Search-based Model Transformation Analysis*

4th Workshop on the Analysis of Model Transformations (AMT) @ MoDELS'15

Manuel Wimmer

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

*This work is co-funded by the European Commission under the ICT Policy Support Programme, grant no. 317859 (ARTIST) as well as by the Christian Doppler Forschungsgesellschaft and the BMWFW, Austria.

Introduction

- **Model Transformations** are **widely used/usable** today
- For **general techniques**
  - Model exchange, diffing, merging, versioning, evolution, execution, annotations, verification, modernization, …
- For **specific application areas**
  - DB, OO, Web, documents, Cloud, social networks, production systems, buildings, …
- To ensure their proper usage, different **properties** are desirable
  - **Classical properties** such as termination, confluence, …
  - **Correctness** w.r.t. specifications
  - **ilities** such as readability, extensibility, maintainability, …

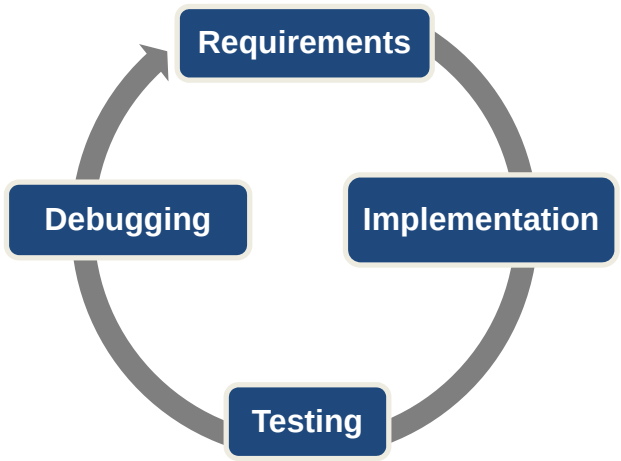
L. Lucio, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. Selim, E. Syriani, M. Wimmer: ***Model transformation intents and their properties***. Software & Systems Modeling, pp. 1–38, 2014.

Analysis of Model Transformation (AMT)

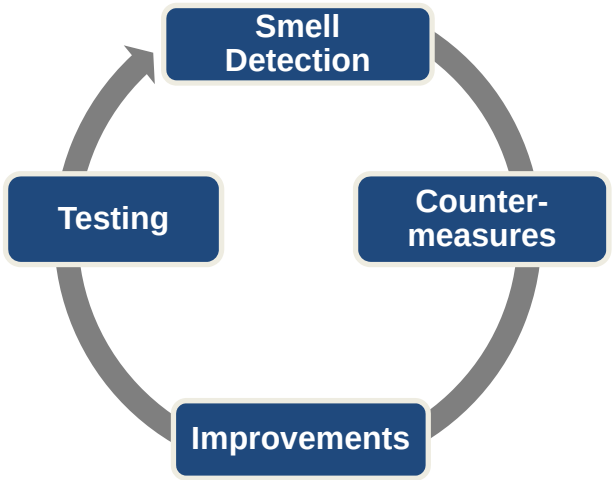
Why?

- Required in **many** different model engineering **processes** and **tasks**
- Our past experiences (excerpt)

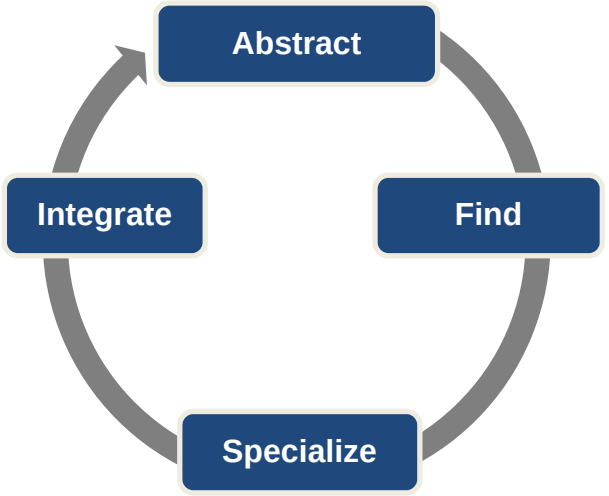
Transformation Testing



Transformation Refactoring



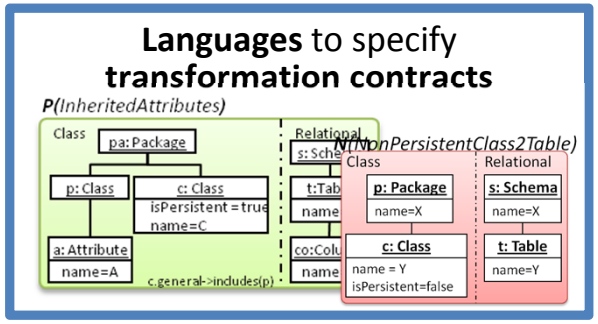
Transformation Reuse

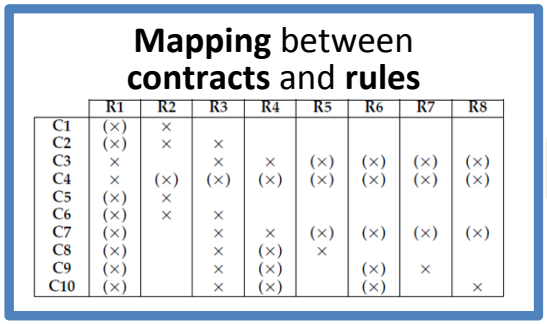


AMT
How?

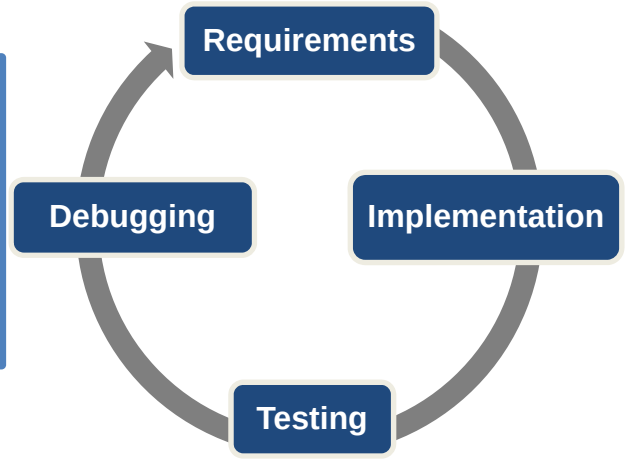
- E. Guerra, J. de Lara, M. Wimmer, G. Kappel, A. Kusel, W. Retschitzegger, J. Schönböck, W. Schwinger: ***Automated verification of model transformations based on visual contracts.*** Journal of Automated Software Engineering 20(1):5-46 (2013).
- L. Burgueño, J. Troya, M. Wimmer, A. Vallecillo***: Static Fault Localization in Model Transformations***. IEEE Transactions on Software Engineering 41(5):490-506 (2015).

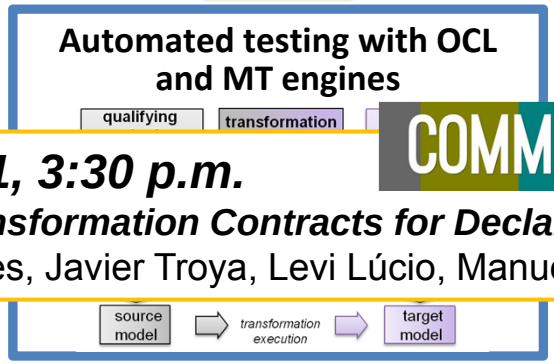
- **Transformation Testing**





	R1	R2	R3	R4	R5	R6	R7	R8
C1	(×)	×						
C2	(×)	×	×					
C3	×		×	×	(×)	(×)	(×)	(×)
C4	×	(×)	(×)	(×)	(×)	(×)	(×)	(×)
C5	(×)	×						
C6	(×)	×	×					
C7	(×)		×	×	(×)	(×)	(×)	(×)
C8	(×)		×	(×)	×			
C9	(×)		×	(×)		(×)	×	
C10	(×)		×	(×)		(×)		×





Thursday Oct 1, 3:30 p.m.

COMMERCIAL BREAK

Fully Verifying Transformation Contracts for Declarative ATL Foundations

Bentley James Oakes, Javier Troya, Levi Lúcio, Manuel Wimmer

AMT

How?

M. Wimmer, S. Martínez Perez, F. Jouault, J. Cabot:
A Catalogue of Refactorings for Model-to-Model Transformations.
Journal of Object Technology 11(2):1-40 (2012).

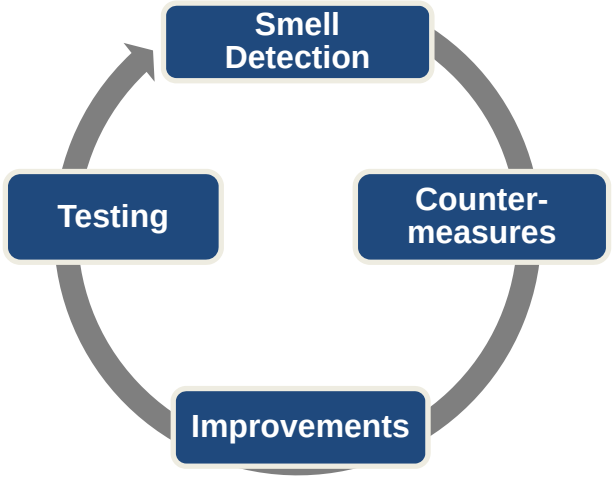
- **Transformation Refactoring**

Transformation Metrics

Metric	*T1*	*T4*	*T6*
LoC	39	44	38
#Elements	122	132	116
#Rules	1	6	6
#ARules	0	2	2
#CRules	1	4	4
#Helpers	1	2	2
#Bindings	10	5	5
#CD	3	0	0
#GD	1	0	0
MEL	13	6	6
Avg Fan-in	0	0,83	0,83
Avg Fan-out	18	6,17	4,8
-internal	1	1	1
-external	17	5,17	3,8
Avg Val-in	1	1,83	1
Avg Val-out	10	1	1
Abstraction	0	0,33	0,33

Including Performance Impacts

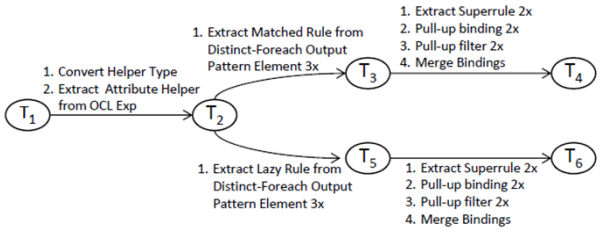
Transformations	Instructions	CPU Time	Speedup
T1	17.125.461	18,07 s	1,00
T2	13.380.985	15,14 s	1,19
T3	256.444.024	186,31 s	0,10
T4	97.064.018	53,20 s	0,34
T5	12.237.967	9,07 s	1,99
T6	12.888.967	10,65 s	1,70



Transformation Refactoring Catalogue

	QVT-R	QVT-O	ETL
Renaming			
Rename In/Out Pattern Element	✓	✓	✓
Rename In/Out Model	✓	✓	✓
Rename Rule/Helper	✓	✓	✓
Restructuring			
Extract Helper/Rule	✓	(✓)	(✓)
Inline Helper/Rule	✓	(✓)	(✓)
Merge Rule	✓	✓	✓
Split Rule	✓	✓	✓
Merge Binding	✓	✓	✓
Split Binding	✓	✓	✓
Convert Rule Type	✓	×	✓
Inheritance-related			
Extract Superrule	×	(✓)	✓
Pull Up Binding	×	✓	✓
Pull Up Filter	×	✓	✓
Eliminate Superrule	×	(✓)	✓
Push Down Binding	×	✓	✓
Push Down Filter	×	✓	✓
OCL related			
Convert Helper Type	×	×	✓
Protect Unsafe Target Navigation	✓	✓	✓
Substituting IF/ELSE Chains with Map	×	✓	✓
Improve Opposite Reference Computations	×	✓	✓
Shorten Navigation By Context Switch	✓	✓	✓
Replace select/first with any	✓	✓	✓
Replace allInstances with Navigation	✓	✓	✓
Introduce Short-Circuits	✓	✓	✓

Refactoring Space Exploration



AMT

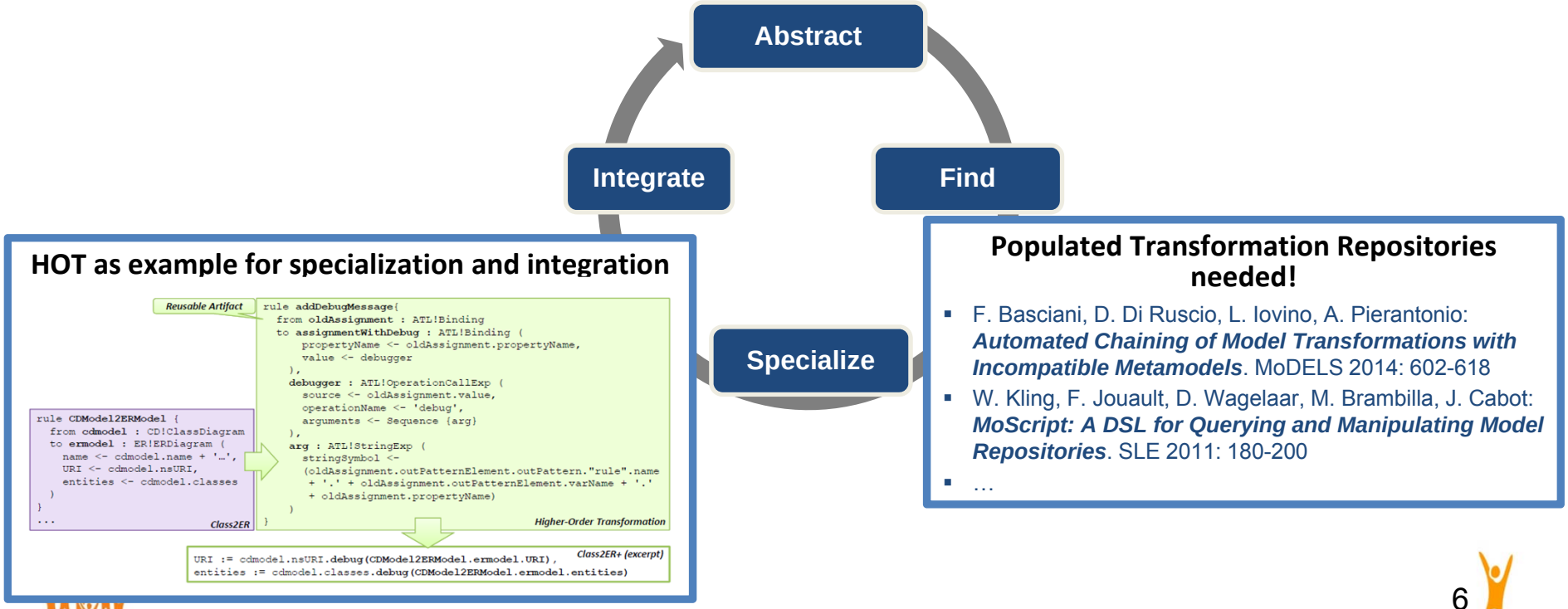
How?

A. Kusel, J. Schönböck, M. Wimmer, G. Kappel, W. Retschitzegger, W.Schwinger:
Reuse in model-to-model transformation languages: are we there yet?
Software & Systems Modeling 14(2):537-572 (2015).

- **Transformation Reuse**

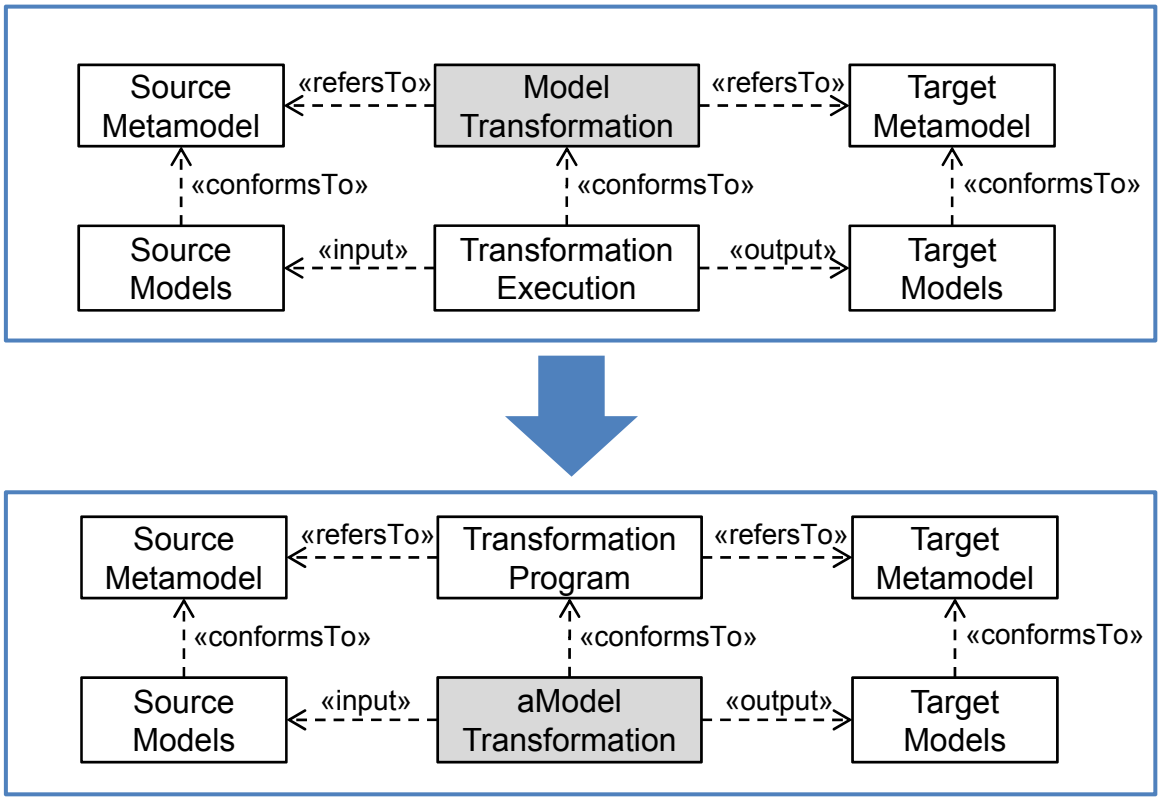
Reuse Mechanisms for Model Transformations

Scope	Specificity	Granularity	Classified Reuse Mechanisms
inter/intra	*concrete/generic*	*small/large*	
intra	concrete	small	Code Scavenging, User-defined Functions, Rule Inheritance
inter	concrete	small	
intra	concrete	large	Module Import, Transformation Product Lines
inter	generic	small	HOT, AOP, Reflection, Generic Functions, Embedded DSLs
intra	generic	small	
inter	generic	large	Generic Transformations, Stand-alone DSLs
intra	generic	large	
inter	concrete	large	Orchestration



AMT
When?

- **Model Transformation Pattern Revisited**

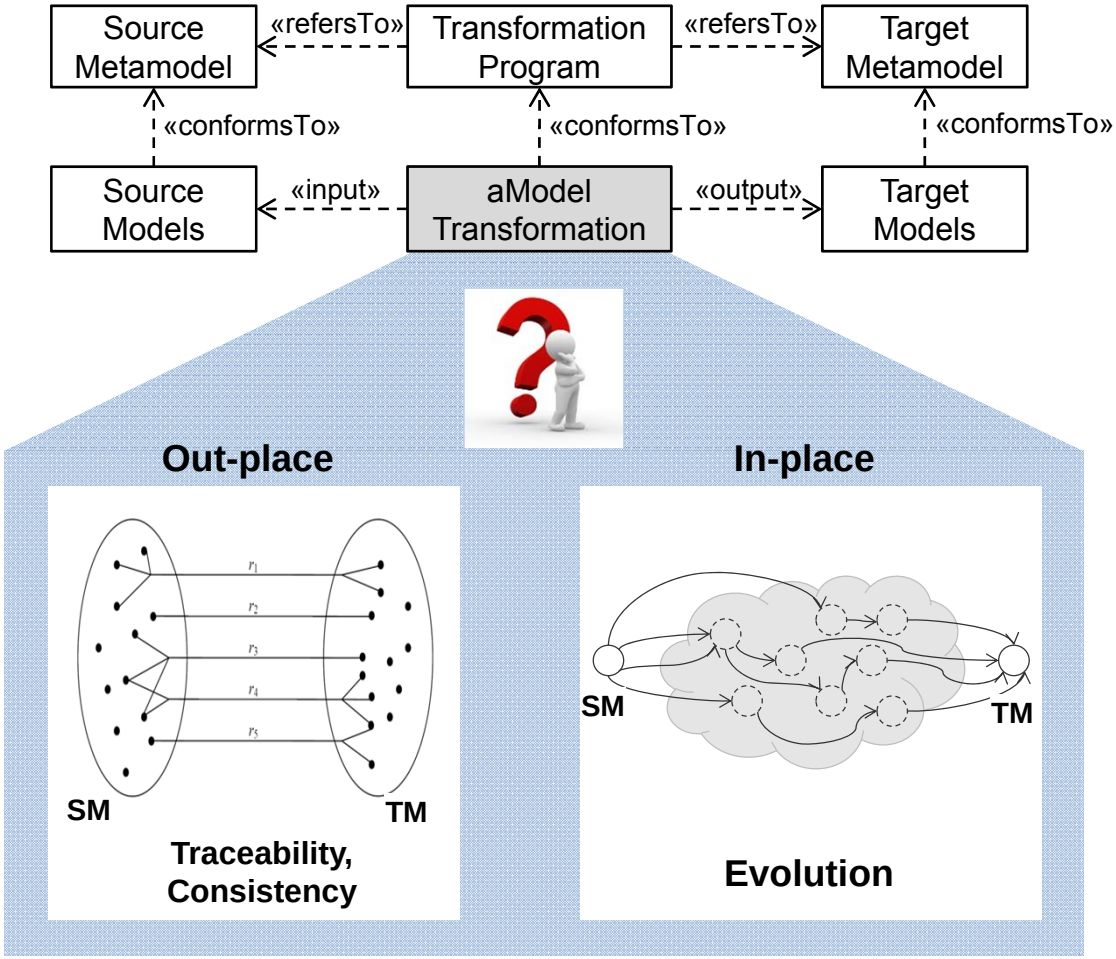


TM = MT(SM)

aMT = F(SM,TM,TP)

A²MT: Analysis of A Model Transformation

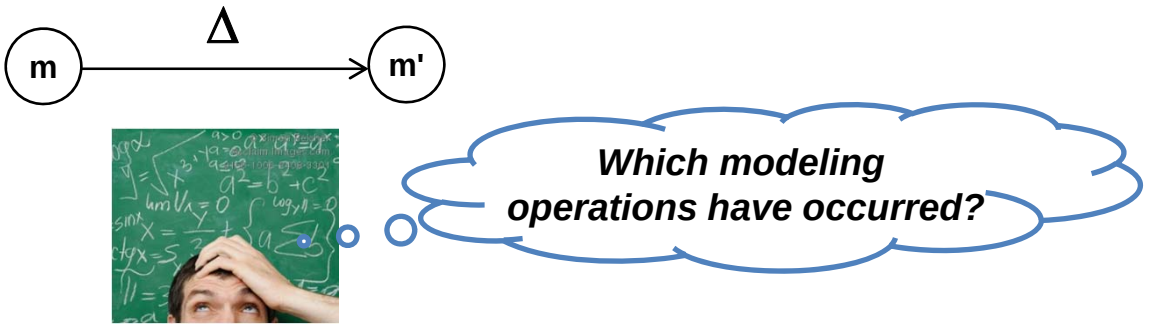
When?



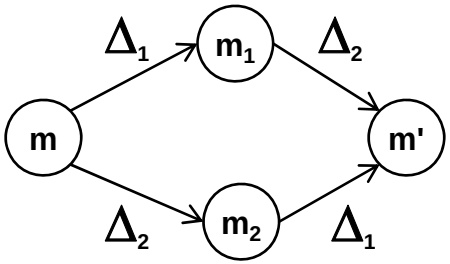
Model Evolution
as
Transformation Reconstruction

Model Evolution as Transformation Reconstruction

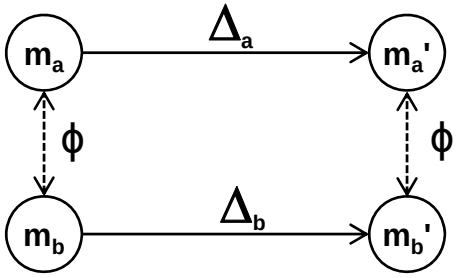
- Basic model evolution problem…



- … occurs in parallel evolution scenarios



Multi-user



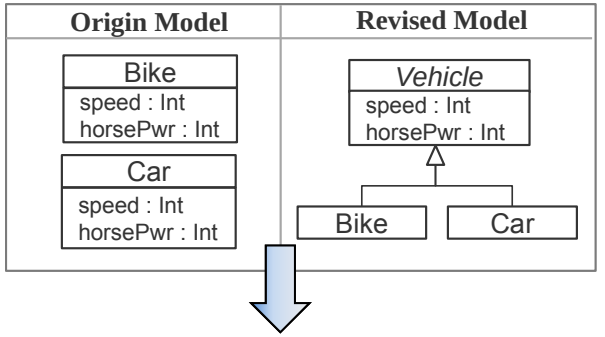
Multi-viewpoint

Operation Detection

Baseline: Atomic Operations

G. Taentzer, C. Ermel, P. Langer, M. Wimmer. ***A fundamental approach to model versioning based on graph modifications: from theory to implementation***. Software & Systems Modeling 13(1):239-272 (2014).

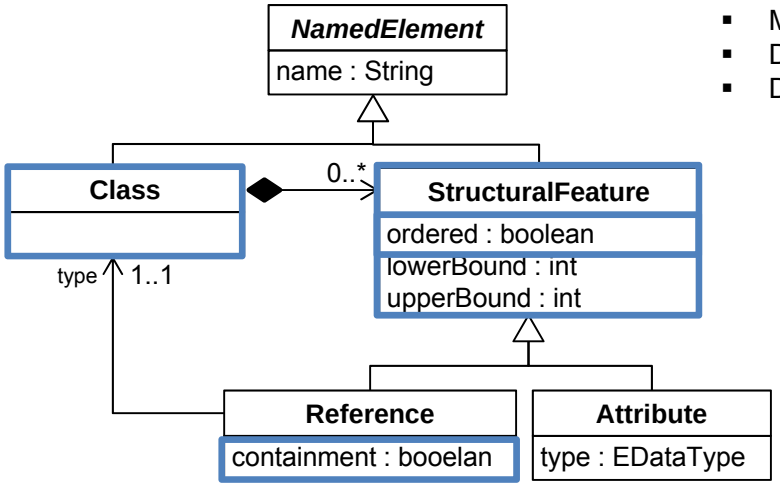
- **Model differencing**
  - **Comparison** of **states** of an artifact
  - **Match function** to find **correspondence** of two elements in compared artifacts
  - **Differences** are converted into **atomic operations**



Atomic changes

- Add Class
- Update Class.superClass
- Update Class.superClass
- Move Attribute
- Move Attribute
- Delete Attribute
- Delete Attribute

- Atomic operation types
  - Add model element
  - Delete model element
  - Feature update
    - Insert feature value
    - Delete feature value
    - Feature order change
  - Move

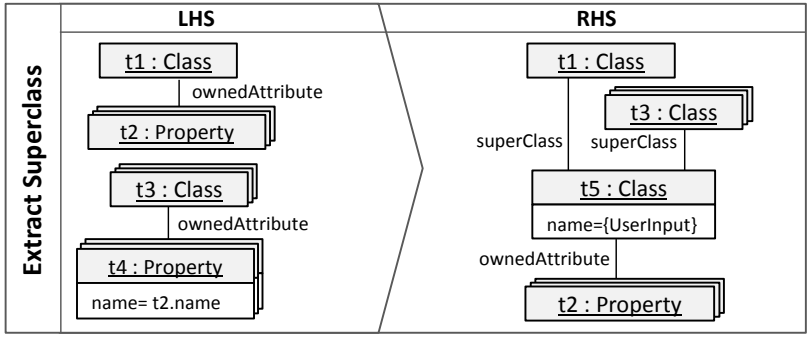


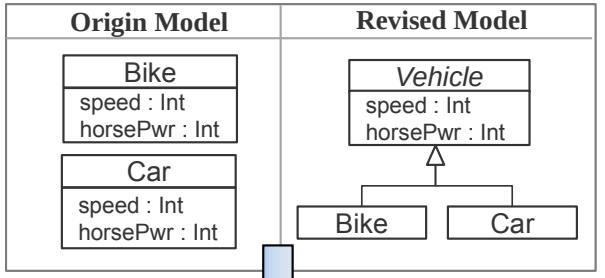
Operation Detection

Extension: Composite Operations

P. Brosch, P. Langer, M. Seidl, K. Wieland, M. Wimmer, G. Kappel, W. Retschitzegger, W. Schwinger: ***An Example Is Worth a Thousand Words: Composite Operation Modeling By-Example***. MoDELS 2009, pp. 271–285.

- **Better understanding the evolution of a model**
  - Going beyond **atomic operations**
  - **Composite operations** (e.g., refactorings)
    - More concise description of evolution
    - Reason about the intent(s) of an evolution
- Composite operations are **model transformations**
  - Preconditions
  - Sequence of atomic operations
  - Postconditions





Atomic changes

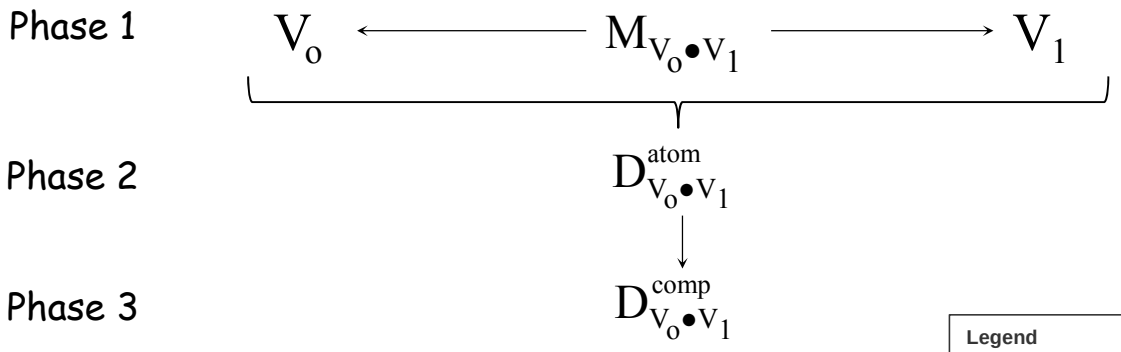
- Add Class
- Update Class.superClass
- Update Class.superClass
- Move Attribute
- Move Attribute
- Delete Attribute
- Delete Attribute

Extract SuperClass

Operation Detection by Model Differencing

P. Langer, M. Wimmer, P. Brosch, M. Herrmannsdörfer, M. Seidl, K. Wieland, G. Kappel: ***A posteriori operation detection in evolving software models***. Journal of Systems & Software 86(2):551–566 (2013).

- **Two**-phase process
  - **Phase 1**: Matching for finding correspondences between objects
  - **Phase 2**: Fine-grained comparison of corresponding objects for finding atomic differences
  - **Phase 3**: Aggregation of atomic differences for detecting composite operation applications



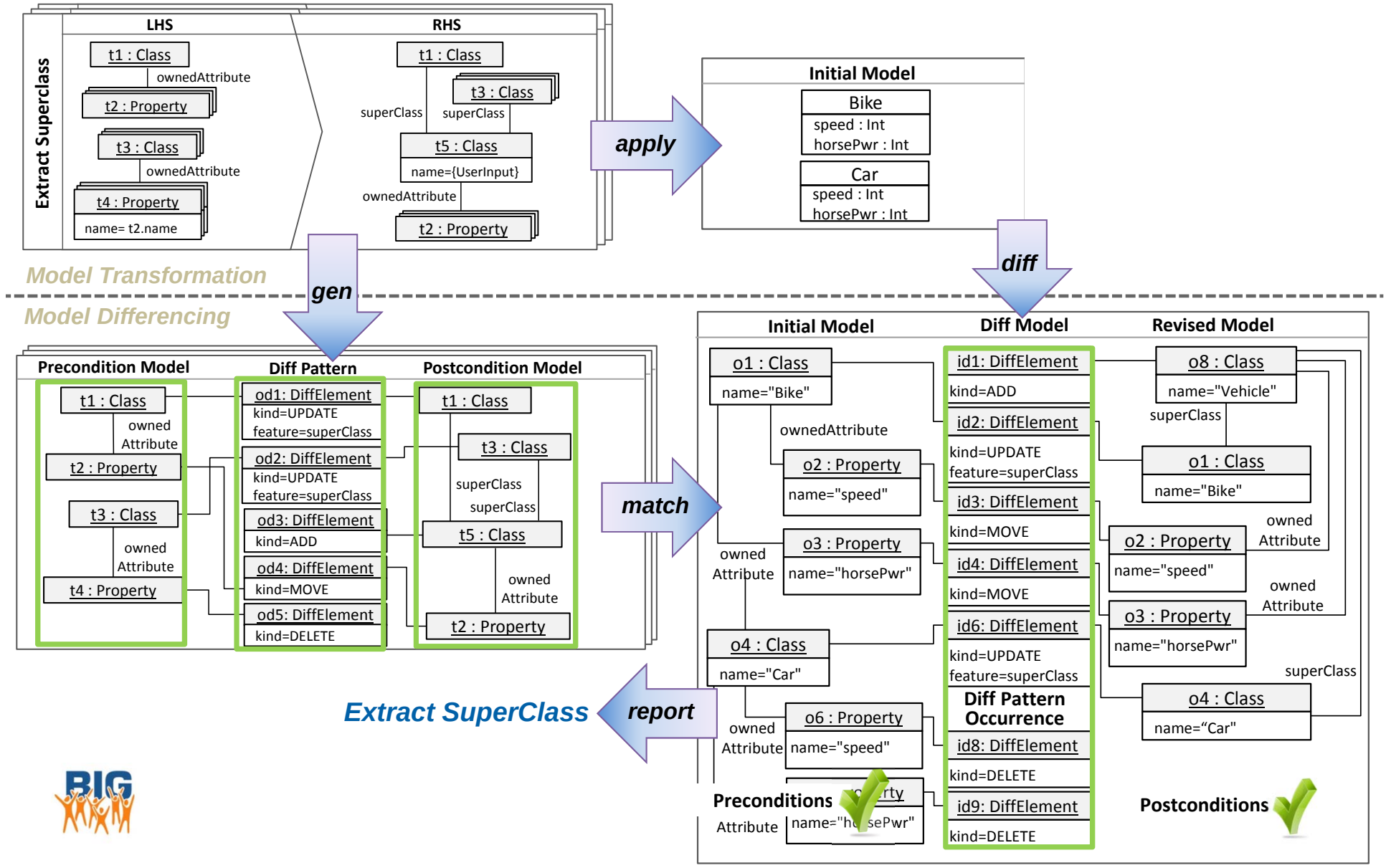
Legend

$M_{V_o \bullet V_1}$ … Match Model

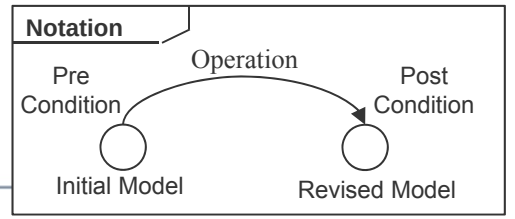
$D^{atom}_{V_o \bullet V_1}$ … Diff Model [atomic]

$D^{comp}_{V_o \bullet V_1}$ … Diff Model [composite]

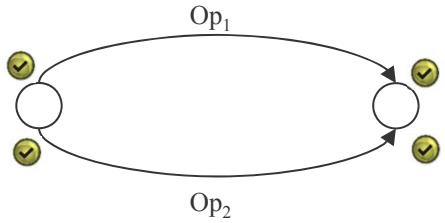
Bridging Model Transformation and Model Differencing



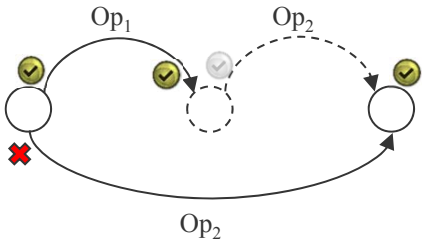
Operation Detection Cases (1/2)



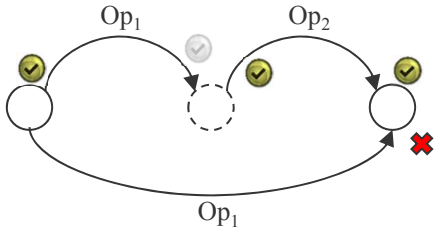
- Independent Operations
  - Op_1
  - Op_2



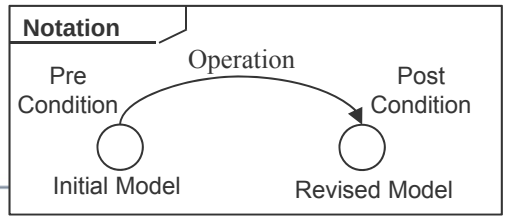
- Overlapping Operations: Type I
  - Op_1
  - ✖ Preconditions of Op_2
  - → Iterative Forward Detection Approach



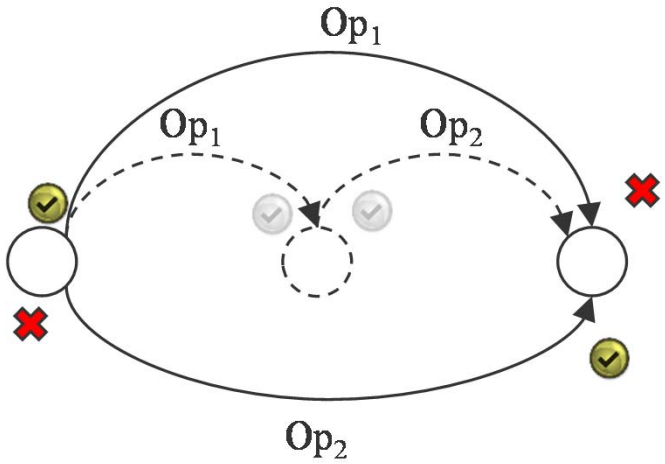
- Overlapping Operations: Type II
  - Op_2
  - ✖ Postconditions of Op_1
  - → Iterative Backward Detection Approach



Operation Detection Cases (2/2)



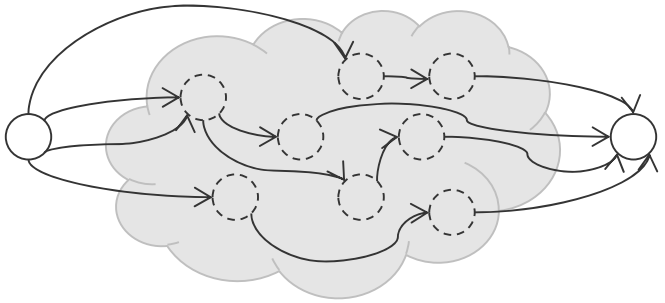
- Overlapping Operation: Type III (Type I + Type II)
  - Neither Op_1 nor Op_2 entirely detectable
  - Preconditions of Op_1 valid
  - Postconditions of Op_2 valid



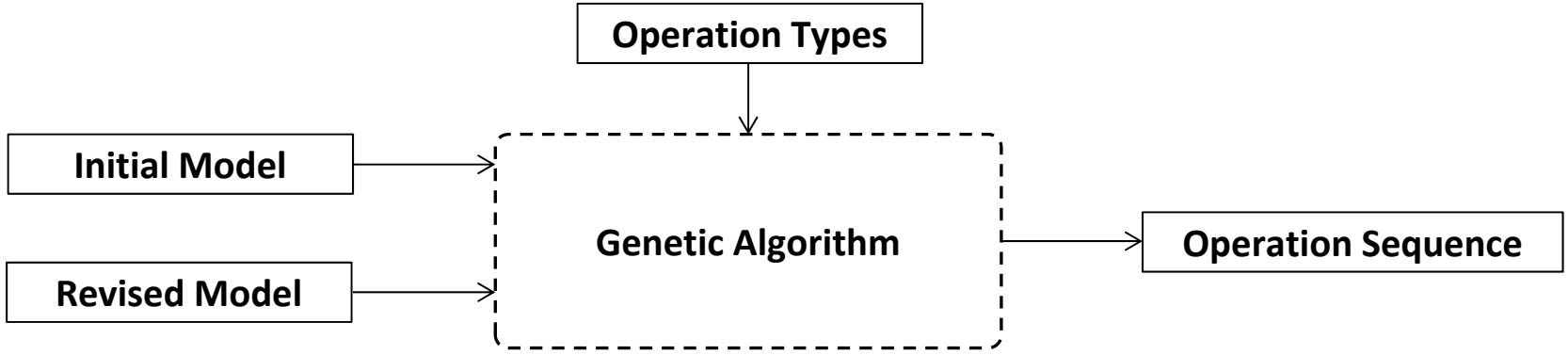
Search-based Operation Detection

A. ben Fadhel, M. Kessentini, P. Langer, M. Wimmer: ***Search-based Detection of High-level Model Changes***. ICSM 2012, pp. 212-221.

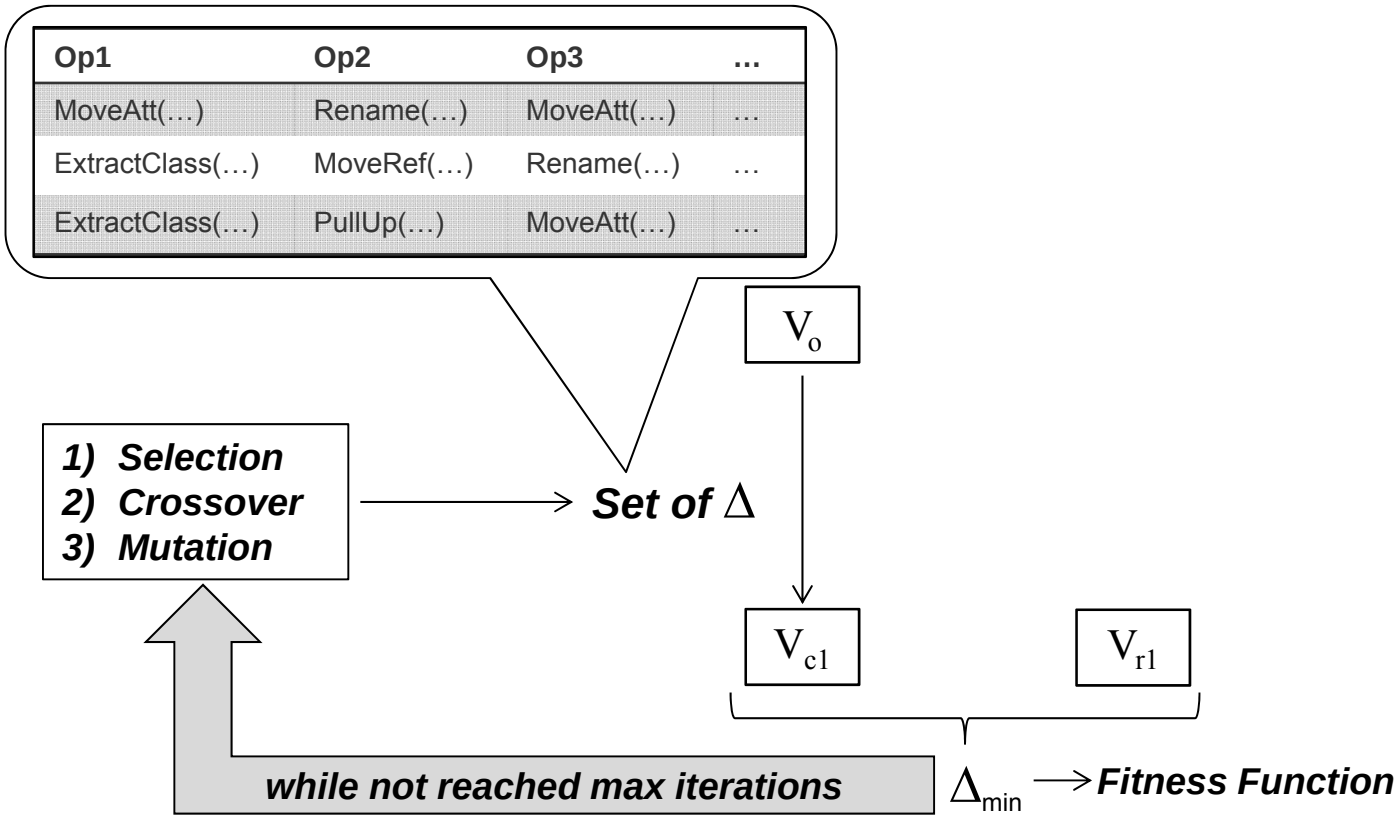
- **Huge Search Space**
  - Multiple combinations
  - Order matters
  - → Meta-heuristic search-based approach



- **Genetic Algorithm** for Operation Detection



Search-based Operation Detection



M. Herrmannsdörfer, D. Ratiu, G. Wachsmuth: ***Language Evolution in Practice: The History of GMF***. SLE 2009, pp. 3-22.

Operation Detection

Evaluation: GMF Case Study

- **Evolution** of the Graphical Modeling Framework (GMF)
  - http://www.eclipse.org/modeling/gmp/
  - Evolution of three Ecore-based metamodels
  - Object-oriented refactoring catalogue
  - Independent and overlapping operation occurrences

Model	# Ops	# Elements	# Values	# Links
GMF Map	8	367, 428	620, 735	683, 784
GMF Graph	24	277, 310	521, 588	629, 695
GMF Gen	93	883, 1295	1385, 2188	1935, 2899

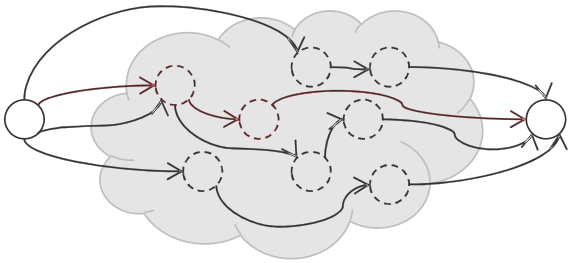
- **Results**

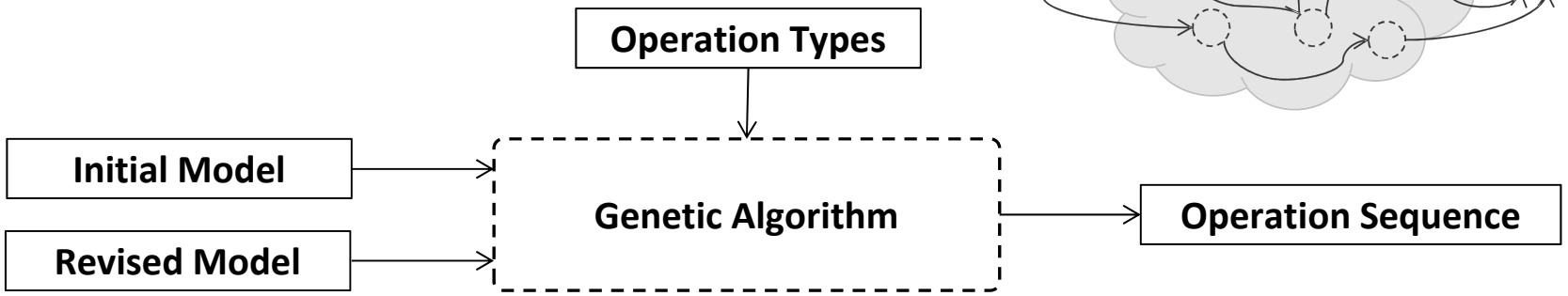
	Matching Approach		Genetic Algorithm	
Model	*Precision*	*Recall*	*Precision*	*Recall*
GMF Map	100%	100%	100%	85%
GMF Graph	100%	50%	95%	87%
GMF Gen	100%	73%	94%	97%

Model Transformation as Optimization Problem

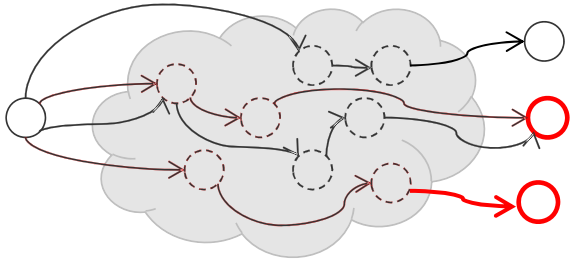
Model Transformation as Optimization Problem

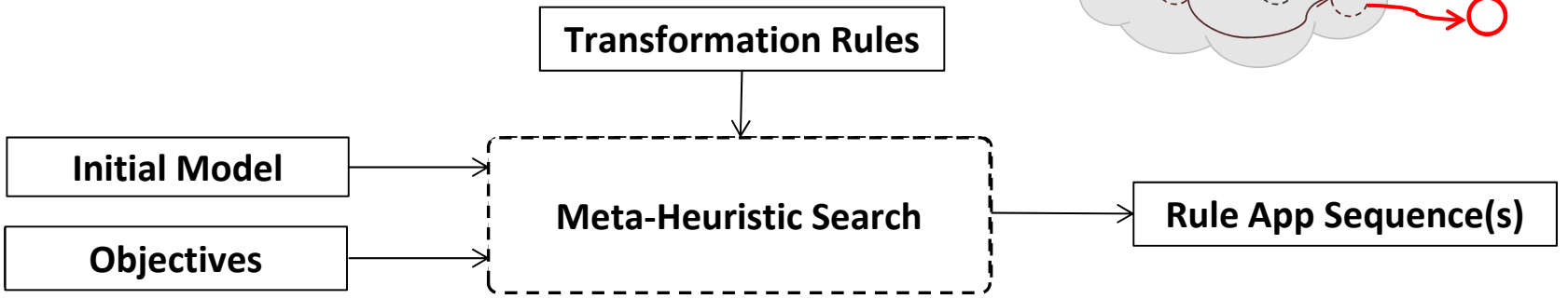
- From *Search-based Operation Detection…*





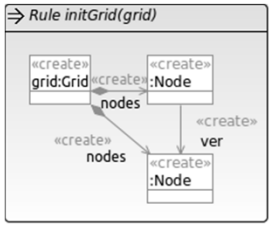
- …to *Search-based Model Transformations*

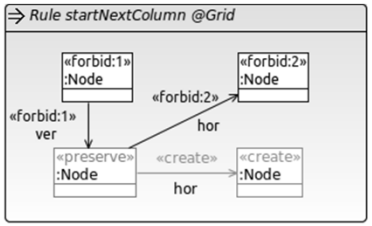


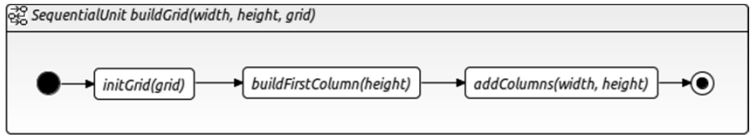


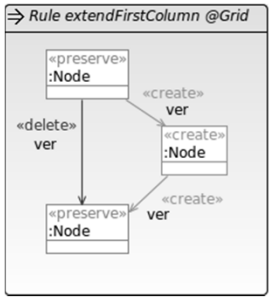
M. Kessentini, P. Langer, M. Wimmer: ***Searching Models, Modeling Search: On the Synergies of SBSE and MDE***; CMSBSE@ICSE 2013, pp. 51-54.

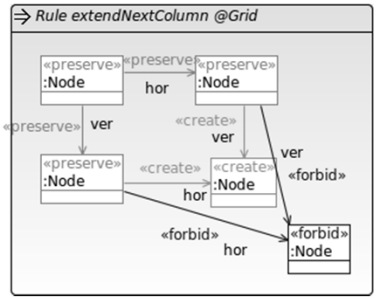
Aim 1: Explicating Transformation Goals

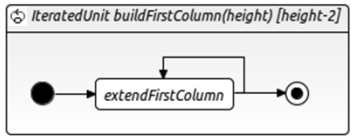


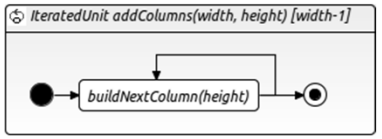


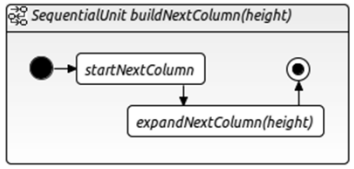


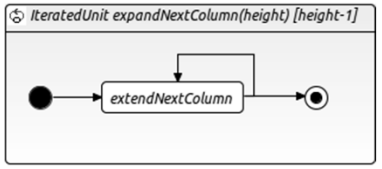


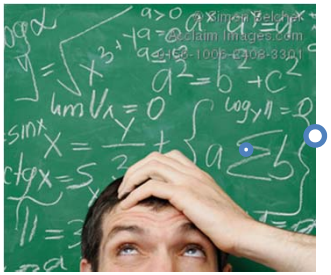






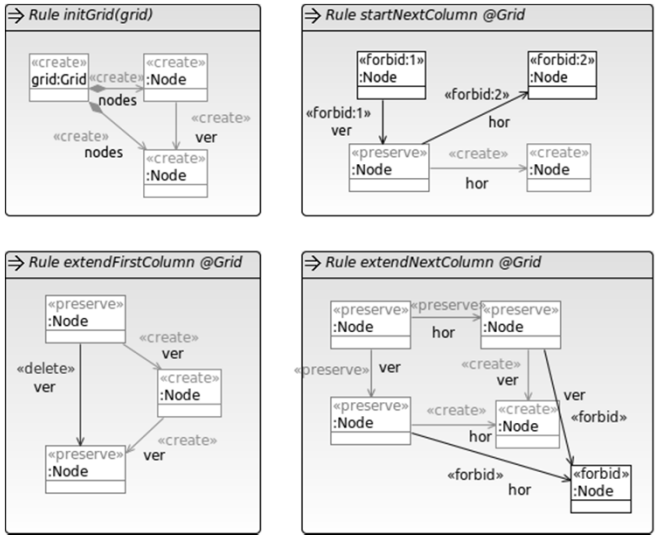


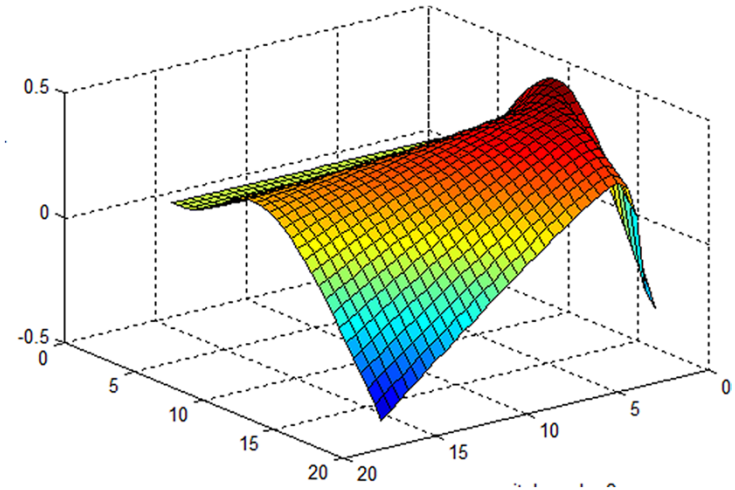


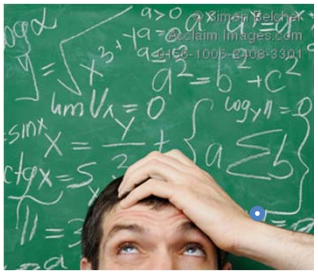


What are the goals of such transformations?

Aim 2: Guided Exploration of Transformation Space



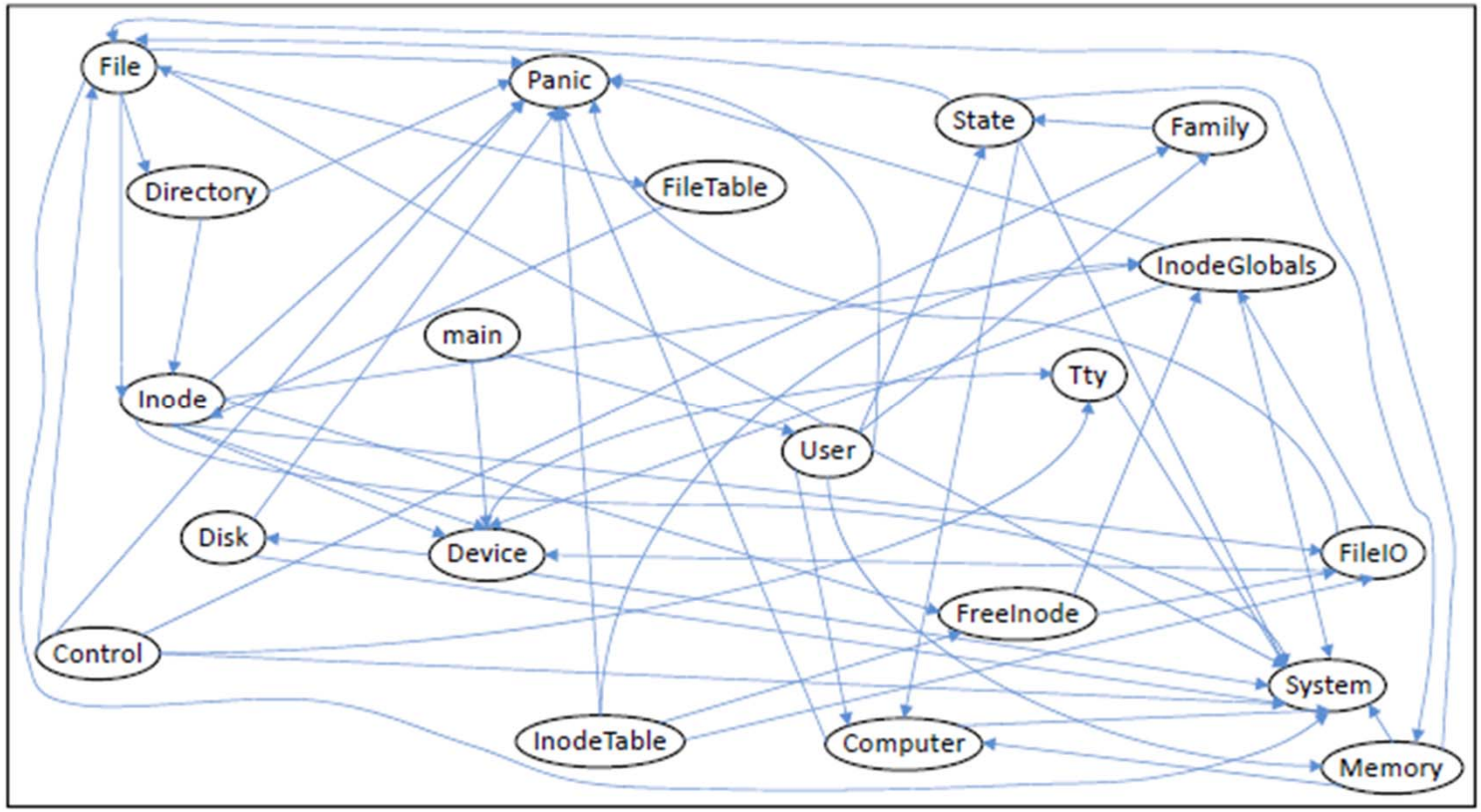




How can I find potential solutions without exhaustively exploring everything?

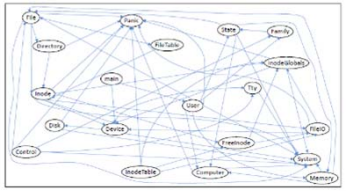
Motivating Example: Modularization Case Study

- mtunis (operating system for educational purposes)

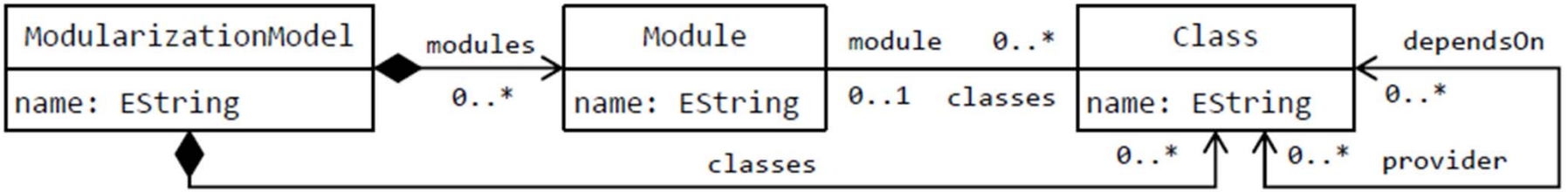


K. Praditwong, M. Harman, X. Yao, ***Software Module Clustering as a Multi-Objective Search Problem***. IEEE Transactions on Software Engineering 37, 2011, pp. 264–282.

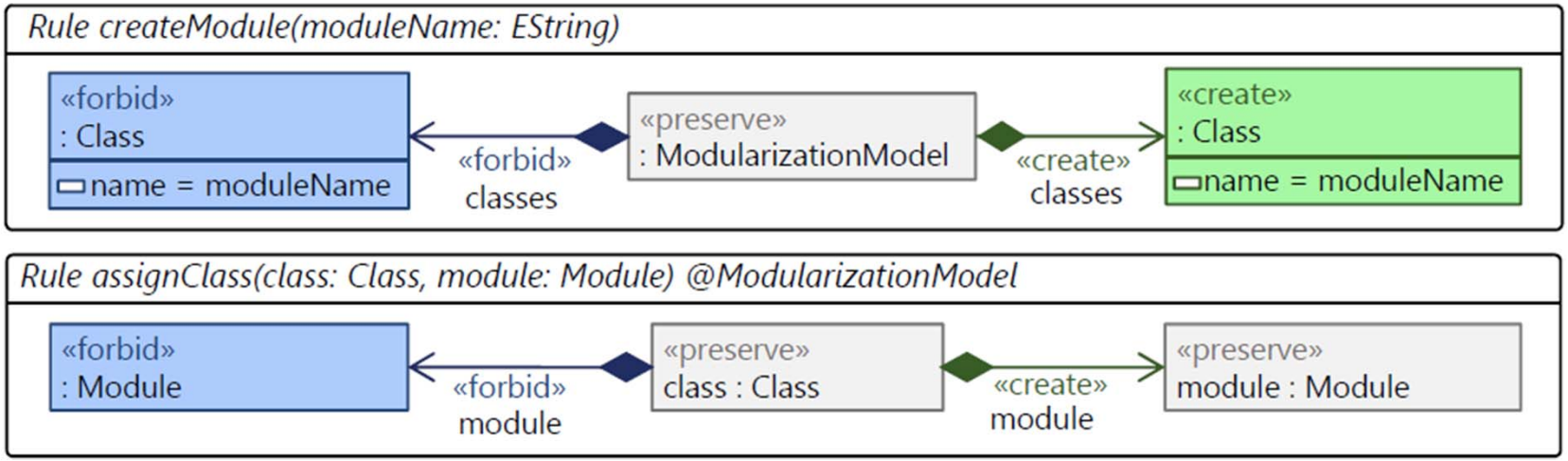
Modularization Case Study



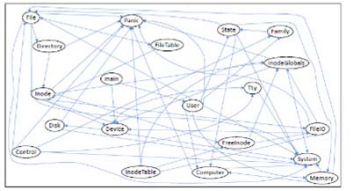
- Represent problem domain through metamodel



- Represent modularization through graph transformation rules



Modularization Case Study



- Goal: find a good OO module design

- Objectives: follow an Equal-Size Cluster Approach (ECA)
  - Coupling : min
  - Cohesion : max
  - Modularization quality (MQ) : max
  - Number of modules : max
  - Difference between max and min number of classes in a module : min

- Complexity corresponds to Bell number
  - $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$
  - mtunis example: 20 classes → 51724158235372 possible solutions

Modularization Case Study

- How can we achieve the objectives?
  - It is not trivial to find a good rule order
  - Rule parameters might have an infinite value range
  - Producing each possible output model is expensive
  - Evaluating each possible output model is expensive as well

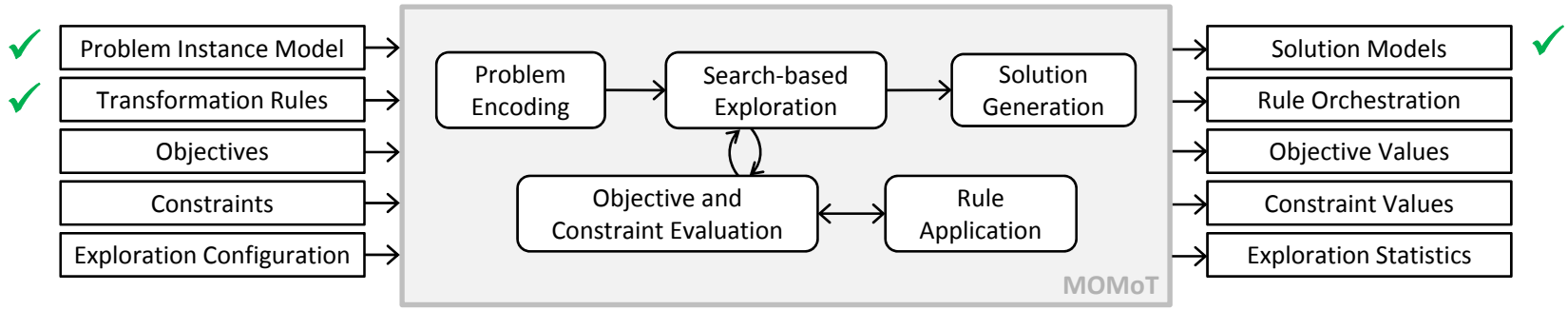
- Is there any way of automating the search for good rule orchestrations?

→ Search-based software engineering (SBSE) [1] to the rescue!
  - Search-based Optimization using Meta-heuristics
  - Many software engineering problems have been solved in this way

- But how may a ***generic***, ***transparent***, ***declarative***, and ***guided*** approach look like?

M. Harman, ***The Current State and Future of Search Based Software Engineering***, FOSE@ICSE 2007, pp. 342–357.

MOMoT – Marrying Search-based Optimization and Model Transformation Technology



- **Bridging two worlds**
  - MDE to model problem domains and create problem instances and model transformations to manipulate problem instances
  - SBSE techniques to efficiently handle potentially infinite search spaces

→ Focus on **reusing** existing technologies and **loose** coupling!

- Henshin Graph Transformation Engine (http://eclipse.org/henshin)
- Extended MOEA Framework (http://moeaframework.org)

M. Fleck, J. Troya, M. Wimmer, ***Marrying Search-based Optimization and Model Transformation Technology***. NasBASE (2015), 1-16.

MOMoT – Objectives

- Objectives specify goals of a transformations
  - Domain-specific objectives are related to the problem domain
    - Example: Minimize coupling between modules
  - Solution-specific objectives are related to the solution representation
    - Example: Minimize the number of rule applications of a solution
  - Provide objectives in SCL based on OCL and Java

```
objectives = { // auto-inject the solution and the root of the model
  Coupling : minimize { // Java-like syntax, access attribute storage
    solution.getAttribute("calculation", typeof(Calculator)).metrics.coupling
  }
  Cohesion : maximize {
    solution.getAttribute("calculation", typeof(Calculator)).metrics.cohesion
  }
  MQ : maximize {
    solution.getAttribute("calculation", typeof(Calculator)).metrics.mq
  }
  MinMaxDiff : minimize {
    solution.getAttribute("calculation", typeof(Calculator)).metrics.mmDiff
  }
  NrModules : maximize "self.modules->size()" // OCL-specification
  Length : minimize new TransformationLengthDimension // generic objective
}
```

MOMoT – Constraints

- Constraints define the validity of solutions
  - Domain-specific constraints are related to the problem domain
    - Example: There should not be empty modules
  - Solution-specific constraints are related to the solution representation
    - Example: The number of rule applications of a solution must be less than 20
  - Provide constraints in SCL based on OCL, Java, and Graph Patterns

```
constraints = { // mark invalid solutions
  UnassignedFeatures : minimize { // Java-like syntax, direct calculation
    (root as ModularizationModel).features.filter[c | c.module == null].size
  }
  EmptyModules : minimize {
    (root as ModularizationModel).features.filter[m | m.classes.empty].size
  }
}
```

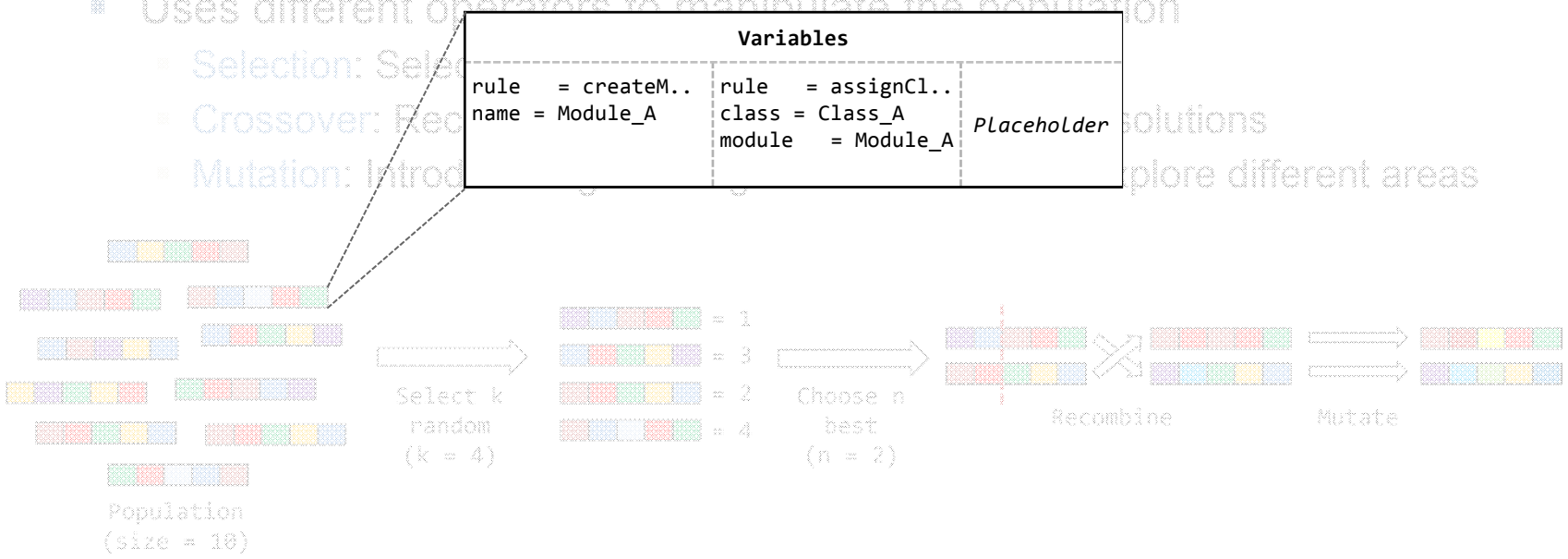
MOMoT – Exploration Configuration

- Exploration Configuration
  - MOMoT is task- and search algorithm-agnostic
  - Reuse existing search algorithms and their possible instantiations
    - Population-based search methods (provided by MOEA)
    - Local Search methods (extensions to MOEA)
  - Provide concrete instantiations of algorithms in SCL

```
algorithms = { // moea, local, and configuration are auto-injected objects
  NSGAIII : moea.createNSGAIII(
    0, 6, // inner and outer divisions
    new TournamentSelection(2), // k == 2
    new OnePointCrossover(1.0), // 1.0 == 100%, always do crossover
    new TransformationPlaceholderMutation(0.15), // 15% mutation
    new TransformationVariableMutation(orchestration.searchHelper, 0.10))
  HillClimbing : local.createHillClimbing( // at most 100 neighbors
    new IncreasingNeighborhoodFunction(configuration.searchHelper, 100),
    new ObjectiveFitnessComparator("MQ"))
  RandomSearch : new RandomSearch( // without auto-injected objects
    configuration.problem,  // problem instance
    configuration.createPopulationGenerator(300), // random solutions
    new NondominatedPopulation) // population class to use
}
```

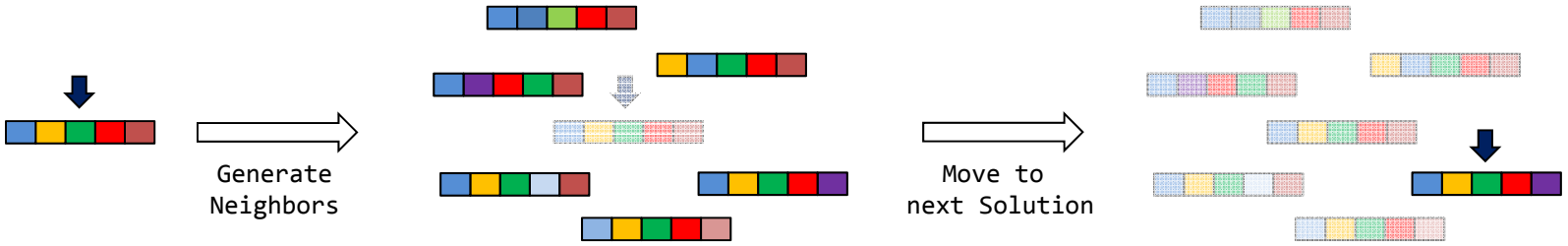
MOMoT – Exploration

- Population-based Search methods
  - Maintains a set of candidate solutions at once (population)
  - Uses different operators to manipulate the population
    - Selection: Sele
    - Crossover: Rec olutions
    - Mutation: Introd plore different areas
  - Examples: ε-MOEA, NSGA-II, NSGA-III, etc.



MOMoT – Exploration

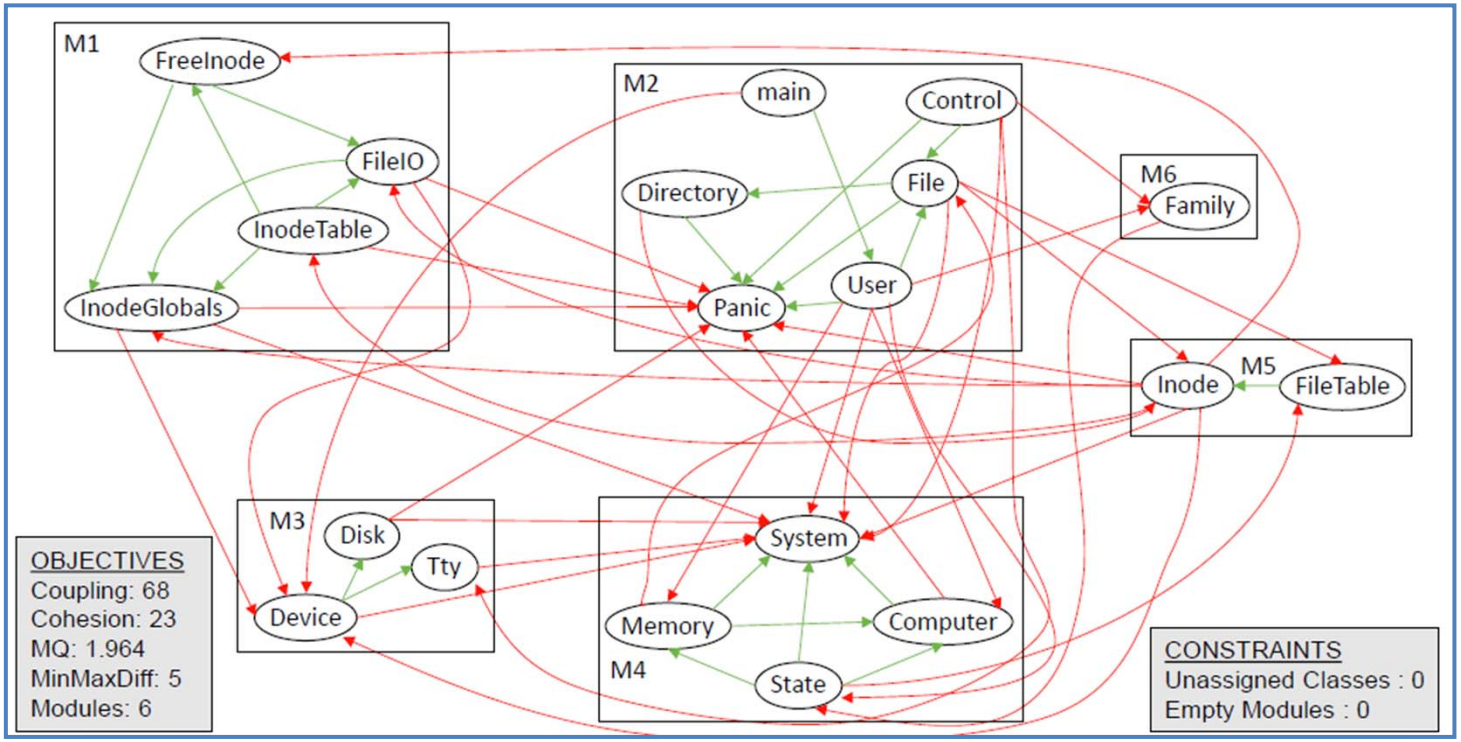
- Local Search methods
  - Maintains one solution at a time
  - Generate neighbor solutions using a neighborhood function
    - Neighbors only differ slightly from the original solution
  - Move to next solution depending on the found neighbors and their fitness



  - Examples: Random Descent, Hill Climbing, Simulated Annealing, etc.

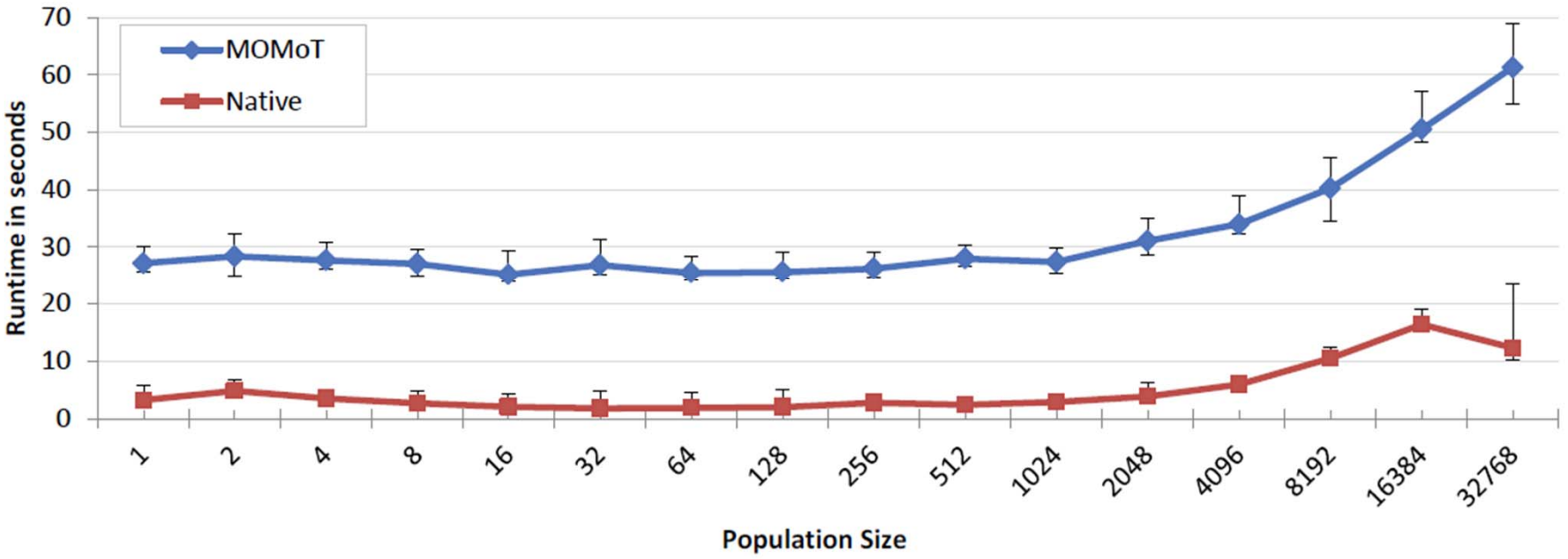
Modularization Case Study: MOMoT Results

- Example solution computed by MOMoT
  - Using NSGA-III
  - Soluton length of 50
  - Population size of 300



Modularization Case Study: MOMoT Results

- Runtime performance
  - Generic MDE encoding vs dedicated, native encoding
  - Average runtime for each encoding
  - MOMoT on average slower by a factor of 3-15 / 10-13



Conclusion
&
Outlook

Conclusion

- New research directions
  - Usage of **search-based techniques** for **transformation analysis**
  - **Analyze** and **guide** individual **transformation executions**

- Several application cases
  - **A-priori**
    - Search-based approaches have been applied for transformation testing
    - Ongoing work: Search-based transformation NFP improvement
  - **On-the-fly**
    - Using objectives to guide the transformation execution
  - **A-posteriori**
    - Reconstruct a transformation
    - Reason about different explanations of possible transformations
    - Generate a transformation program (MTBE)
  - …

Outlook

- **Reproduction studies** for evaluating MOMoT
  - **Model Matching**: Precise and complete match model
  - **Model Diffing**: Short paths explaining the complete evolution
  - **Model Merging**: Apply as many operations as possible while ensuring general constraints

- **Comparative studies** with other emerging search-based transformation approaches
  - Model search algorithms as model transformations [1]
  - Enhance transformation engine with specific search algorithm [2]
  - Compile models to typical search encoding representation [3]

[1] J. Denil, M. Jukss, C. Verbrugge, H. Vangheluwe, ***Search-Based Model Optimization Using Model Transformations.*** SAM 2014, pp. 80–95.

[2] H. Abdeen, D. Varro, H. A. Sahraoui, A. S. Nagy, C. Debreceni, A. Hegedüs, A. Horvath, ***Multi-objective optimization in rule-based design space exploration***, ASE 2014, pp. 289–300.

[3] D. Efstathiou, J. R. Williams, S. Zschaler, ***Crepe complete: Multi-objective optimisation for your models***. CMSEBA@MODELS, 2014.

Thank you!

Comments? Questions? Feedback?

Manuel Wimmer
wimmer@big.tuwien.ac.at


MOMoT on github
http://martin-fleck.github.io/momot

…