Multimodel Management and Repair

I-MDE-A Seminar

Adrian Rutle https://dblp.uni-trier.de/pid/00/5718.html

Ángela Barriga Rodríguez, Alejandro Rodríguez Tena
Patrick Stünkel, Tima Kräuter, Fernando Macías

Madrid – May 16, 2023

Western Norway University of Applied Sciences

Bakcground

Prof. at HVL (Western Norway University of Applied Sciences)

Worked on Model-driven software engineering since 2007

Interests: diagrammatic modeling, model transformation, multilevel modeling, multimodeling, behaviour modeling, softcat and DPF, model repair \subset model management, robotics, “ML/AI”.

Model driven engineering in a nutshell

- The 3 A's: Abstraction, Analysis, Automation
- Features: Mapping, Reduction and Pragmatic
- Ingredients: Models, Metamodels and Model transformations
- Slogans: «everything is a model», «to model or to code, it is not a question»

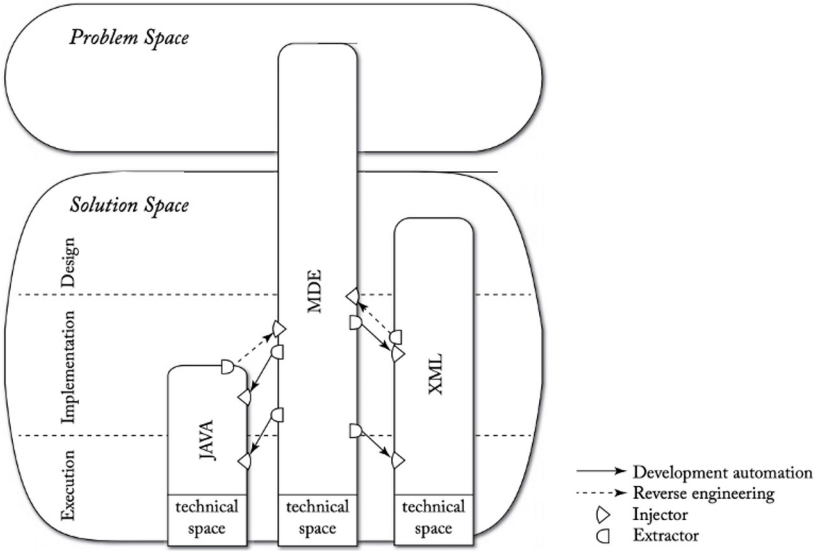
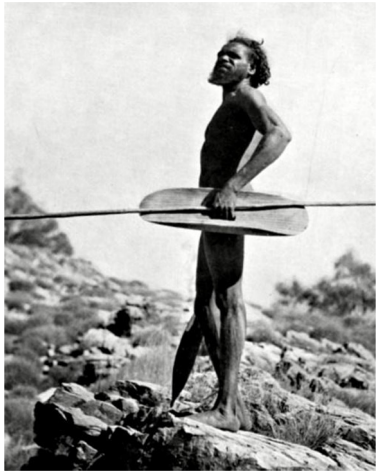
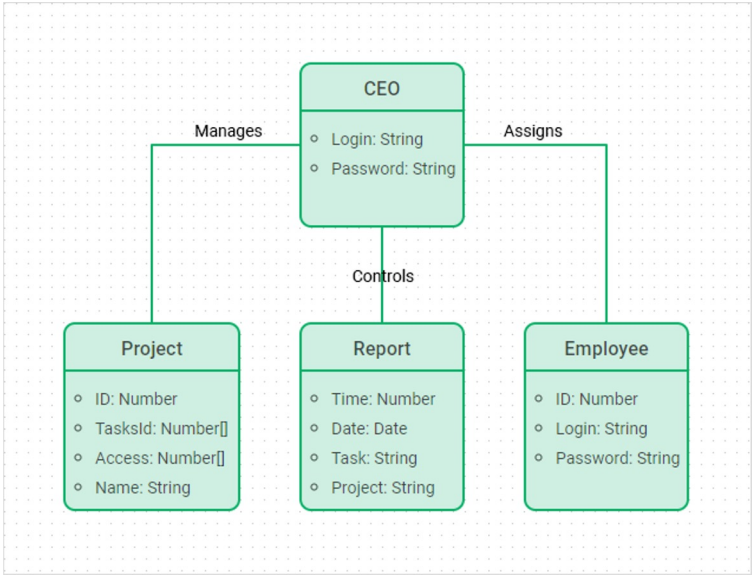


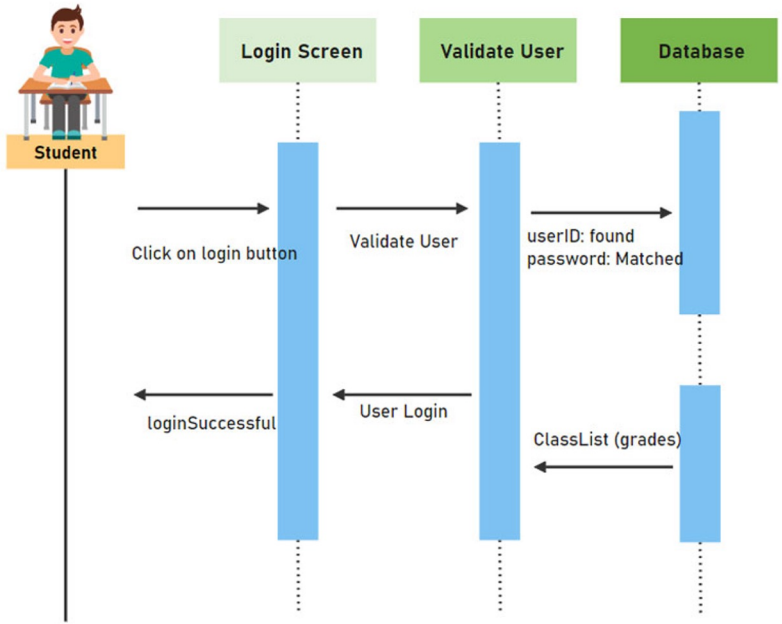
Figure 2.3: Technical spaces examples and coverage.



What is a model?

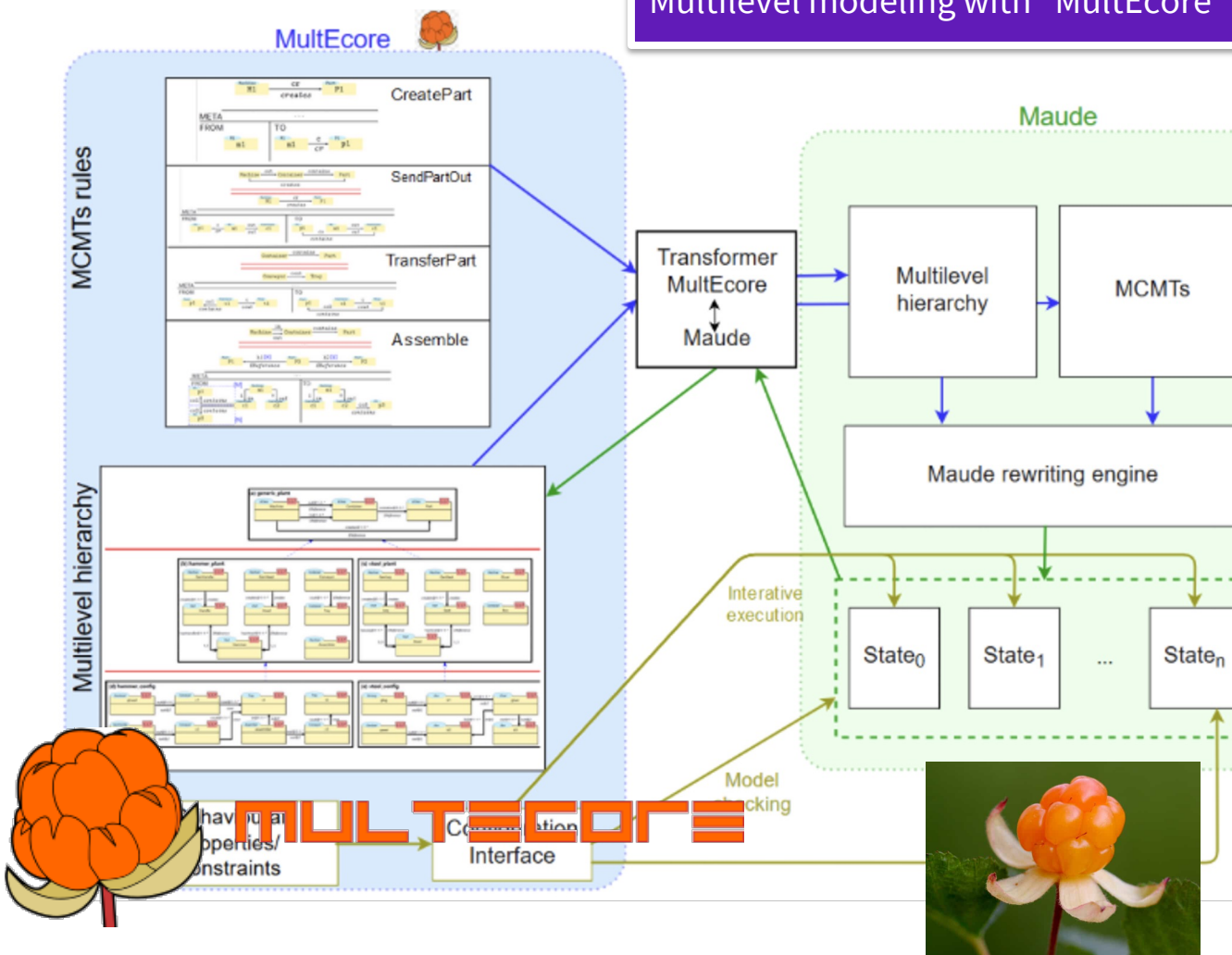
What mathematicians call “specification”


CEO
Login: String
Password: String
Manages
Assigns
Controls
Project
ID: Number
TasksId: Number[]
Access: Number[]
Name: String
Report
Time: Number
Date: Date
Task: String
Project: String
Employee
ID: Number
Login: String
Password: String


Student
Login Screen
Validate User
Database
Click on login button
Validate User
userID: found
password: Matched
loginSuccessful
User Login
ClassList (grades)

Levels of abstraction

Multilevel modeling with “MultEcore”

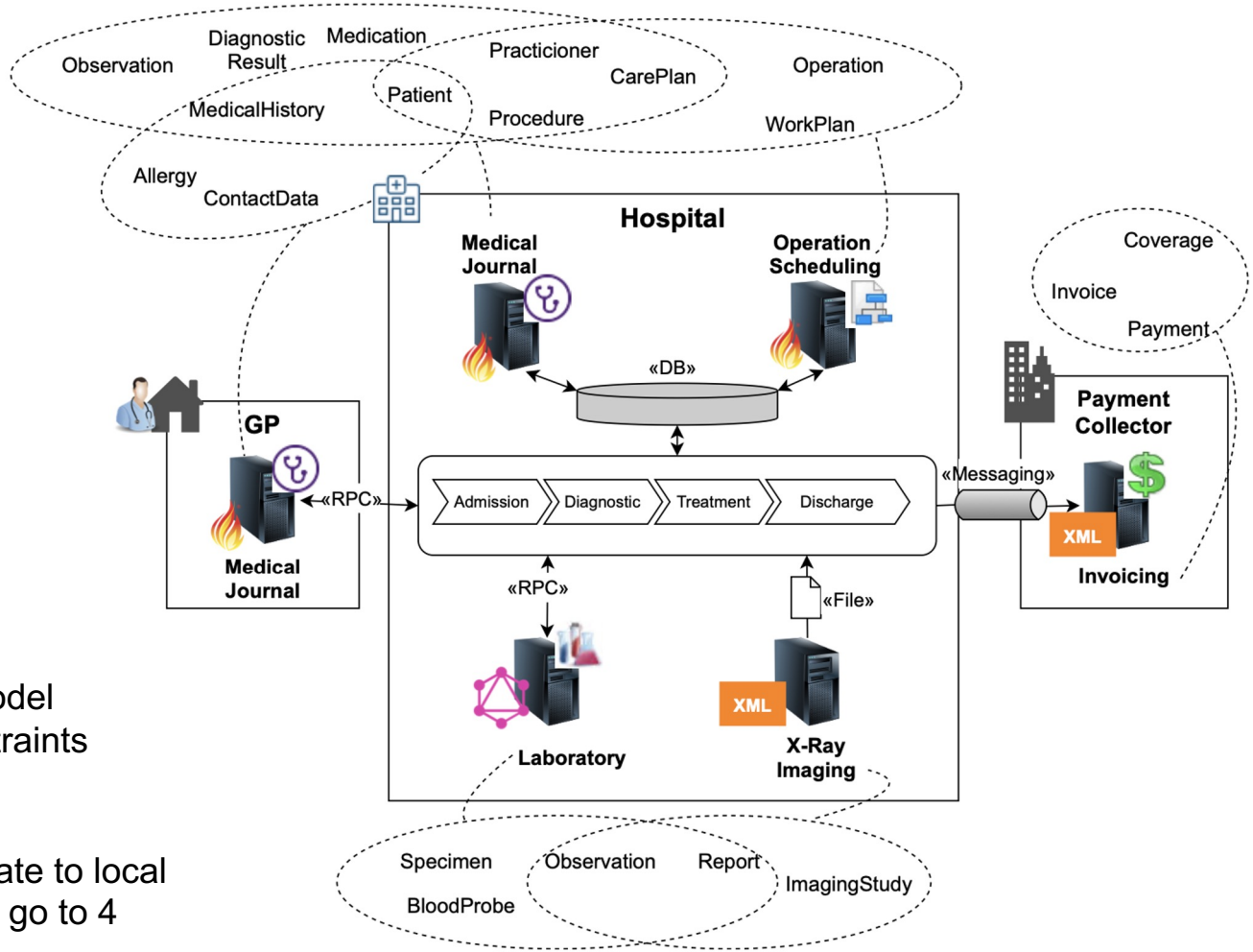


Model Management

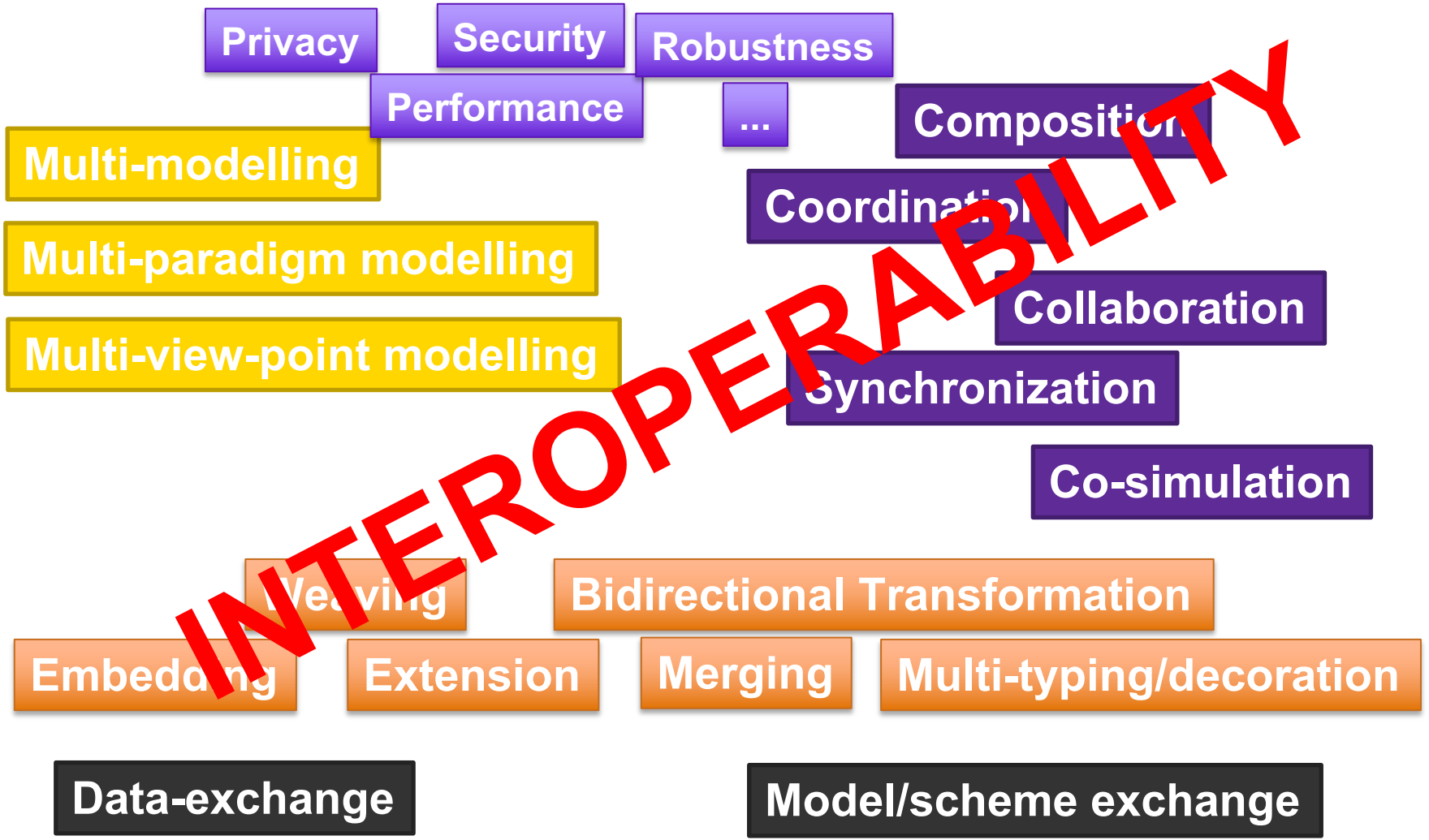
- Version control
- Global management (Multi-models)
- Persistence
- Interchange
- Comparison
- Co-evolution
- Quality
- Collaborative modeling
- Repair/restoration
- Composition

Global management

- Structure
- Behaviour
- Both



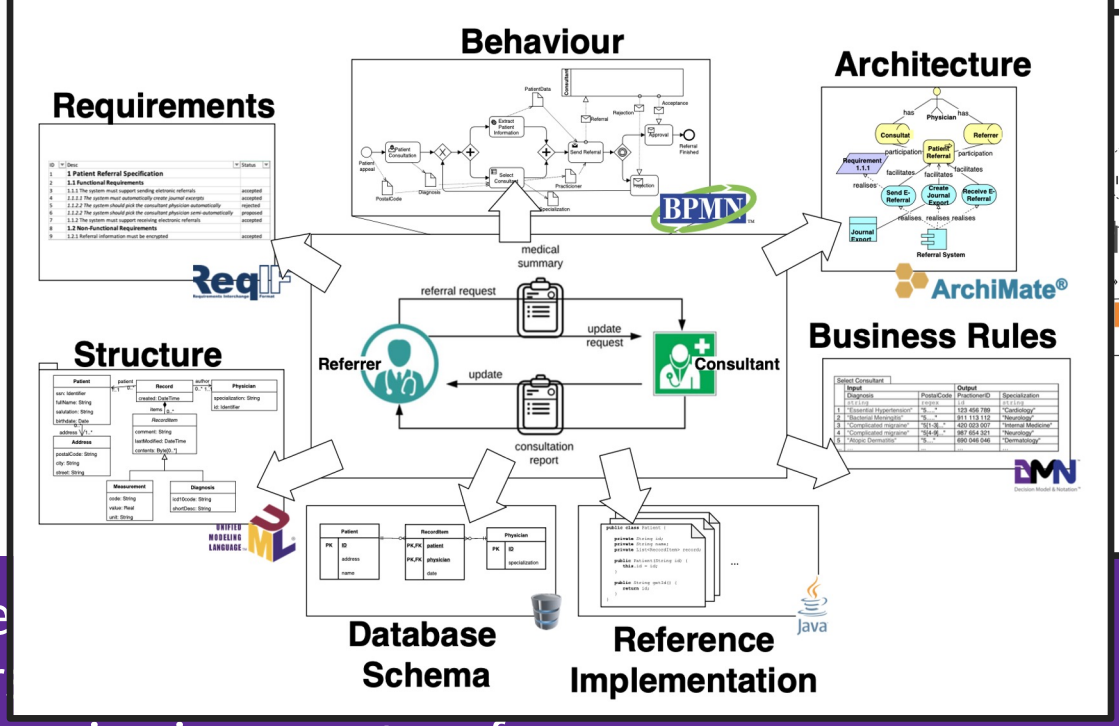
1. Define correspondence
2. «Put them together»/global model
3. Define consistency rules/constraints
4. Check the rules/constraints
   If not consistent, try to repair
   If global repair, propagate update to local
5. After each local/global update, go to 4

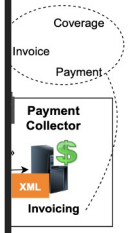

Privacy
Security
Robustness
Performance
...
Composition
Multi-modelling
Coordination
Multi-paradigm modelling
Collaboration
Multi-view-point modelling
Synchronization
INTEROPERABILITY
Co-simulation
Weaving
Bidirectional Transformation
Embedding
Extension
Merging
Multi-typing/decoration
Data-exchange
Model/scheme exchange

Interoperability

Different systems operate «together»

Fields





Multi-modelling: multiple systems (parts exist fir
Multi-viewpoint modelling: viewing a system from different perspectives (hole exist first)
Multi-paradigm modelling: languages of the parts belong to different paradigms (often continous systems, CPSs, real-time systems, robotics)

Methods

Composition: mostly languages and models (create whole)
Coordination: mostly wrt. behavior
Collaboration: often data and message/event exchange
Synchronization: independent systems with notification
Co-simulation: often for CPSs

Techniqu

To merge

Comprehensive syst

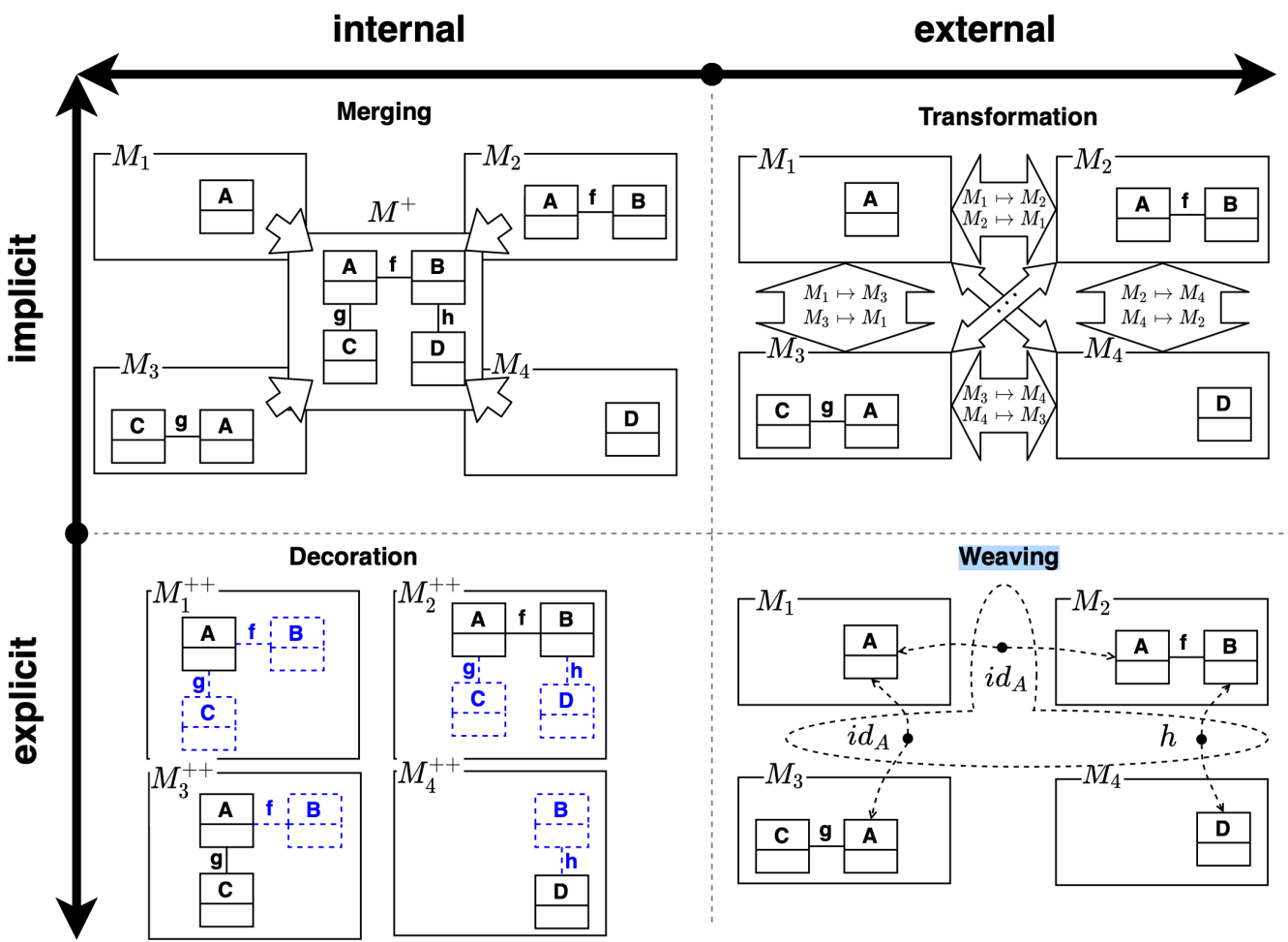
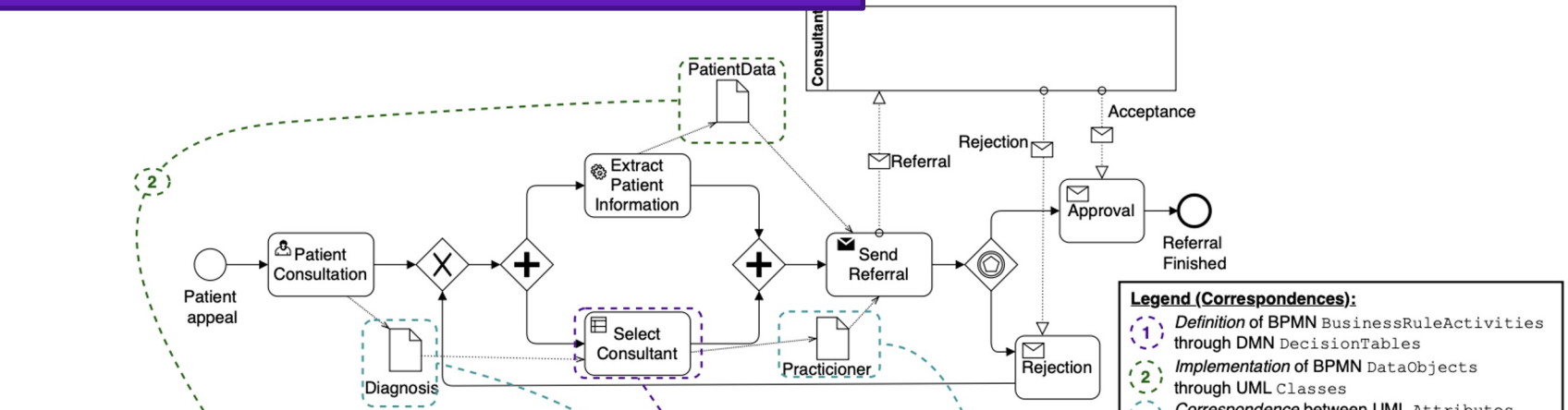
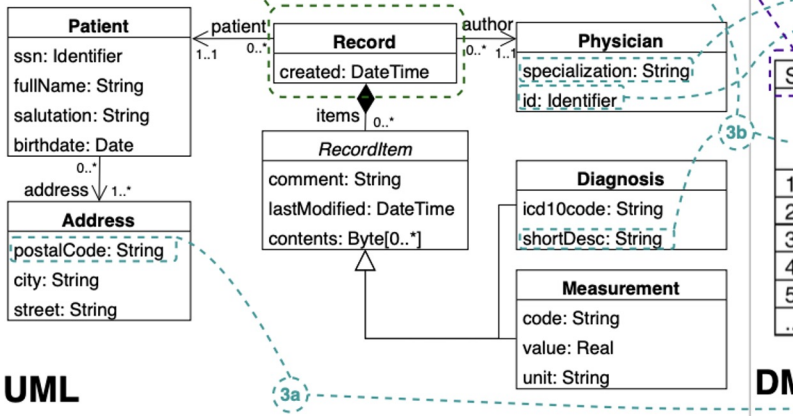
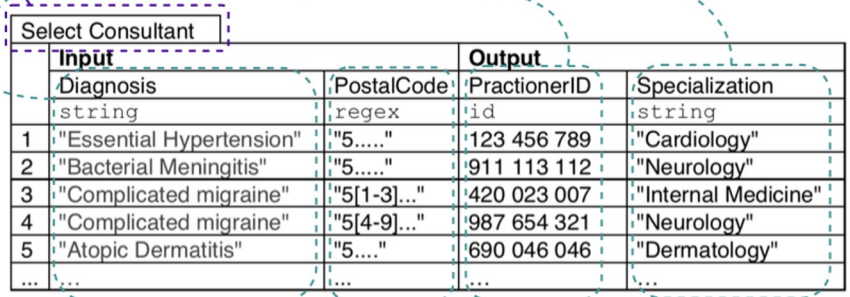


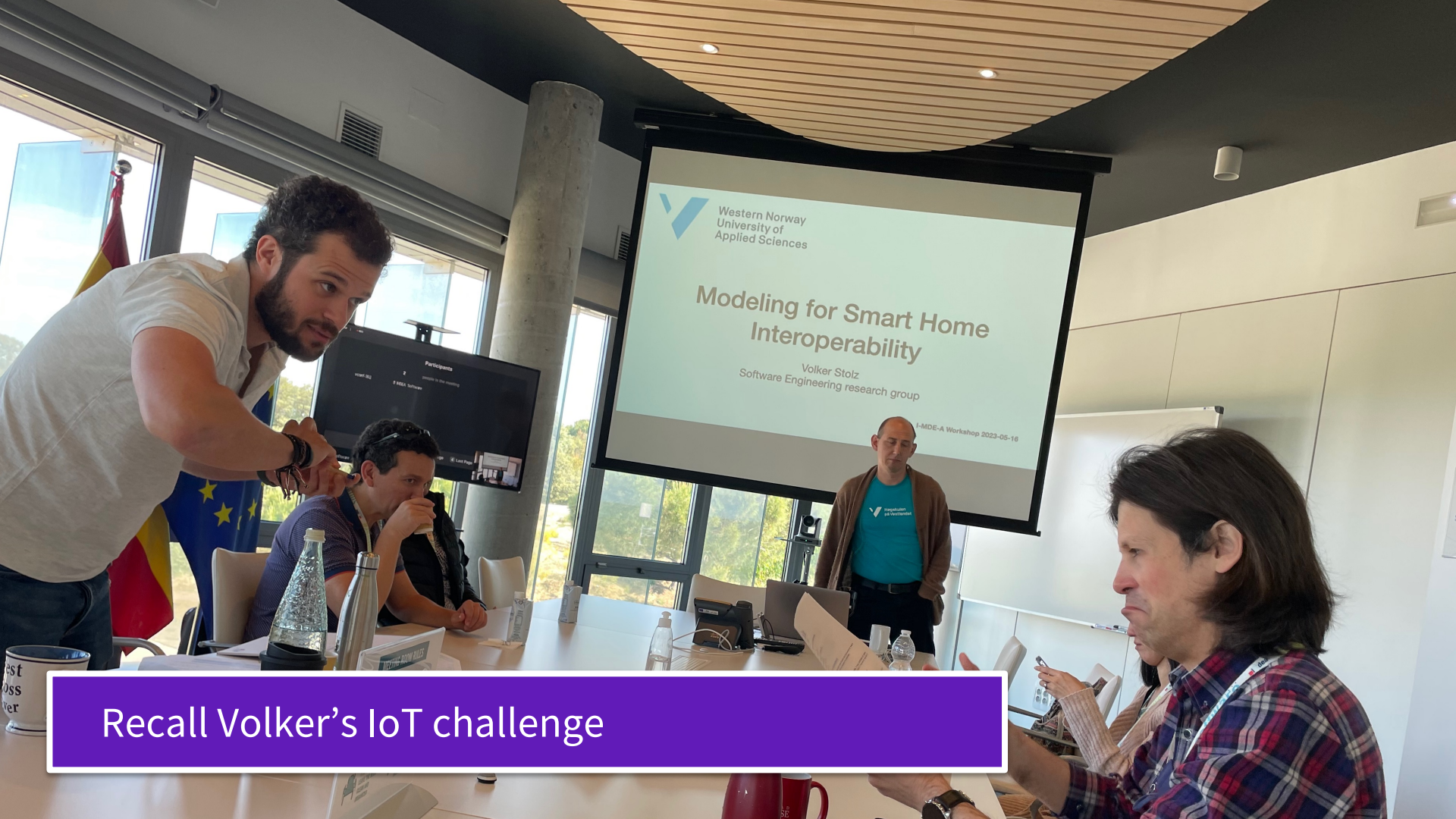
Fig. 4.7: Trace-based Commonality Representation

Example: commonality leading to comprehensive system









Recall Volker's IoT challenge

WIRELESS
ALARM FAMILY
ONE ALARM
TRIGGERS ALL

65 mm Photoelectric Smoke Alarm
Wireless Family

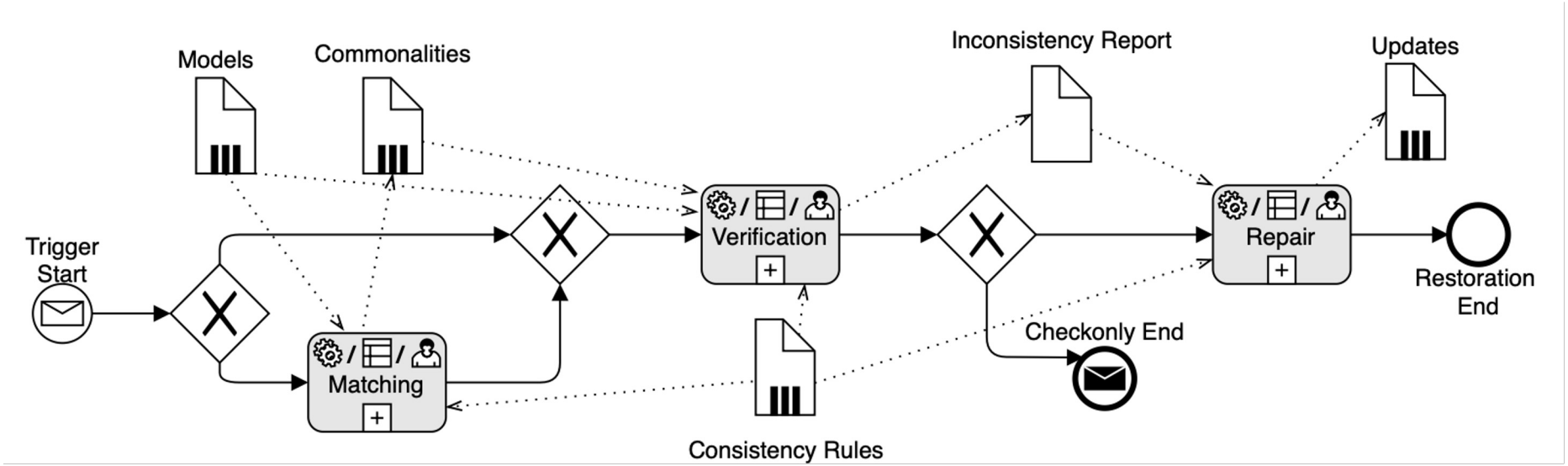
65 mm Thermal Heat Alarm
Wireless Family

78 mm Photoelectric Smoke Alarm
Wireless Family

IFTTT

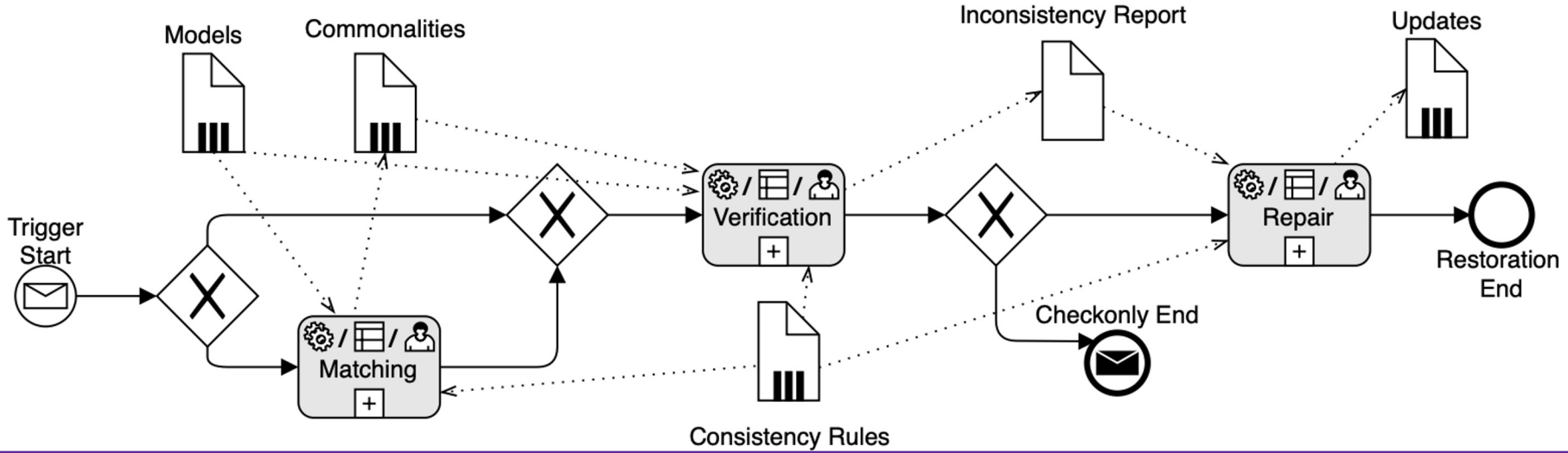
Yale Doorman Classic Connected

General Scheme: Consistency management



Feature model

Organising concepts in the steps



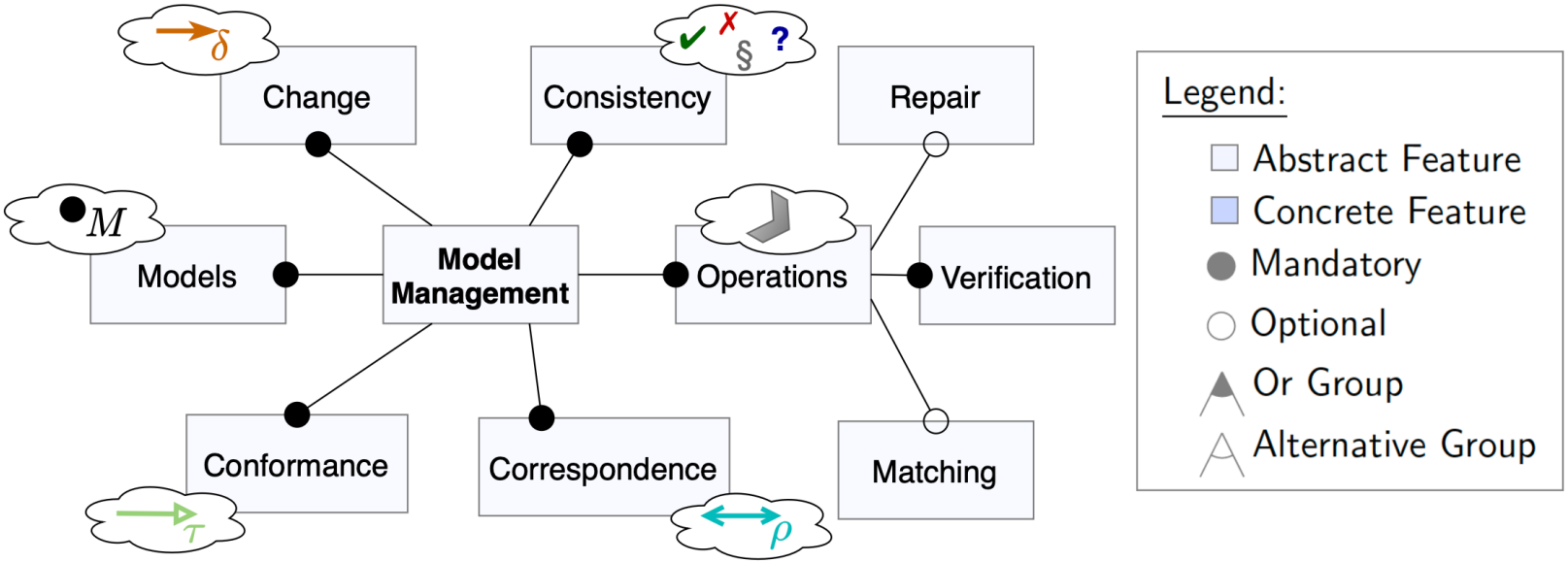
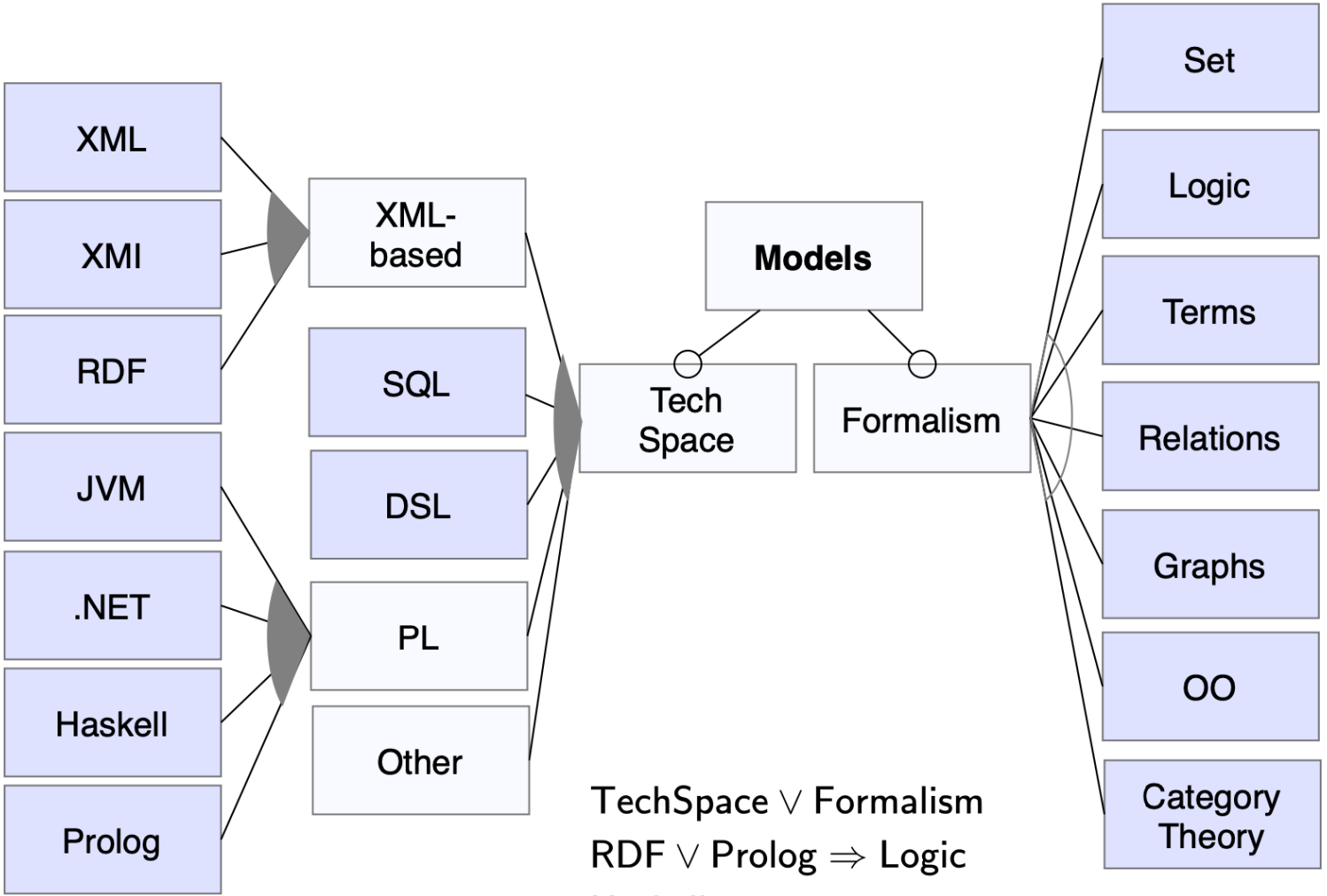


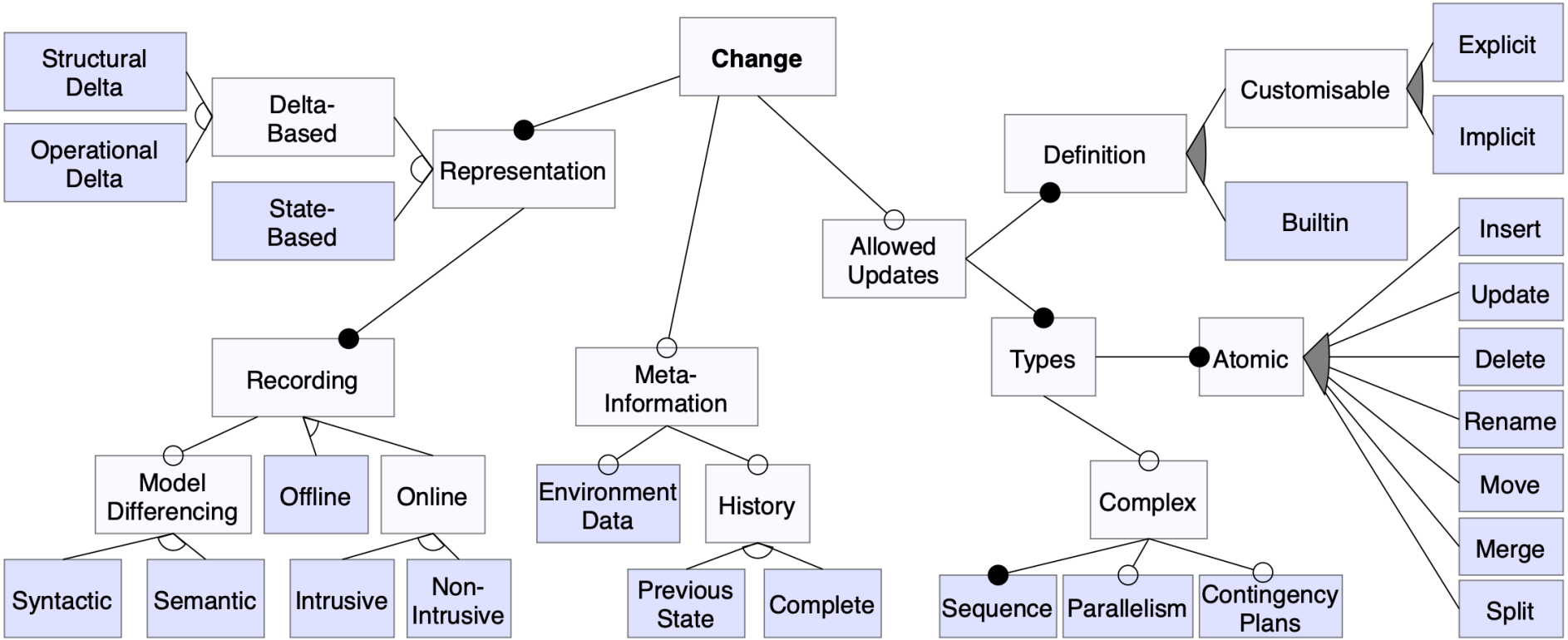
Fig. 4.1: Feature Model Overview



$\text{TechSpace} \vee \text{Formalism}$

$\text{RDF} \vee \text{Prolog} \Rightarrow \text{Logic}$

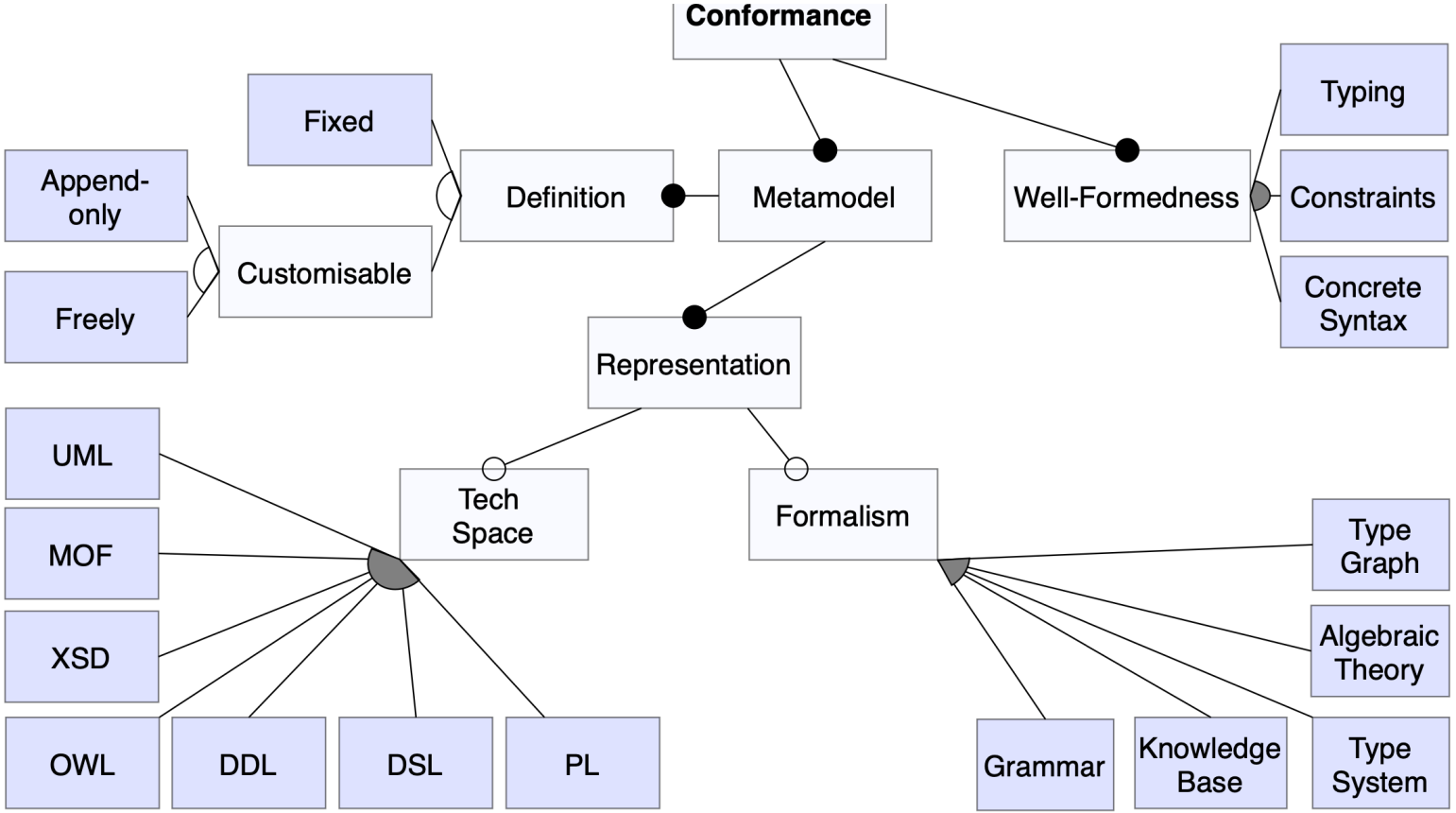
$\text{Haskell} \Rightarrow \text{Terms}$



Offline $\wedge$ Delta-Based $\Rightarrow$ Model Differencing

Operational Delta $\wedge$ Model Differencing $\Rightarrow$ Semantic

Delta-Based $\Rightarrow$ Previous State



TechSpace $\vee$ Formalism

UML $\Rightarrow$ Fixed $\vee$ Append only

DDL $\Rightarrow$ SQL

XSD $\Rightarrow$ XML

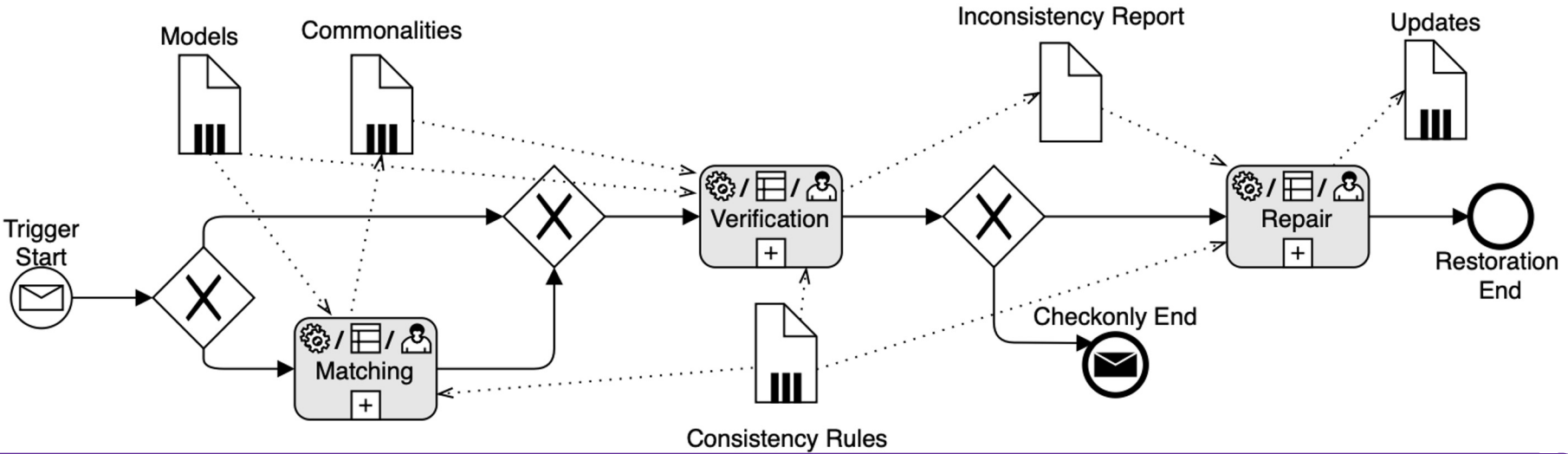
OWL $\Rightarrow$ RDF $\wedge$ Knowledge Base

Algebraic Theory $\vee$ Type System $\Rightarrow$ Terms

Type Graph $\Rightarrow$ Graphs

Knowledge Base $\Rightarrow$ Logic

Correspondence and commonality



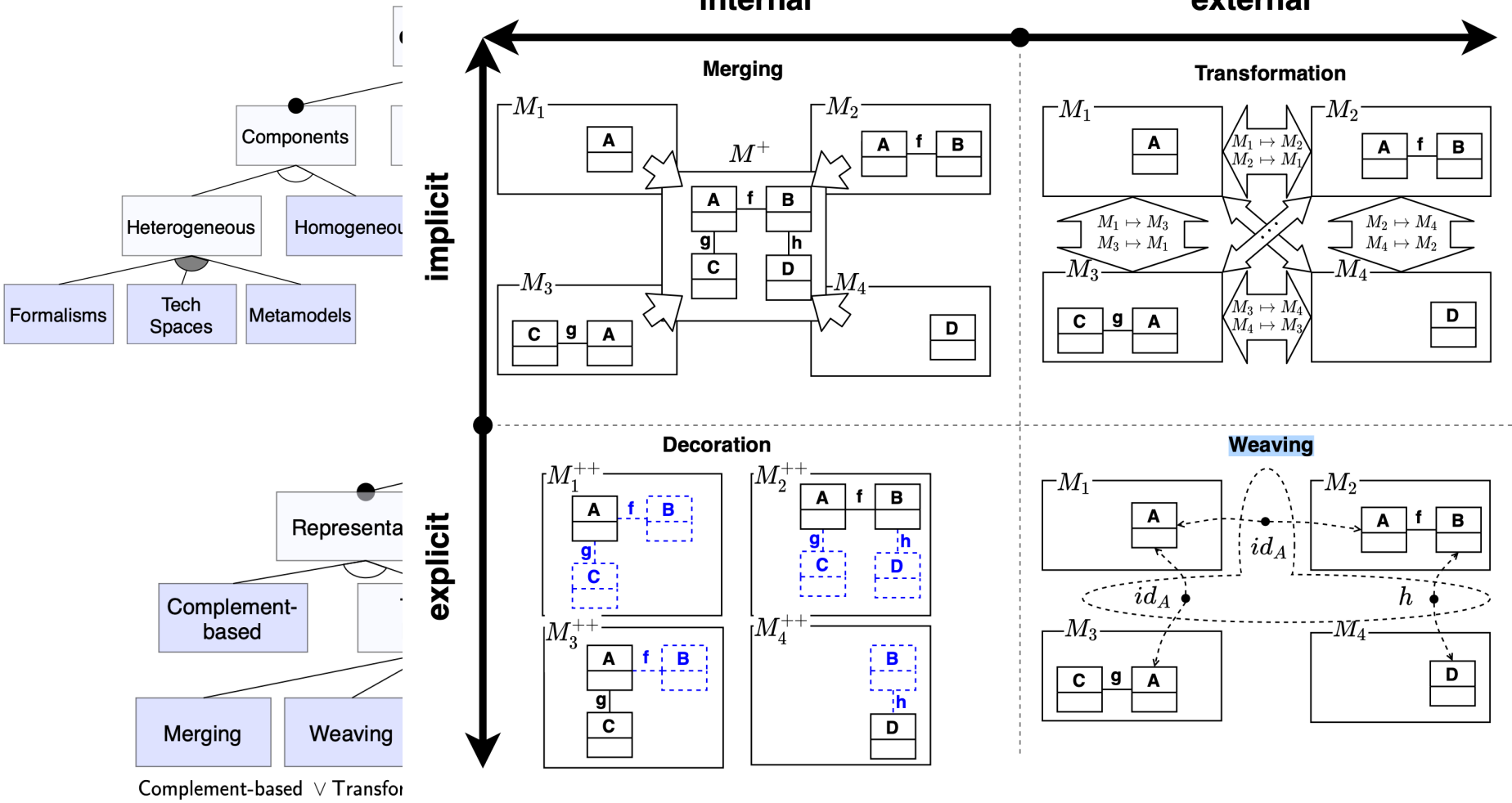
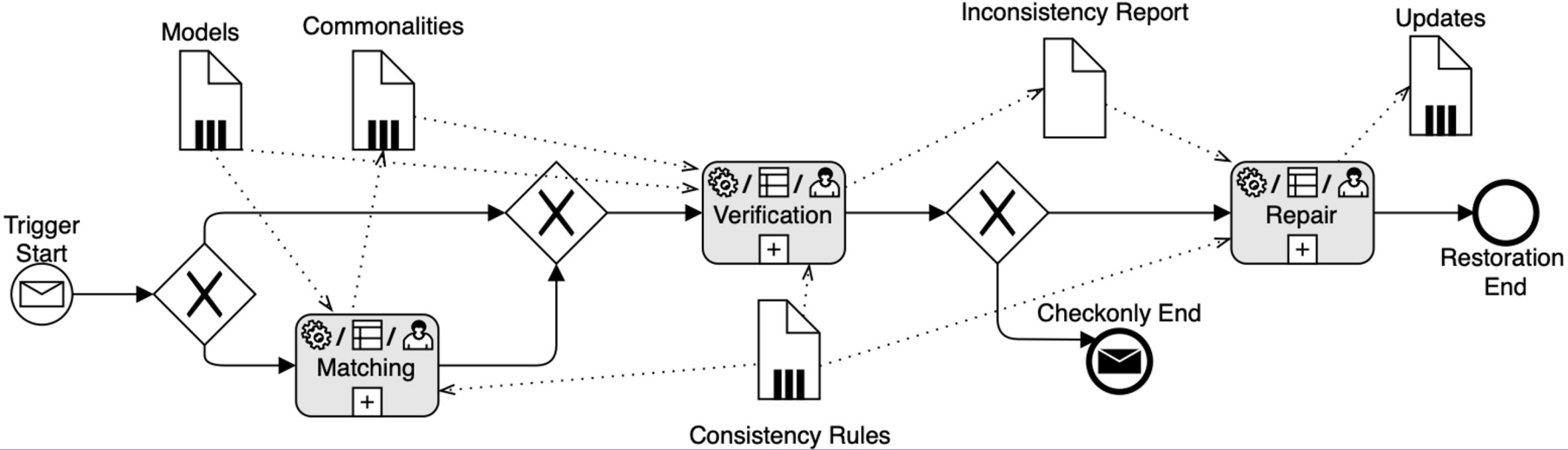
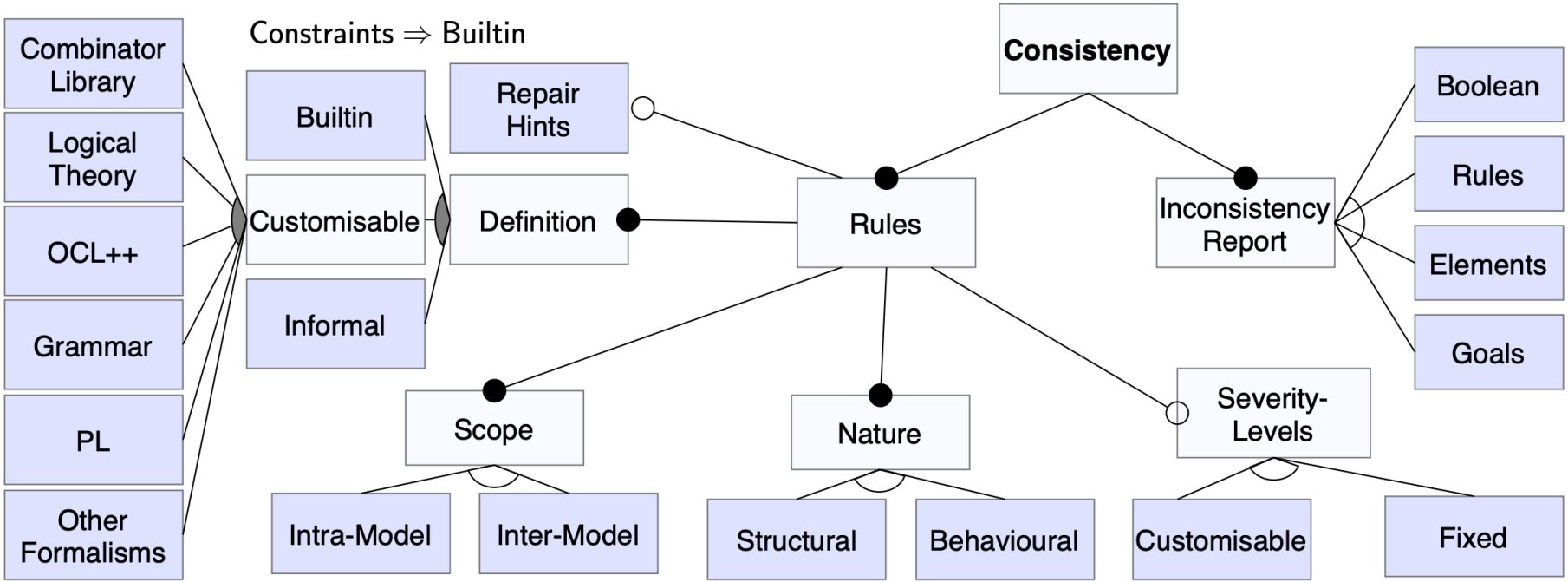
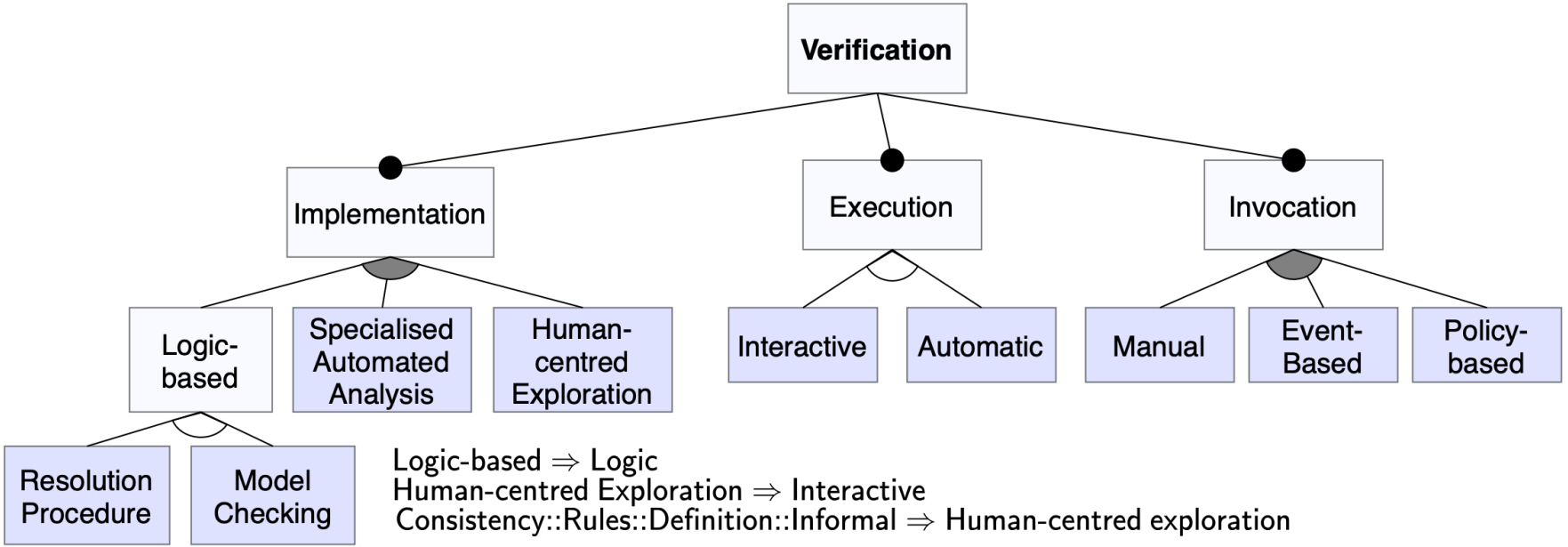


Fig. 4.7: Trace-based Commonality Representation

Define consistency

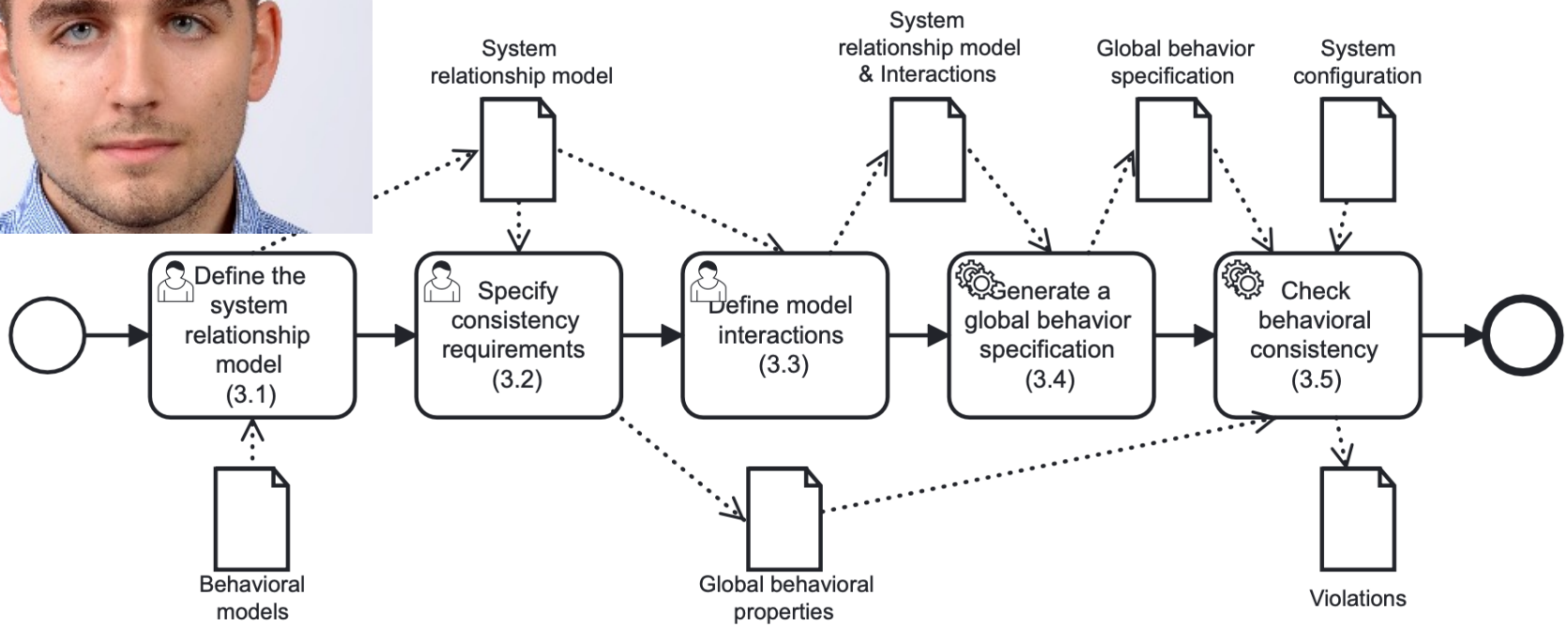



Combinator Library
Logical Theory
OCL++
Grammar
PL
Other Formalisms
Constraints ⇒ Builtin
Builtin
Customisable
Informal
Repair Hints
Definition
Consistency
Rules
Inconsistency Report
Boolean
Rules
Elements
Goals
Scope
Intra-Model
Inter-Model
Nature
Structural
Behavioural
Severity-Levels
Customisable
Fixed

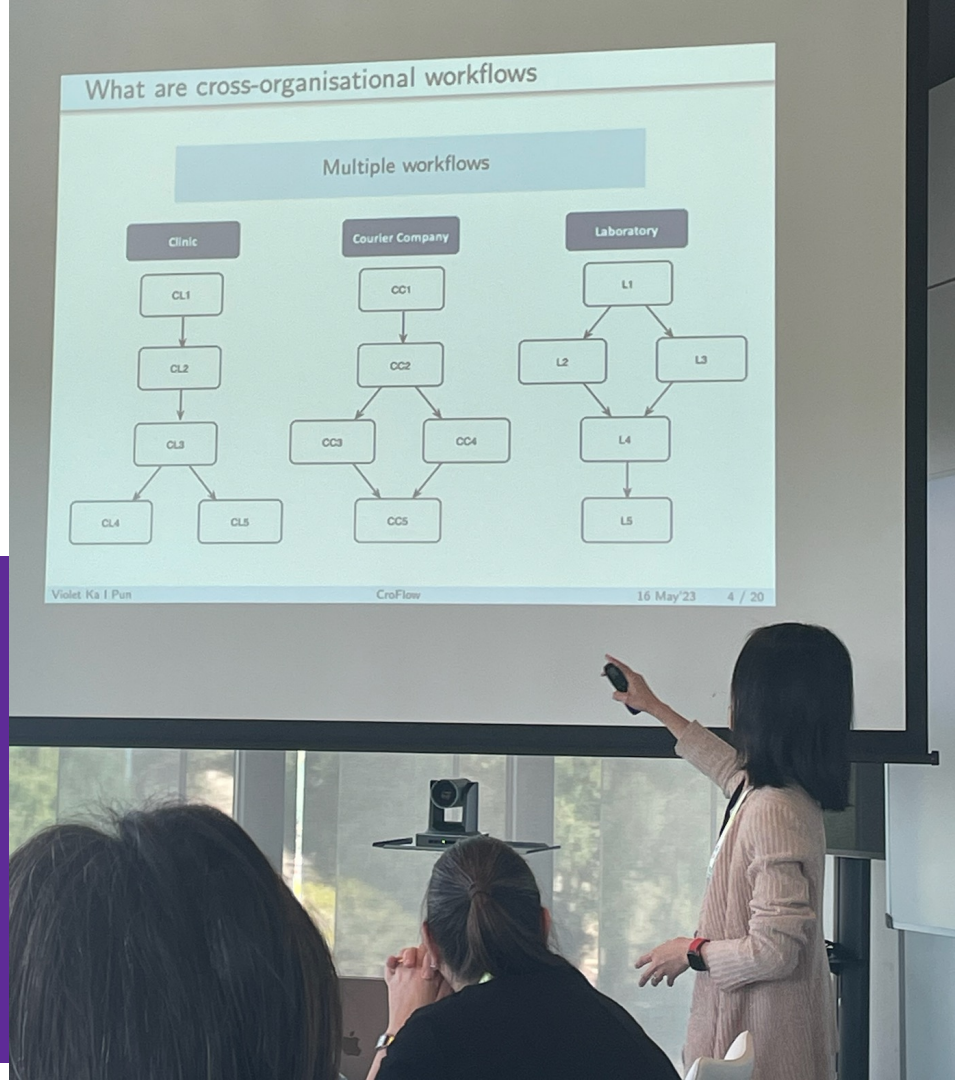


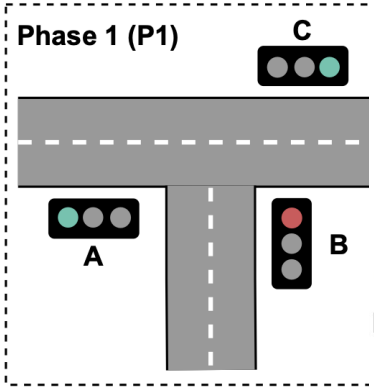
Logic-based ⇒ Logic
Human-centred Exploration ⇒ Interactive
Consistency::Rules::Definition::Informal ⇒ Human-centred exploration

In case of behavior



Recall Violet's CroFlow cases





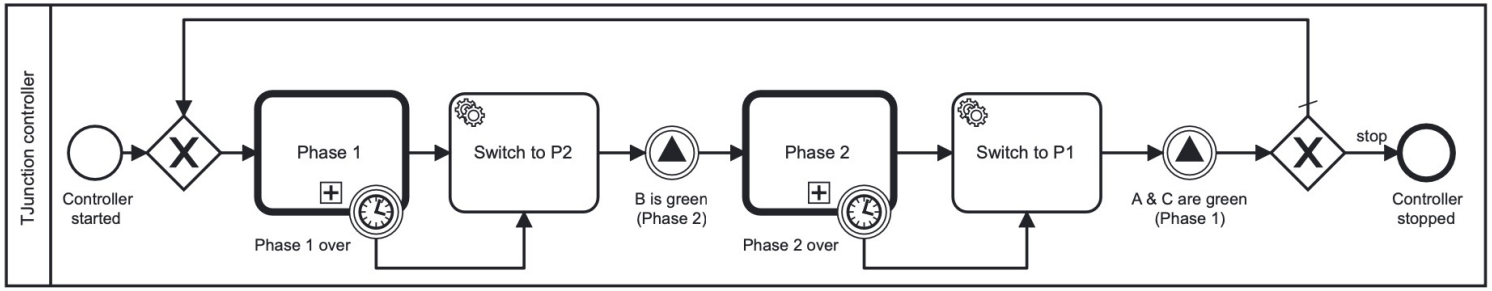
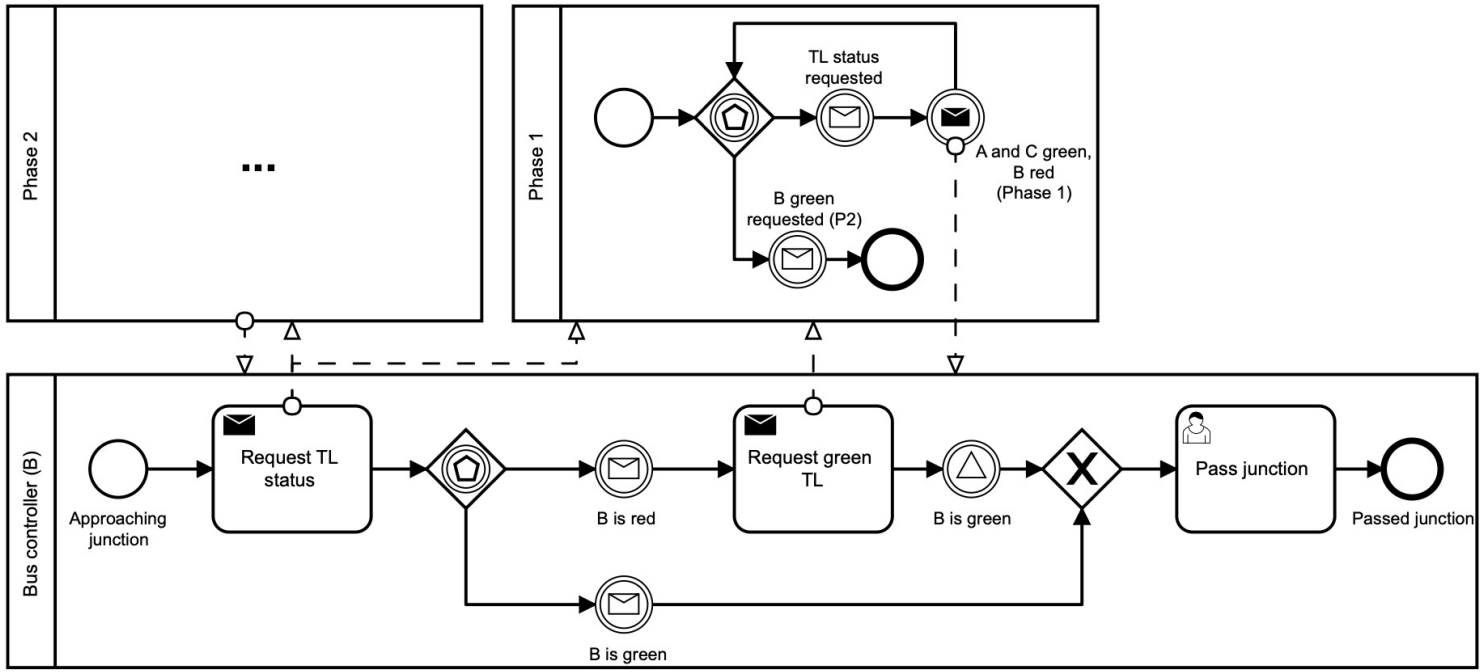


Figure 3 Model for a TJunction controller

Traffic light



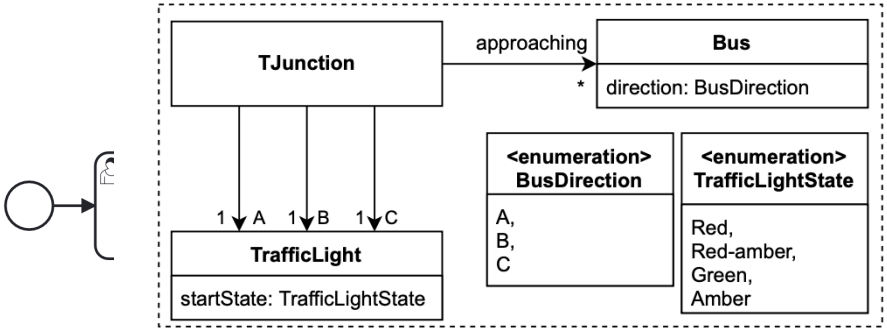


Figure 5 System relationship model of the traffic management system

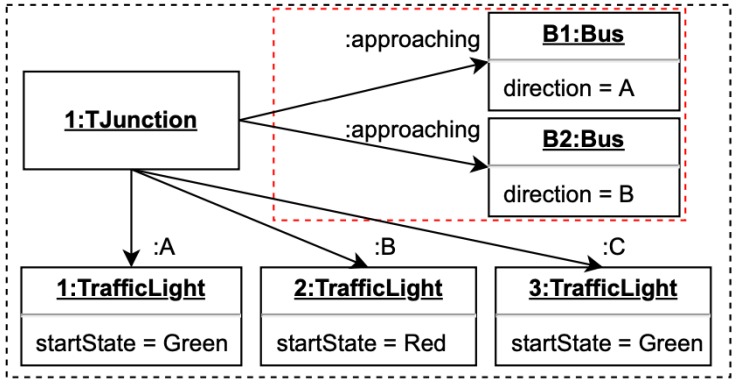


Figure 6 Test system configuration

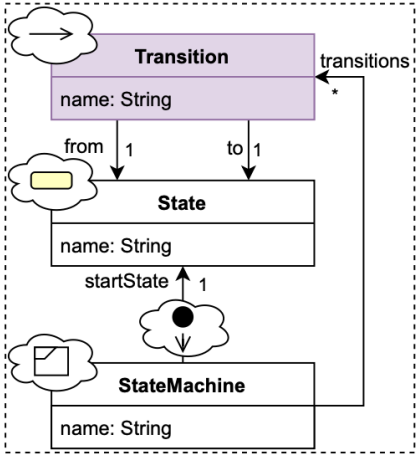
$$\Box\neg((A_{green} \vee A_{amber}) \wedge (B_{green} \vee B_{amber})) \quad (1)$$

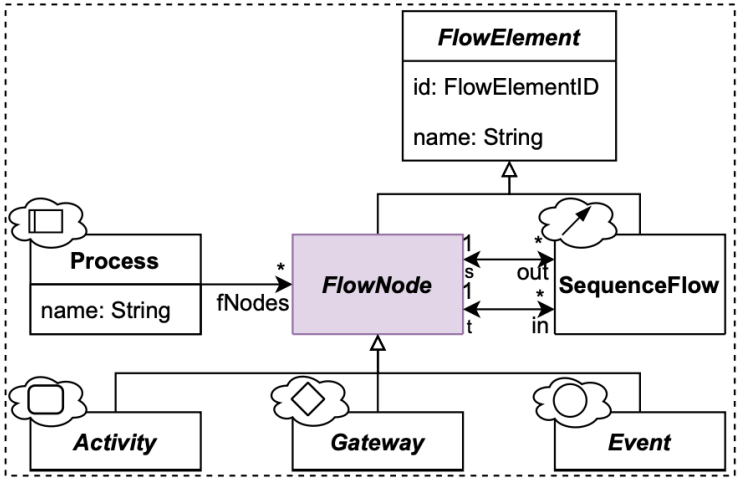
$$\Box\neg((C_{green} \vee C_{amber}) \wedge (B_{green} \vee B_{amber})) \quad (2)$$

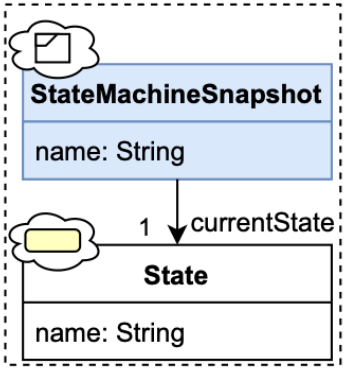
$$\Box\neg(B1_{passing} \wedge (A_{red} \vee A_{red-amber})) \quad (3)$$

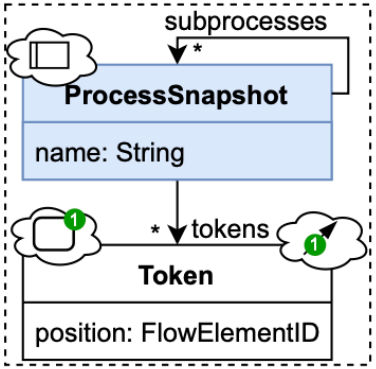
$$\Box\neg(B2_{passing} \wedge (B_{red} \vee B_{red-amber})) \quad (4)$$

SRM and global rules/constraints









Premises: The «states» and «state changing elements» in the participating languages

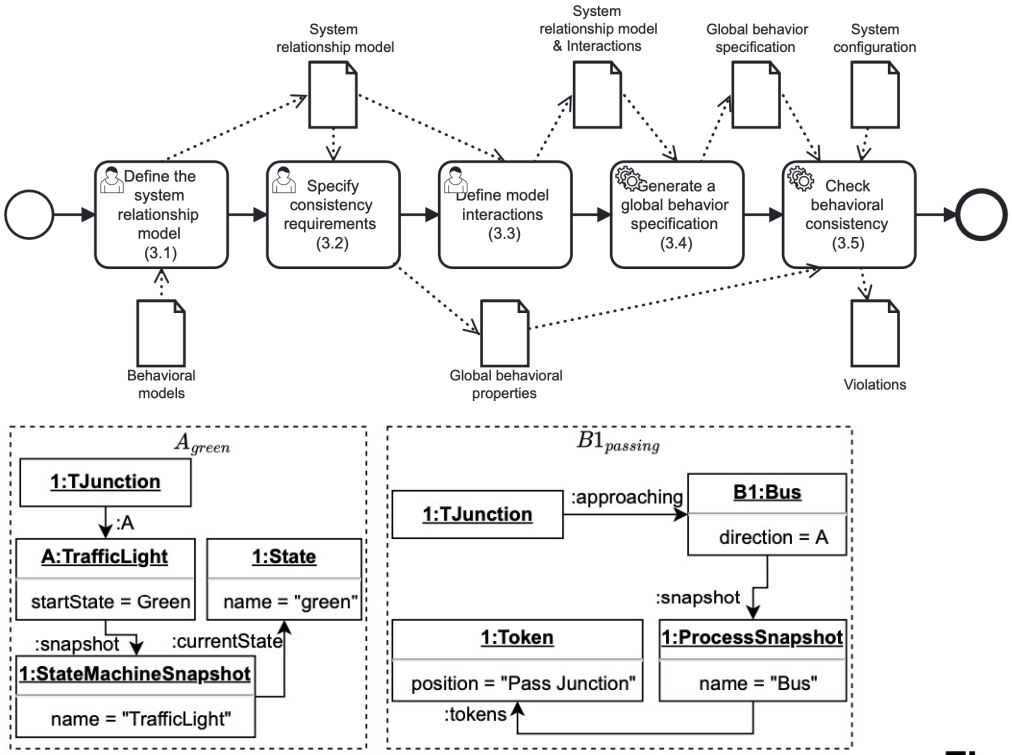


Figure 11 Atomic propositions A_{green} and $B1_{passing}$

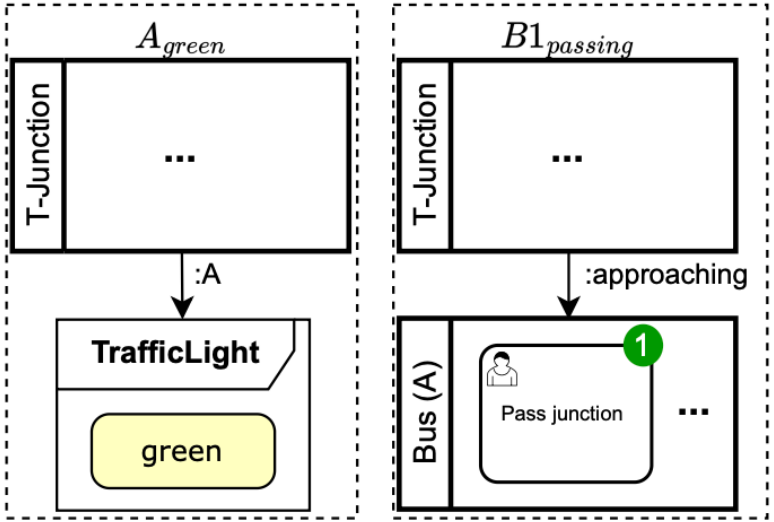
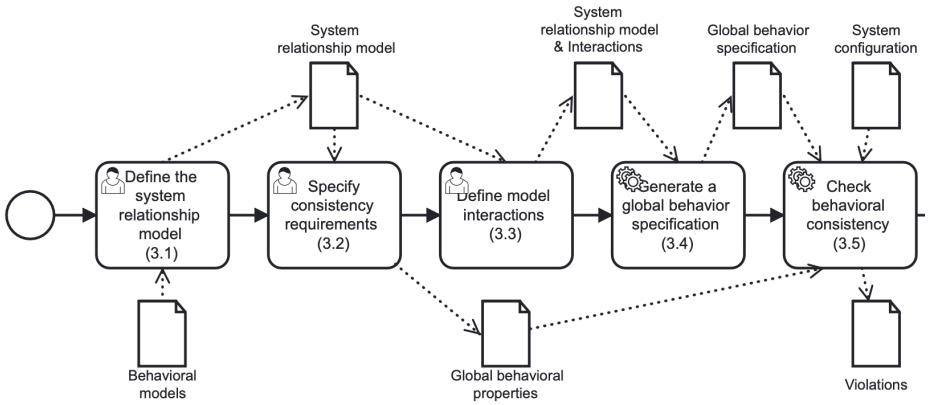


Figure 12 Concrete syntax for A_{green} and $B1_{passing}$

Property specification




```
1  synchronize(TJunction.Switch_to_P1,
2              TJunction.A.turn_green,
3              TJunction.B.turn_red,
4              TJunction.C.turn_green)
5
6  synchronize(TJunction.Switch_to_P2,
7              TJunction.A.turn_red,
8              TJunction.B.turn_green,
9              TJunction.C.turn_red)
```

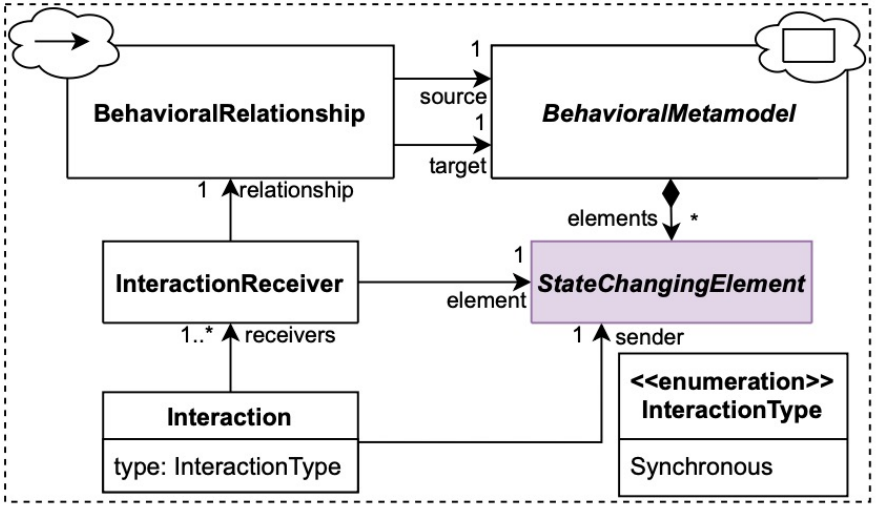


Figure 13 System relationship metamodel

Define interactions

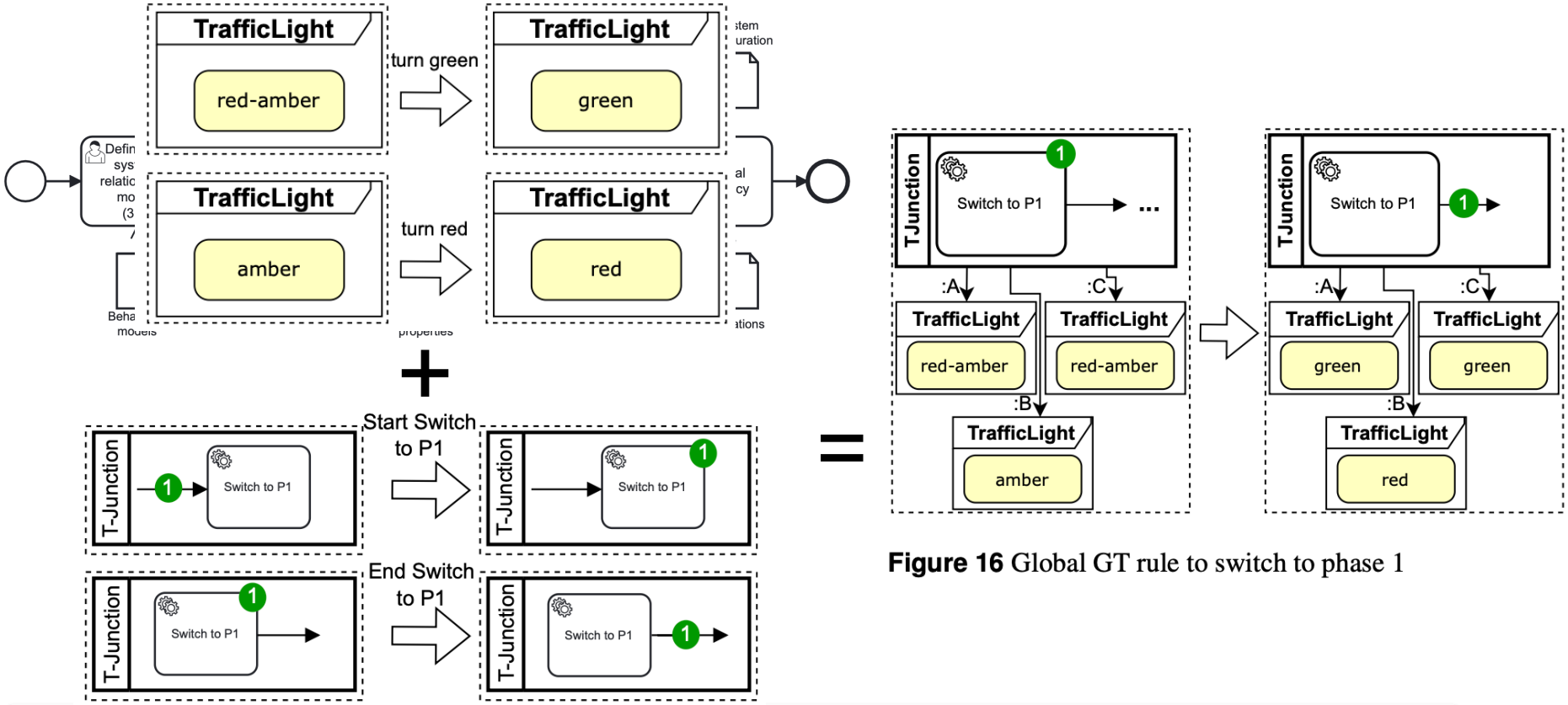


Figure 16 Global GT rule to switch to phase 1

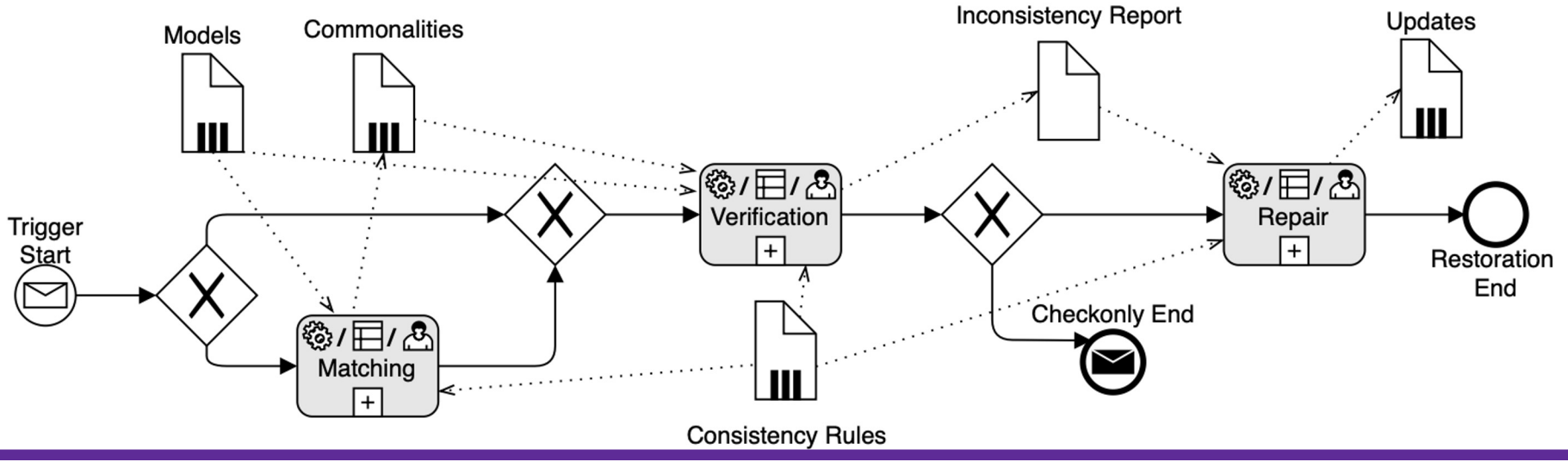
Generate global behavior

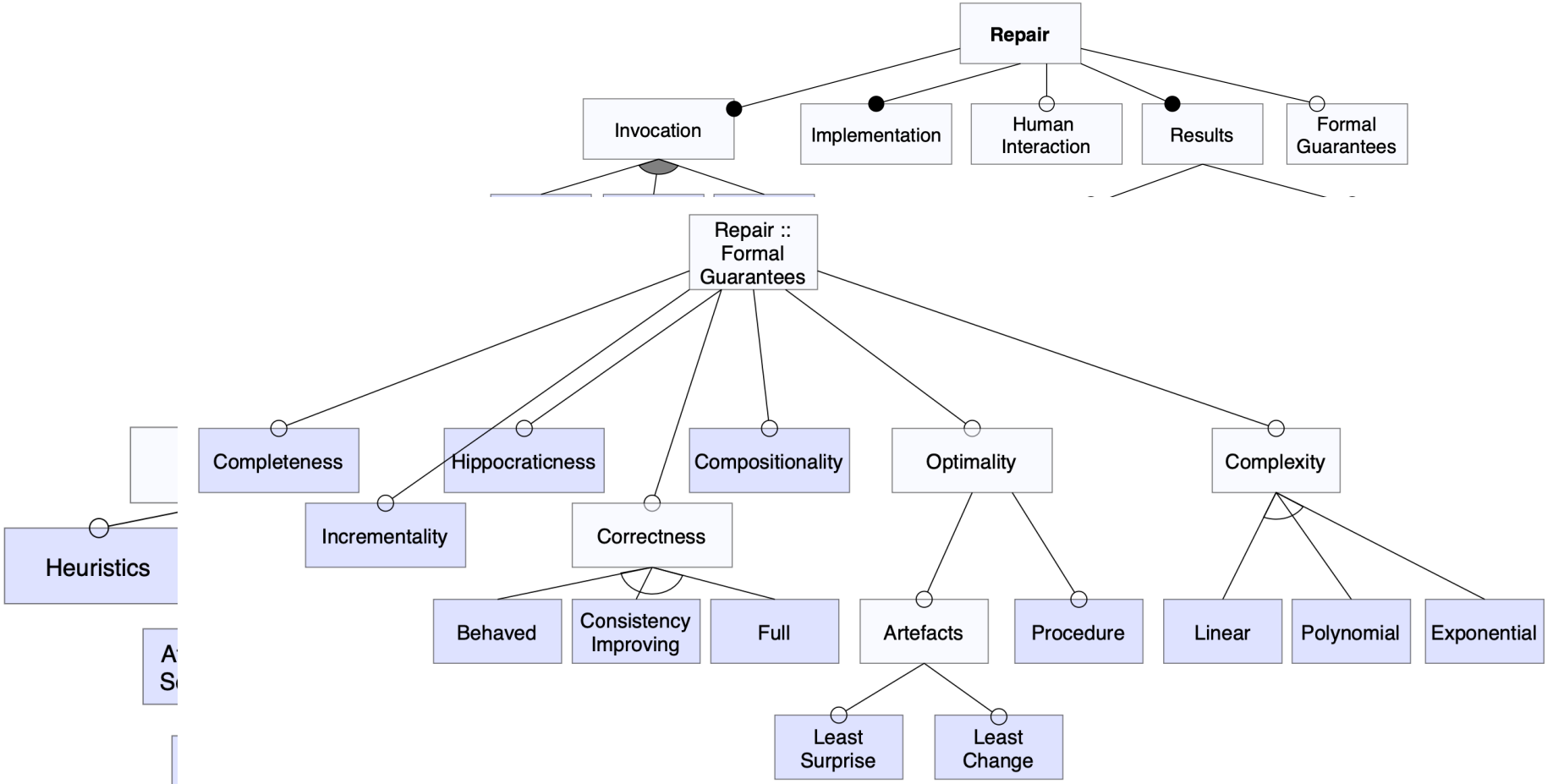
Formalization/Foundations

Structure: Categorty theory
Behavior: Coalgebra

Æ æ æ æ!

When things go wrong: repair





Semantic ⇒ Consistency::Rules::Definition::Grammar

Solver ⇒ Logic

PARMOREL:

Personalized and automatic repair of models using reinforcement learning

General approach:

Multi-models or not, this should work!

Ph.D.: Angela B. Rodriguez

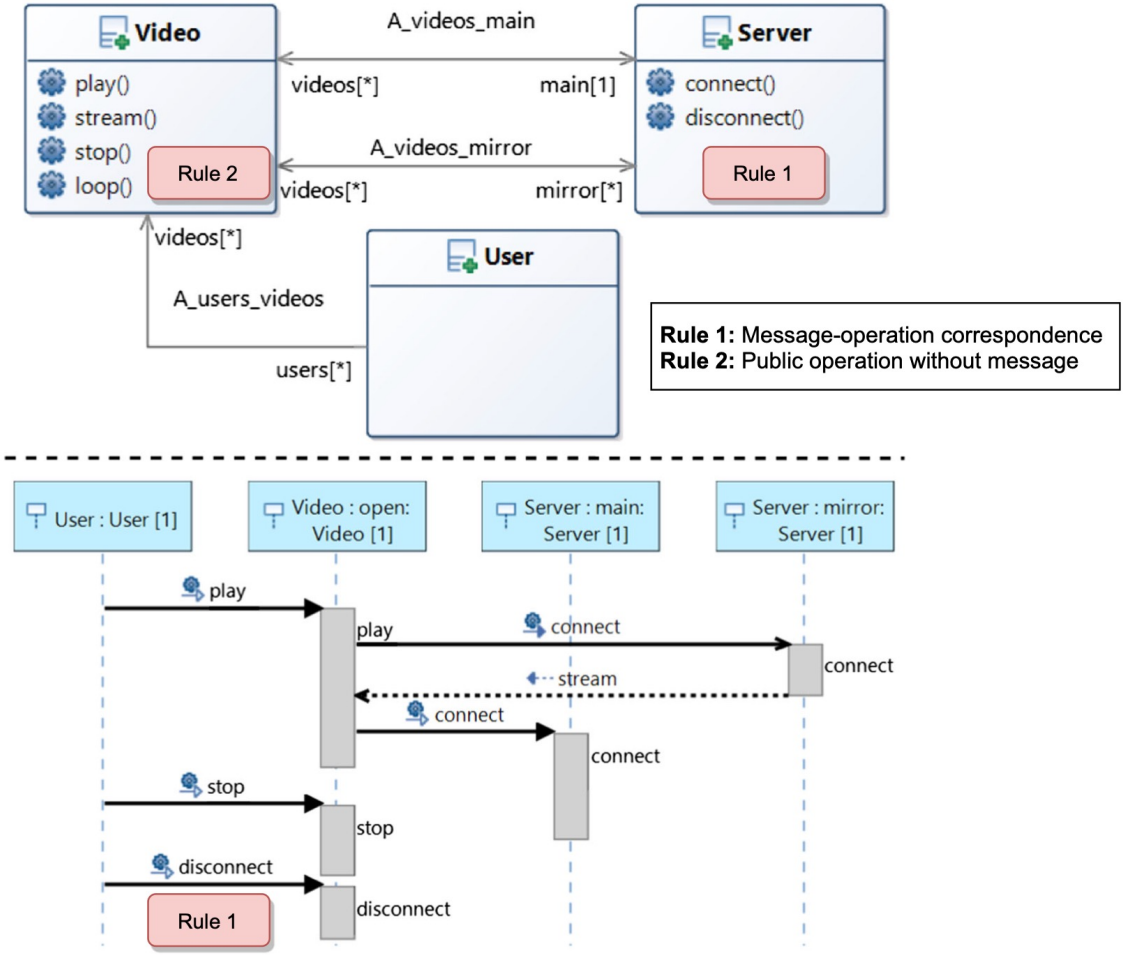
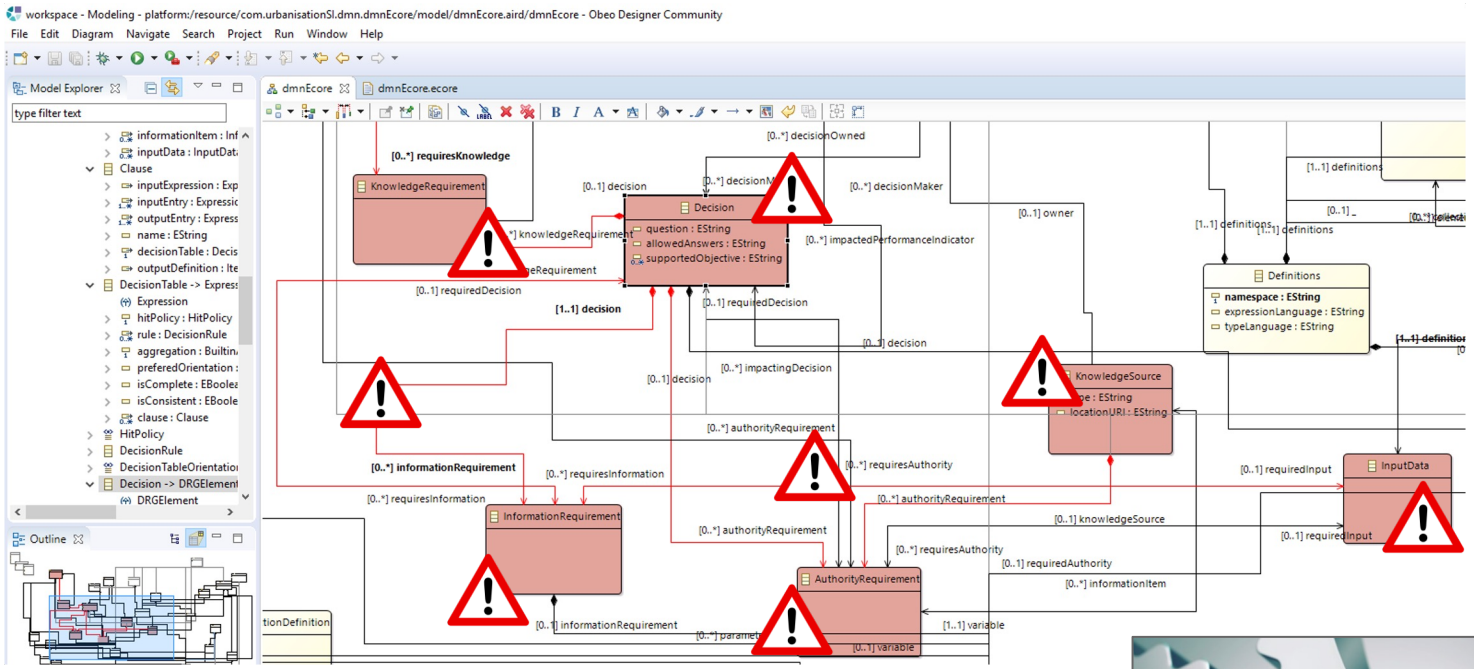
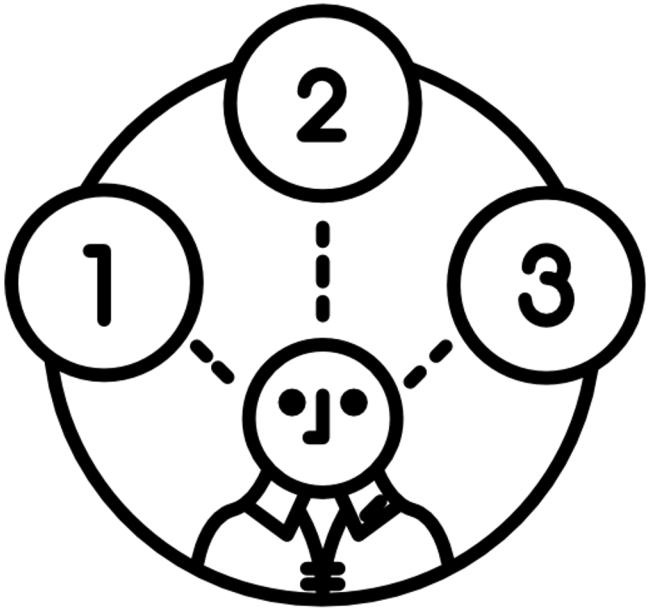


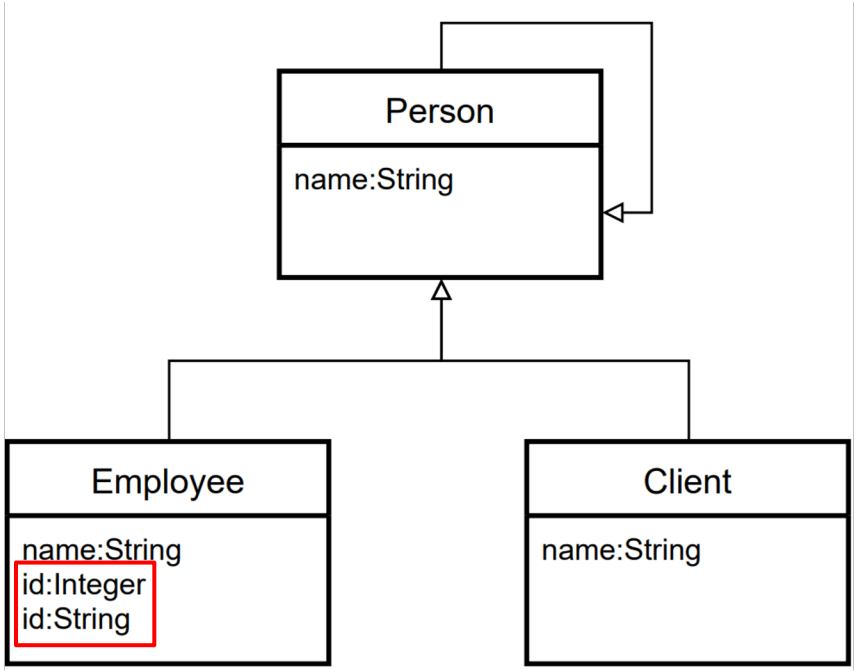
Fig. 8 Running example for inconsistencies between UML class and sequence diagrams


workspace - Modeling - platform:/resource/com.urbanisationSI.dmn.dmnEcore/model/dmnEcore.aird/dmnEcore - Obeo Designer Community
File Edit Diagram Navigate Search Project Run Window Help
Model Explorer
type filter text
Outline
dmnEcore
dmnEcore.ecore
KnowledgeRequirement
Decision
question : EString
allowedAnswers : EString
supportedObjective : EString
Definitions
namespace : EString
expressionLanguage : EString
typeLanguage : EString
KnowledgeSource
InputData
InformationRequirement
AuthorityRequirement
[0..*] requiresKnowledge
[0..1] decision
[0..*] decisionOwned
[0..*] decisionMaker
[0..1] owner
[0..*] impactedPerformanceIndicator
[0..1] requiredDecision
[1..1] decision
[0..*] impactingDecision
[0..*] authorityRequirement
[0..*] informationRequirement
[0..*] requiresInformation
[0..*] requiresAuthority
[0..1] requiredInput
[0..1] knowledgeSource
[0..1] requiredAuthority
[0..*] informationItem
[1..1] variable
[0..1] variable
[1..1] definitions

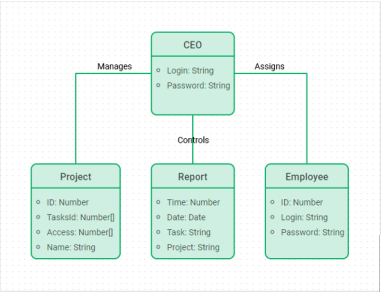

process
automation

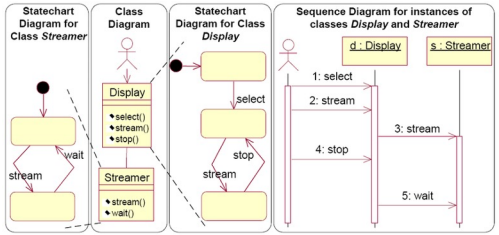
Each issue → multiple solutions.
Solutions are domain and user-dependant.

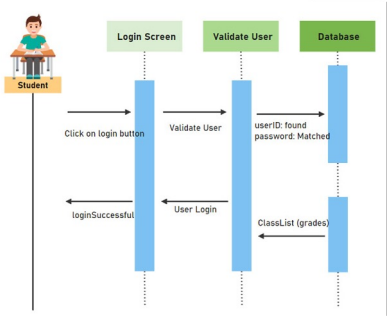


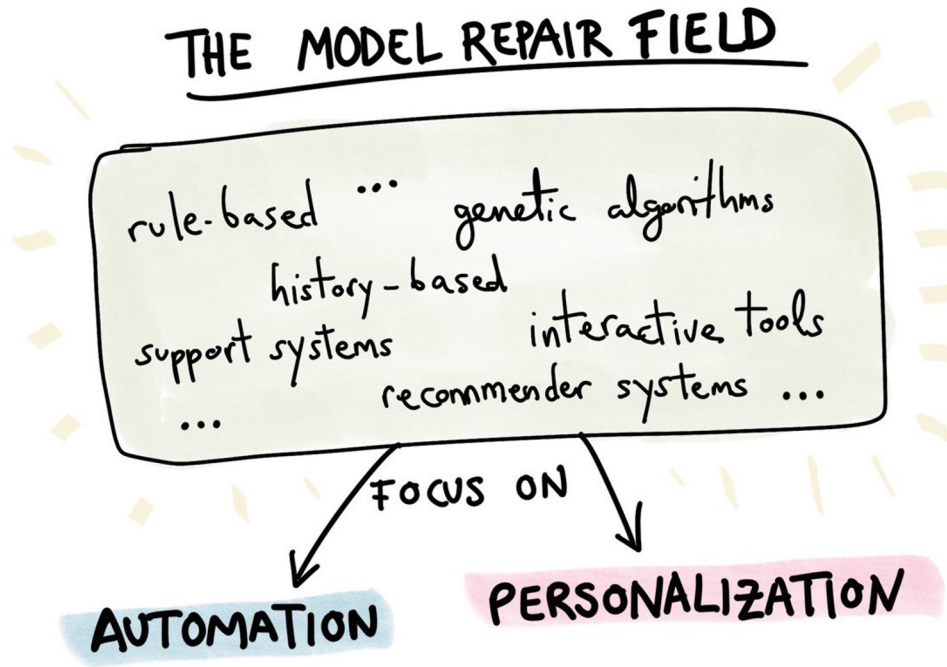

Person
name:String
Employee
name:String
id:Integer
id:String
Client
name:String

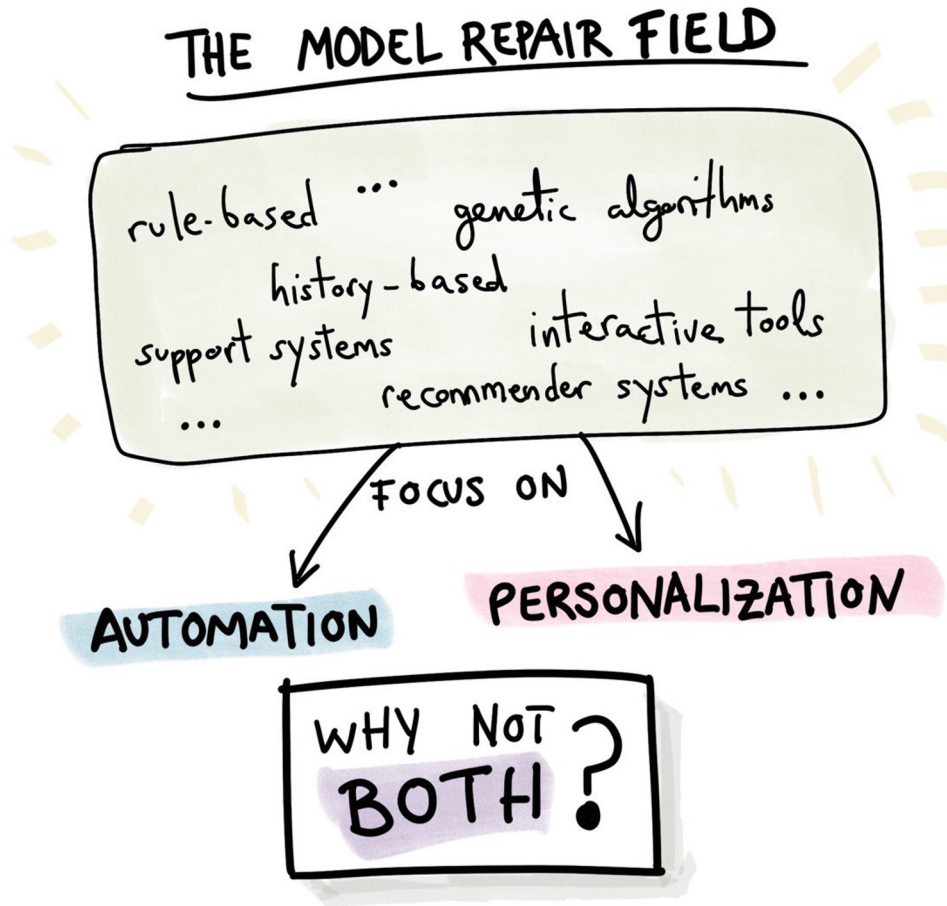
Each issue ➜ multiple solutions.
Solutions are domain and user-dependant.
Different types of issues and models.

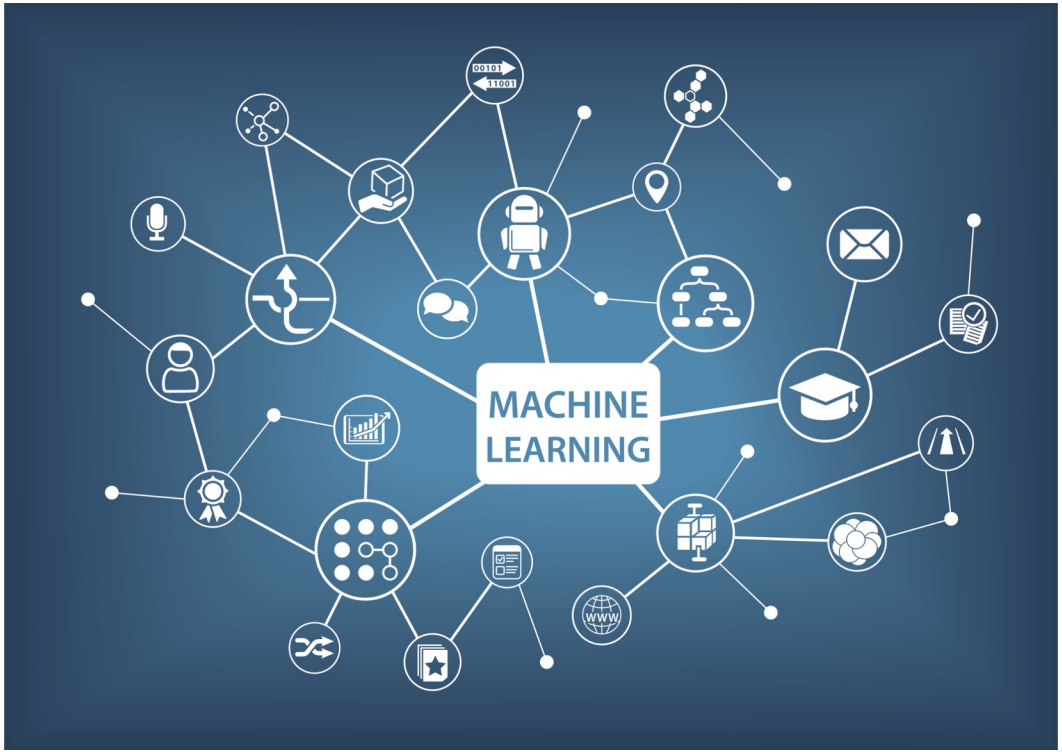


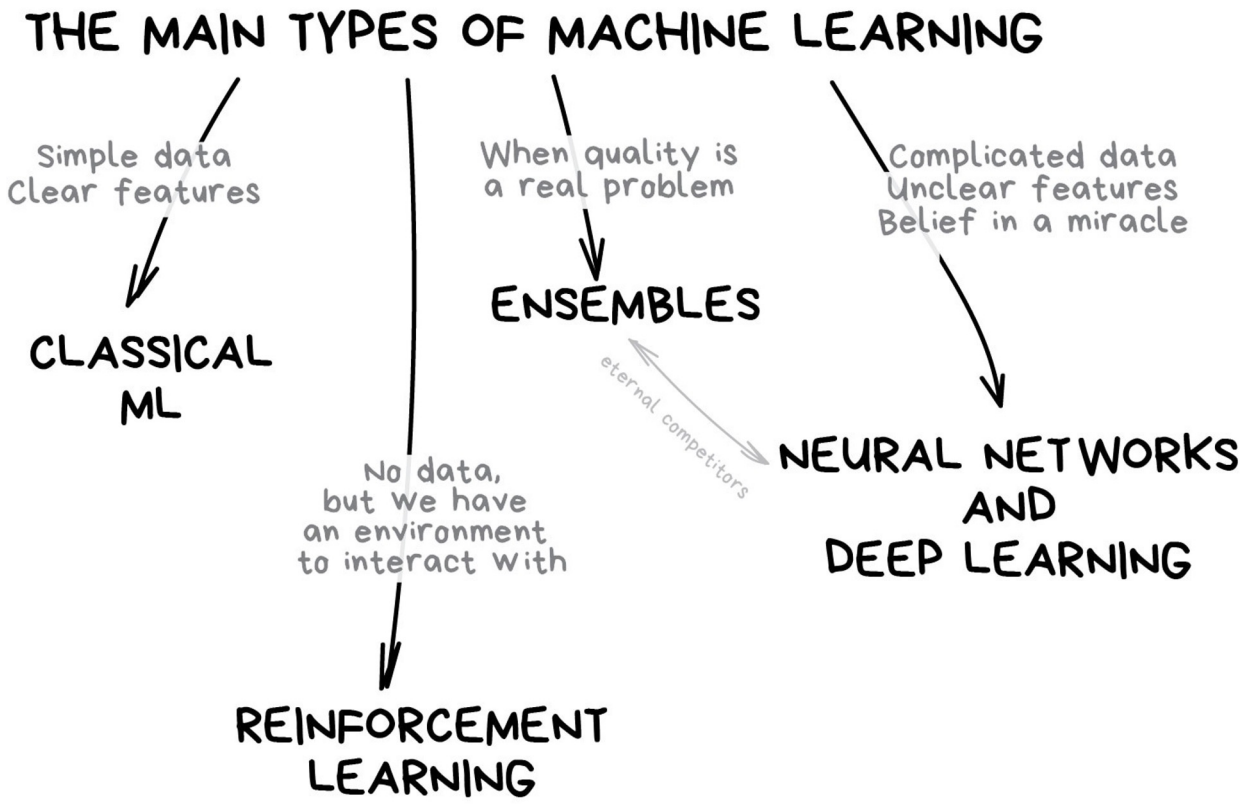





THE MODEL REPAIR FIELD
rule-based ...
genetic algorithms
history-based
support systems
interactive tools
recommender systems ...
...
FOCUS ON
AUTOMATION
PERSONALIZATION

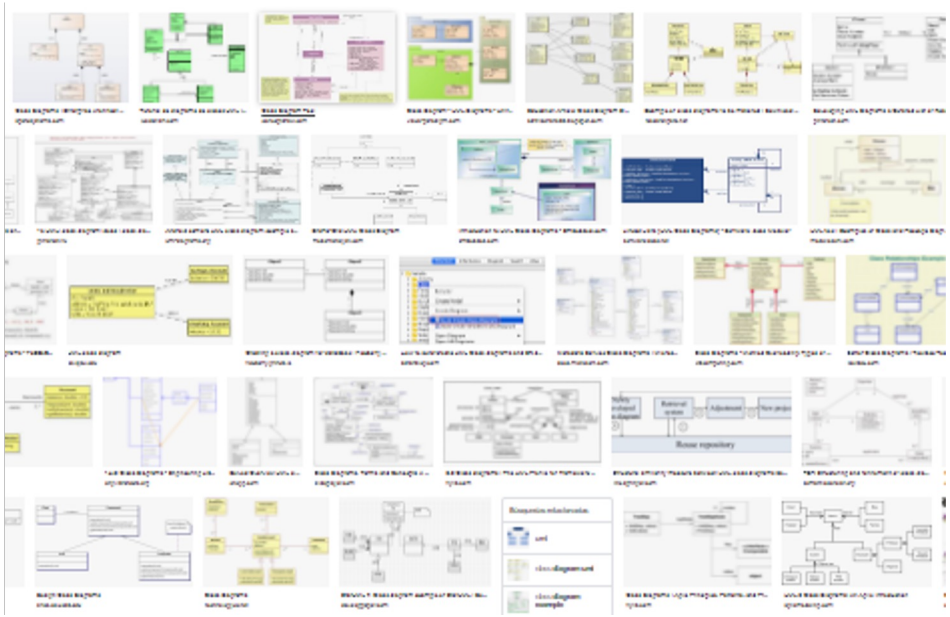

THE MODEL REPAIR FIELD
rule-based ...
genetic algorithms
history-based
support systems
interactive tools
recommender systems ...
...
FOCUS ON
AUTOMATION
PERSONALIZATION
WHY NOT BOTH?

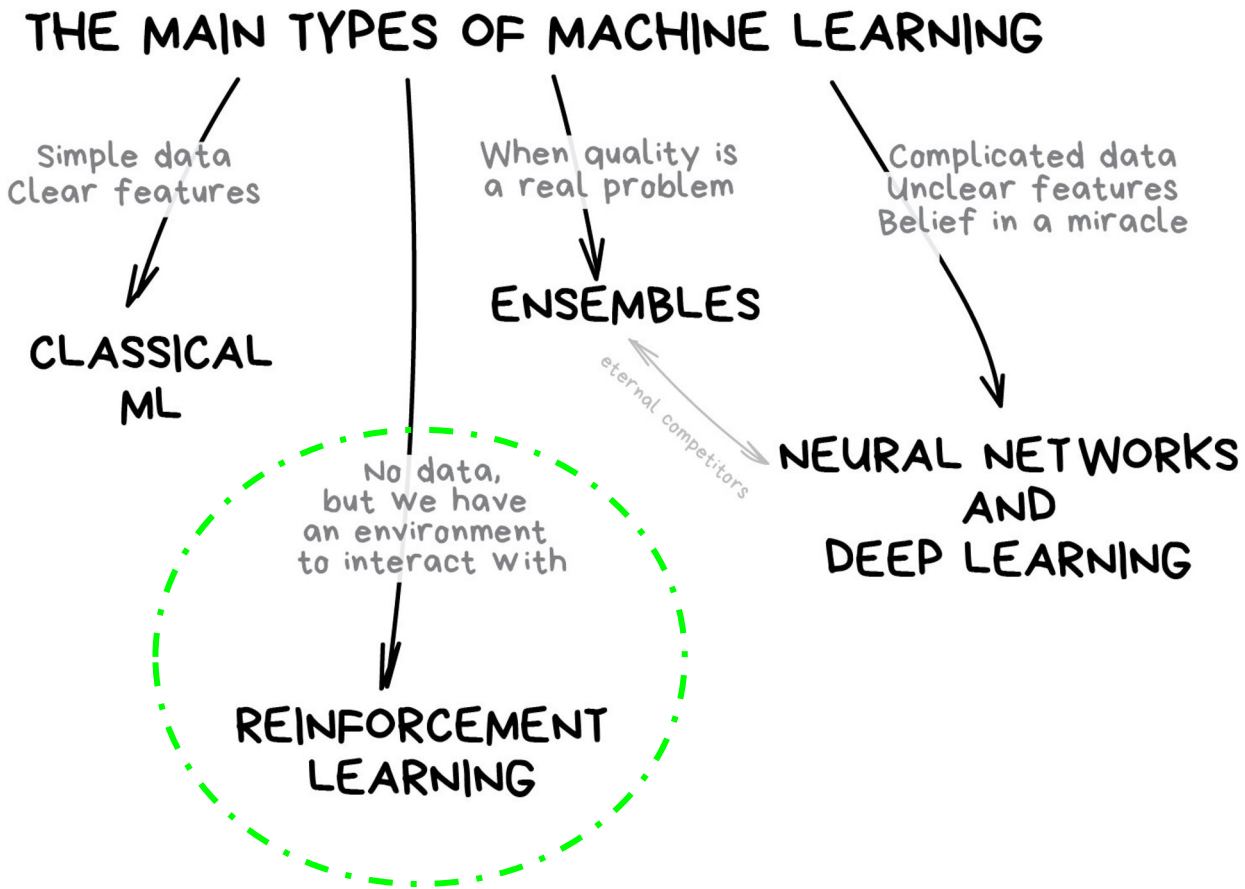

MACHINE
LEARNING
WWW


THE MAIN TYPES OF MACHINE LEARNING
Simple data
Clear features
CLASSICAL ML
No data,
but we have
an environment
to interact with
REINFORCEMENT LEARNING
When quality is
a real problem
ENSEMBLES
eternal competitors
Complicated data
Unclear features
Belief in a miracle
NEURAL NETWORKS AND DEEP LEARNING

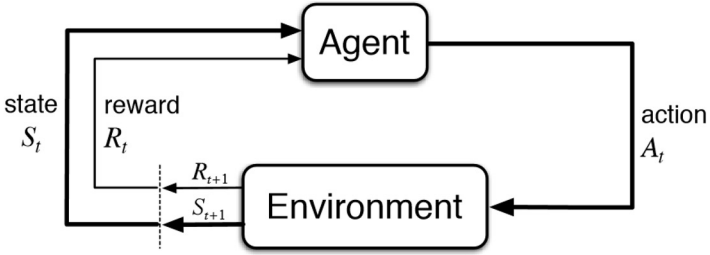
SORRY

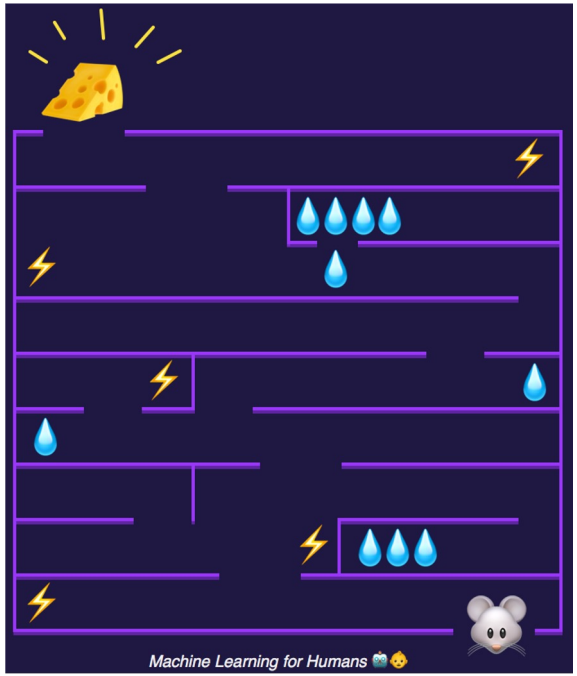
NO DATA FOUND



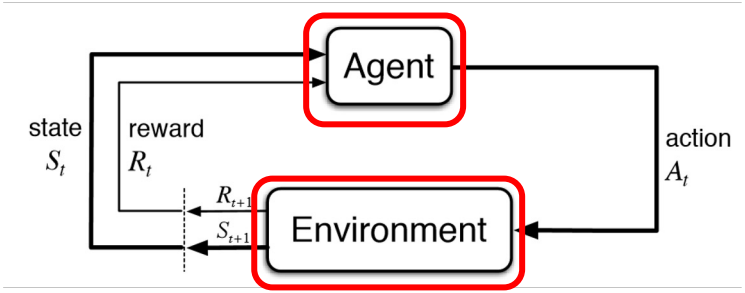

THE MAIN TYPES OF MACHINE LEARNING
Simple data
Clear features
CLASSICAL ML
No data,
but we have
an environment
to interact with
REINFORCEMENT LEARNING
When quality is
a real problem
ENSEMBLES
eternal competitors
Complicated data
Unclear features
Belief in a miracle
NEURAL NETWORKS AND DEEP LEARNING

Reinforcement Learning



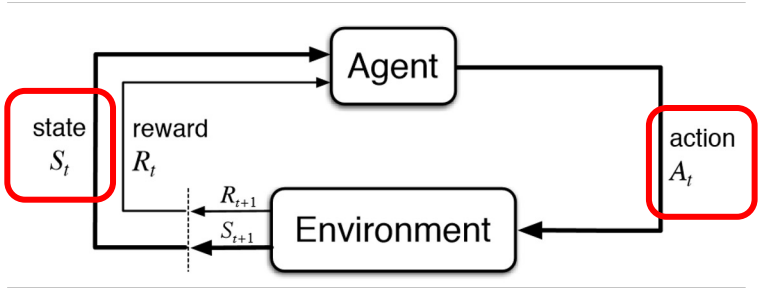


How can we apply RL algorithms in model repair?



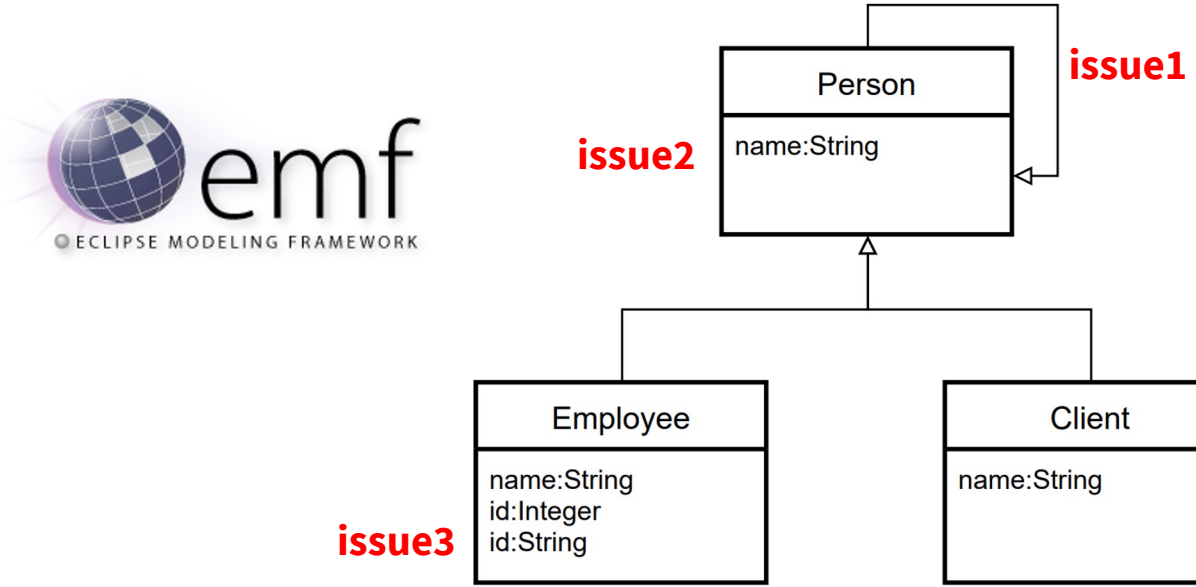
Agent: The RL algorithm.

Environment: The model to repair.



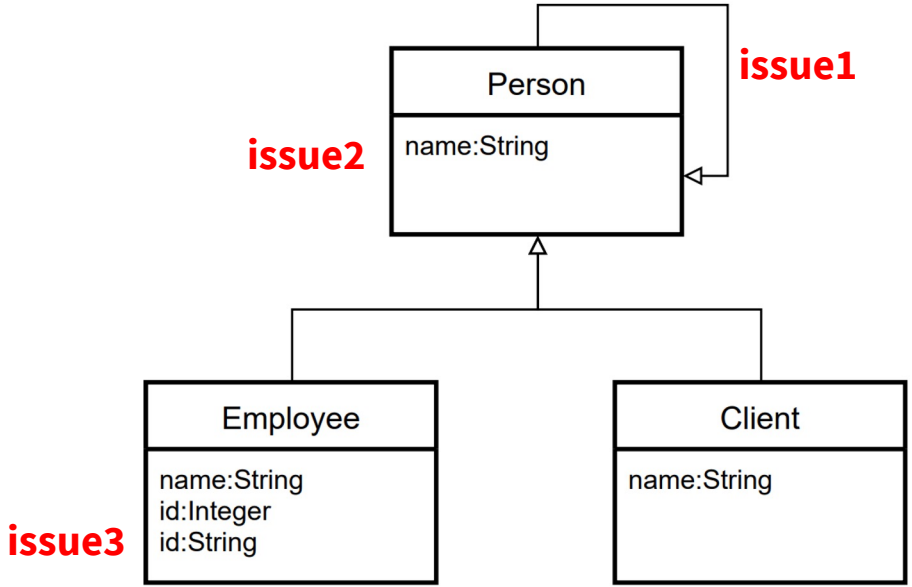
State space: The state space is defined by the set of issues present in the model.

Action space: The set of editing actions able to repair a model.

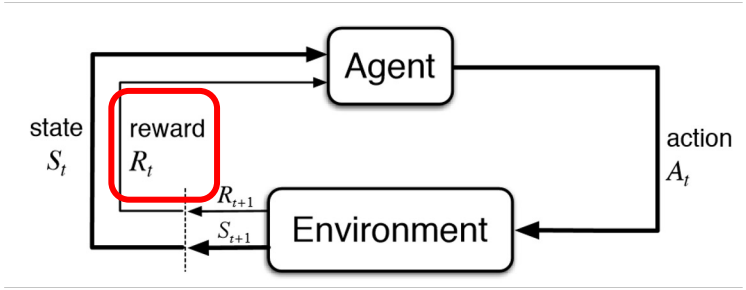


Initial state: {issue1, issue2, issue3}

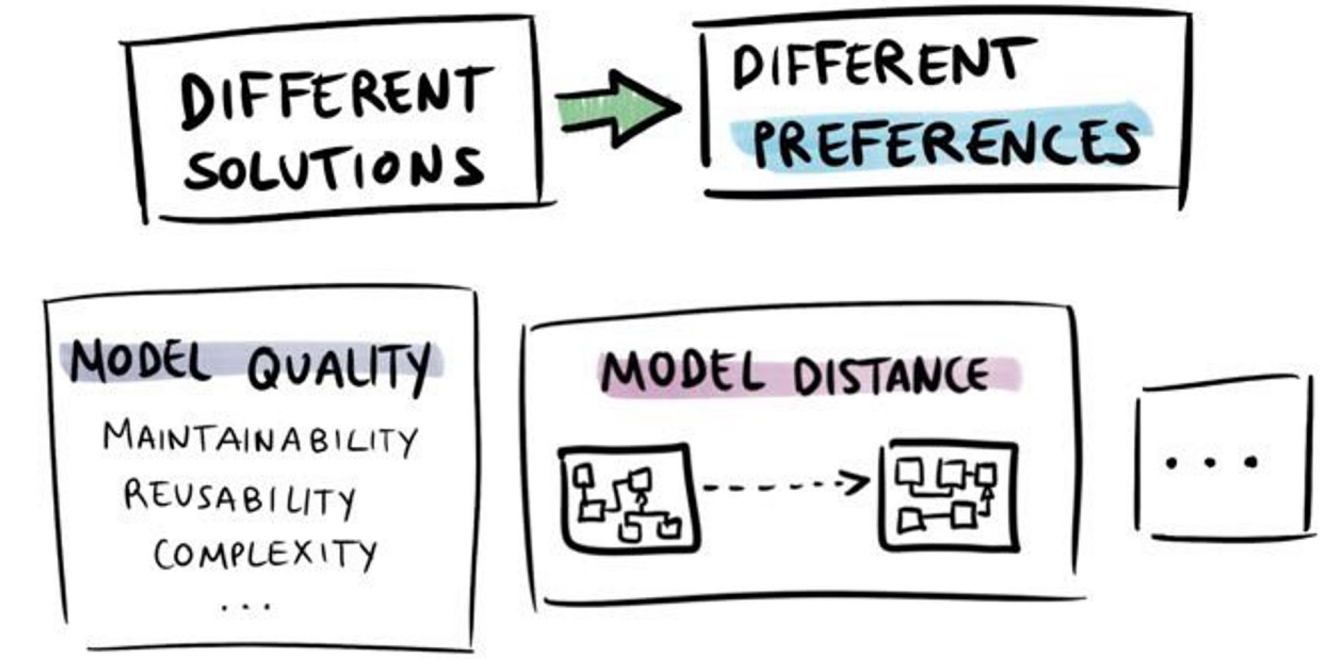
Final state: {}

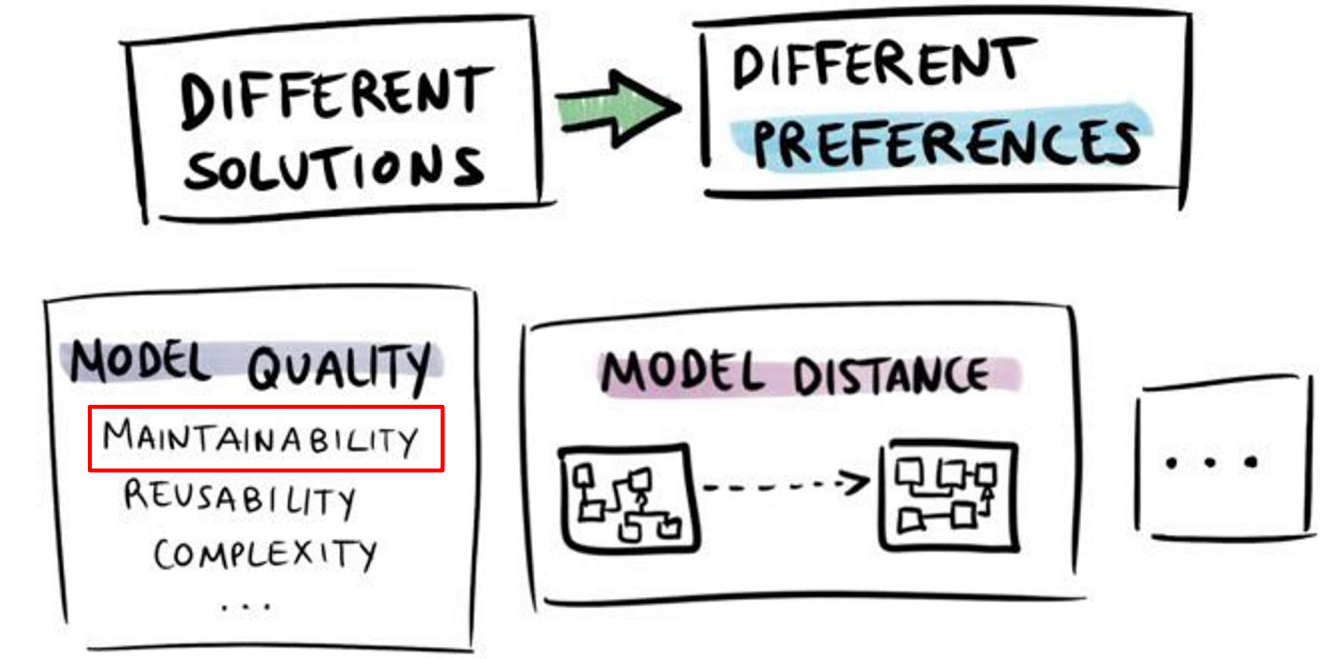


Action space: deleteReference | Class | Attribute, setName(*name*), addReference | Class | Attribute

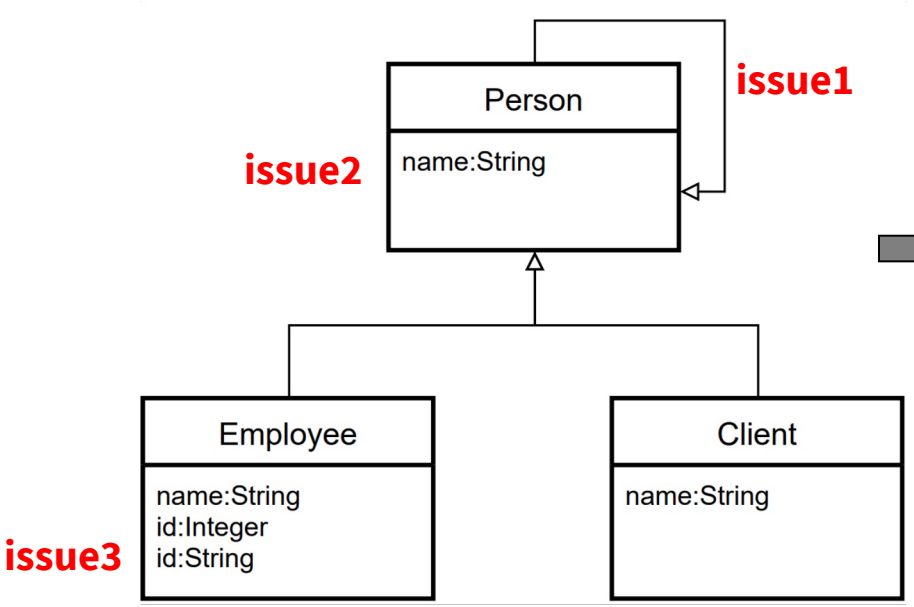


Reward: Rewards can be adapted to align with user preferences to personalize the repair result.


DIFFERENT SOLUTIONS
DIFFERENT PREFERENCES
MODEL QUALITY
MAINTAINABILITY
REUSABILITY
COMPLEXITY
...
MODEL DISTANCE
...


DIFFERENT SOLUTIONS
DIFFERENT PREFERENCES
MODEL QUALITY
MAINTAINABILITY
REUSABILITY
COMPLEXITY
...
MODEL DISTANCE
...

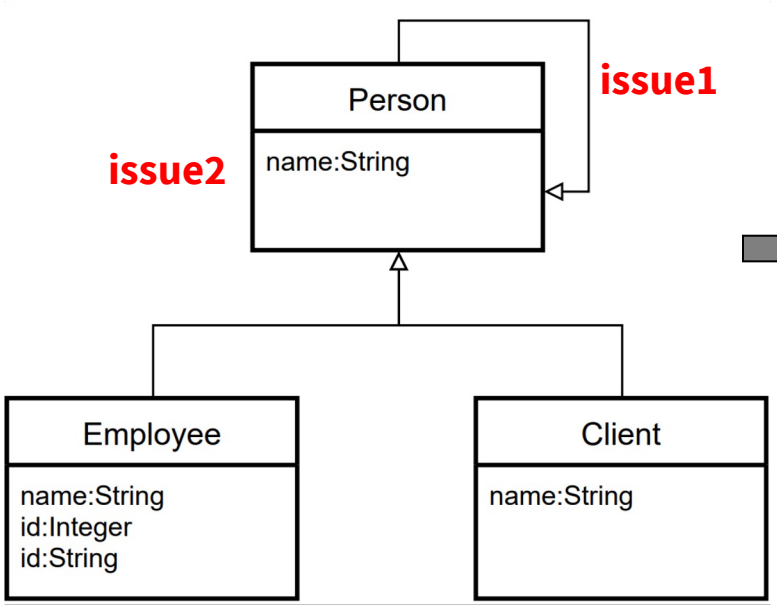
Maintainability



Q-table

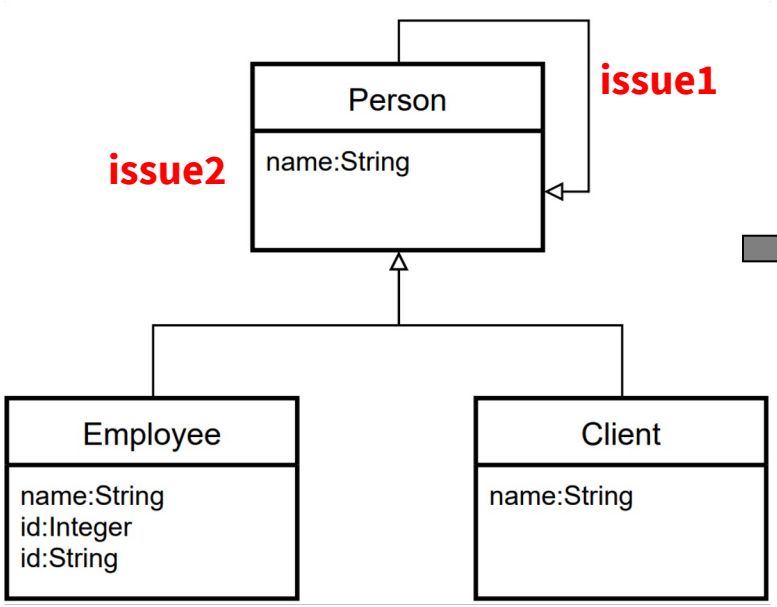
Entries	

Maintainability



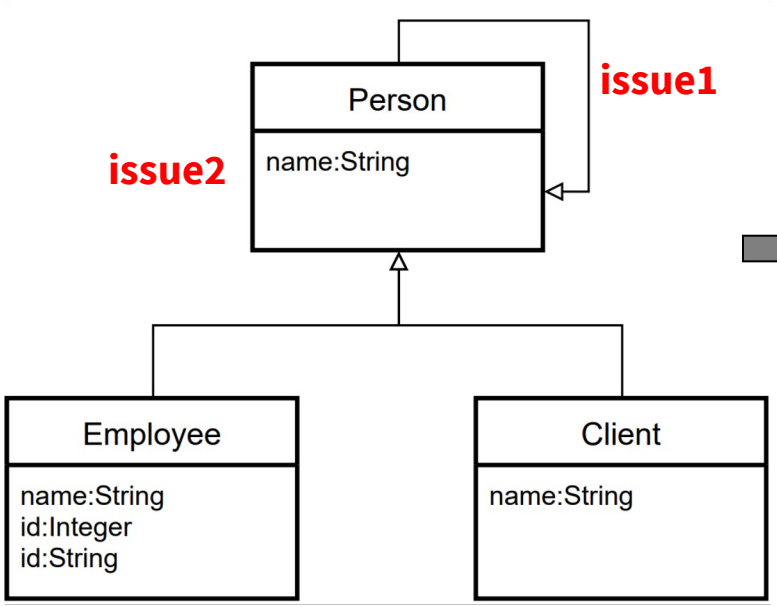
Entries	
entry1: issue1, deleteReference	
entry2: issue1, addClass	

Maintainability



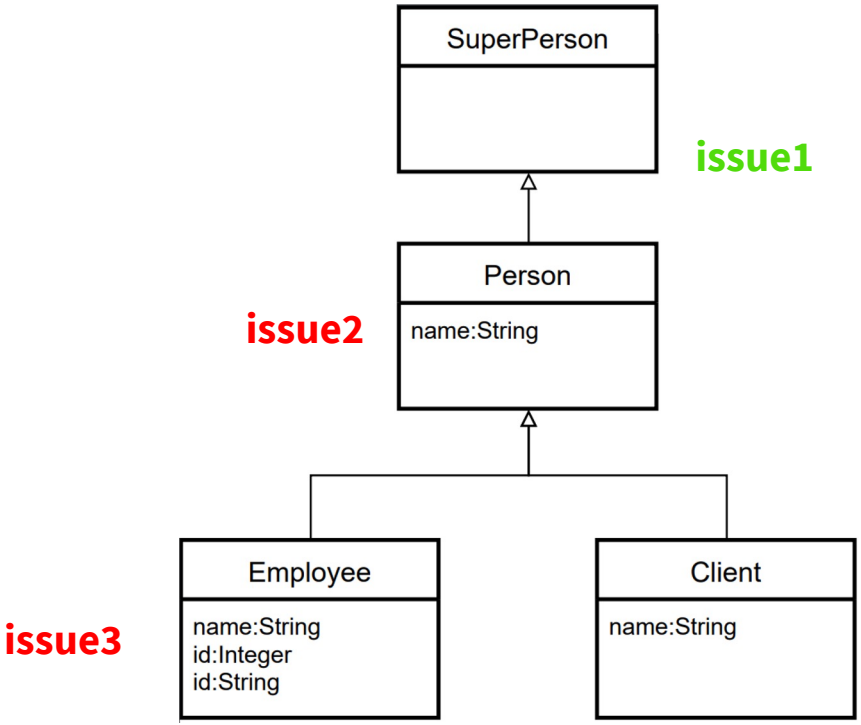
Entries	Q-value
entry1: issue1, deleteReference	
entry2: issue1, addClass	

Maintainability



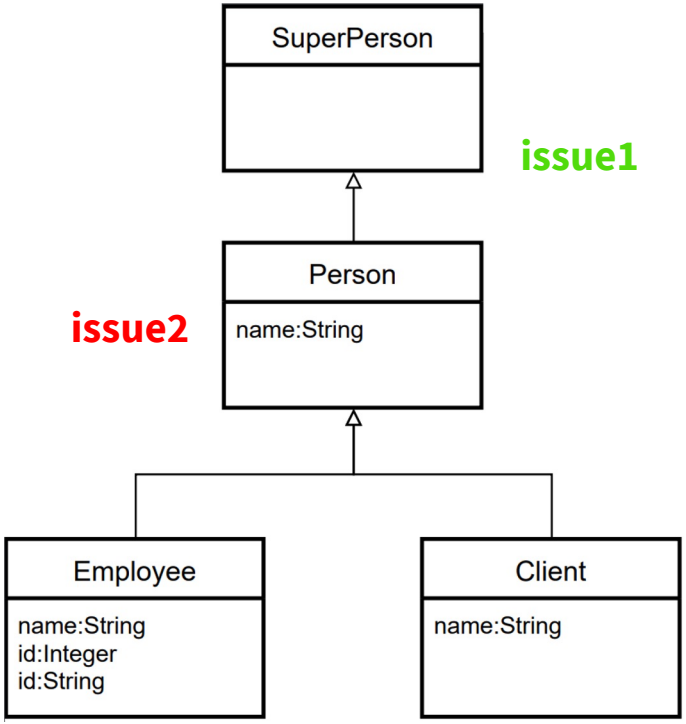
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	-500

Maintainability



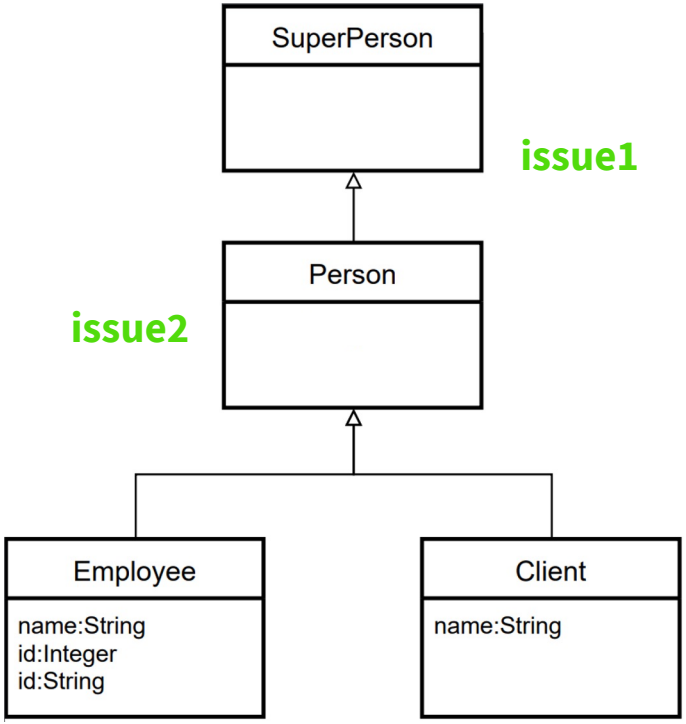
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	0

Maintainability



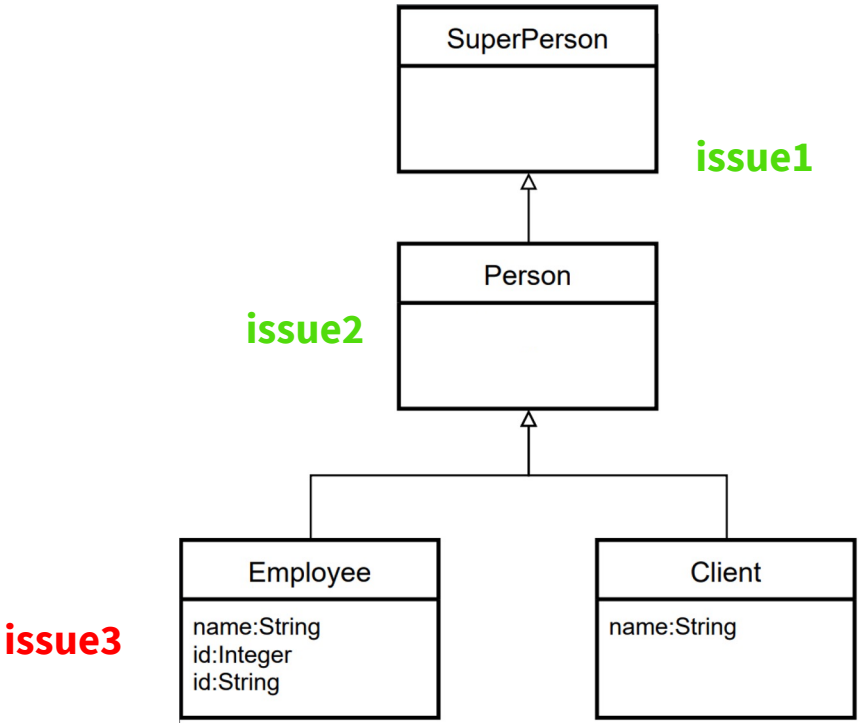
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	0
entry3: issue2, deleteAttribute(superClass)	-500
entry4: issue2, deleteAttribute(children)	-500

Maintainability



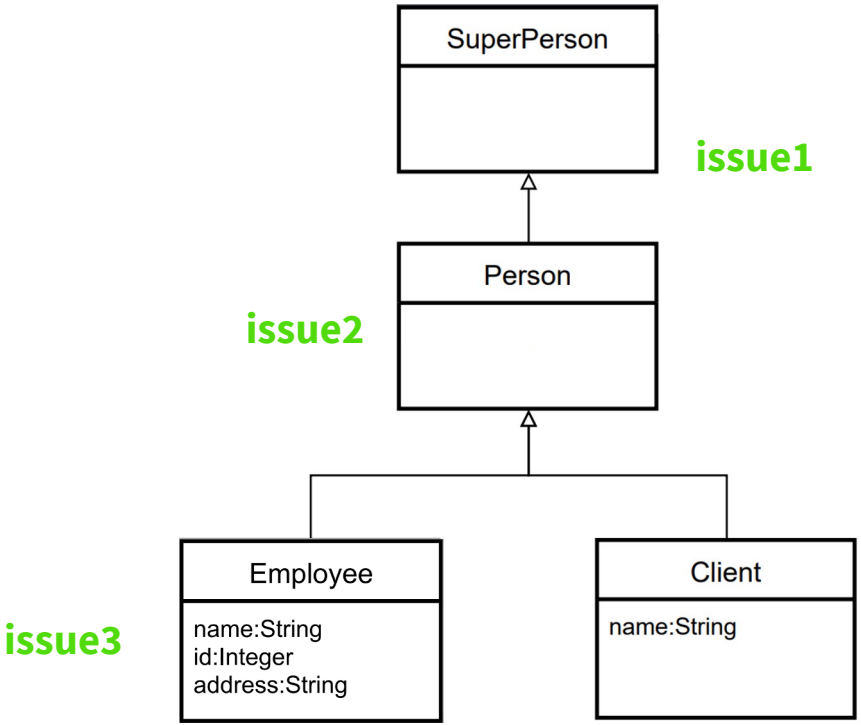
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	0
entry3: issue2, deleteAttribute(superClass)	0
entry4: issue2, deleteAttribute(children)	-500

Maintainability

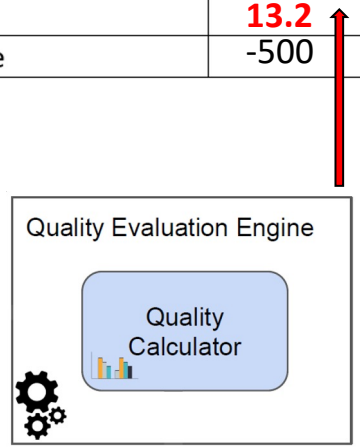


Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	0
entry3: issue2, deleteAttribute(superClass)	0
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	-500
entry6: issue3, deleteAttribute	-500

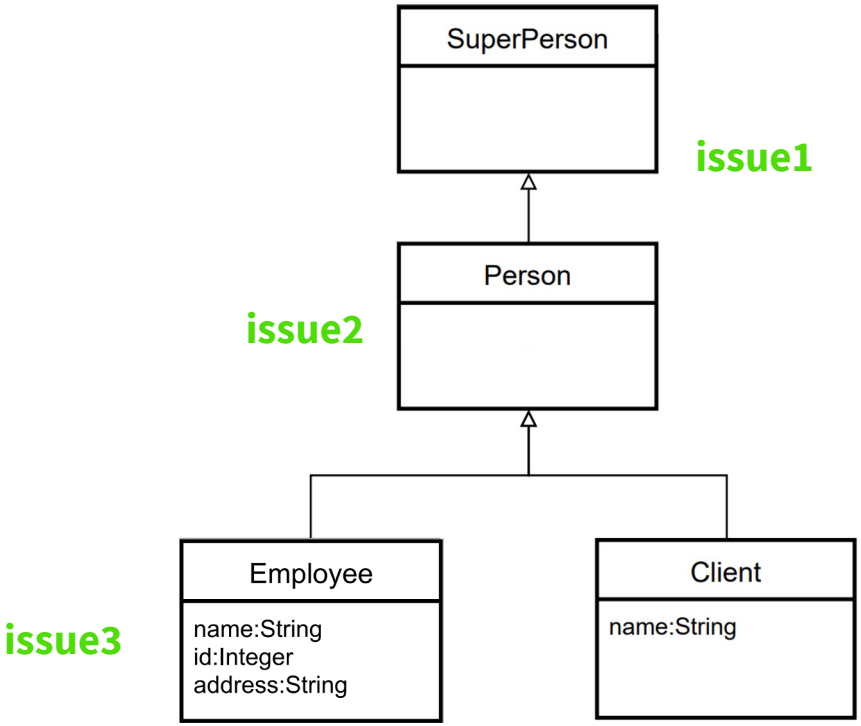
Maintainability



Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	**0**
entry3: issue2, deleteAttribute(superClass)	**0**
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	**13.2**
entry6: issue3, deleteAttribute	-500



Maintainability



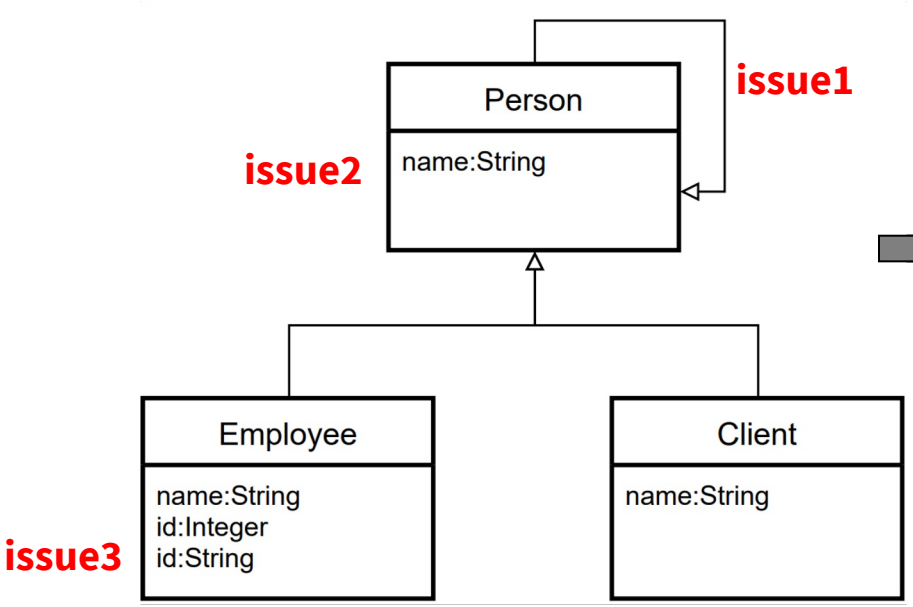
Repaired model #1

Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	**0**
entry3: issue2, deleteAttribute(superClass)	**0**
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	**13.2**
entry6: issue3, deleteAttribute	-500

Sequence #1:

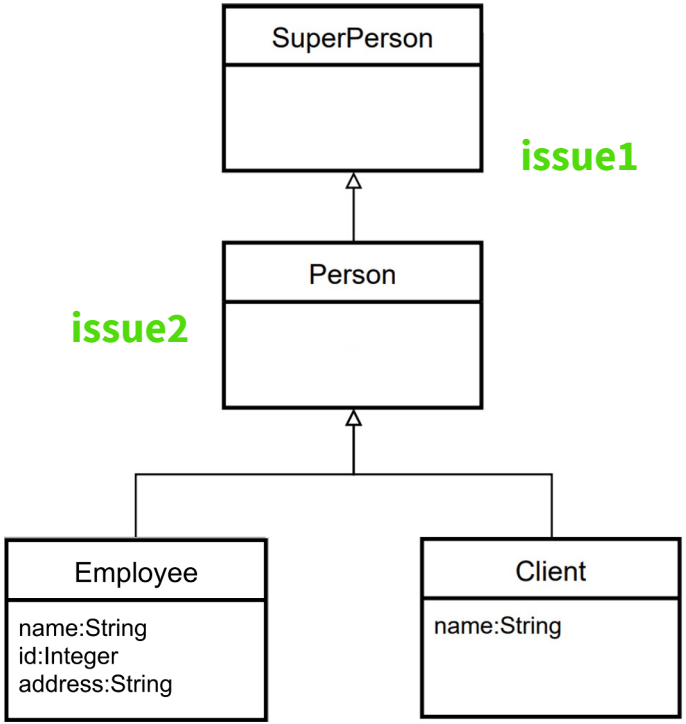
{[issue1, addClass],[issue2, deleteAttribute(superClass)], [issue3, setName]}

Maintainability



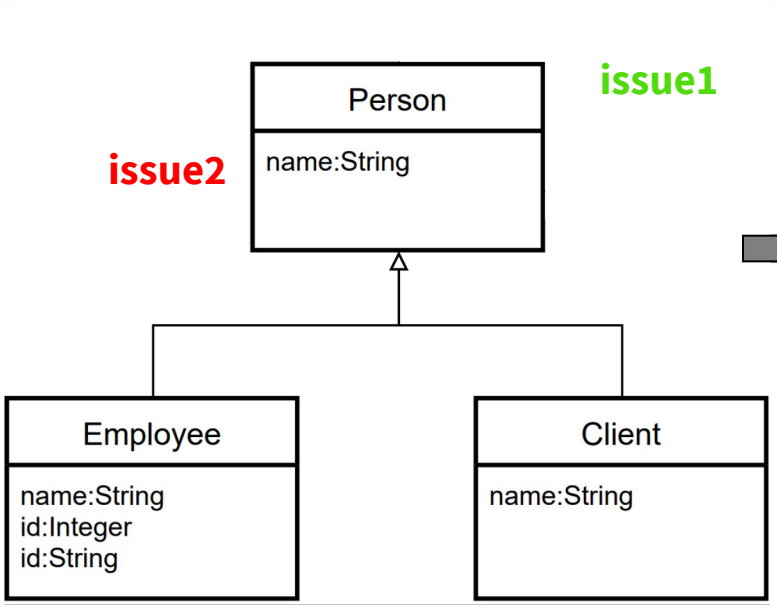
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	**0**
entry3: issue2, deleteAttribute(superClass)	**0**
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	**13.2**
entry6: issue3, deleteAttribute	-500

Maintainability



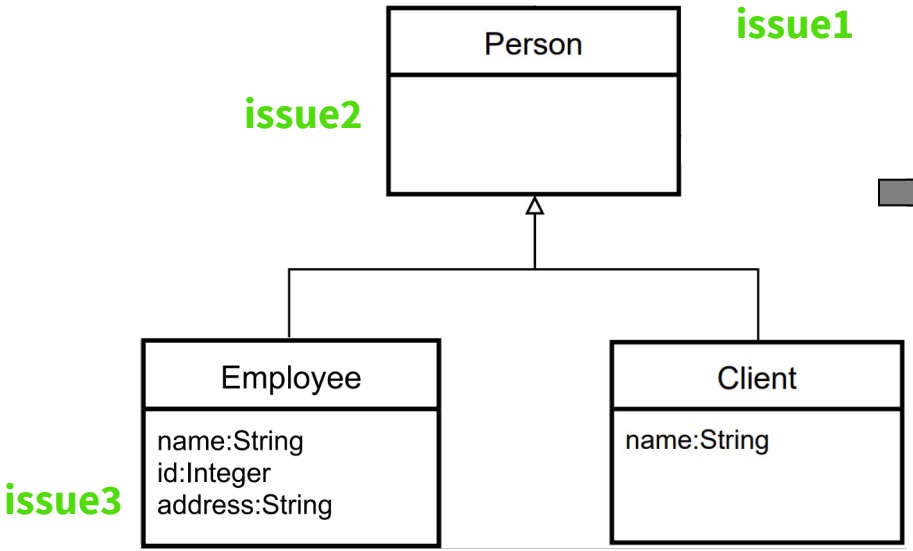
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	**7.8**
entry3: issue2, deleteAttribute(superClass)	**8.4**
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	**26.4**
entry6: issue3, deleteAttribute	-500

Maintainability



Entries	Q-value
entry1: issue1, deleteReference	**18.9**
entry2: issue1, addClass	7.8
entry3: issue2, deleteAttribute(superClass)	8.4
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	26.4
entry6: issue3, deleteAttribute	-500

Maintainability



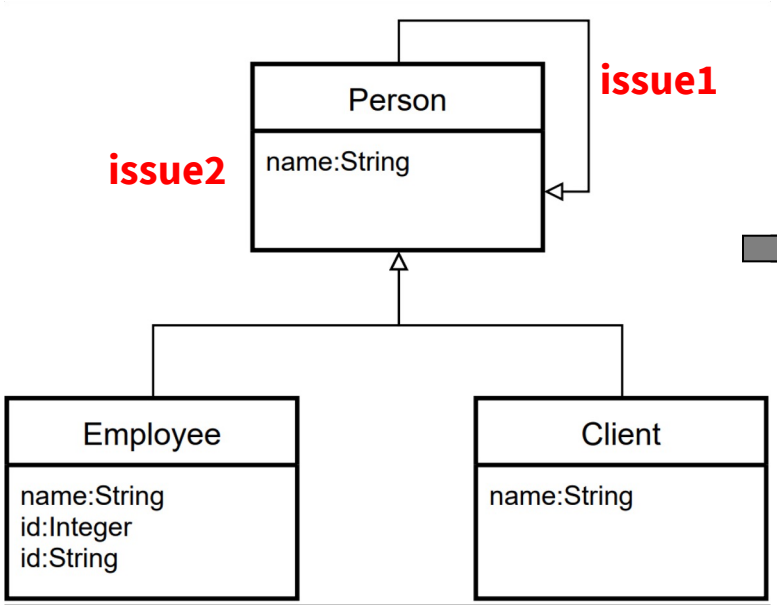
Entries	Q-value
entry1: issue1, deleteReference	**18.9**
entry2: issue1, addClass	7.8
entry3: issue2, deleteAttribute(superClass)	**12.5**
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	**39.6**
entry6: issue3, deleteAttribute	-500

Repaired model #2

Sequence #2:

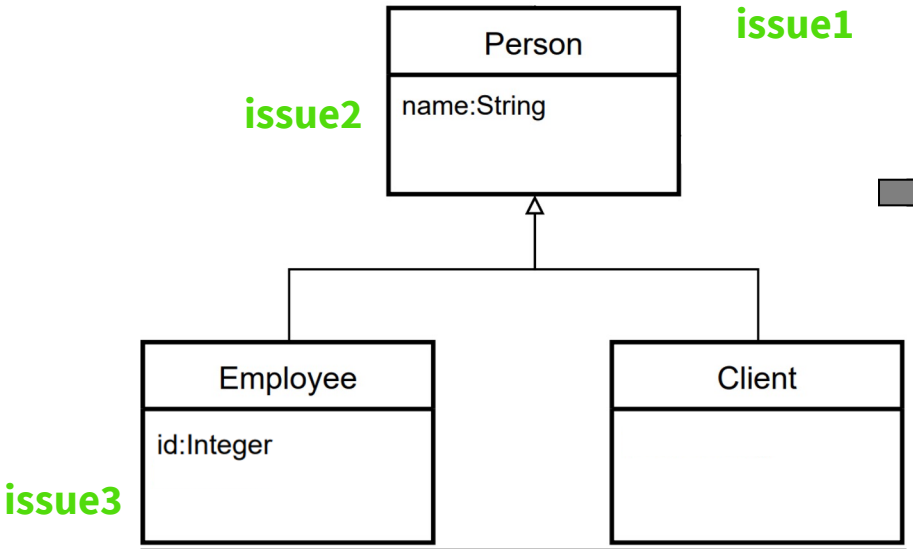
{[issue1, deleteReference], [issue2, deleteAttribute(superClass)], [issue3, setName]}

Maintainability



Entries	Q-value
entry1: issue1, deleteReference	**18.9**
entry2: issue1, addClass	7.8
entry3: issue2, deleteAttribute(superClass)	**12.5**
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	**39.6**
entry6: issue3, deleteAttribute	-500

Maintainability



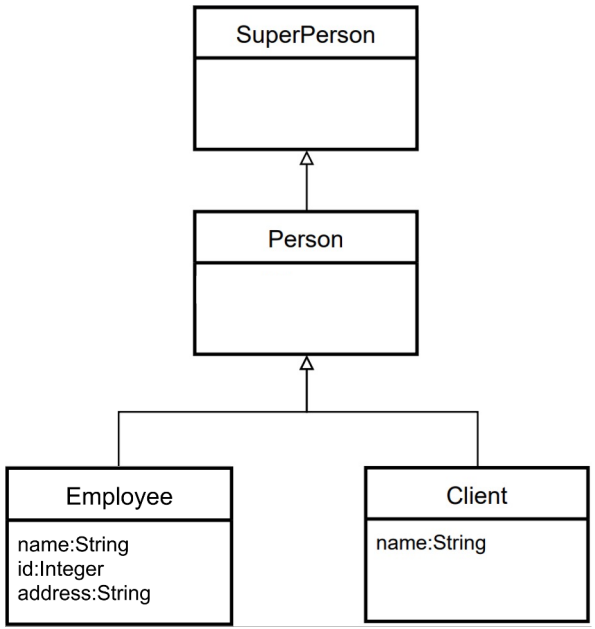
Entries	Q-value
entry1: issue1, deleteReference	**37.8**
entry2: issue1, addClass	7.8
entry3: issue2, deleteAttribute(superClass)	12.5
entry4: issue2, deleteAttribute(children)	**23.4**
entry5: issue3, setName	39.6
entry6: issue3, deleteAttribute	**45.7**

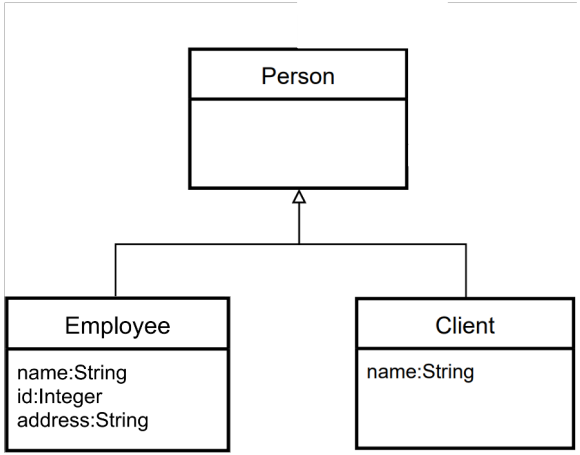
Repaired model #3

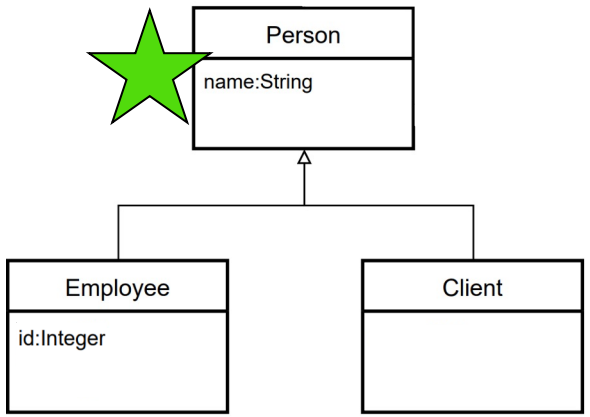
Sequence #3:

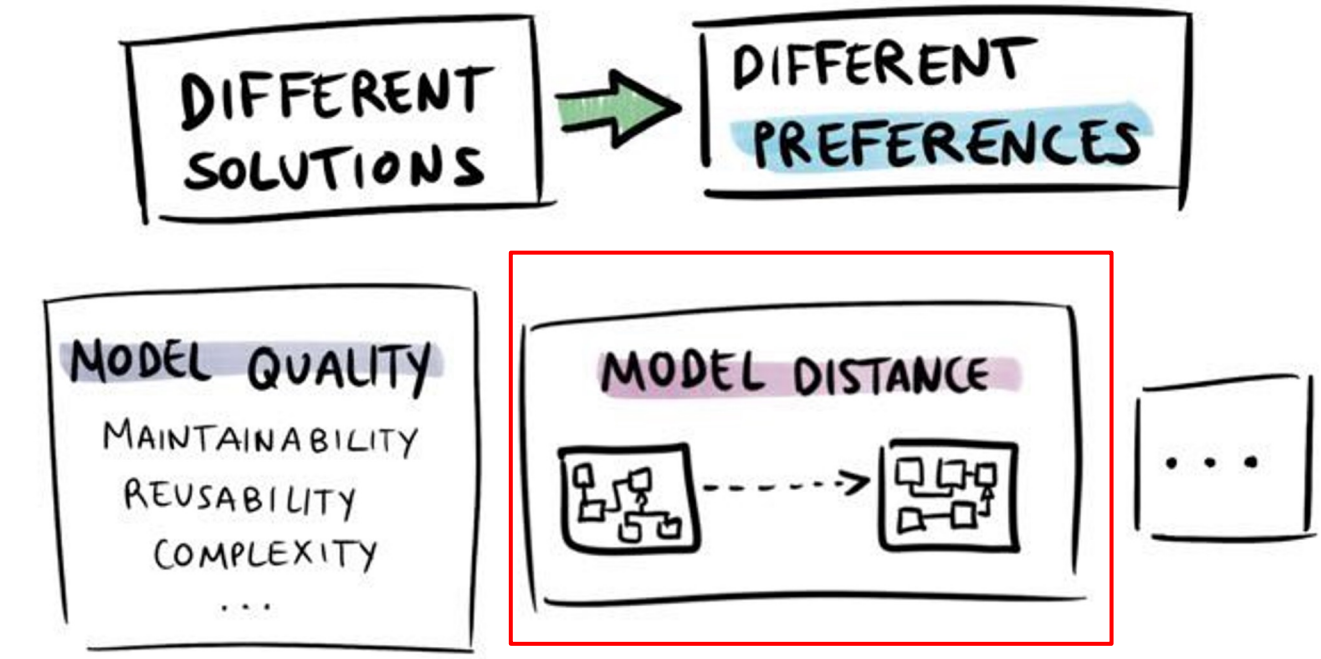
{[issue1, deleteReference], [issue2, deleteAttribute(children)], [issue3, deleteAttribute]}

Best model w.r.t. maintainability

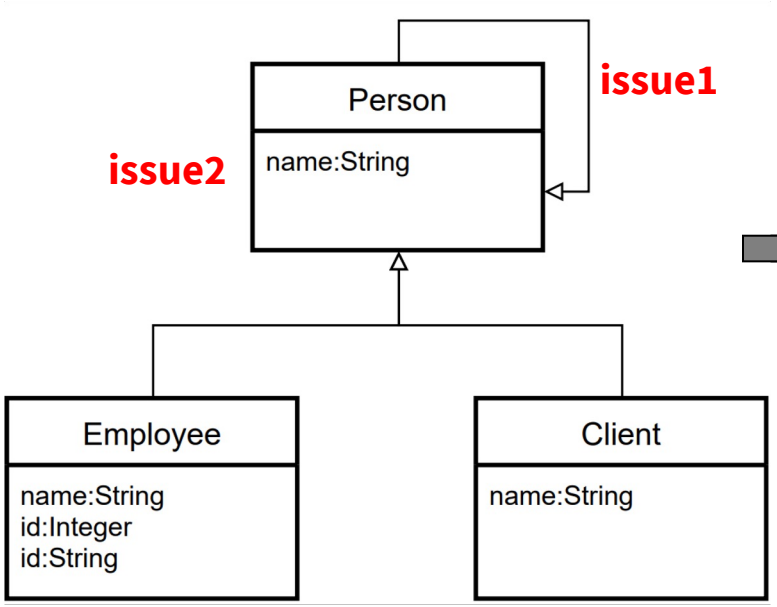






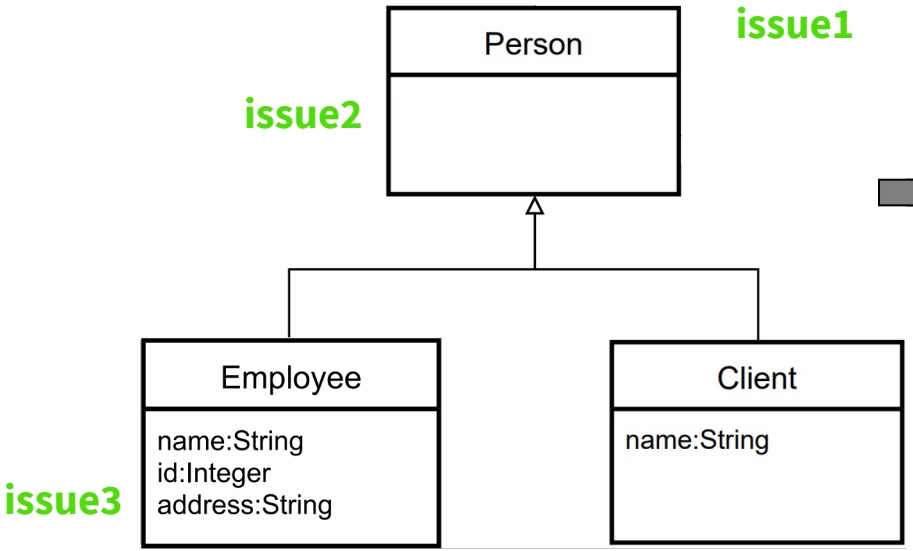

DIFFERENT SOLUTIONS
DIFFERENT PREFERENCES
MODEL QUALITY
MAINTAINABILITY
REUSABILITY
COMPLEXITY
...
MODEL DISTANCE
...

Model distance



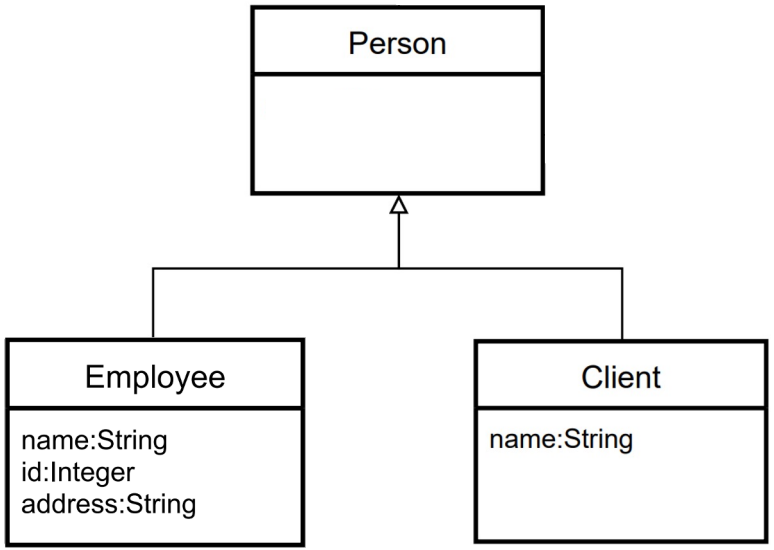
Entries	Q-value
entry1: issue1, deleteReference	-500
entry2: issue1, addClass	-500
entry3: issue2, deleteAttribute(superClass)	-500
entry4: issue2, deleteAttribute(children)	-500
entry5: issue3, setName	-500
entry6: issue3, deleteAttribute	-500

Model distance

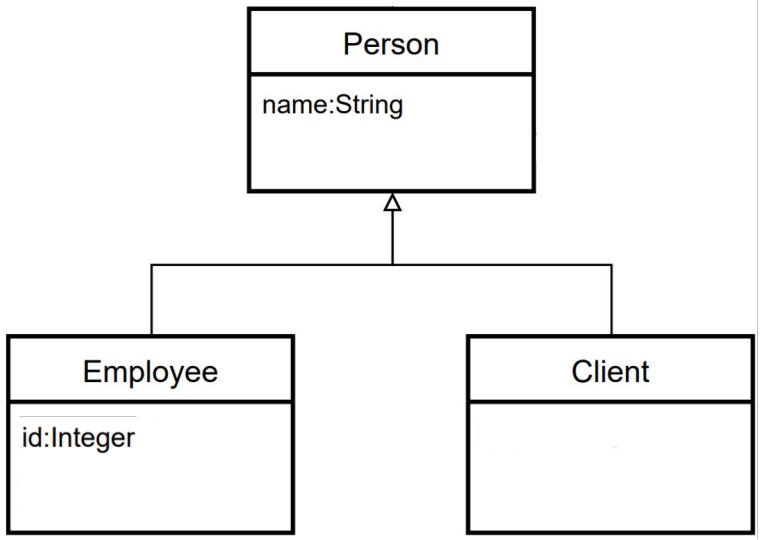


Entries	Q-value
entry1: issue1, deleteReference	**53.8**
entry2: issue1, addClass	10.2
entry3: issue2, deleteAttribute(superClass)	**36.5**
entry4: issue2, deleteAttribute(children)	15.6
entry5: issue3, setName	**68.6**
entry6: issue3, deleteAttribute	18.6

Model distance



Maintanibility

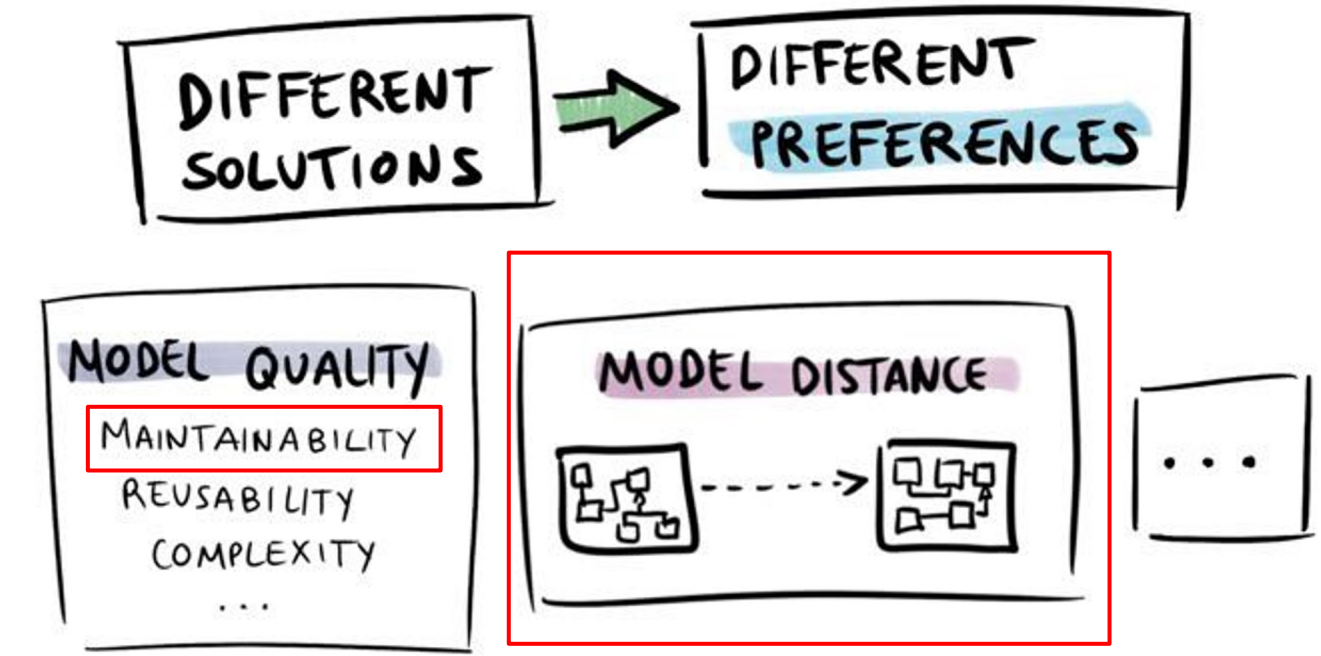


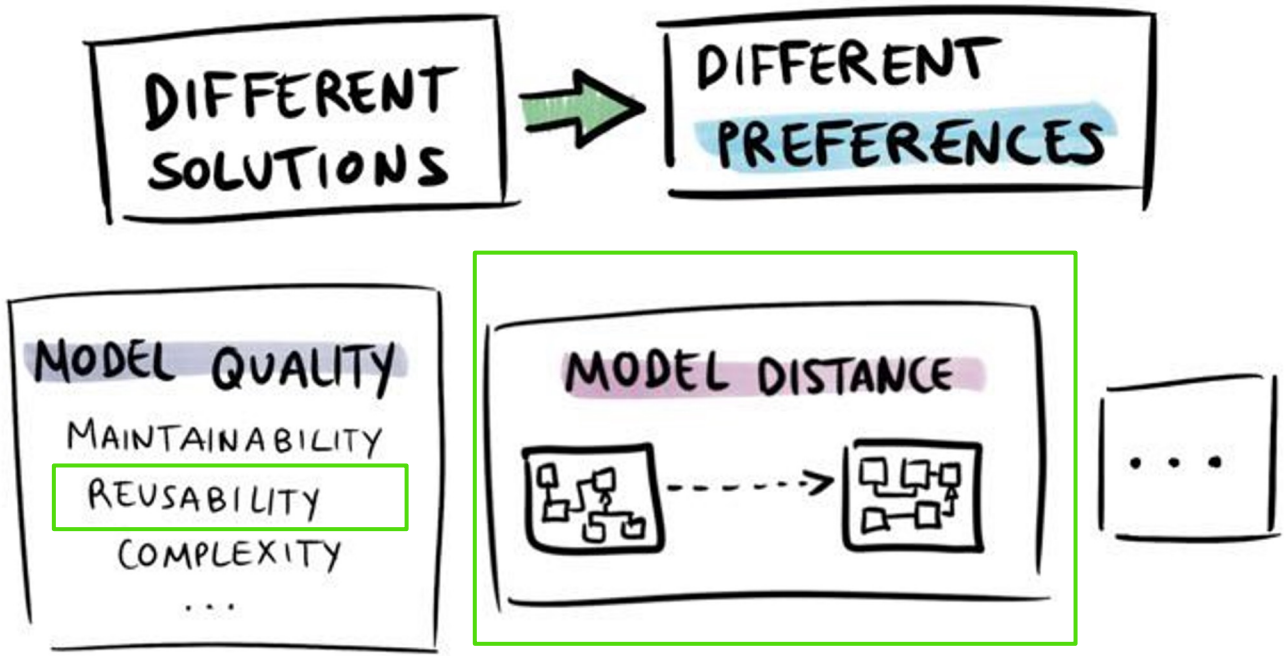
Too personalized… learning and automation?

Is the Q-table reusable?

It depends...

		Preferences	
		Same	Different
Models	Same	✓	✗
	Different	✓	✗


DIFFERENT SOLUTIONS
DIFFERENT PREFERENCES
MODEL QUALITY
MAINTAINABILITY
REUSABILITY
COMPLEXITY
...
MODEL DISTANCE
...


DIFFERENT SOLUTIONS
DIFFERENT PREFERENCES
MODEL QUALITY
MAINTAINABILITY
REUSABILITY
COMPLEXITY
...
MODEL DISTANCE
...

Maintainability AND Model distance

Entries	Q-value
★ entry1: issue1, deleteReference	**37.8**
entry2: issue1, addClass	7.8
entry3: issue2, deleteAttribute(superClass)	12.5
★ entry4: issue2, deleteAttribute(children)	**23.4**
entry5: issue3, setName	39.6
★ entry6: issue3, deleteAttribute	**45.7**

Entries	Q-value
★ entry1: issue1, deleteReference	**53.8**
entry2: issue1, addClass	10.2
★ entry3: issue2, deleteAttribute(superClass)	**36.5**
entry4: issue2, deleteAttribute(children)	15.6
★ entry5: issue3, setName	**68.6**
entry6: issue3, deleteAttribute	18.6

In isolation vs together

Entries	Q-value
★ entry1: issue1, deleteReference	**67.8**
entry2: issue1, addClass	26.5
entry3: issue2, deleteAttribute(superClass)	43.5
★ entry4: issue2, deleteAttribute(children)	**47.4**
★ entry5: issue3, setName	**72.4**
entry6: issue3, deleteAttribute	34.7

Maintainability AND Model distance

Entries	Q-value	Maint	Distance
entry1: issue1, deleteReference	67.8	**36.2**	**31.6**
entry2: issue1, addClass	26.5		
entry3: issue2, deleteAttribute(superClass)	43.5		
entry4: issue2, deleteAttribute(children)	47.4		
entry5: issue3, setName	72.4		
entry6: issue3, deleteAttribute	34.7		

Qvalue decomposition

Maintainability AND Model distance

Entries	Q-value	Maint	Distance
entry1: issue1, deleteReference	67.8	**36.2**	31.6
entry2: issue1, addClass	26.5	**13.5**	13
entry3: issue2, deleteAttribute(superClass)	43.5	11.1	**32.4**
entry4: issue2, deleteAttribute(children)	47.4	**35.2**	12.2
entry5: issue3, setName	72.4	30.3	**42.1**
entry6: issue3, deleteAttribute	34.7	**22.3**	12.4

Transfer learning

Entries	Q-value	Maint	Distance
entry1: issue1, deleteReference	67.8	**36.2**	31.6
entry2: issue1, addClass	26.5	**13.5**	13
entry3: issue2, deleteAttribute(superClass)	43.5	11.1	**32.4**
entry4: issue2, deleteAttribute(children)	47.4	**35.2**	12.2
entry5: issue3, setName	72.4	30.3	**42.1**
entry6: issue3, deleteAttribute	34.7	**22.3**	12.4

x 0.2

Entries	Q-value
entry1: issue1, deleteReference	
entry2: issue1, addClass	
entry3: issue2, deleteAttribute(superClass)	
entry4: issue2, deleteAttribute(children)	
entry5: issue3, setName	
entry6: issue3, deleteAttribute	

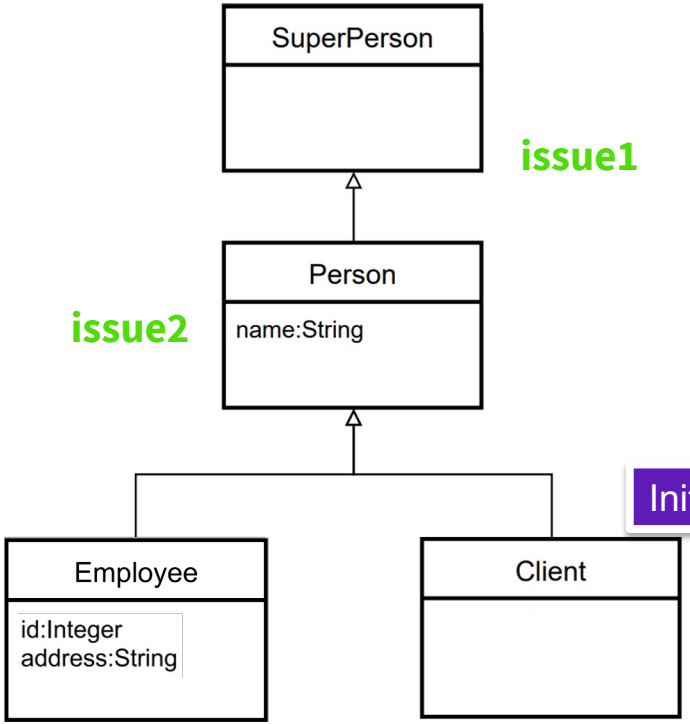
Transfer learning

Entries	Q-value	Maint	Distance
entry1: issue1, deleteReference	67.8	36.2	31.6
entry2: issue1, addClass	26.5	13.5	13
entry3: issue2, deleteAttribute(superClass)	43.5	11.1	32.4
entry4: issue2, deleteAttribute(children)	47.4	35.2	12.2
entry5: issue3, setName	72.4	30.3	42.1
entry6: issue3, deleteAttribute	34.7	22.3	12.4

x 0.2

Entries	Q-value
entry1: issue1, deleteReference	**6.3**
entry2: issue1, addClass	2.6
entry3: issue2, deleteAttribute(superClass)	**6.4**
entry4: issue2, deleteAttribute(children)	2.4
entry5: issue3, setName	**8.4**
entry6: issue3, deleteAttribute	2.4

Reusability and Model distance



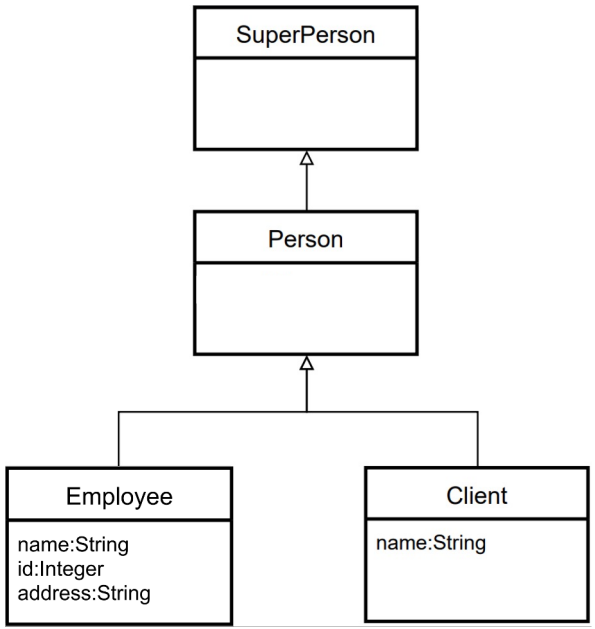
Entries	Q-value
entry1: issue1, deleteReference	**6.3**
entry2: issue1, addClass	2.6
entry3: issue2, deleteAttribute(superClass)	**6.4**
entry4: issue2, deleteAttribute(children)	2.4
entry5: issue3, setName	**8.4**
entry6: issue3, deleteAttribute	2.4

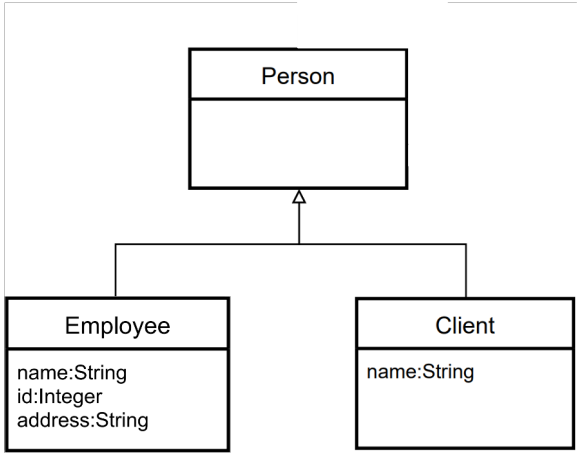
Initialized with model distance

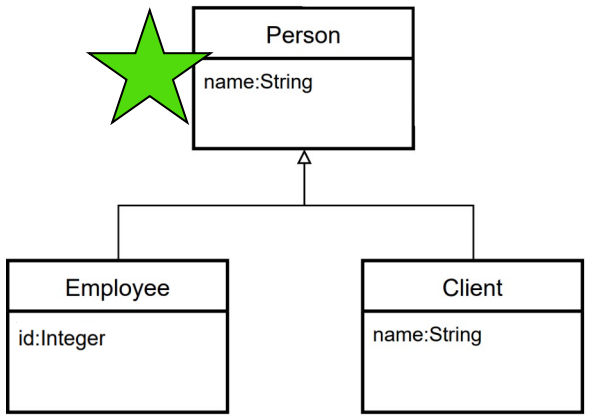
Entries	Q-value
entry1: issue1, deleteReference	47.8
★ entry2: issue1, addClass	**66.5**
entry3: issue2, deleteAttribute(superClass)	43.5
★ entry4: issue2, deleteAttribute(children)	**47.4**
★ entry5: issue3, setName	**68.4**
entry6: issue3, deleteAttribute	34.7

What if preferences are not enough?

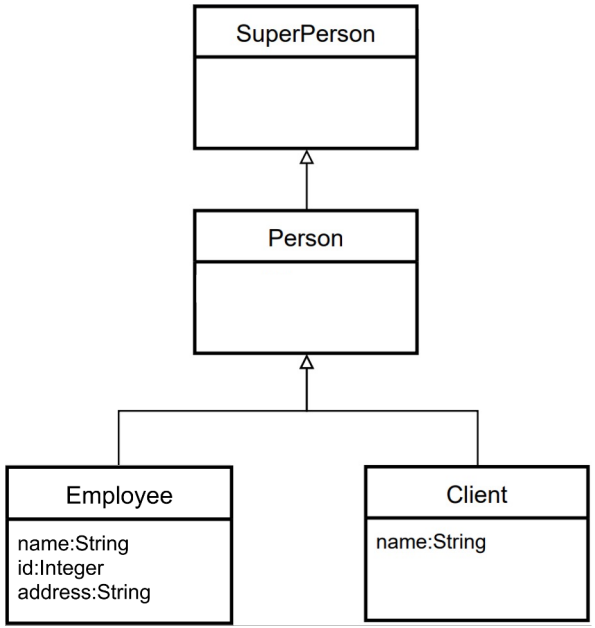
Maintainability

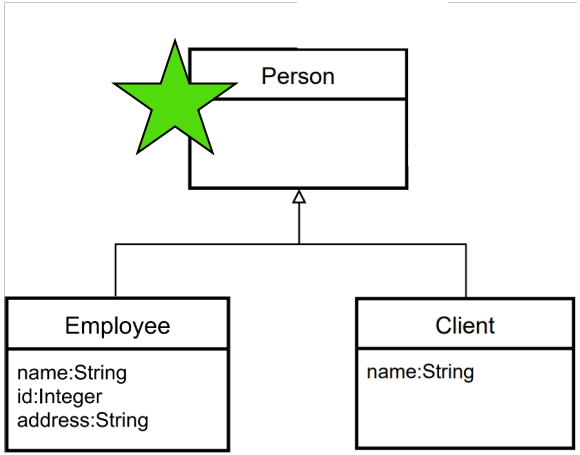


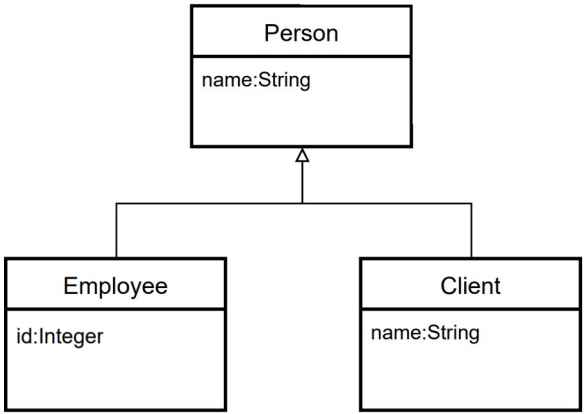




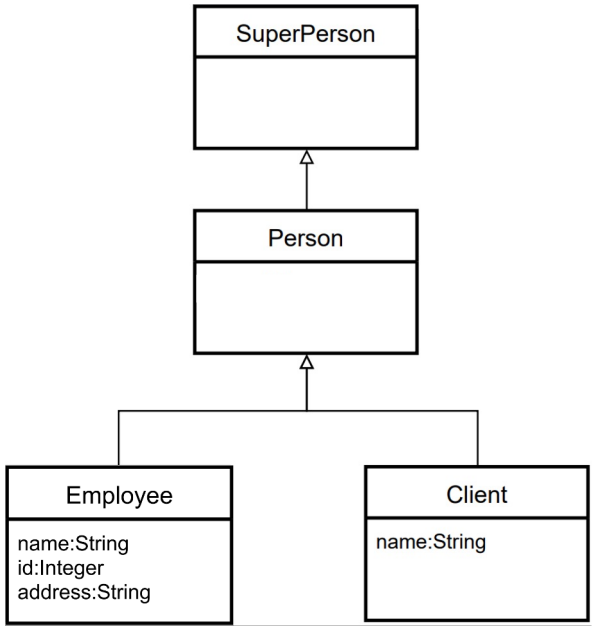
Maintainability - user feedback

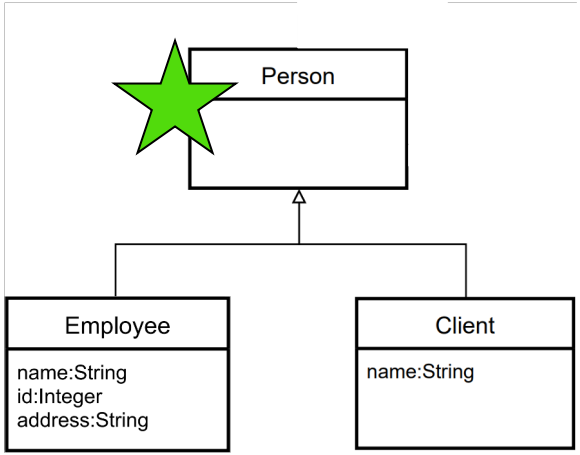


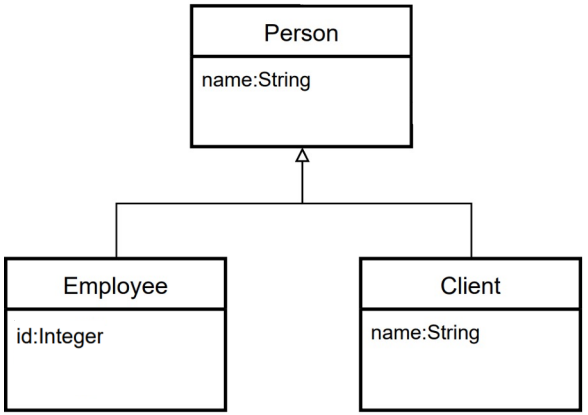




Maintainability - user feedback



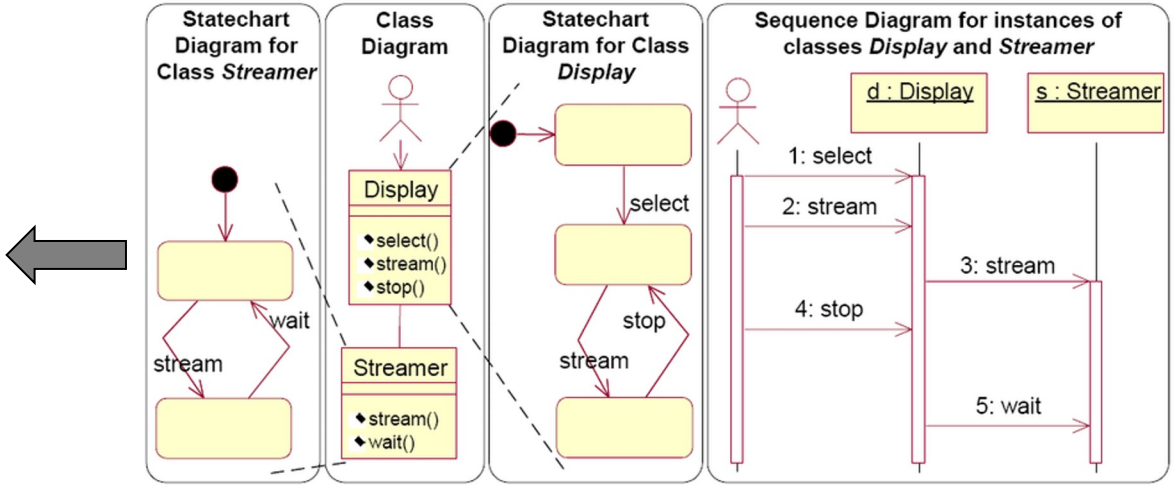




Extra rewards to selected sequence.

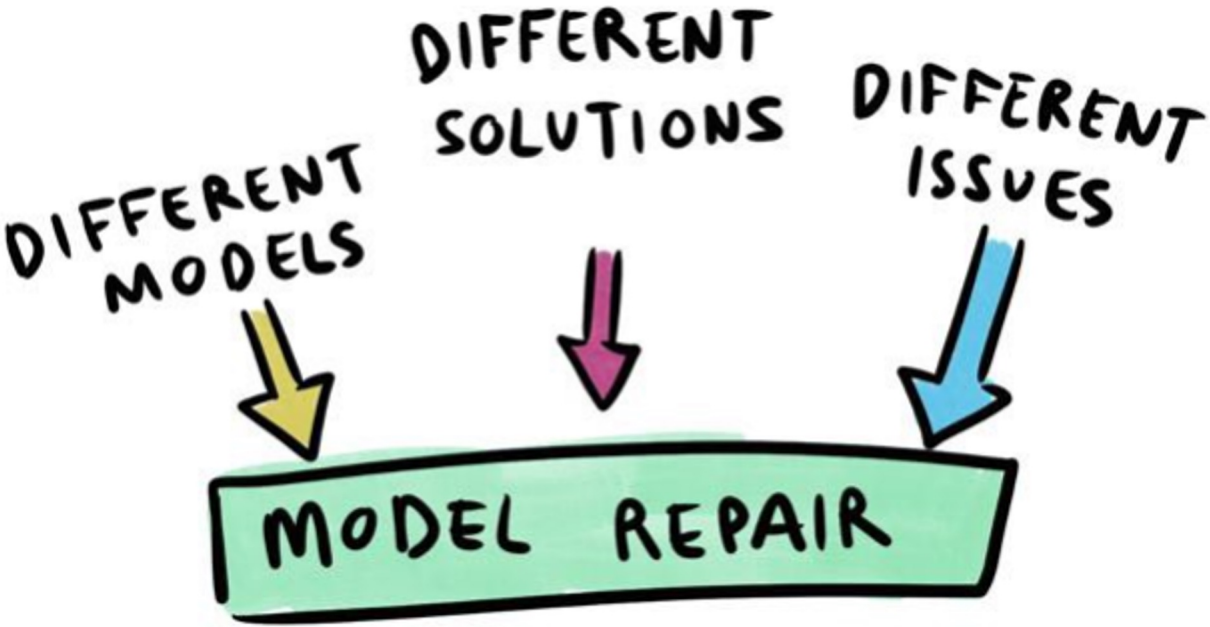
Multimodeling case

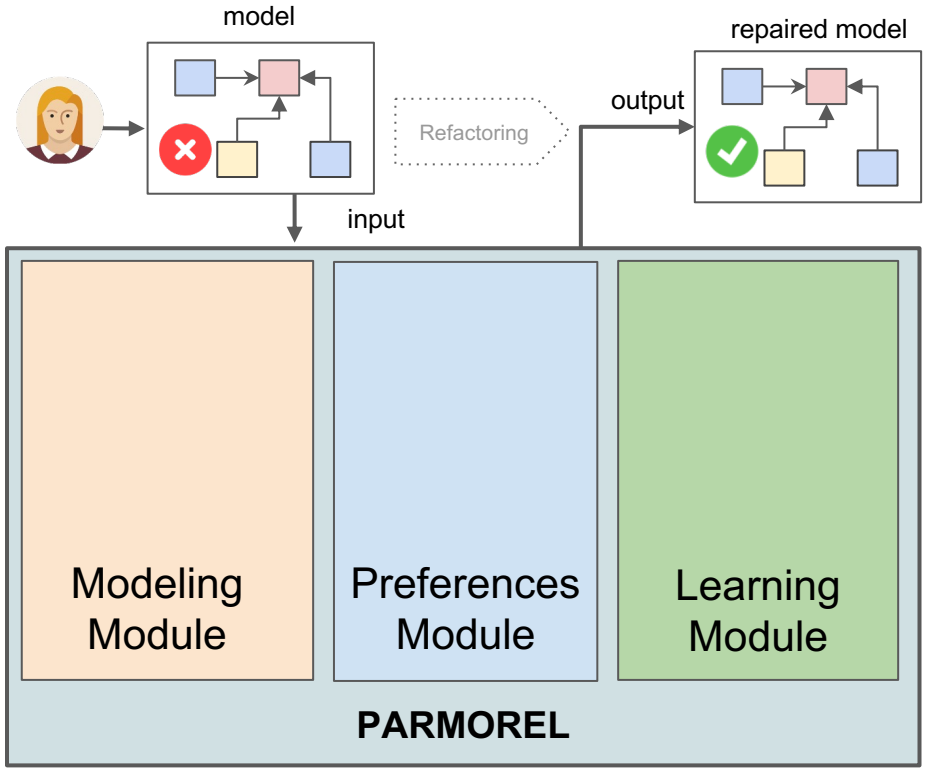
Parmorel fills the gaps, but does it make sense?

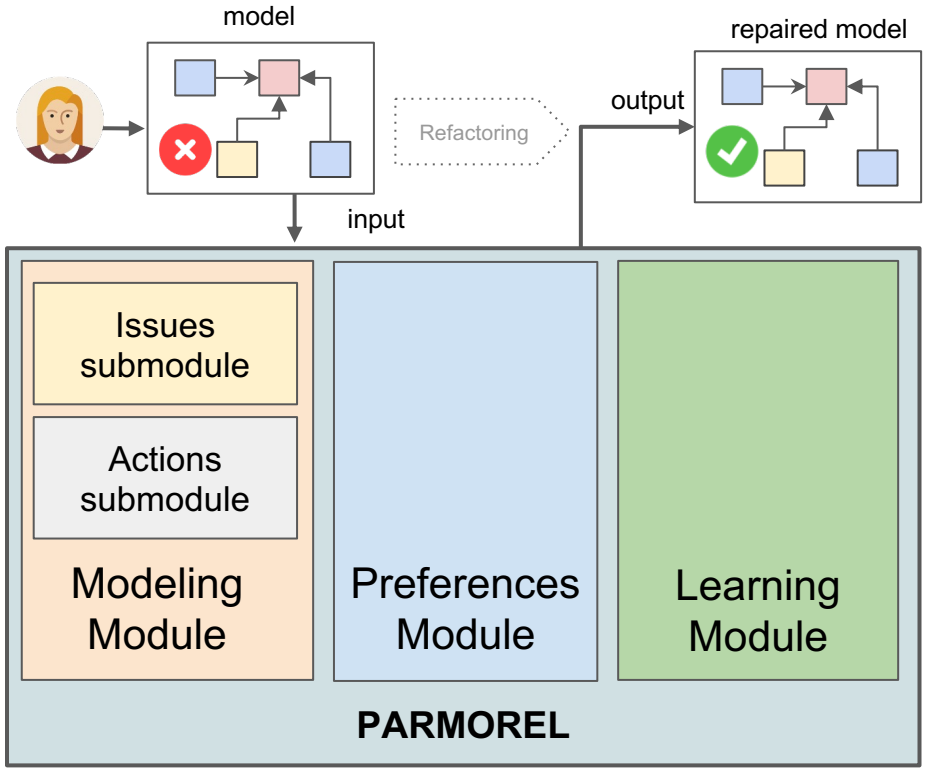


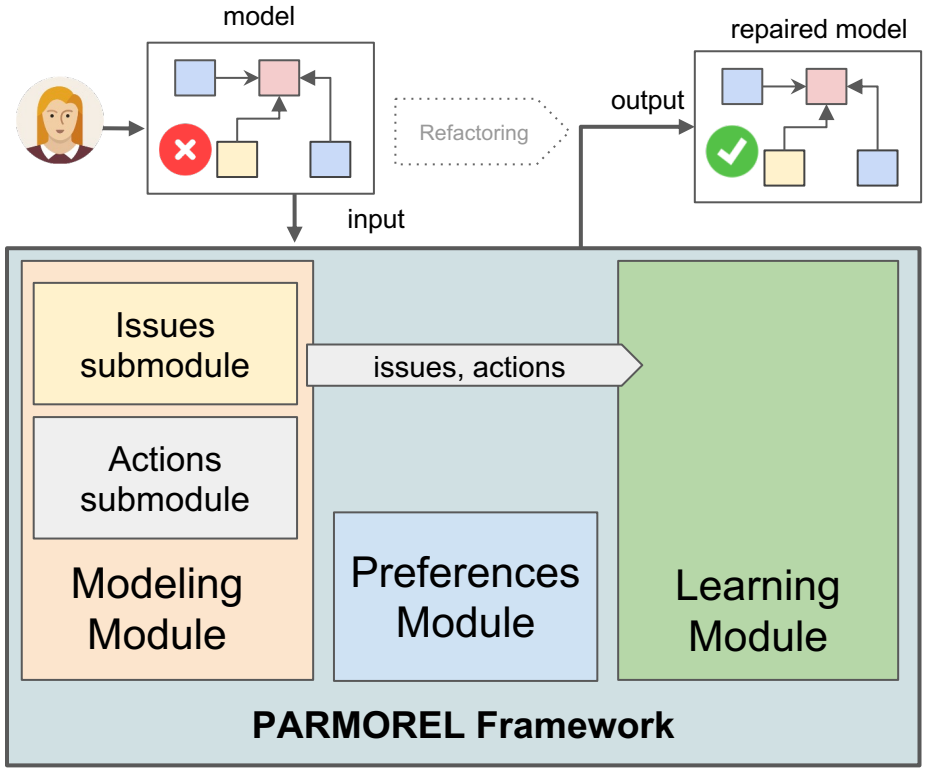
The extensible framework

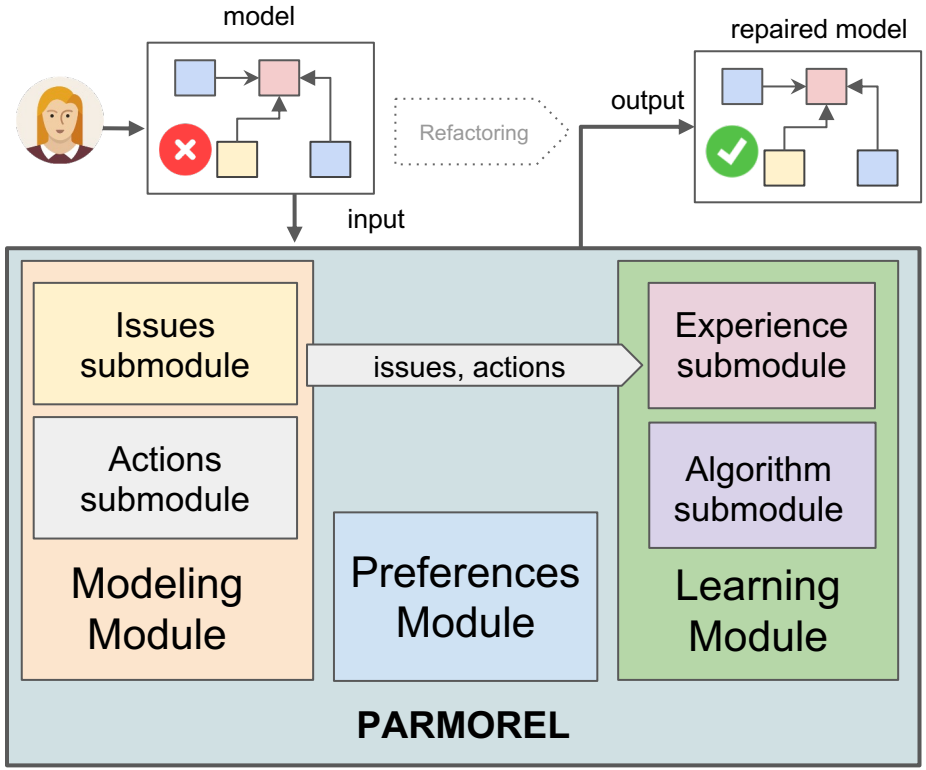
The challenge

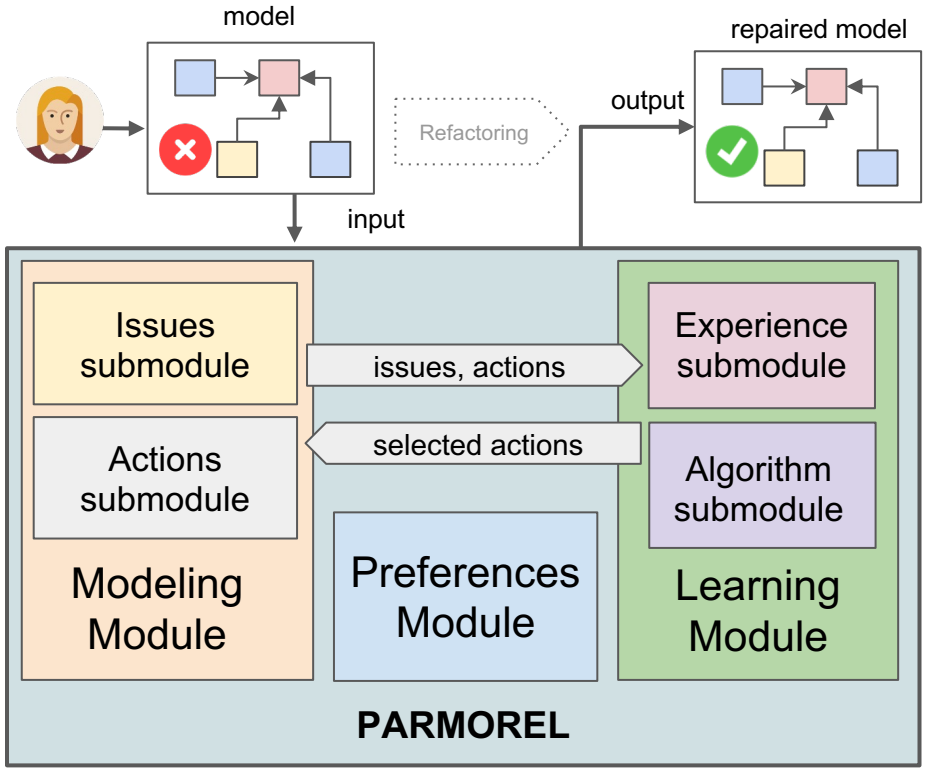


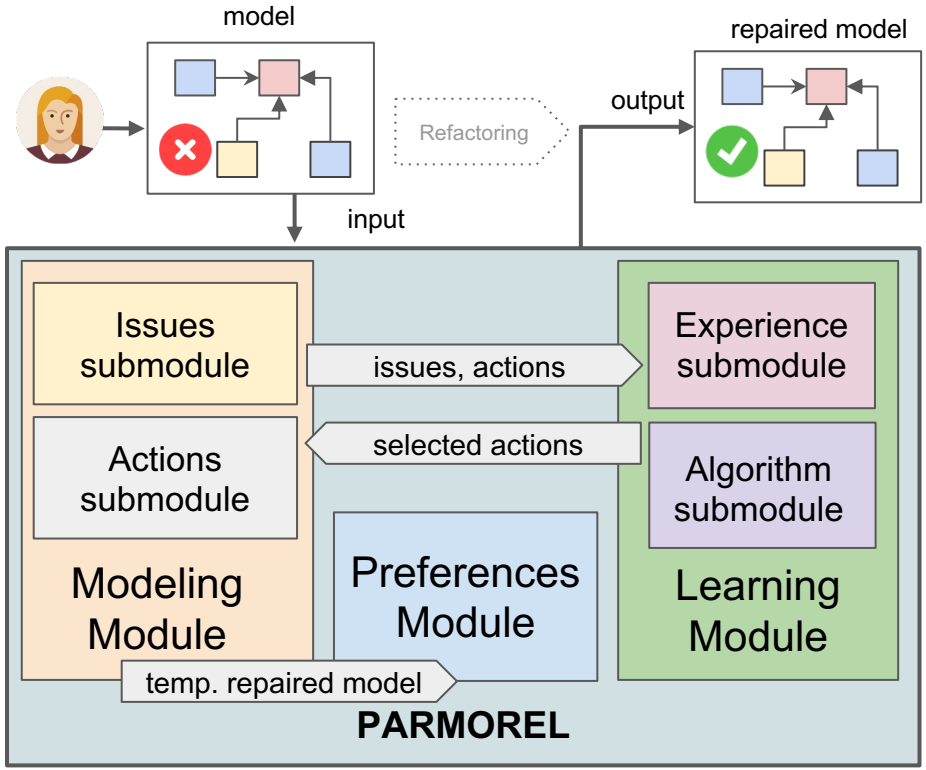

model
repaired model
Refactoring
output
input
Modeling Module
Preferences Module
Learning Module
PARMOREL

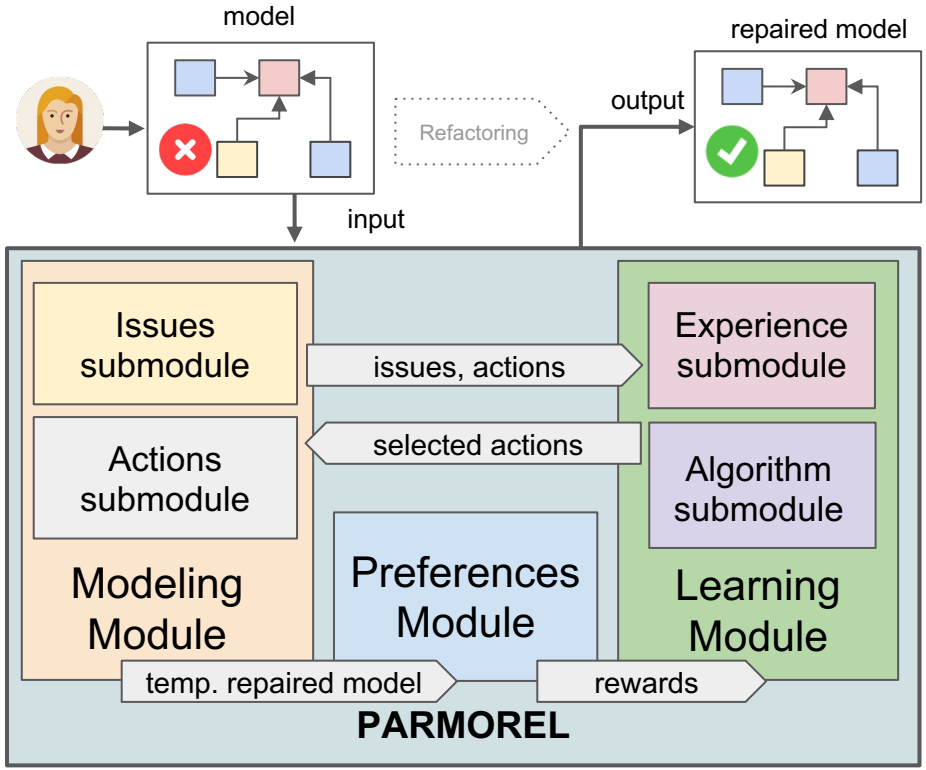

model
repaired model
Refactoring
output
input
Issues submodule
Actions submodule
Modeling Module
Preferences Module
Learning Module
PARMOREL


model
repaired model
Refactoring
output
input
Issues submodule
issues, actions
Actions submodule
Modeling Module
Preferences Module
Learning Module
PARMOREL Framework

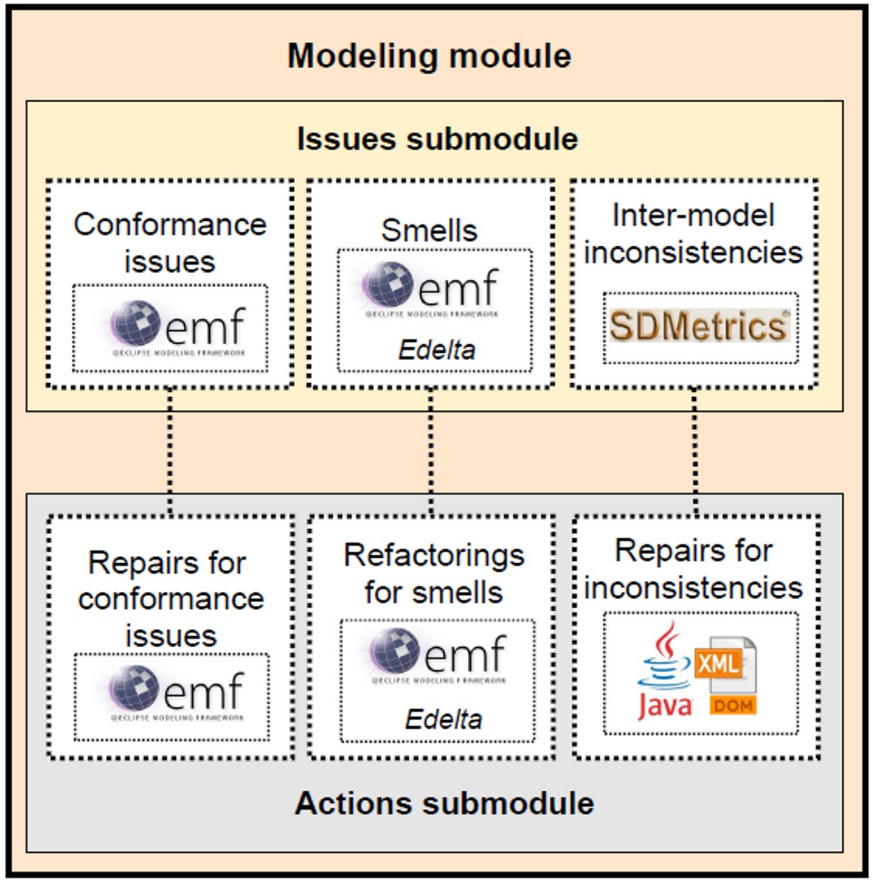

model
repaired model
Refactoring
output
input
Issues submodule
Actions submodule
Modeling Module
issues, actions
Preferences Module
Experience submodule
Algorithm submodule
Learning Module
PARMOREL


model
repaired model
Refactoring
output
input
Issues submodule
issues, actions
Experience submodule
Actions submodule
selected actions
Algorithm submodule
Modeling Module
Preferences Module
Learning Module
PARMOREL

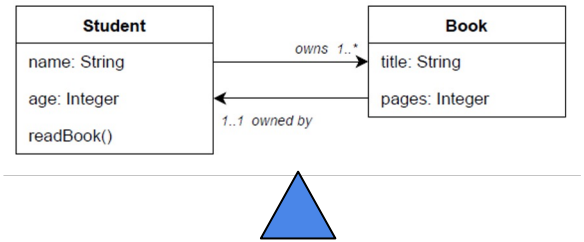

model
repaired model
Refactoring
output
input
Issues submodule
issues, actions
Experience submodule
Actions submodule
selected actions
Algorithm submodule
Modeling Module
Preferences Module
Learning Module
temp. repaired model
PARMOREL

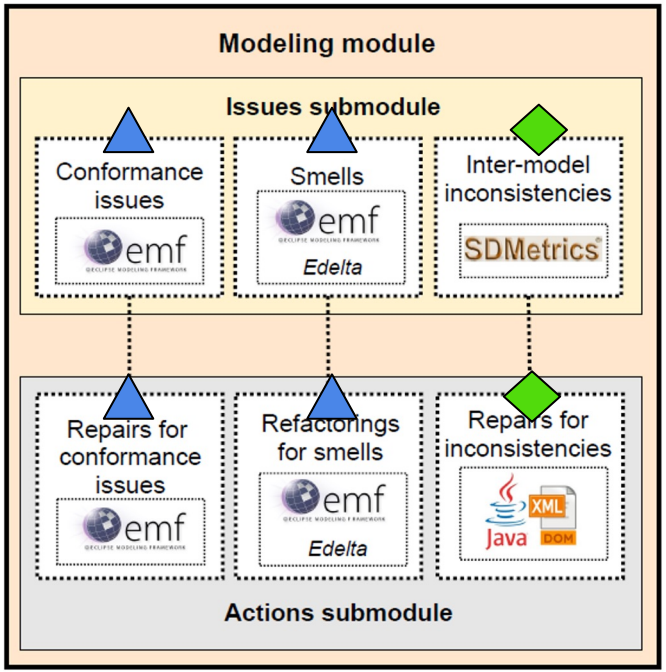

model
repaired model
Refactoring
output
input
Issues submodule
issues, actions
Experience submodule
Actions submodule
selected actions
Algorithm submodule
Modeling Module
Preferences Module
Learning Module
temp. repaired model
rewards
PARMOREL

Extensions

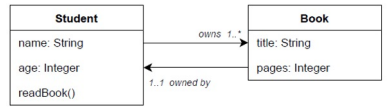

Modeling module
Issues submodule
Conformance issues
emf
Smells
emf
Edelta
Inter-model inconsistencies
SDMetrics
Repairs for conformance issues
emf
Refactorings for smells
emf
Edelta
Repairs for inconsistencies
Java
XML
DOM
Actions submodule

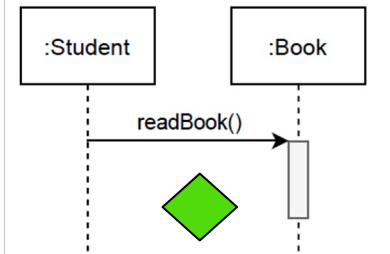
Ecore

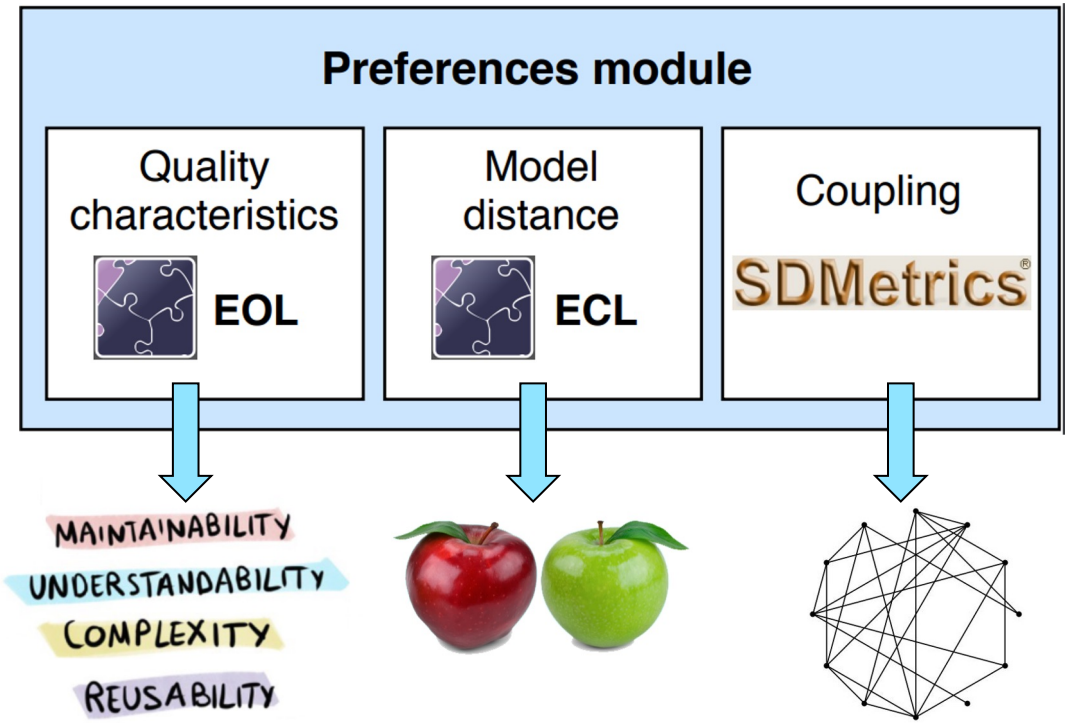

Student
name: String
age: Integer
readBook()
owns 1..*
1..1 owned by
Book
title: String
pages: Integer

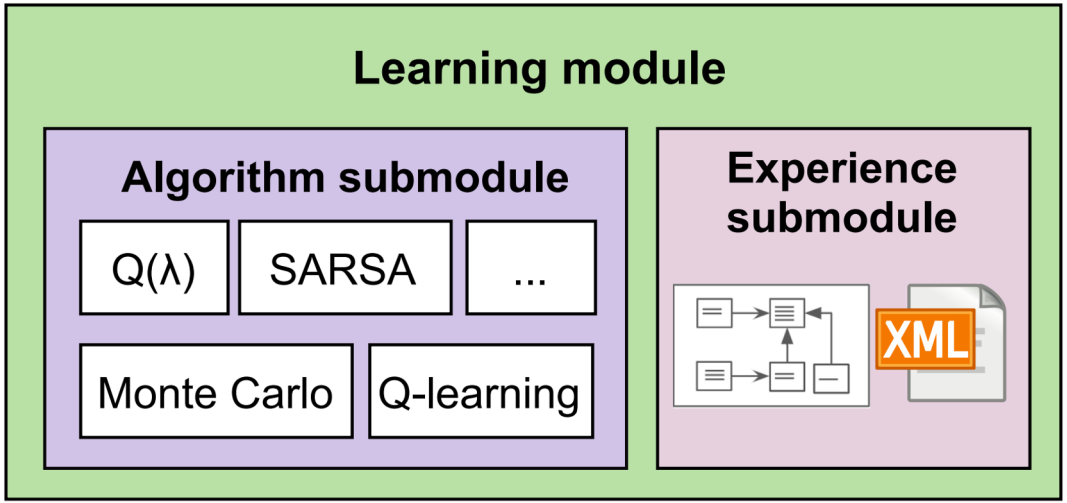

Modeling module
Issues submodule
Conformance issues
emf
Smells
emf
Edelta
Inter-model inconsistencies
SDMetrics
Repairs for conformance issues
emf
Refactorings for smells
emf
Edelta
Repairs for inconsistencies
Java
XML
DOM
Actions submodule

UML


Student
name: String
age: Integer
readBook()
owns 1..*
1..1 owned by
Book
title: String
pages: Integer


:Student
:Book
readBook()


Preferences module
Quality characteristics
EOL
Model distance
ECL
Coupling
SDMetrics®
MAINTAINABILITY
UNDERSTANDABILITY
COMPLEXITY
REUSABILITY


Learning module
Algorithm submodule
Q(λ)
SARSA
...
Monte Carlo
Q-learning
Experience submodule
XML

Summary?

Thank you for your attention!