Incremental MTL vs. GPLs: Class into Relational Database Schema Case

Sandra Greiner, Stefan Höppner, Frédéric Jouault, **Théo Le Calvar**, and Mickaël Clavreul

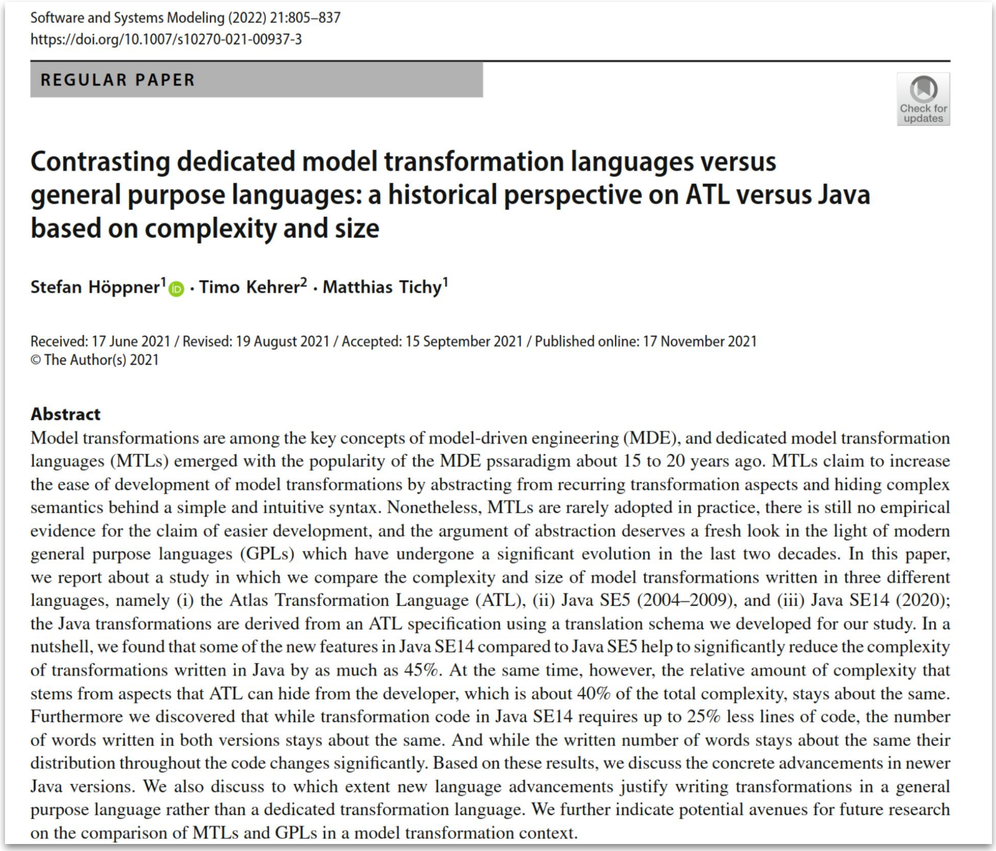
TTC 2023

Detailed Outline

- Context
    - Objectives
    - Background: Stefan’s work on MTL vs. GPL
- Case Presentation
    - Quick overview
    - Challenges of the Case
        - Homogenizing annotations
            - we should probably count only what is really transformation specific
                - excluding model loading?
                - excluding driver?
        - The binding deactivation problem
            - ideally adding a corresponding change model
- Solution Reviews
    - Overview (as a table?)
        - “properties/features” of each solution
        - (Stefan’s) syntactic complexity diagrams
    - MTLs
        - ATL
        - ATOL
        - BxtendDSL
    - Embedded MTLs
        - NMF
    - GPLs
        - Java
        - C#
    - Other languages
        - Cheptre
- Towards a Journal Paper / Call for More Solutions

Why use a GPL for transformation when MTL performs better?

Or vice versa?



Software and Systems Modeling (2022) 21:805–837
https://doi.org/10.1007/s10270-021-00937-3

REGULAR PAPER

Check for updates

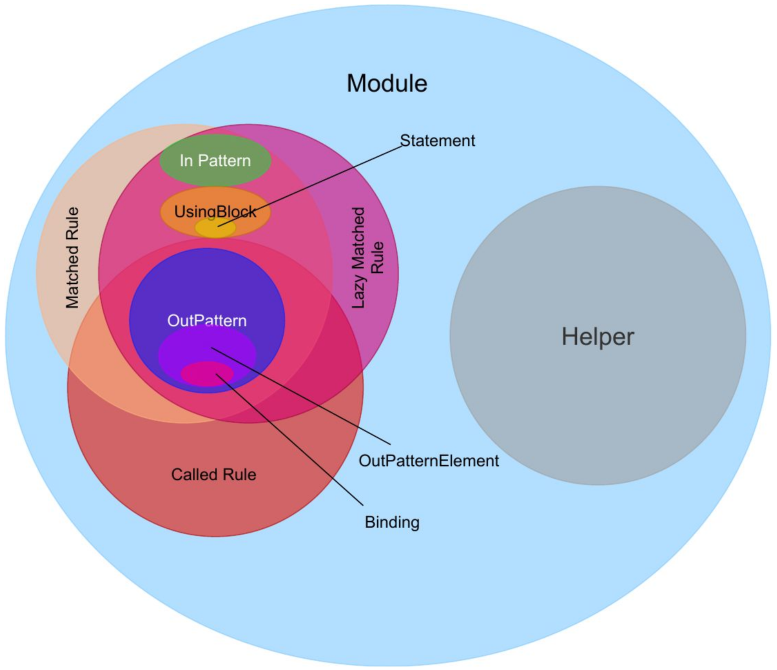
Contrasting dedicated model transformation languages versus general purpose languages: a historical perspective on ATL versus Java based on complexity and size

Stefan Höppner[1] · Timo Kehrer[2] · Matthias Tichy[1]

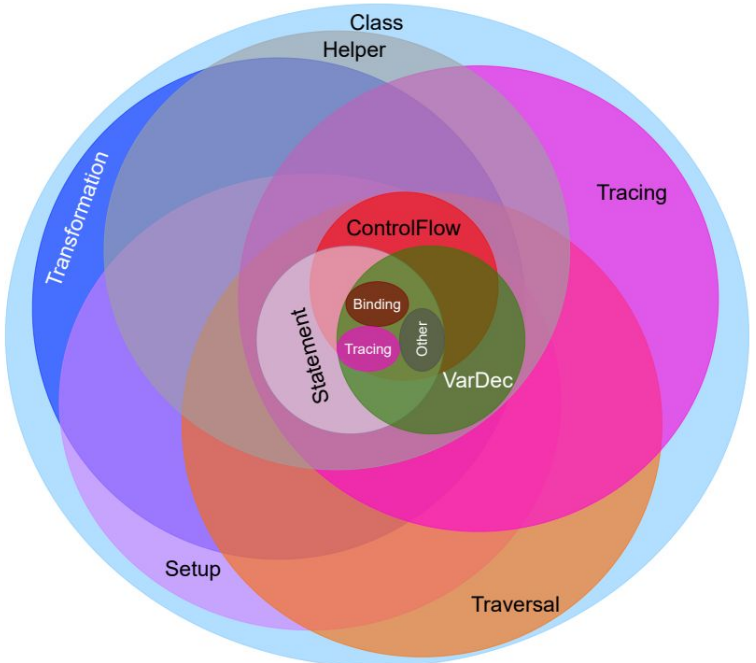
Received: 17 June 2021 / Revised: 19 August 2021 / Accepted: 15 September 2021 / Published online: 17 November 2021

Abstract
Model transformations are among the key concepts of model-driven engineering (MDE), and dedicated model transformation languages (MTLs) emerged with the popularity of the MDE pssaradigm about 15 to 20 years ago. MTLs claim to increase the ease of development of model transformations by abstracting from recurring transformation aspects and hiding complex semantics behind a simple and intuitive syntax. Nonetheless, MTLs are rarely adopted in practice, there is still no empirical evidence for the claim of easier development, and the argument of abstraction deserves a fresh look in the light of modern general purpose languages (GPLs) which have undergone a significant evolution in the last two decades. In this paper, we report about a study in which we compare the complexity and size of model transformations written in three different languages, namely (i) the Atlas Transformation Language (ATL), (ii) Java SE5 (2004–2009), and (iii) Java SE14 (2020); the Java transformations are derived from an ATL specification using a translation schema we developed for our study. In a nutshell, we found that some of the new features in Java SE14 compared to Java SE5 help to significantly reduce the complexity of transformations written in Java by as much as 45%. At the same time, however, the relative amount of complexity that stems from aspects that ATL can hide from the developer, which is about 40% of the total complexity, stays about the same. Furthermore we discovered that while transformation code in Java SE14 requires up to 25% less lines of code, the number of words written in both versions stays about the same. And while the written number of words stays about the same their distribution throughout the code changes significantly. Based on these results, we discuss the concrete advancements in newer Java versions. We also discuss to which extent new language advancements justify writing transformations in a general purpose language rather than a dedicated transformation language. We further indicate potential avenues for future research on the comparison of MTLs and GPLs in a model transformation context.

Outcome: MTL vs GPL for Batch Transformations



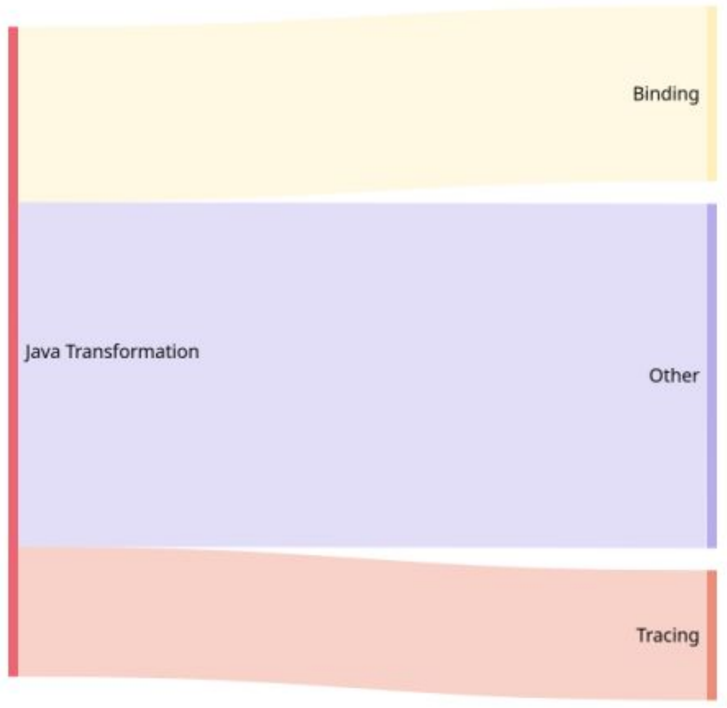
Code Structure: ATL



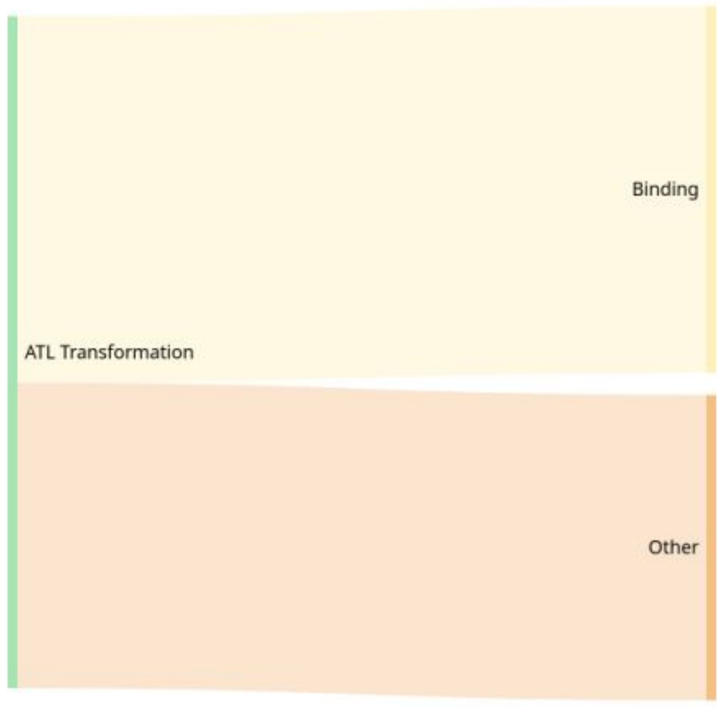
Code Structure: Java

[Höppner et al 2022]

ATL vs Java

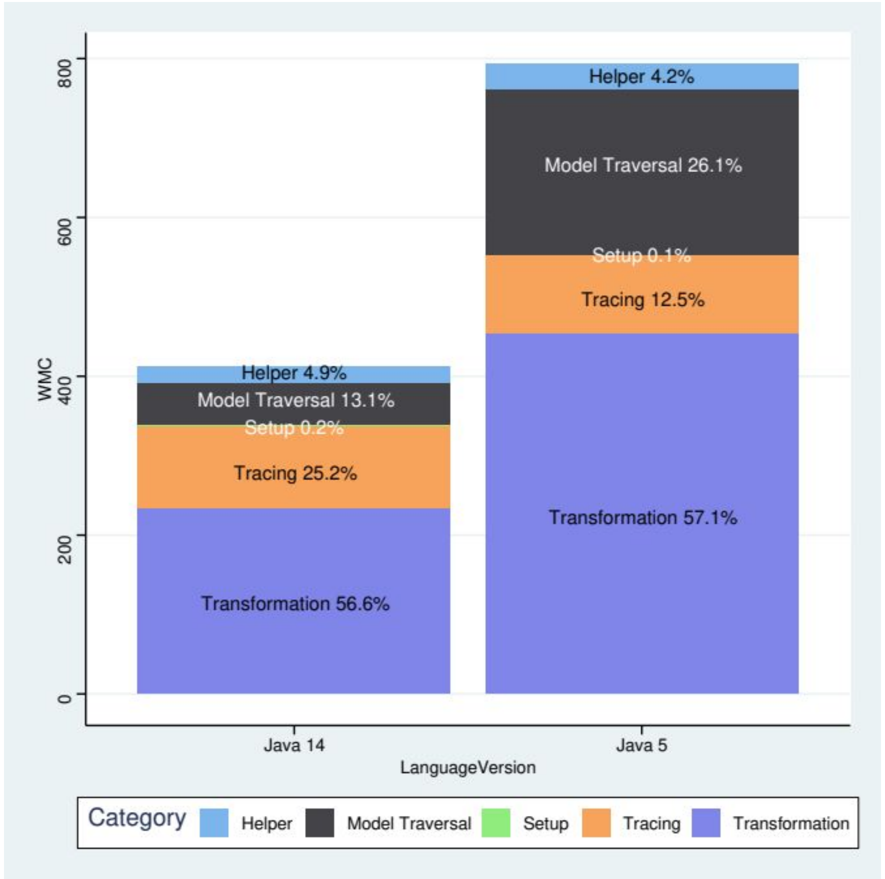


Java Transformation
Binding
Other
Tracing



ATL Transformation
Binding
Other

Historical Advancement in Java



Objectives: Compare MTL and GPL?

Do MTLs perform better than GPLs in *incremental* scenarios?

RQ 1 Distribution of transformation size over different parts in GPL vs MTL

RQ 2: Error rate of GPL vs MTL in **incremental transformations?**

RQ 3: Situations where MTL better than GPLs for incremental transformations?

→a multitude of heterogeneous solutions required

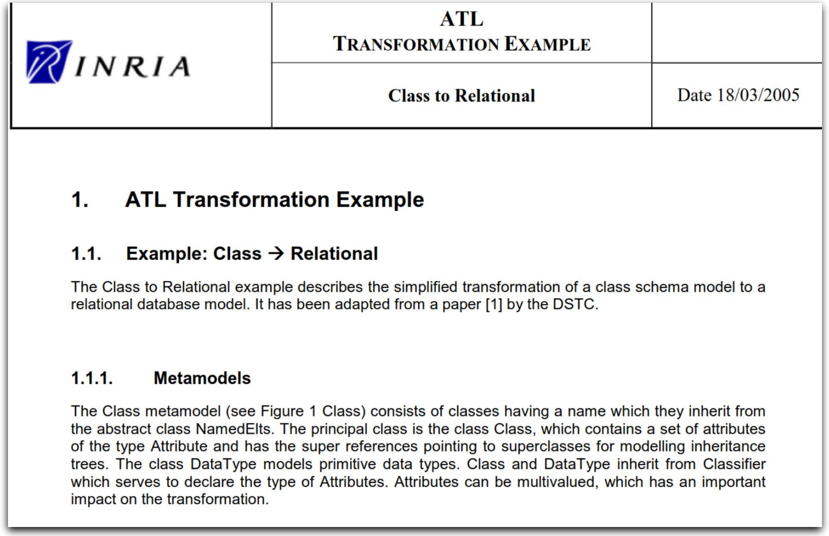
→a case which is popular and easy to solve

Objectives

→ Compare GPL and MTL transformation in **incremental** situations

Requires:

Several solutions which are **annotated**



INRIA	ATL TRANSFORMATION EXAMPLE	
	Class to Relational	Date 18/03/2005

1. ATL Transformation Example

1.1. Example: Class → Relational

The Class to Relational example describes the simplified transformation of a class schema model to a relational database model. It has been adapted from a paper [1] by the DSTC.

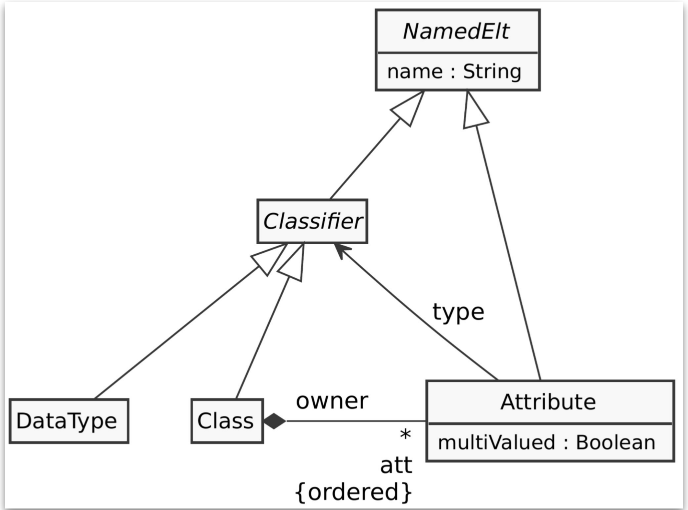
1.1.1. Metamodels

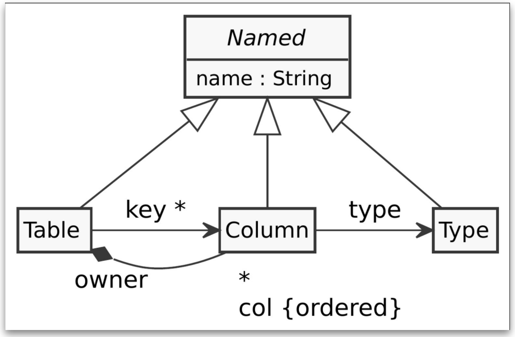
The Class metamodel (see Figure 1 Class) consists of classes having a name which they inherit from the abstract class NamedElts. The principal class is the class Class, which contains a set of attributes of the type Attribute and has the super references pointing to superclasses for modelling inheritance trees. The class DataType models primitive data types. Class and DataType inherit from Classifier which serves to declare the type of Attributes. Attributes can be multivalued, which has an important impact on the transformation.

→ **Class 2 Relational in incremental mode**

- **Well-known/popular**
- **"Easy" to solve**

Class 2 Relational Incremental





Completeness and **Correctness** and **Labeling**

Correct: commuting batch and incremental transformations

Complete: support different incremental behaviors

Labeling: for analyzing the transformation complexity

Class 2 Relational Incremental: Completeness

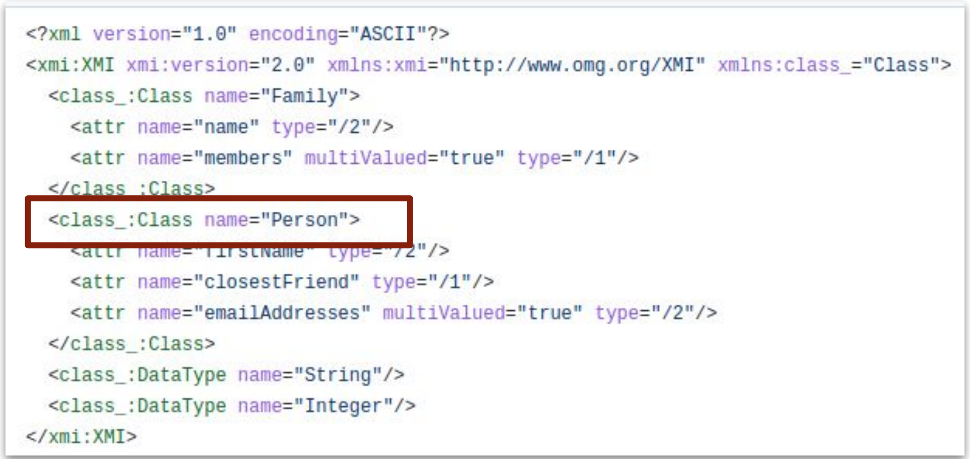
Criteria:

1) Change value of EAttribute

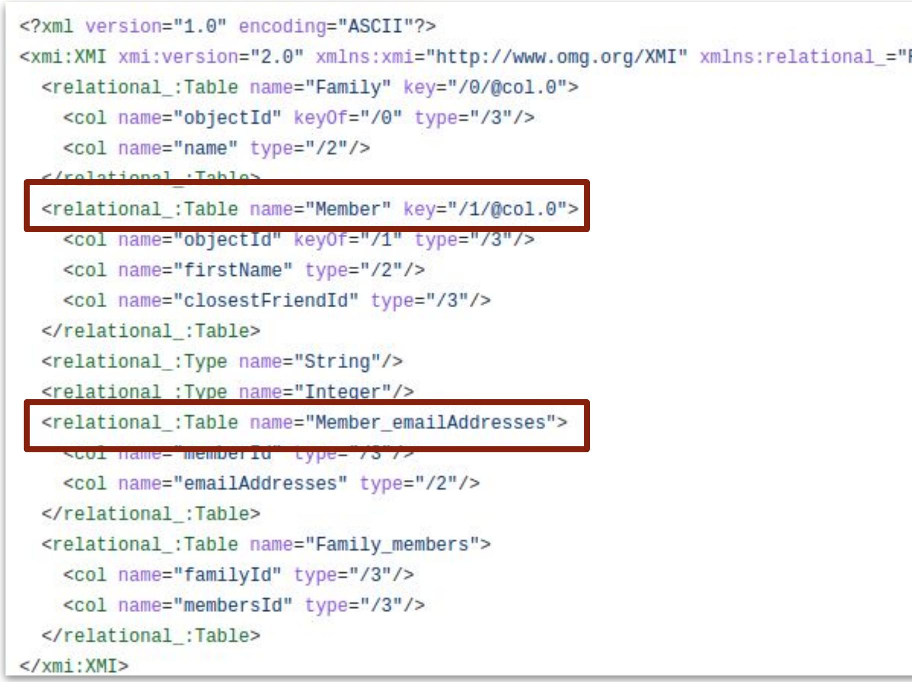
2) Change value of EReference

3) Create and Delete EObjects

Change name of class Person

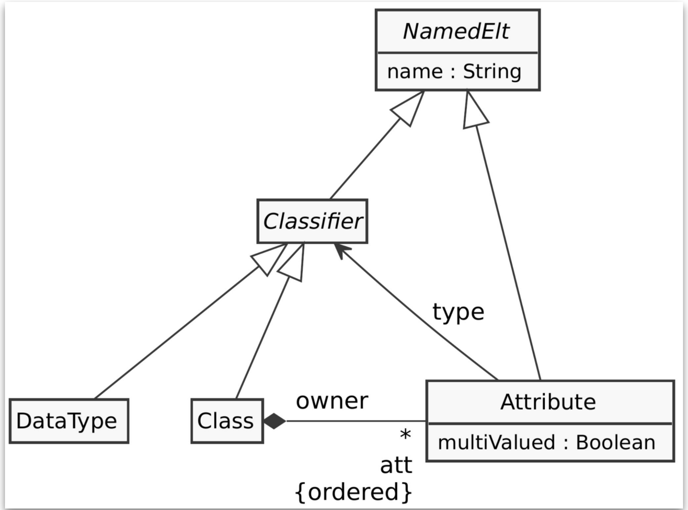


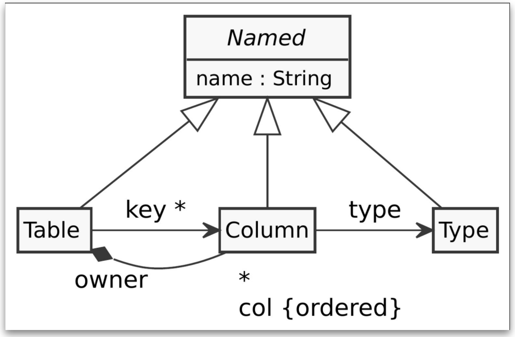
```xml
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:class_="Class">
  <class_:Class name="Family">
    <attr name="name" type="/2"/>
    <attr name="members" multiValued="true" type="/1"/>
  </class_:Class>
  <class_:Class name="Person">
    <attr name="firstName" type="/2"/>
    <attr name="closestFriend" type="/1"/>
    <attr name="emailAddresses" multiValued="true" type="/2"/>
  </class_:Class>
  <class_:DataType name="String"/>
  <class_:DataType name="Integer"/>
</xmi:XMI>
```



```xml
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:relational_="
  <relational_:Table name="Family" key="/0/@col.0">
    <col name="objectId" keyOf="/0" type="/3"/>
    <col name="name" type="/2"/>
  </relational_:Table>
  <relational_:Table name="Member" key="/1/@col.0">
    <col name="objectId" keyOf="/1" type="/3"/>
    <col name="firstName" type="/2"/>
    <col name="closestFriendId" type="/3"/>
  </relational_:Table>
  <relational_:Type name="String"/>
  <relational_:Type name="Integer"/>
  <relational_:Table name="Member_emailAddresses">
    <col name="memberId" type="/3"/>
    <col name="emailAddresses" type="/2"/>
  </relational_:Table>
  <relational_:Table name="Family_members">
    <col name="familyId" type="/3"/>
    <col name="membersId" type="/3"/>
  </relational_:Table>
</xmi:XMI>
```

Class 2 Relational Incremental





Completeness and **Correctness** and **Labeling**

Correct: commuting batch and incremental transformations

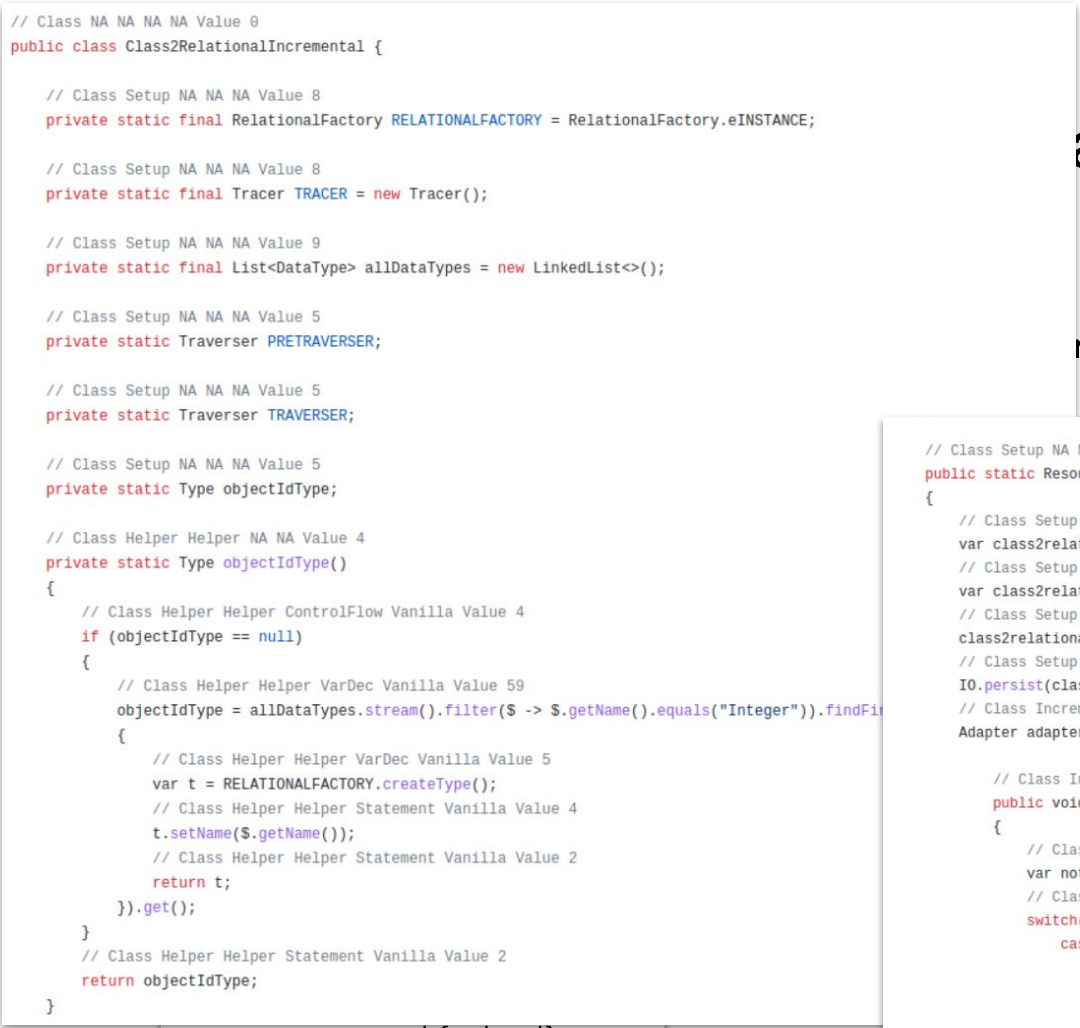
Complete: support different incremental behaviors

Labeling: for analyzing the transformation complexity

abeling

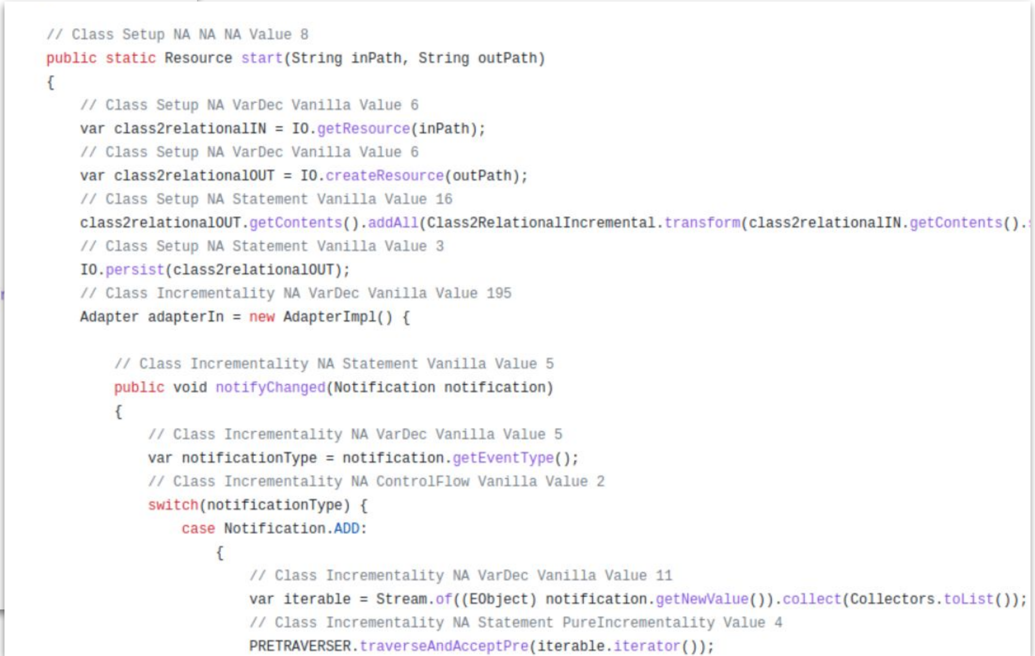
: source code for making transformation work

rsal: source code for traversing input to find element(s) to transform



```
// Class NA NA NA NA Value 0
public class Class2RelationalIncremental {

    // Class Setup NA NA NA Value 8
    private static final RelationalFactory RELATIONALFACTORY = RelationalFactory.eINSTANCE;

    // Class Setup NA NA NA Value 8
    private static final Tracer TRACER = new Tracer();

    // Class Setup NA NA NA Value 9
    private static final List<DataType> allDataTypes = new LinkedList<>();

    // Class Setup NA NA NA Value 5
    private static Traverser PRETRAVERSER;

    // Class Setup NA NA NA Value 5
    private static Traverser TRAVERSER;

    // Class Setup NA NA NA Value 5
    private static Type objectIdType;

    // Class Helper Helper NA NA Value 4
    private static Type objectIdType()
    {
        // Class Helper Helper ControlFlow Vanilla Value 4
        if (objectIdType == null)
        {
            // Class Helper Helper VarDec Vanilla Value 59
            objectIdType = allDataTypes.stream().filter($ -> $.getName().equals("Integer")).findFi
            {
                // Class Helper Helper VarDec Vanilla Value 5
                var t = RELATIONALFACTORY.createType();
                // Class Helper Helper Statement Vanilla Value 4
                t.setName($.getName());
                // Class Helper Helper Statement Vanilla Value 2
                return t;
            }).get();
        }
        // Class Helper Helper Statement Vanilla Value 2
        return objectIdType;
    }
```

col {ordered}



```
// Class Setup NA NA NA Value 8
public static Resource start(String inPath, String outPath)
{
    // Class Setup NA VarDec Vanilla Value 6
    var class2relationalIN = IO.getResource(inPath);
    // Class Setup NA VarDec Vanilla Value 6
    var class2relationalOUT = IO.createResource(outPath);
    // Class Setup NA Statement Vanilla Value 16
    class2relationalOUT.getContents().addAll(Class2RelationalIncremental.transform(class2relationalIN.getContents().
    // Class Setup NA Statement Vanilla Value 3
    IO.persist(class2relationalOUT);
    // Class Incrementality NA VarDec Vanilla Value 195
    Adapter adapterIn = new AdapterImpl() {

        // Class Incrementality NA Statement Vanilla Value 5
        public void notifyChanged(Notification notification)
        {
            // Class Incrementality NA VarDec Vanilla Value 5
            var notificationType = notification.getEventType();
            // Class Incrementality NA ControlFlow Vanilla Value 2
            switch(notificationType) {
                case Notification.ADD:
                    {
                        // Class Incrementality NA VarDec Vanilla Value 11
                        var iterable = Stream.of((EObject) notification.getNewValue()).collect(Collectors.toList());
                        // Class Incrementality NA Statement PureIncrementality Value 4
                        PRETRAVERSER.traverseAndAcceptPre(iterable.iterator());
```

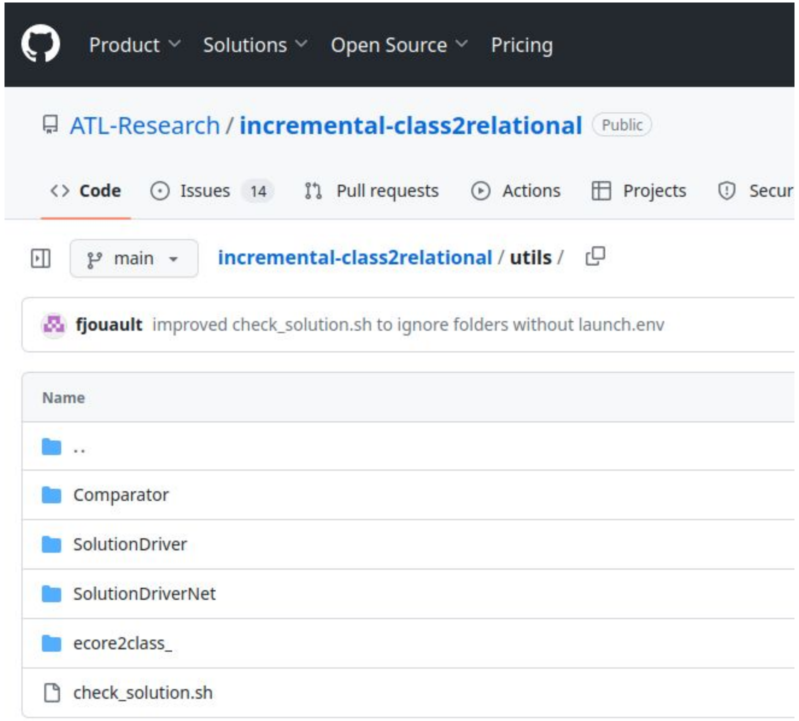
Solution(s) overview

- GPL: 2 (Java, C#)
- MTL: 2 (ATL, Cheptre)
- Hybrid: 2 (BxtendDSL, ATOL)

Solution(s) evaluation

- C#
  - correctness1 -> order (objects+attributes) et pas de clés
  - correctness2 -> order et pas de clés
  - correctness3-> order et pas de clés
  - correctness4 -> pas de clés (batch), table multivaluée présente (inc)
  - correctness5 -> order et pas de clés
  - correctness6 -> pas de clés,
    - (exp1) attr type null non supprimé
    - (exp2) attr type null pas de type
  - correctness7 -> pas de clés, order
    - (exp1) ok
  - correctness8 -> order, pas de clés, pas d'ajout de la classe à root
  - correctness9 -> order, pas de clés
    - (exp1) new attr non ignoré
    - (exp2) attr présent dans la table Person
  - correctness10 -> order, pas de clés
    - (exp1) attr ignoré
  - correctness11 -> order, pas de clés
    - (exp1) ok
  - correctness12 -> order, pas de clés
    - (exp1) ok
  - correctness13 -> order, pas de clés
    - (exp2) attr membersId has no type
  -

Solution(s) evaluation


Product Solutions Open Source Pricing

ATL-Research / incremental-class2relational Public

Code Issues 14 Pull requests Actions Projects Secur

main incremental-class2relational / utils /

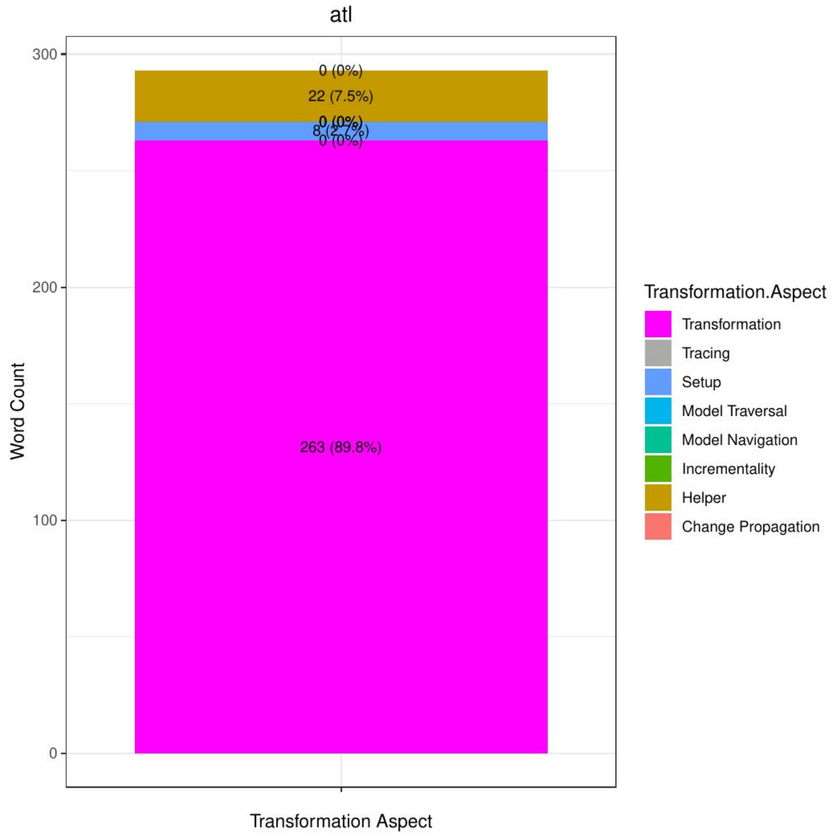
fjouault improved check_solution.sh to ignore folders without launch.env

Name
..
Comparator
SolutionDriver
SolutionDriverNet
ecore2class_
check_solution.sh

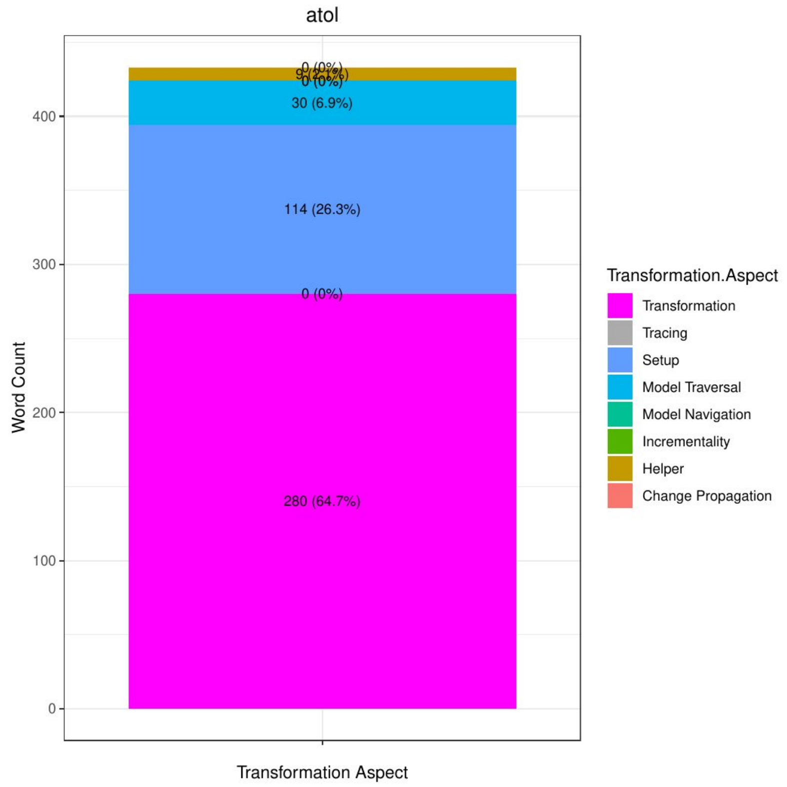
NMF

- correctness1 -> ok
- correctness2 -> table attribute ‘name’ is present but empty (exp1 has no ‘name’ attribute)
- correctness3-> ok
- correctness4 -> ok
- correctness5 -> emailAdresses column is deleted but firstname attribute is present (?)
- correctness6 -> order types + attributes
  - (exp1)
  - (exp2) closestFriend is owned by Person but no type
- correctness7 ->
  - (exp1) ok
- correctness8 -> ok (error given but comes from next run)
- correctness9 -> order types + attributes
  - (exp1)
  - (exp2) attribute name is different(?)
- correctness10 ->
  - (exp1) ok
- correctness11 ->
  - (exp1) ok
- correctness12 ->
  - (exp1) ok
- correctness13 -> order types + attributes
  - (exp2) attr members no type
-

Labeling Analysis

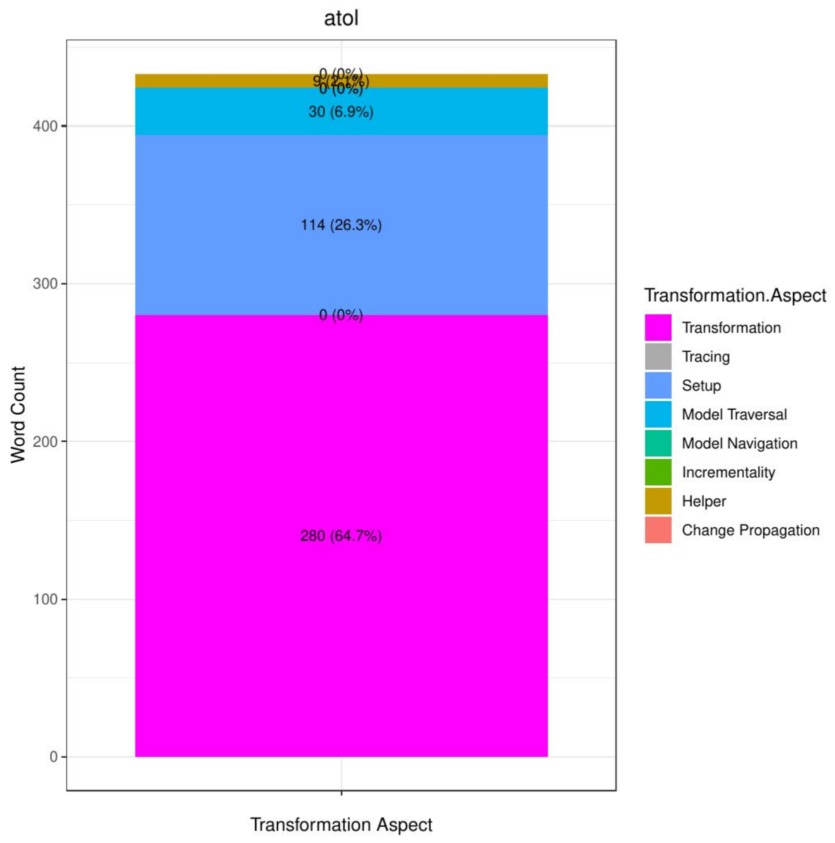


ATL

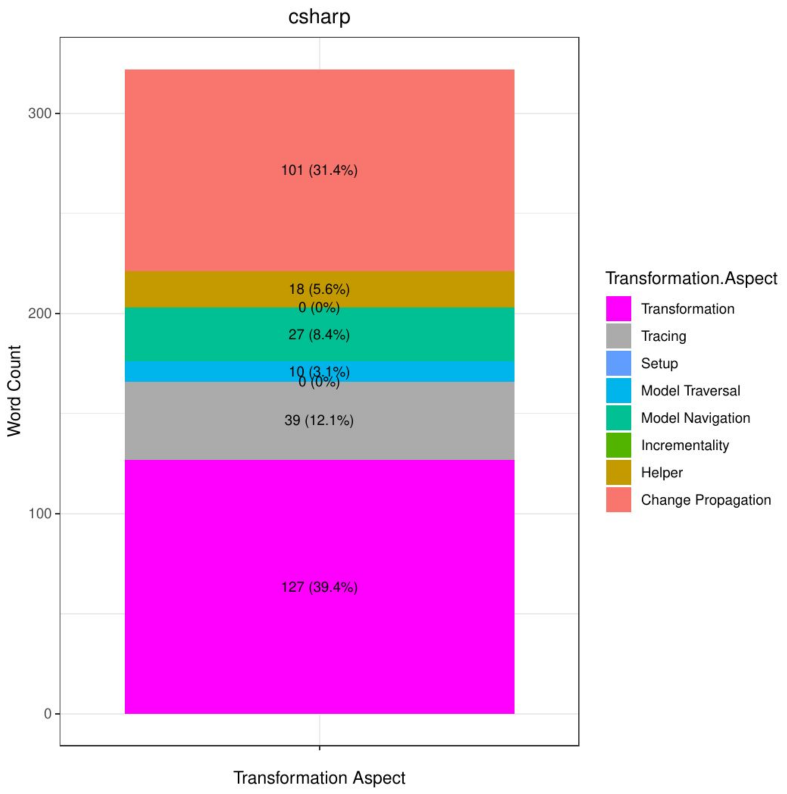


ATOL

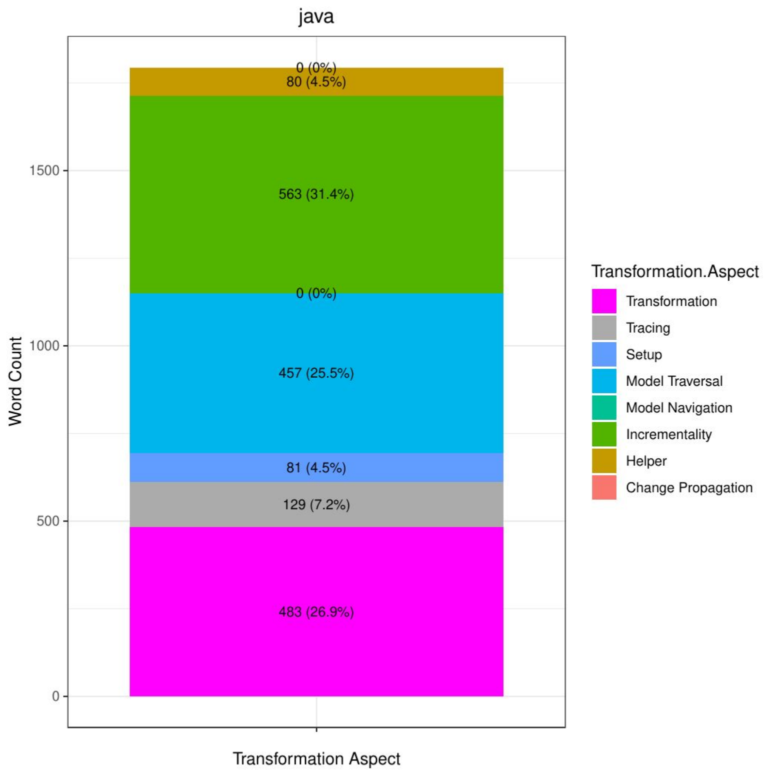
ATOL


atol
0 (0%)
9 (2.1%)
0 (0%)
30 (6.9%)
114 (26.3%)
0 (0%)
280 (64.7%)
Word Count
0
100
200
300
400
Transformation Aspect
Transformation.Aspect
Transformation
Tracing
Setup
Model Traversal
Model Navigation
Incrementality
Helper
Change Propagation

Labeling Analysis

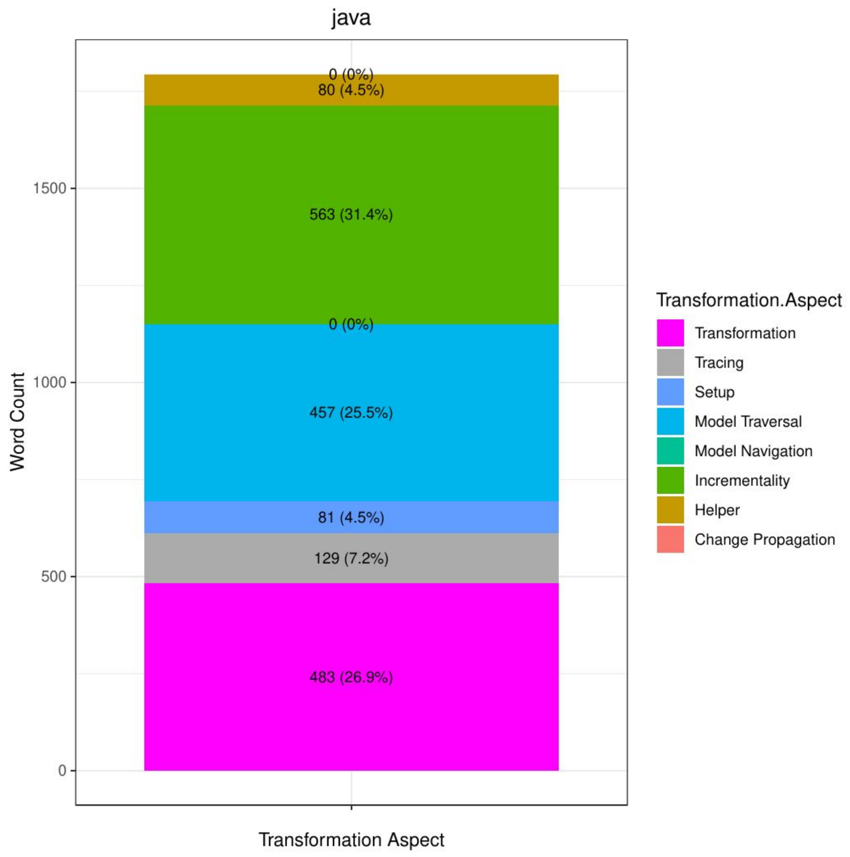


C#

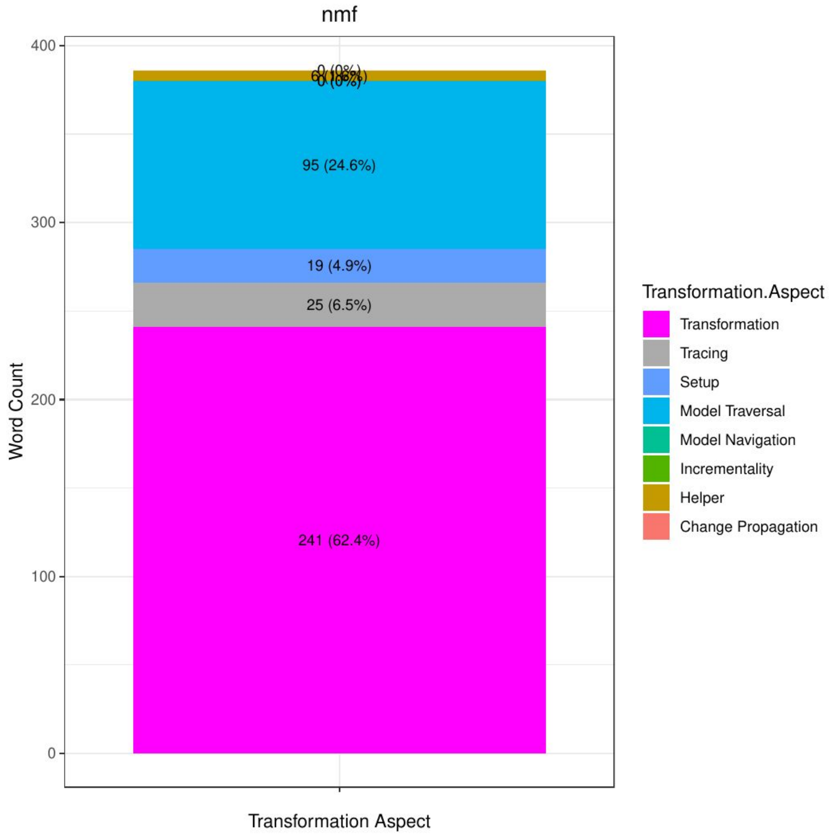


Java

Java


java
0 (0%)
80 (4.5%)
563 (31.4%)
0 (0%)
457 (25.5%)
81 (4.5%)
129 (7.2%)
483 (26.9%)
1500
1000
500
0
Word Count
Transformation Aspect
Transformation.Aspect
Transformation
Tracing
Setup
Model Traversal
Model Navigation
Incrementality
Helper
Change Propagation

NMF


nmf
Word Count
Transformation Aspect
0 (0%)
6 (1.6%)
0 (0%)
95 (24.6%)
19 (4.9%)
25 (6.5%)
241 (62.4%)
0
100
200
300
400
Transformation.Aspect
Transformation
Tracing
Setup
Model Traversal
Model Navigation
Incrementality
Helper
Change Propagation

Thanks for listening :) !

Awards?

Most complete

Best in GPL

Best in MTL

Best overall

…