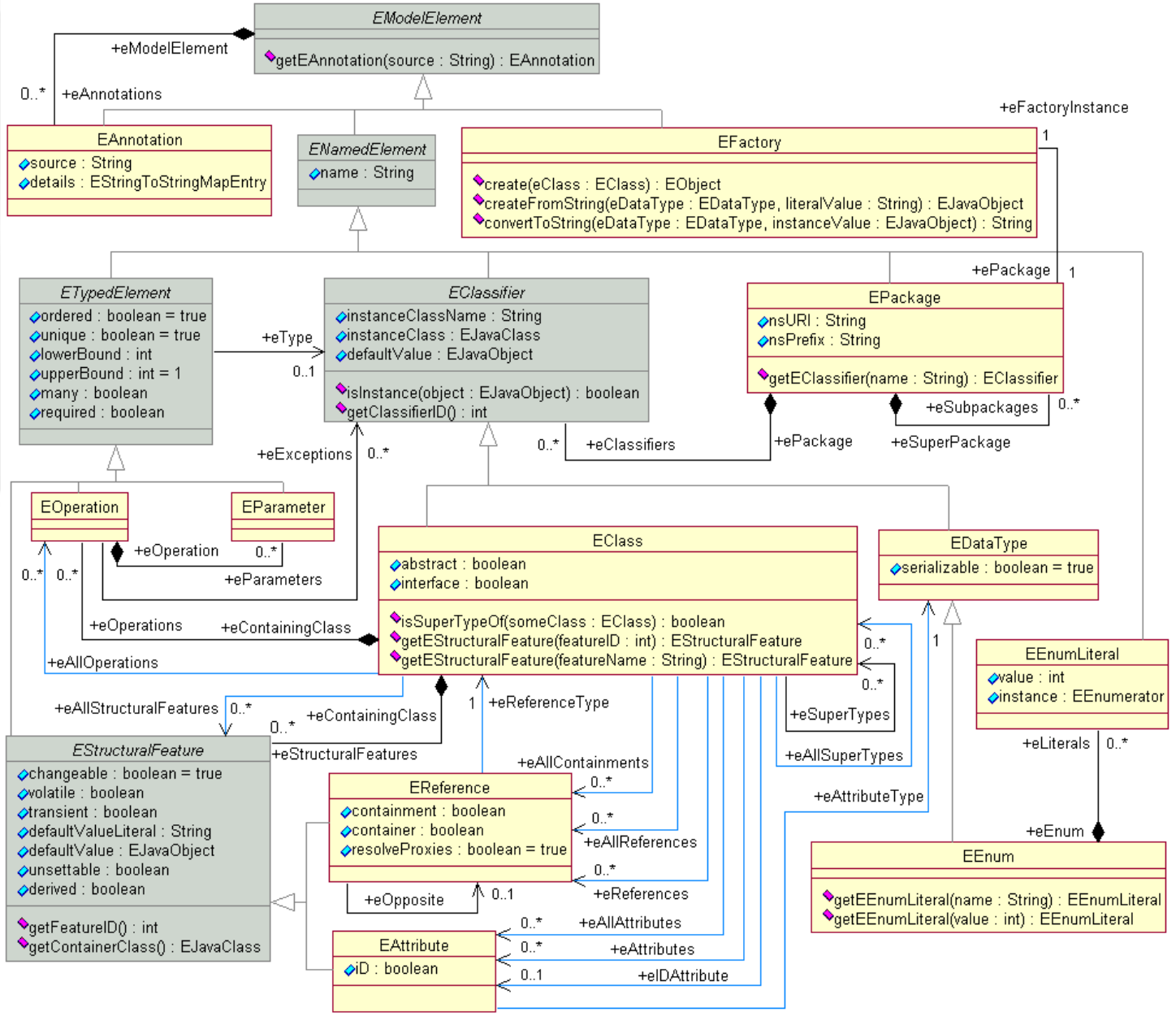
A Rule-Based Approach to the Semantic Lifting of Model Differences in the context of Model Versioning

Automatisierte Erzeugung von Transformationsregeln zur Optimierung von Modelldifferenzen

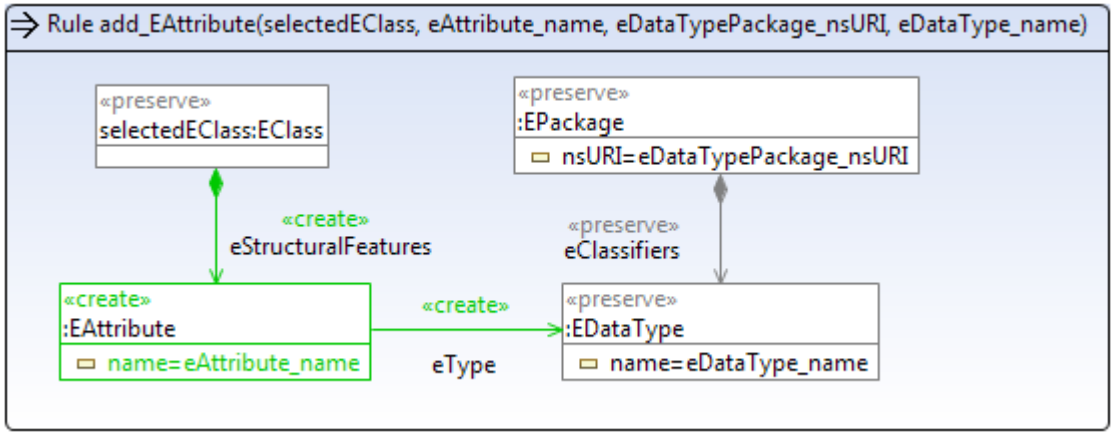
Auswahl des Metamodells



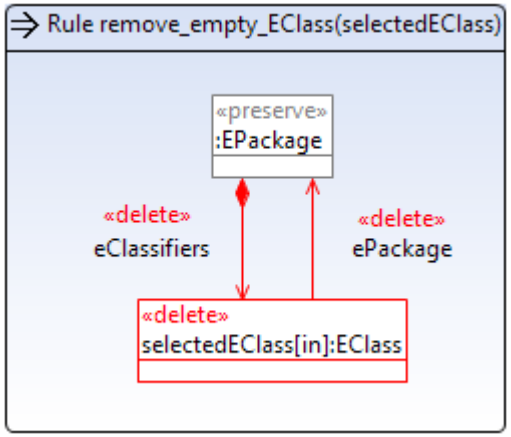
Editieren eines Modells

- Grafischer-Editor
- Tree-Based-Editor
- EMF-Refactor
  - Erweitert den Tree-Based-Editor um benutzerdefinierte Refactorings.
  - Neben Java bietet der EMF-Refactor eine Schnittstelle um Refactorings mit Hilfe von Henshinmodellen zu formulieren.
- Henshin
  - Henshin ist ein In-Place-Modell-Transformationssprache für EMF.
  - Transformation von Ecore basierenden Modellen zur Laufzeit.
  - Grafischer- und Tree-Based-Editor zur Erstellung.
  - Das Henshin Metamodell basiert auf Ecore.

Erstellen eines Transformationssystems



- Graph LHS
  - Node selectedEClass:EClass
  - Node :EDataType
  - Node :EPackage
  - Edge (eClassifiers) _ -> _
- Graph RHS
  - Node selectedEClass:EClass
  - Node :EAttribute
  - Node :EDataType
  - Node :EPackage
  - Edge (eStructuralFeatures) selectedEClass -> _
  - Edge (eType) _ -> _
  - Edge (eClassifiers) _ -> _



- Graph LHS
  - Node selectedEClass:EClass
  - Node :EPackage
  - Edge (eClassifiers) _ -> selectedEClass
  - Edge (ePackage) selectedEClass -> _
- Graph RHS
  - Node :EPackage

Erstellen der Edit-Rules

- Die Henshin Transformationssysteme werden im folgenden Kontext des Semantic Liftings als Edit-Rules bezeichnen.
- Die Edit-Rules zu einem bestimmten Metamodell müssen zunächst von Hand erzeugt werden.
- Welche Edit-Rules werden benötigt?
  - Ziel ist es ein Basissatz an Edit-Rules zu erzeugen, mit denen ein Modell aufgebaut und abgeändert werden kann.

Erstellen der Edit-Rules

- Erstellen eines vollständigen Satzes *Atomic-Edit-Rules*
  - Eine Regel darf sich nicht weiter in kleinere Regeln zerlegen lassen.
  - Eine Regel gibt die kleinstmögliche Änderung an einem Modell an bei der das Modell valid bleibt.
    - Valid bedeutet in diesem Zusammenhang, dass aus dem EMF-Modell ausführbarer Code erzeugt werden kann.
    - Bedenke: Der Ecore-Editor lässt auch Änderungen zu die das Modell inkonsistent werden lassen. (z.B. Klassen ohne Namen zu erzeugen.)
  - Erstelle eine Atomic-Edit-Rule für jedes Element im Model:
    - Klassen -> entfernen, hinzufügen, verschieben
    - Referenzen -> entfernen, hinzufügen, umbiegen (required)
    - Attribute -> Wertänderung je Attribute
  - Abstrakte Klassen selbst (nicht deren Attribute und Referenzen), abgeleitete Referenzen und nicht veränderbare Attribute usw. müssen nicht betrachtet werden.

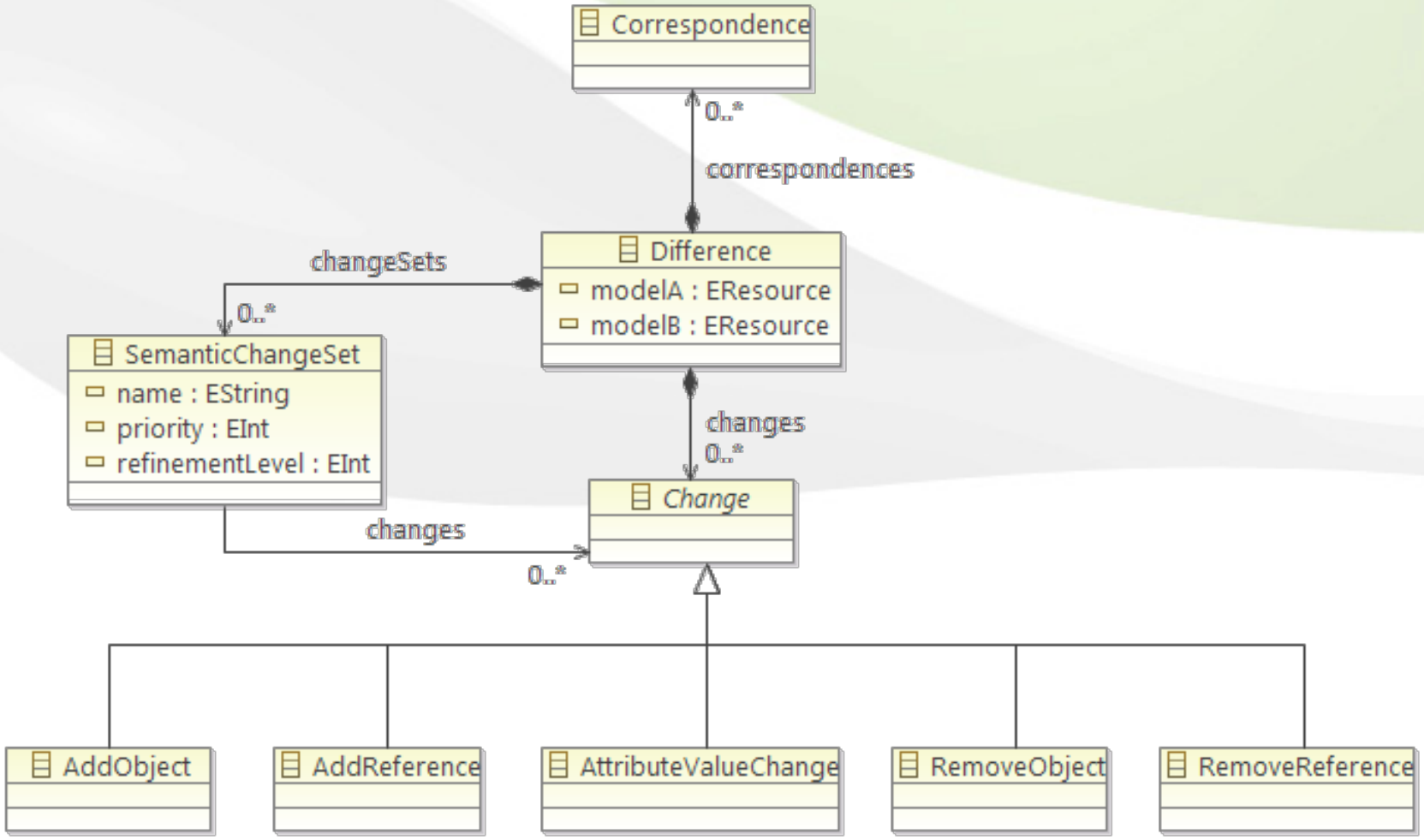
Erstellen der Edit-Rules

- 59 + (11) Atomic-Edit-Rules:
  - Add <Class> : 11
  - Remove <Class> : 11
  - (Move <Class> : 11)
  - Add Reference <Class> <Reference> : 5
  - Remove Reference <Class> <Reference> : 5
  - Change Reference <Class>: 3 (Required References)
  - Attribute Value Change <Class> <Attribute> : 23
- 18 Advanced-Edit-Rules
  - Create sub EClass
  - Pull-Up EAttribute
  - ….
- Ausblick: Automatische Generierung der Atomic-Edit-Rules.

Model Differencing Pipline - Matching

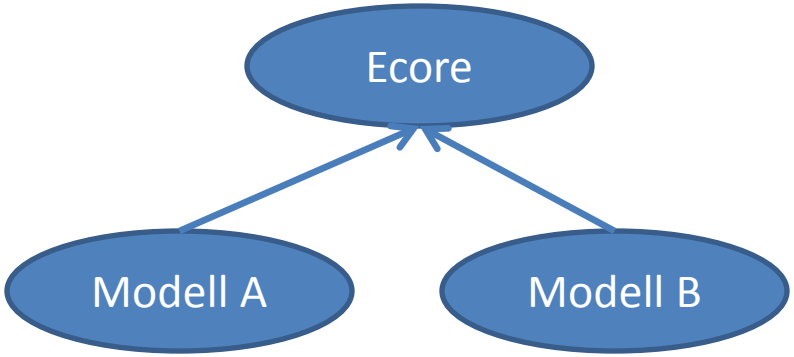
- SiDiff
- Oder anderer Matcher
- Die Qualität des Liftings hängt direkt von der Qualität des Matchings ab.
  - Aus Benutzersicht macht es einen Unterschied, ob z.B. ein Attribute aus einer Klasse gelöscht wurde und ein gleiches Attribut in einer anderen Klasse wieder eingefügt wurde oder ob das selbe Attribute verschoben wurde.

Model Differencing Pipline - Difference Derivation



Model Differencing Pipline - Difference Derivation

- Ein Modell kann Klassen aus anderen Modellen referenzieren.
  - Das Ecore-Modell stellt z.B. eine Type-Library zur Verfügung die häufig für Attribute verwendet wird.
  - Das Problem besteht darin, dass für das referenzierte Modell keine Matchings also auch keine Correspondences erzeugt werden.
  - Die Correspondences werden aber für das Sematic Lifting benötigt.



Model Differencing Pipline - Difference Derivation

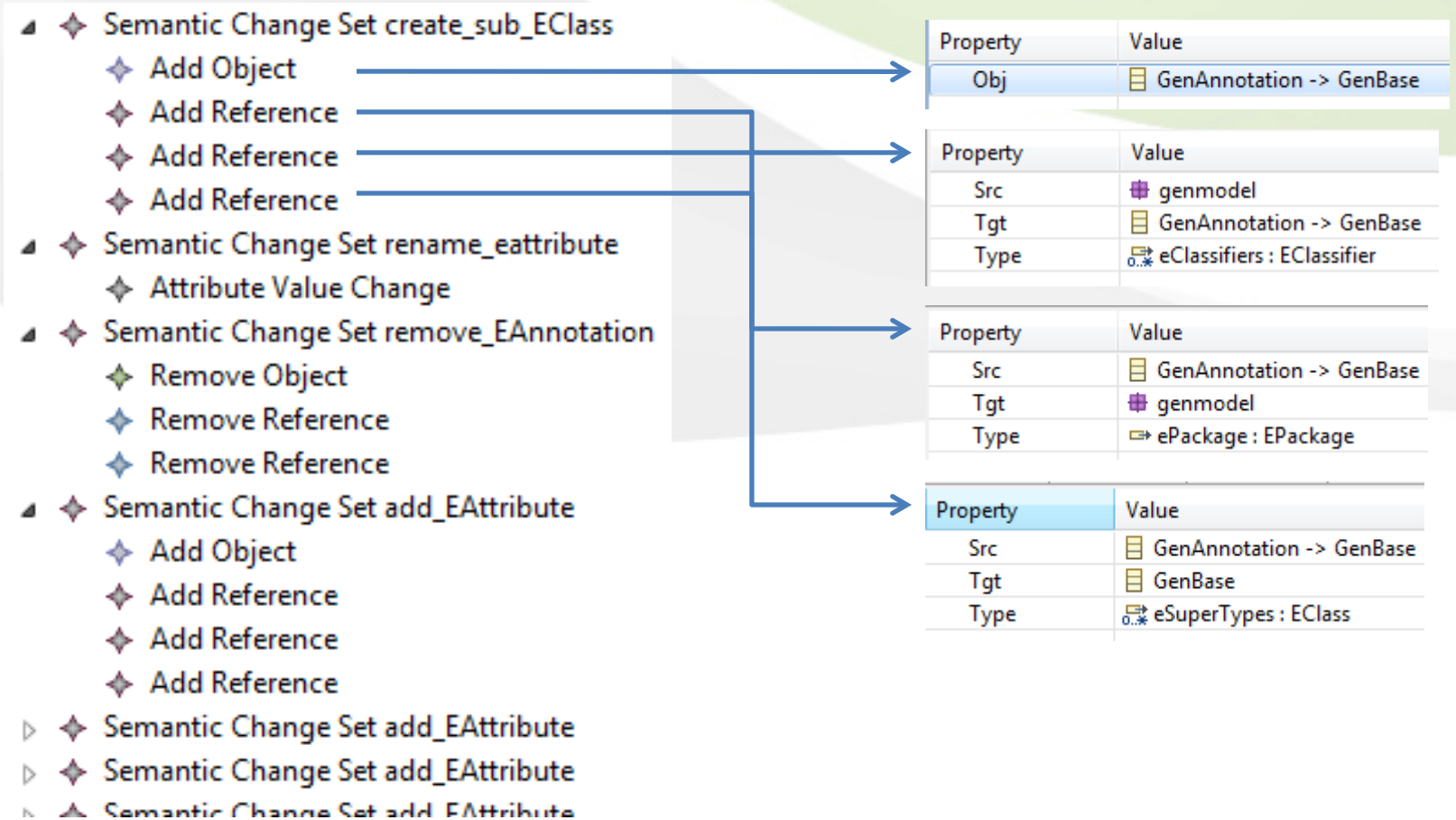
- Merge Imports:
  1. Finde alle externen Referenzen in Modell A und B
  2. Erstelle eine Kopie aller Referenzierten Modelle
  3. Erzeuge eine Correspondence zwischen Original Modell Element und der Kopie.
     - Filter: Nur für referenzierte Elemente und deren Container.
  4. Biege alle externen Referenzen in Modell B auf die Kopie um.
  5. Biege alle externen Referenzen der AddReference Changes auf die Kopie um.
  6. Nach Lifting -> Mache alle Änderung an der Difference und an Model B wieder rückgängig.

Model Differencing Pipline - Semantic Lifting

- Ziel des Semantic Liftings (technisch):
  - Gruppierung der Low-Level Changes zu User-Level Changes.
  - Eine Gruppierung wird als Sematic Change Set bezeichnet.
  - Die Gruppierung wird durch die zuvor erstellten Edit-Rules vorgegeben.
  - Die Gruppierung erfolgt automatisch über Henshin Transformationsregeln. Wir werden diese Transformationsregeln im folgenden als Recognition-Rules bezeichnen.
  - Die Recognition-Rules werden automatisch aus den Edit-Rules generiert.

Model Differencing Pipline - Semantic Lifting

- Gruppieren der Low-Level Changes:

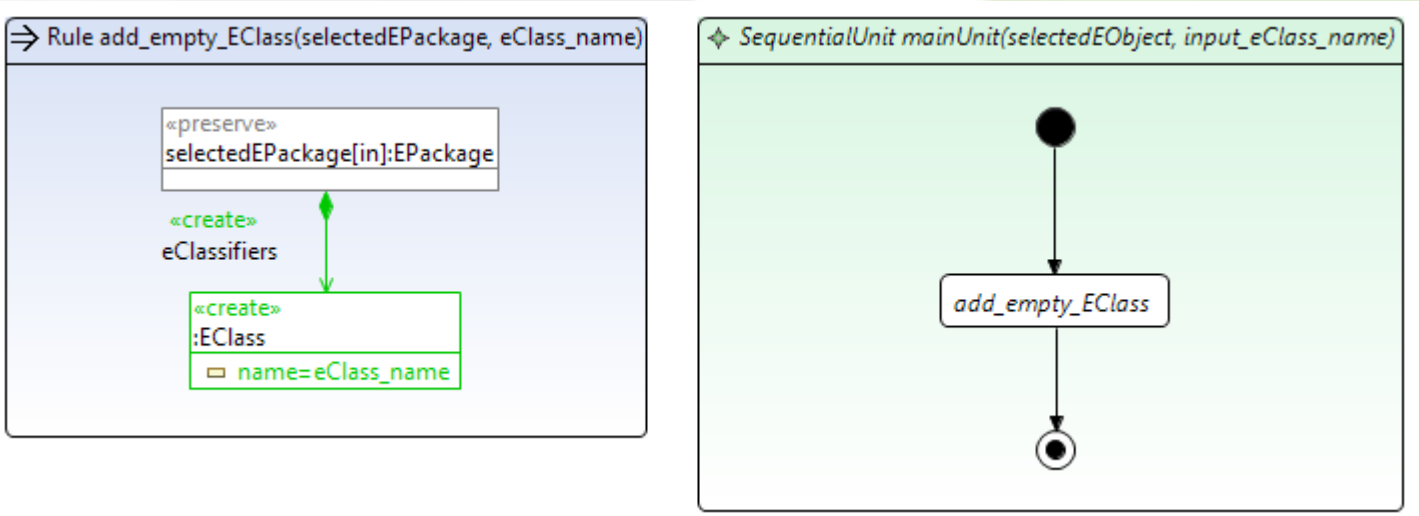


Automatische Generierung der Recognition-Rules

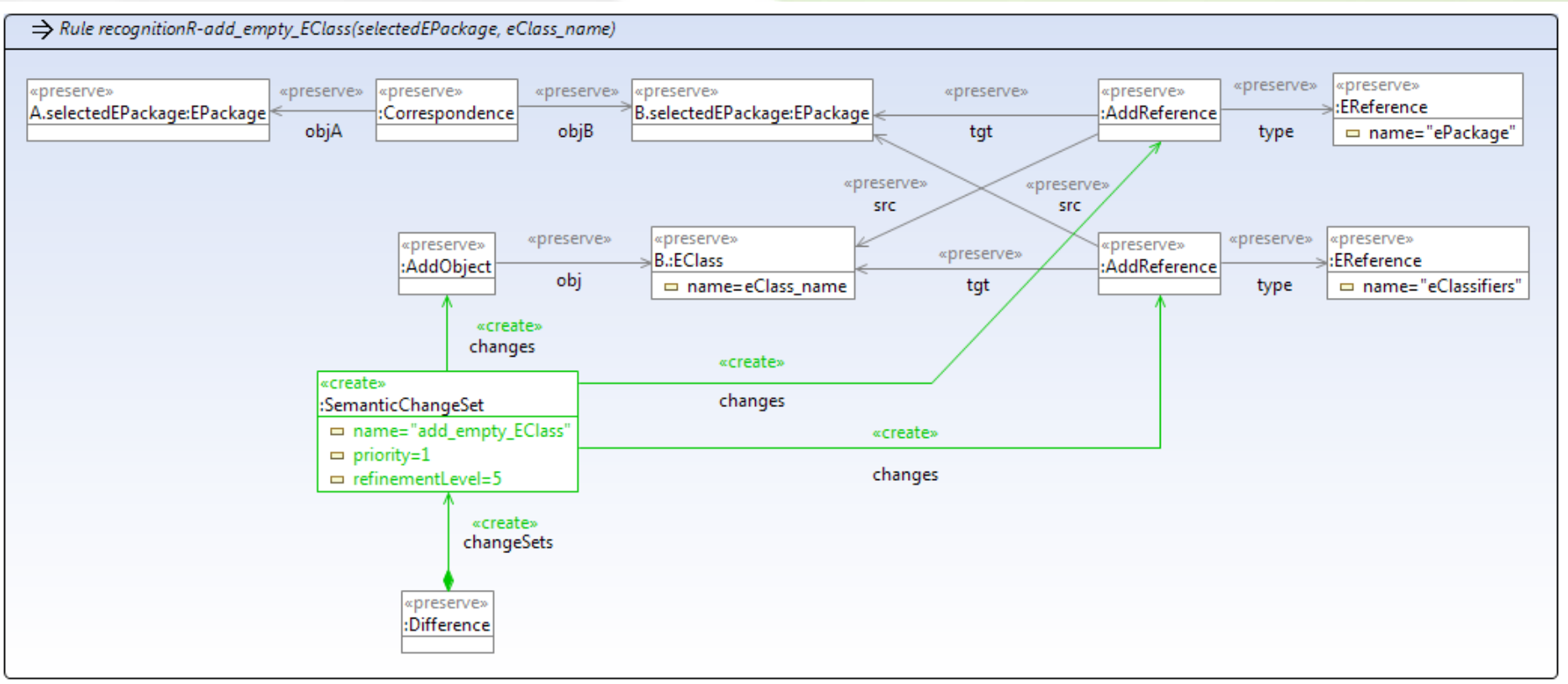
- Generieren der einzelnen Patterns:
  - Henshin Modell Elemente:
    - Nodes, Edges, Attributes
  - Henshin Stereotypen:
    - <<create>>, <<delete>>, <<preserve>>

- Amalgamation Units
  - Generieren der Kernel- und Multi-Recognition-Rules
  - Einbetten der Kernel-Rule in die Multi-Rules

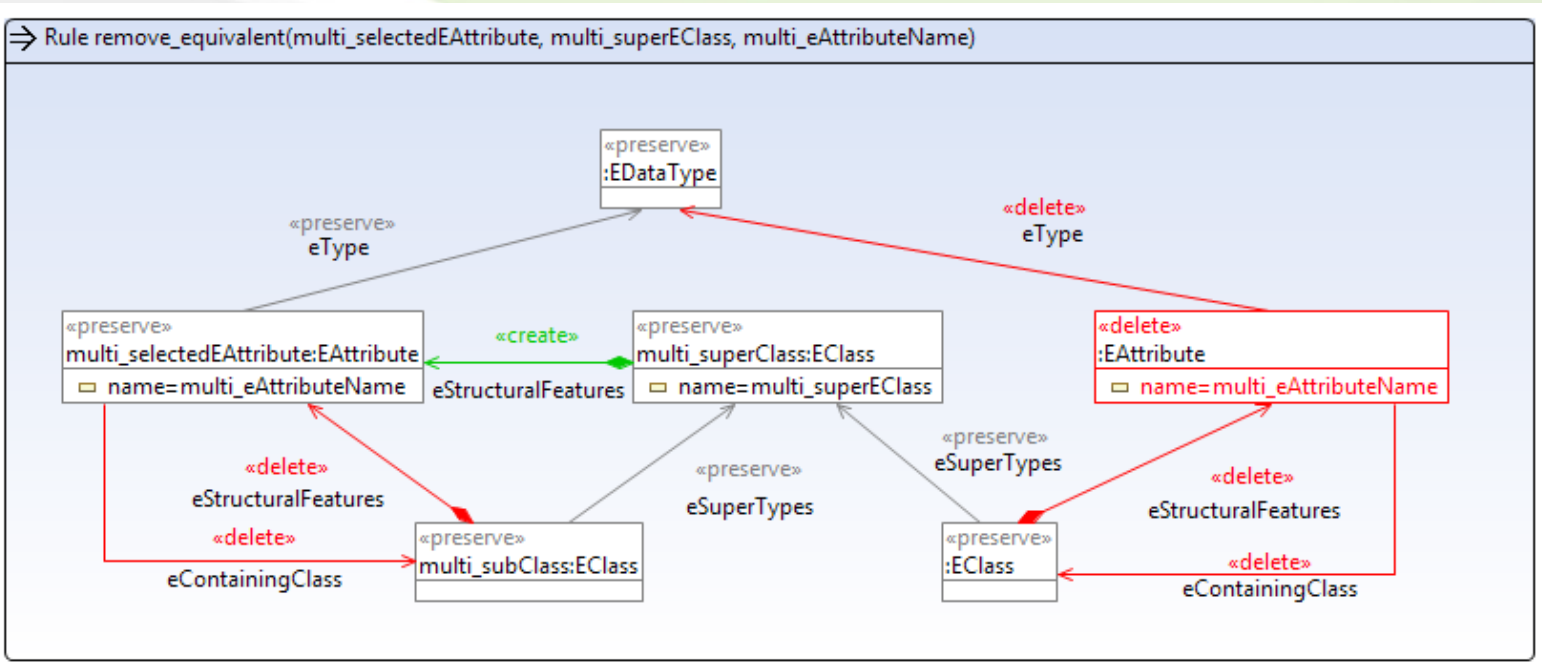
Automatische Generierung der Recognition-Rules

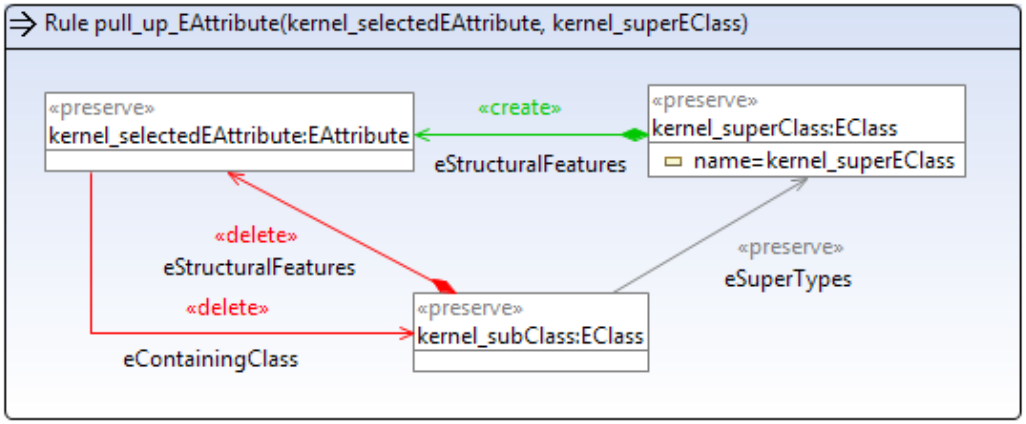


Automatische Generierung der Recognition-Rules

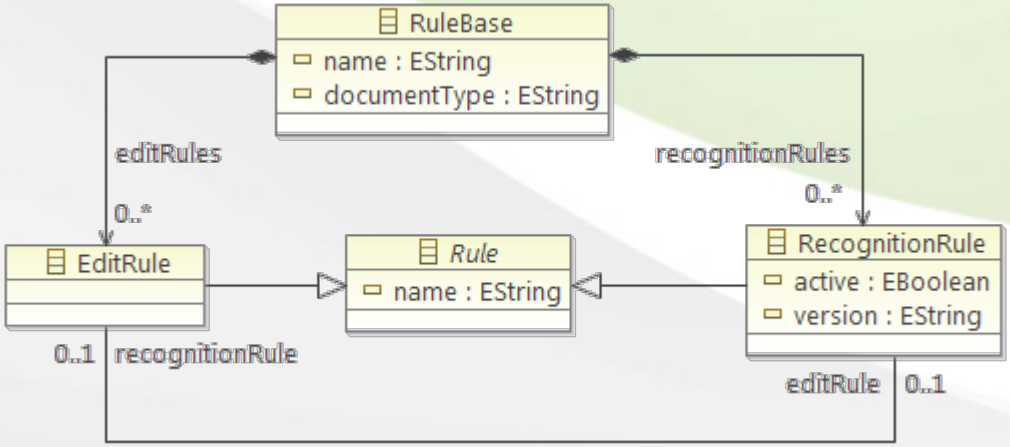


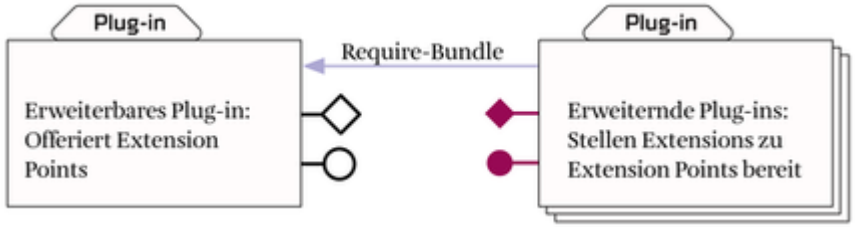
Automatische Generierung der Recognition-Rules





Verwalten der Rulebases





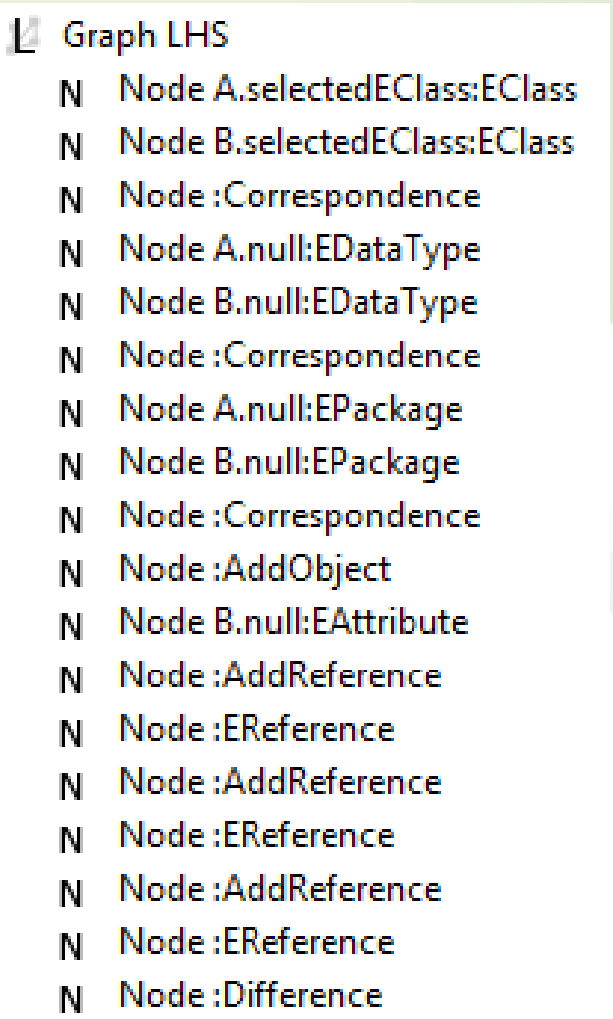
- Name
- Document Type
- Liste der auszuführenden Regeln

Recognition-Engine

- Aufbauen des Henshin Arbeitsgraphen:
  - Technische Modell Differenz
  - Modell A und B
  - Das Metamodell von Modell A und B (In unserem Fall Ecore)
  - Ecore als Meta-Metamodell
- Filtern der Recognition-Rules: (optimierung)
  1. Zählen der Low-Level Changes in der Difference.
  2. Zählen der Low-Level Change Nodes pro Recognition-Rule.
  3. Hat eine Recognition-Rule mehr Nodes von einem Low-Level Change Typ als zu diesem Typ Low-Level Changes in der Difference existieren kann die Regel ausgelassen werden.

Recognition-Engine

- Sortieren der Recognition-Rule:
  - Low-Level Change Nodes werden nach Häufigkeit in der Difference im oberen Teil der Liste sortiert.
  - Alle Correspondences werden in den unteren Teil der Liste verschoben.
  - Alle anderen Nodes verbleiben im mittleren Teil der Liste.
- Optimierung verkürzt die Rechenzeit gerade bei großen Modellen vom Minuten- in den Sekundenbereich.



Recognition-Engine

- Anschließend werden die Recognition-Rules blockweise an Threads übergeben und Parallel ausgeführt.
- D.h. es werden für jede Regel alle Matches im Graph gesucht und für jeden gefundenen Match wird beim anwenden der Regel ein Semantic Change Set erzeugt.
- Die dabei entstehenden Semantic Change Sets können sich Teilweise oder ganz überschneiden.
- D.h. der selbe Low-Level Change kann in mehreren Semantic Change Sets auftauchen.

Post-Processing

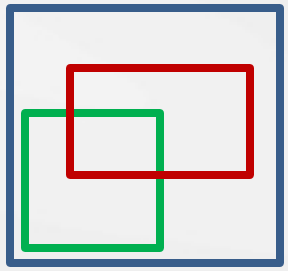
- Ziel des Post-Processings:
  - Eleminieren aller Überschneidungen von Semantic Change Sets.
  - Erreichen einer möglichst guten Überdeckung der Low-Level Changes.
  - Es sollten möglichst die großen Semantic Change Sets erhalten bleiben.
- Algorithmus arbeitet mit 2 Sets:
  - PCS_D: Noch zu bearbeitende Semantic Change Sets
  - PCS_min: Semantic Change Sets die erhalten bleiben.

Post-Processing

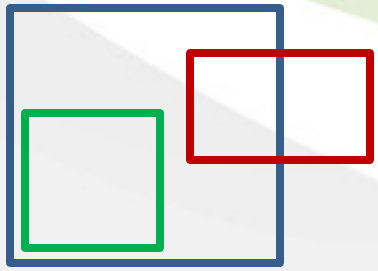
1. Übernehme nur jeweils ein SCS bei äquivalenten SCS in PCS_D.
   - Bevorzuge SCS mit höherer Priorität.
     - Die Prioritäten können benutzerdefiniert sein.
     - Amalgamation Units haben nach der Generierung standardmäßig eine niedrigere Priorität als einfache Regeln.
     - -> Problem: Amalgamation Kernel-Rules sind häufig identisch mit einfachen Atomic-Recognition-Rules. Bei Amalgamation Units muss es aber nicht zwangsläufig einen Match für die Multi-Rules geben.
   - Bevorzuge SCS mit höherem Refinment-Level
     - Das Refinment-Level wird beim generieren der Recognition-Rule berechnet.
     - -> Summiere für jeden Node der Edit-Rule die Anzahl aller Supertypen => eine Regel mit vielen Supertypen ist „spezieller“ als eine Regel mit wenig Supertypen, die das gleiche SCS erzeugt.

Post-Processing

2. Übernehme alle sich nicht überlappenden SCS in PCS_min
3. Finde SCS bei denen alle überschneidenden SCS Teilmengen dieses SCS sind. Entferne alle Teilmengen aus PCS_D und übernehme das SCS in PCS_min

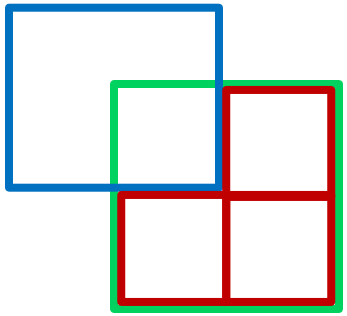


O.K.

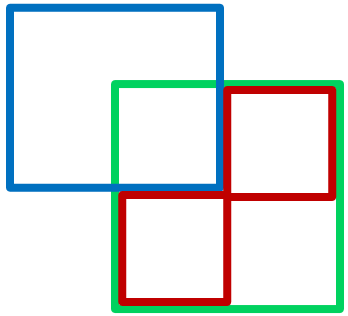


nicht O.K.

4. Finde SCS die Low-Level Changes enthalten die in keinem anderen SCS enthalten sind.
   - Können alle überlappenden SCS aus PCS_D entfernt werden ohne ungruppierte Low-Level Changes zu erzeugen kann das SCS in PCS_min übernommen werden.



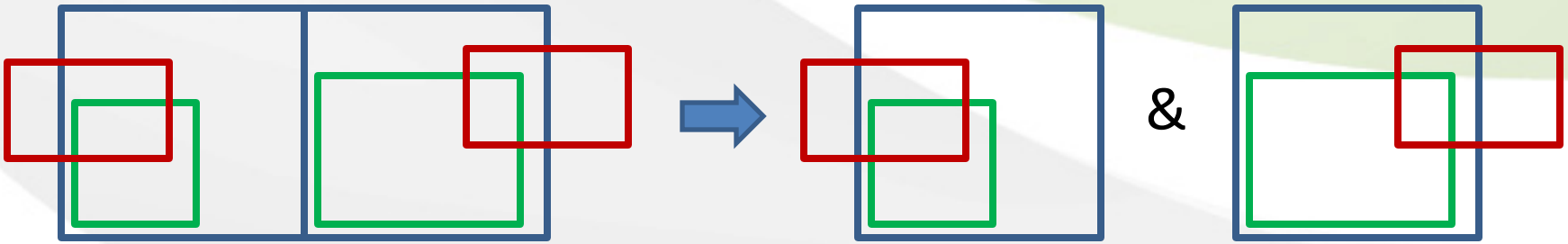
O.K.



nicht O.K.

Post-Processing

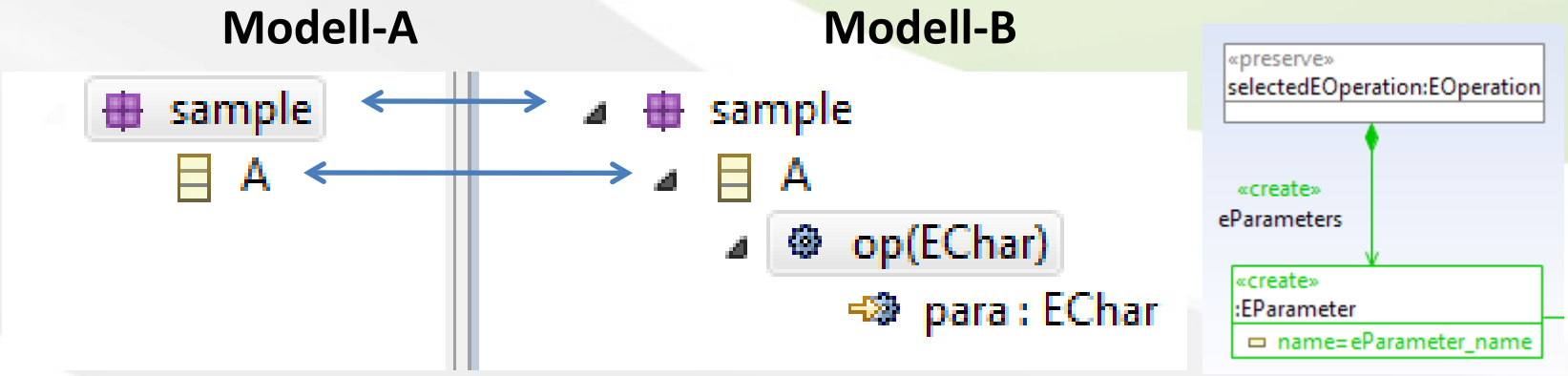
5. Zerlege alle restlichen sich noch überschneidenden SCS in PCS_D in disjunkte Gruppen.



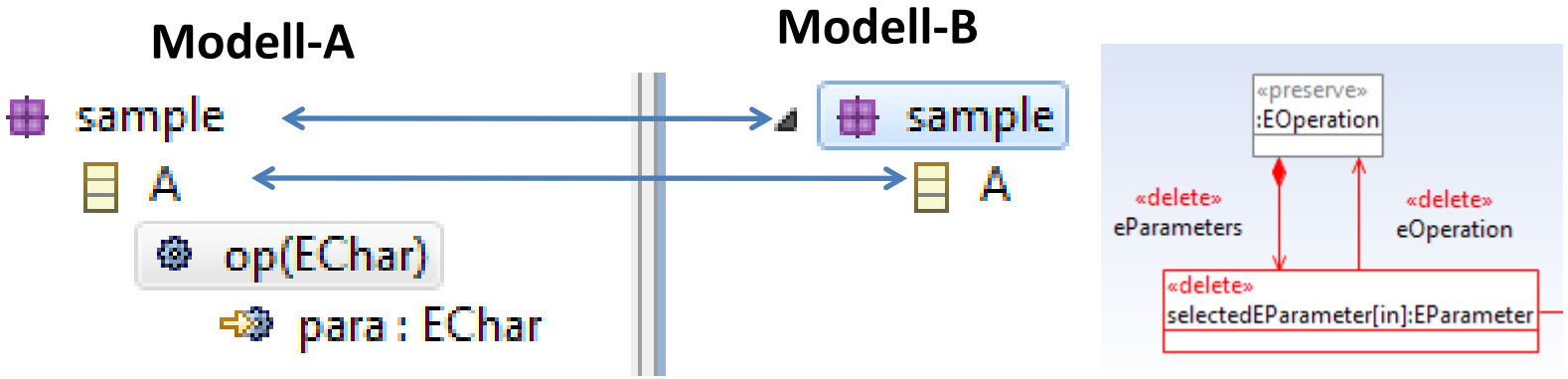
6. Berechne kombinatorisch (limitiert) die optimale Auswahl der zu erhaltenden SCS unter den zuvor gegebenen Optimierungskriterien:
   - Eleminieren aller Überschneidungen von Semantic Change Sets.
   - Erreichen einer möglichst guten Überdeckung der Low-Level Changes.
   - Es sollten möglichst die großen Semantic Change Sets erhalten bleiben.

Sequential-Lifting

- Create-Use-Dependence



- Removed-Used-Dependence



Sequential-Lifting

1. Führe alle Low-Level Changes aus, die in einem Semantic Change Sets liegen.
   - AddObjects/References werden Modell A hinzugefügt.
   - RemoveObjects/References werden Modell B hinzugefügt.
   - AttributeValueChanges werden in Modell A auf den neuen Wert gesetzt.
2. Entferne alle ausgeführten Semantic Change Sets incl. deren Low-Level Changes aus der Difference.
3. Füge für alle entfernten Add/RemoveObjects Correspondences in die Difference ein.
4. Starte die Recognition-Engine.
5. Starte das Post-Processing.
6. Wiederhole Schritt 1-5 solange wie in einem durchlauf neue Semantic Change Sets erzeugt werden oder bis keine Low-Level Changes mehr vorhanden sind.

Live-Demo