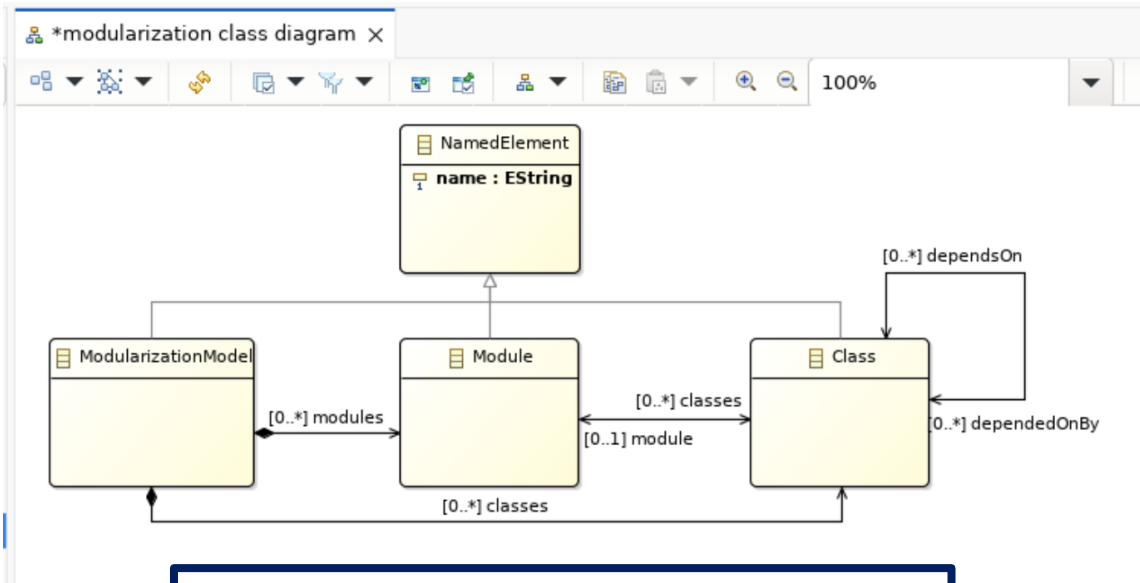


Look at the rules — then set up the search yourself



Hands-on: Software modularization

Assign classes to modules: low coupling, high cohesion



Assign 20 classes to modules

5.17×10^{13}

possible partitions

(Bell number $B_{20} = 51\,724\,158\,235\,372$)

Metamodel

- `ModularizationModel` contains `modules`, `classes`
- `Class`: `module`, `dependsOn`, `dependedBy`

Rules (look only)

- `createModule(moduleName)` — name must be new
- `assignClass(module, clazz)` — class not yet assigned
- `reassignClass(module, clazz)` — move a class

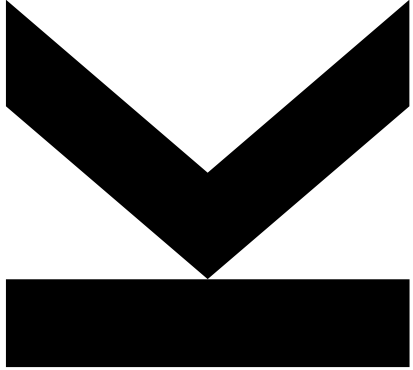
Objectives & constraints

- min **coupling** · max **cohesion** · max **MQ**
- no unassigned classes · no empty modules

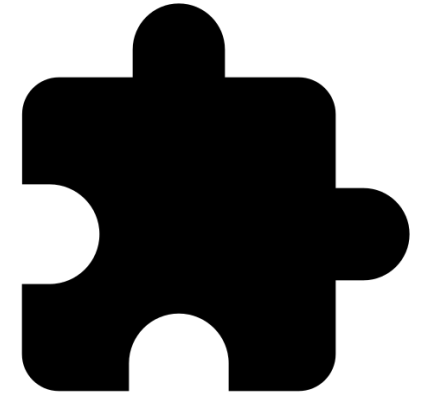
Instances: `mdg_noModules.xmi` (8 classes) · `mtunis.xmi` (20 classes)

Bottom left: mtunis dependency graph. K. Praditwong, M. Harman, X. Yao: Software Module Clustering as a Multi-Objective Search Problem. IEEE TSE 37(2), pp. 264–282, 2011.

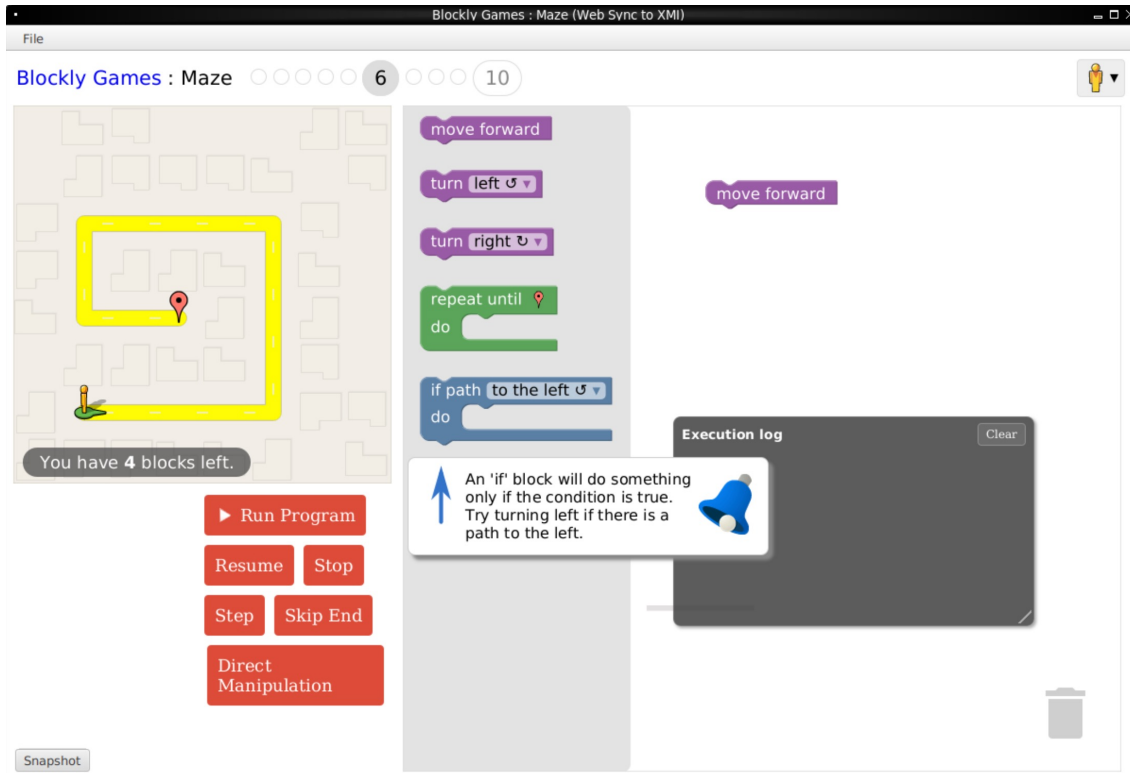
5 Showcase: Blockly Maze



MDO behind a graphical user interface



Blockly Maze: the same ingredients



Metamodel

- `blocky.ecore`: Game, Level, GridMap, Cell, Body, Statement, Loop, IfStmt, ...

Transformation rules

- insert containers, populate them with moves / loops / ifs, delete containers
- search uses composite units
`CreateThenInsertContainerThenPopulate`,
`DeleteContainerAnywhere`

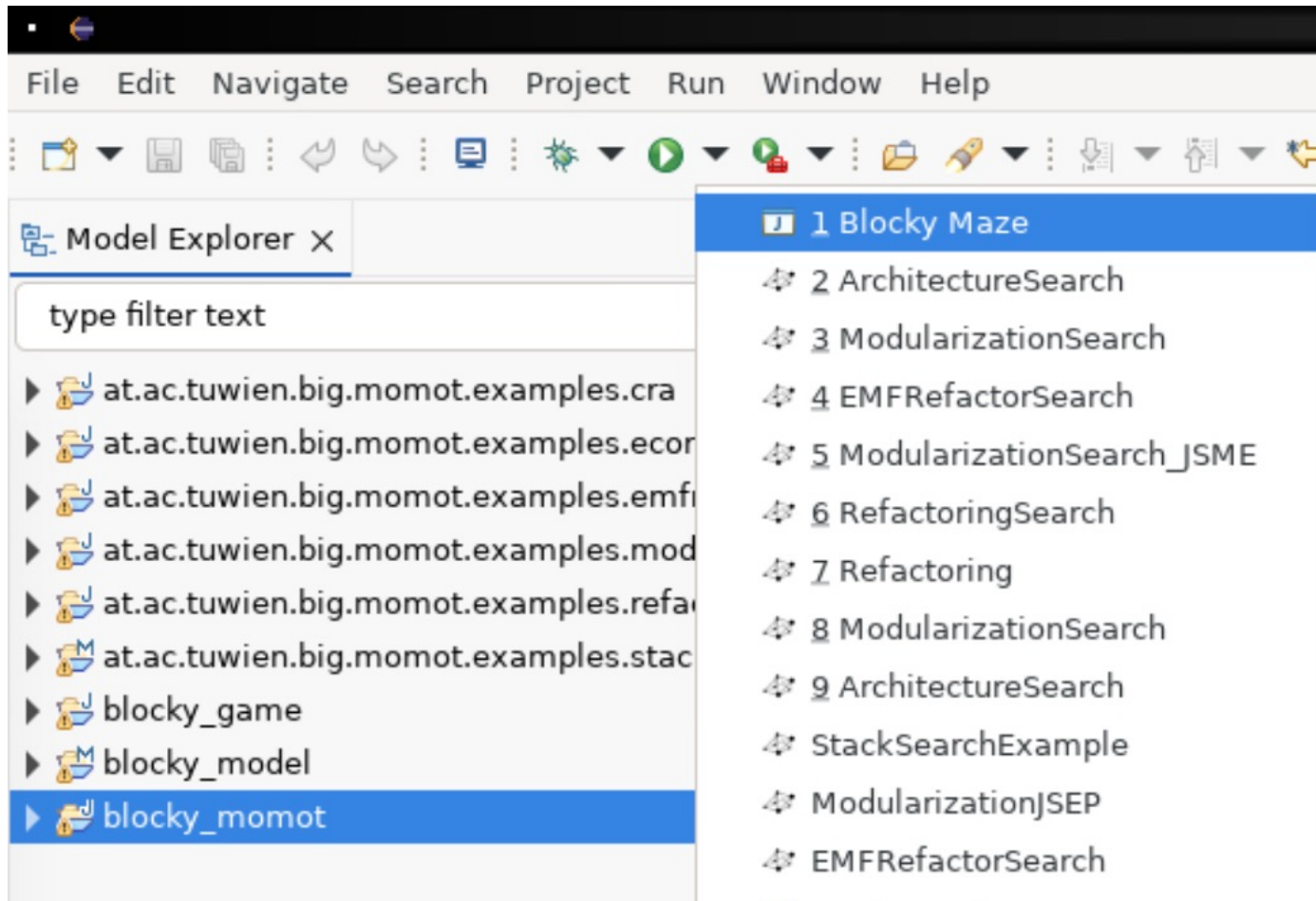
Objectives (headless simulator)

- $\max \text{GoalReached} \cdot \min \text{Edits} \cdot \min \text{ShortestPath} \cdot \min \text{closestToGoal}$

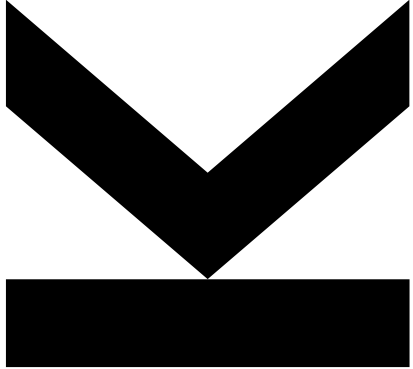
Algorithm

- NSGA-II

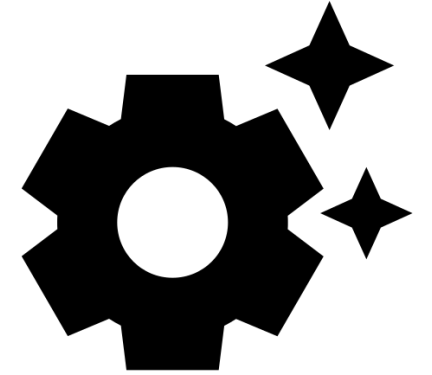
Demo: from playing to searching



6 Competitor tools



Alternatives to MOMoT and to Henshin



The MDO tool landscape

Same ingredients — different searcher, different guarantees

Metaheuristic search

- **MOMoT** — sequences of rule applications
- **MDEOptimiser** — rules as mutation operators
- **VIATRA-DSE** — trajectories in a rule-induced state space
- No guarantees · scales to large spaces

Exact / solver-based

- **GIPS / GIPSL** — graph patterns compiled to ILP
- **Refinery** — logic-based partial models
- **Clafer, Alloy** — constraints over configurations
- Optimality or unsatisfiability · scope-bounded

Learning-based

- **RL in MOMoT** — agent picks the next rule
- **RL + human guidance** — preferences in the loop
- **LLM-assisted MDE** — propose, explain, repair
- Reusable policies · evaluation still open

The ingredients stay the same — metamodel, rules, objectives. What changes is **who searches** and **with what guarantees**.

What is the search variable?

The decisive difference between the tools

MOMoT

sequence of rule applications

```
shiftLeft(S4→S3,4) · shiftLeft(S3→S2,2) · ...
```

MDEOptimiser

the model itself, mutated by rules

```
model → rule → model' → rule → model''
```

VIATRA-DSE

trajectory through the state space

```
state0 → state1 → state2 → ... (goal states)
```

GIPS / Refinery

variables derived from pattern matches

```
 $x_{ij} \in \{0,1\}$ , maximise  $c^T x$  s.t.  $Ax \leq b$ 
```

Consequence: what is **reachable**, what is **explainable**, and what can be **guaranteed** differ per tool.

The two closest relatives

MDEOptimiser

King's College London · EMF + Henshin

- Optimisation spec in its own **DSL** (objectives, constraints, operators)
- Henshin rules act as **mutation operators** on the model
- Work on automatically generating consistency-preserving operators
- Same stack as MOMoT — different **encoding decision**

VIATRA-DSE

BME Budapest · EMF + VIATRA queries

- Exploration of a **rule-induced state space** towards goal states
- Incremental graph queries ([VQL](#)) for fast re-evaluation
- Multi-objective search (NSGA-II) and trajectory-level objectives
- Solved the same CRA benchmark at TTC 2016

A. Burdusel, S. Zschaler, D. Strüder: MDEOptimiser — A Search Based Model Engineering Tool. MODELS '18 Companion, pp. 12–16.

Á. Hegedüs, Á. Horváth, D. Varró: A model-driven framework for guided design space exploration. Autom. Softw. Eng. 22(3), pp. 399–436, 2015.

Exact and solver-based alternatives

When “good enough” is not enough

GIPS / GIPSL

eMoflon · TU Darmstadt

- Graph patterns + objectives compiled into an integer linear program
- Solved by an off-the-shelf ILP solver → proven optimum
- Cost: the problem must fit a linear formulation

Refinery

graph solver (VIATRA Solver line)

- Logic-based refinement of partial models
- Generates consistent, well-formed models under constraints
- Complementary: generate candidates, then optimise

Clafer / Alloy

constraint-based modelling

- Feature/class models with attributes and constraints
- Multi-objective optimisation over configurations (Clafer)
- Scope-bounded analysis, strong tool support

Rule of thumb: metaheuristics when the space is huge and the objectives are messy; solvers when the model is small enough and the formulation is clean.

S. Ehmes, M. Kratz, A. Schürr et al.: Graph-Based Specification and Automated Construction of ILP Problems (GIPS). arXiv:2212.11629, GCM 2022. · Refinery: Graph Solver as a Service. ICSE 2024 companion, doi:10.1145/3639478.3640045.

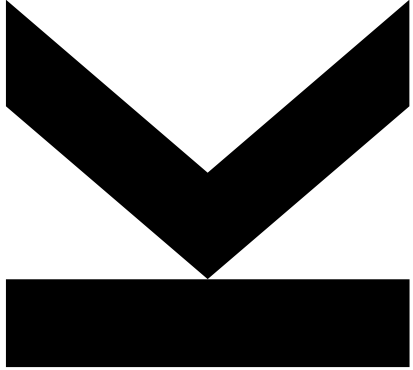
Alternatives to Henshin

Transformation engines you could plug into an MDO loop

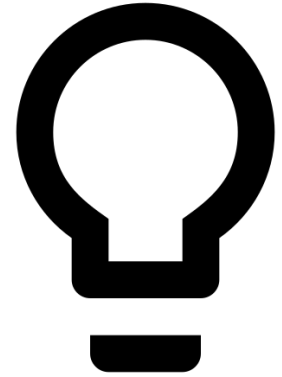
Engine	Paradigm	Why it matters here
Henshin	algebraic graph transformation on EMF	what MOMoT uses; graphical + textual syntax, state-space tools
VIATRA (3)	reactive, incremental graph queries	fastest re-evaluation; basis of VIATRA-DSE
eMoflon::IBeX / ::Neo	triple graph grammars, bidirectional	consistency and bx; hosts the GIPS tooling
AGG	classical attributed graph transformation	critical-pair analysis: conflicts and dependencies of rules
GrGen.NET	high-performance graph rewriting (.NET)	speed-oriented option outside EMF
GROOVE	graph transformation + state space	when you want all reachable models, not an optimum
ATL / QVT / Epsilon	declarative, mostly out-place	standard MDE transformation stacks; not designed as search operators
NMF	.NET modelling, incremental	non-Java option; solved the TTC CRA case

Henshin: T. Arendt et al., MODELS 2010, LNCS 6394, pp. 121–135 · D. Strüber et al., ICGT 2017, LNCS 10373, pp. 196–208. · VIATRA: G. Bergmann et al., ICMT 2015, LNCS 9152, pp. 101–110.

7 Recent developments



What happened in the last two years



Recent developments (2024–2026)

Transformation chains 2024

- MOMoT applied to **chains** of transformations, not single searches
- Eisenberg, Sahay, Di Ruscio, Iovino, Wimmer, Pierantonio
- [Inf. Softw. Technol. 174:107500](#)

Reinforcement learning 2025

- From single- to multi-objective RL-based model transformation
- RL with **uncertain human guidance** (MODELS 2025)
- [SoSyM 24\(3\):891–921](#) · [arXiv:2506.20883](#)

LLMs in MDE 2025

- Systematic picture of what LLMs are actually used for in MDE
- Model repositories, recommenders, completion — and MDE for LLMs
- [SoSyM 24\(3\):923–948](#)

MDO beyond software 2024

- Model-driven optimization applied to **business processes**
- Taentzer, Kosiol, Lambers
- [PETRI NETS 2024, LNCS 14628, pp. 24–35](#)

Where the field seems to be heading

Open problems worth a thesis

1

Search-space design

The rule set and the sequence length decide what is reachable. Almost no tooling tells you that your optimum is unreachable.

2

Scalability

Rule application dominates evaluation cost — incremental matching, parallel evaluation, surrogate models.

3

Explainability

The transformation sequence is already an explanation; turning it into a rationale a human accepts is open.

4

Humans in the loop

Preference-guided and interactive search; RL with uncertain human guidance is a first step.

5

LLMs as operators

Proposing rules, seeding populations, explaining fronts — with fitness claims that must be checked, not trusted.

6

Runtime / digital twins

Plans computed on a model that drifts: repair, replan, or re-seed the search.

Take-home messages

- 1 MDO searches over **transformation sequences** — the model stays a model
- 2 Three ingredients: **metamodel, rules, objectives** — plus an algorithm
- 3 Rules and sequence length **define the reachable space**
- 4 No guarantees → **multiple runs, random baseline, indicators**

Continue:

jku-win-se.github.io/MOMoT2 (docs, case studies, Eclipse update site)
github.com/hadiDHD/fame-26-mdo-workspace

This tutorial would not have been possible without

Manuel Wimmer

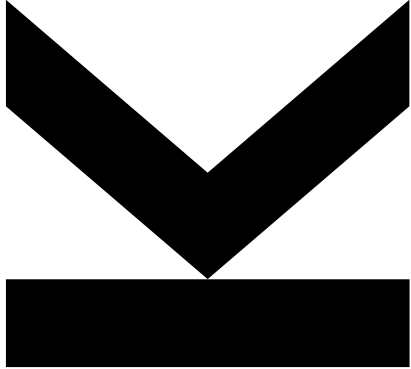
MohammadHadi Dehghani

Martin Eisenberg

...and many other colleagues.

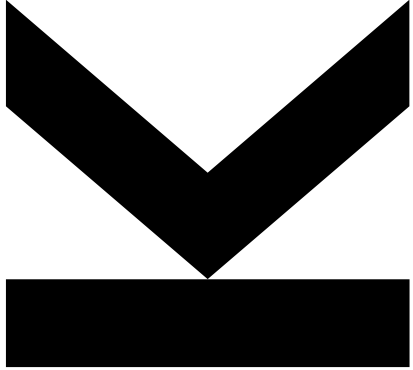


Model-Driven Optimization Hands-on with MOMoT

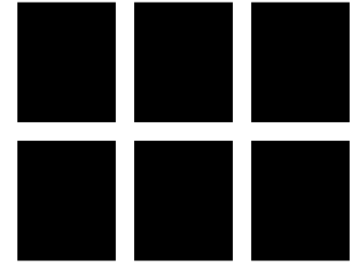


FAME 2026, University of Málaga · 1 October 2026 · Stefan Klikovits · JKU Linz
<https://klikovits.net>

8 Related work



The neighbourhood: fields, benchmarks, foundations



Neighbouring fields

What MDO borrows from, and what borrows from MDO

Search-based software engineering (SBSE) — The umbrella: metaheuristics for software engineering problems. MDO = SBSE with the model as the first-class artefact.

M. Harman, FOSE 2007, pp. 342–357

Software module clustering — The multi-objective remodularisation formulation used in the hands-on exercise.

K. Praditwong, M. Harman, X. Yao, IEEE TSE 37(2), pp. 264–282, 2011

Design-space exploration (embedded systems) — Same problem shape — architectures, mappings, deployments — usually without an MDE layer.

Opt4J / OpenDSE and related toolchains

Model finding & consistent generation — Generate valid models under constraints instead of improving an existing one.

Refinery, Alloy

Graph transformation theory — Rules, matches, application conditions, confluence — the formal basis of the operators.

R. Heckel, G. Taentzer: Graph Transformation for Software Engineers. Springer, 2020

Benchmarks and comparisons

How the community compares MDO approaches

Benchmark problems

- **TTC 2016 — Class Responsibility Assignment:** nine tool solutions to one problem (incl. MOMoT, VIATRA-DSE, NMF, SDMLib, UML-RSDS)
- **Module clustering (mtunis, MDG suite):** coupling / cohesion / MQ, used in the hands-on exercise
- **Stack load balancing:** the toy with a known optimum — useful precisely because it is verifiable
- **Native vs. MDE encoding:** same problem, hand-written encoding — the honest runtime comparison

Reporting discipline

- **Quality indicators:** hypervolume, additive epsilon, IGD — never single runs
- **Baselines:** random search first; then the algorithm you advocate
- **Statistics:** ≥ 30 runs, significance tests, effect sizes
- **Reachability:** report whether the optimum is reachable by your rules at all

The MOMoT editor already warns when `nrRuns` is too small — the tool enforces part of this discipline for you.

Reading list

All references verified against publisher metadata, 29 September 2026

R. Bill, M. Fleck, J. Troya, T. Mayerhofer, M. Wimmer: A local and global tour on MOMoT. SoSyM 18(2), pp. 1017–1046, 2019. doi:10.1007/s10270-017-0644-3

M. Fleck, J. Troya, M. Wimmer: Search-based model transformations. J. Softw. Evol. Process 28(12), pp. 1081–1117, 2016. doi:10.1002/smr.1804

M. Eisenberg, A. Sahay, D. Di Ruscio, L. Iovino, M. Wimmer, A. Pierantonio: Multi-objective model transformation chain exploration with MOMoT. Inf. Softw. Technol. 174:107500, 2024. doi:10.1016/j.infsof.2024.107500

M. Eisenberg, M. Wimmer: From single-objective to multi-objective reinforcement learning-based model transformation. SoSyM 24(3), pp. 891–921, 2025. doi:10.1007/s10270-024-01233-6

A. Burdusel, S. Zschaler, D. Strüber: MDEOptimiser — A Search Based Model Engineering Tool. MODELS '18 Companion, pp. 12–16. doi:10.1145/3270112.3270130

Á. Hegedüs, Á. Horváth, D. Varró: A model-driven framework for guided design space exploration. Autom. Softw. Eng. 22(3), pp. 399–436, 2015. doi:10.1007/s10515-014-0163-1

G. Taentzer, J. Kosiol, L. Lambers: On the Application of Model-Driven Optimization to Business Processes. PETRI NETS 2024, LNCS 14628, pp. 24–35. doi:10.1007/978-3-031-61433-0_2

J. Di Rocco, D. Di Ruscio, C. Di Sipio, P. T. Nguyen, R. Rubel: On the use of large language models in model-driven engineering. SoSyM 24(3), pp. 923–948, 2025. doi:10.1007/s10270-025-01263-8

I. Boussaïd, P. Siarry, M. Ahmed-Nacer: A survey on search-based model-driven engineering. Autom. Softw. Eng. 24(2), pp. 233–294, 2017. doi:10.1007/s10515-017-0215-4

J. Denil, M. Jukss, C. Verbrugge, H. Vangheluwe: Search-Based Model Optimization Using Model Transformations. SAM 2014, LNCS 8769, pp. 80–95. doi:10.1007/978-3-319-11743-0_6

Full list with tool websites and the unverified items: [RelatedWork.md](#) in the tutorial folder.