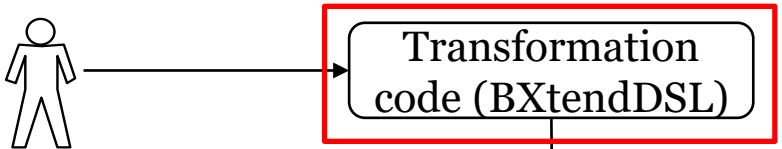
A BXtendDSL Solution to the Incremental MTL vs. GPLs Case

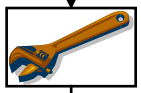
Thomas Buchmann

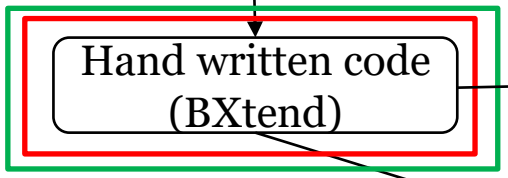
BXtendDSL

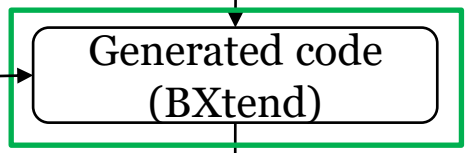
- Small and lightweight external DSL
- Using BXtendDSL the transformation developer essentially declares correspondences between elements of source and target models
- BXtendDSL is **intentionally** incomplete
  - Usually it is not possible to solve a transformation completely on the declarative level (as this would require a more expressive and comprehensive language)
  - Rather, from a transformation definition written in BXtendDSL code on top of the BXtend framework is generated
  - Subsequently, the generated code is extended with manually written imperative code

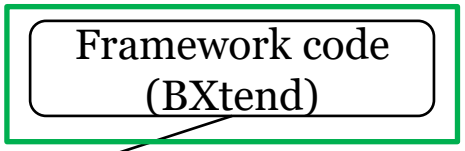


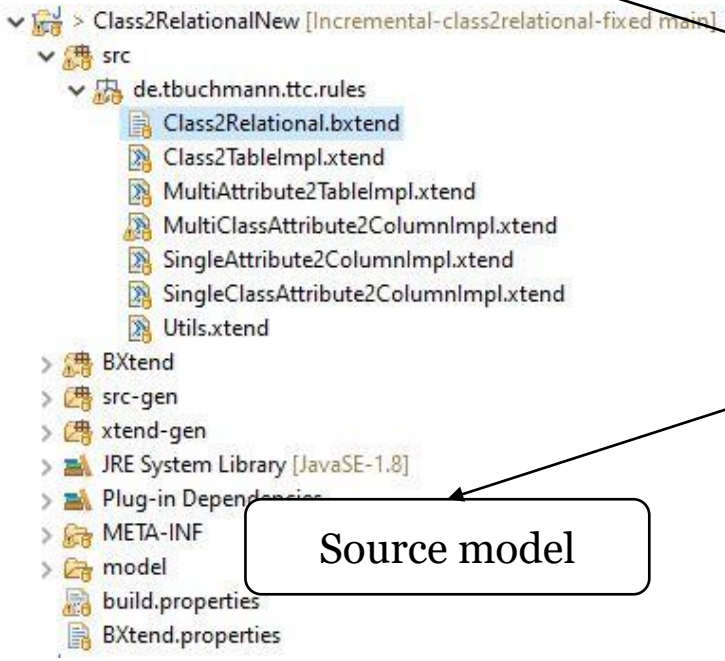
Transformation developer

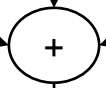


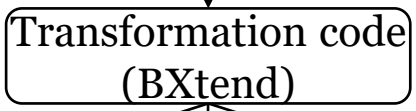


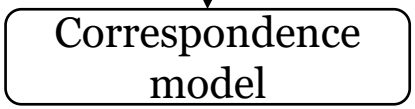












BXtendDSL Solution: Declarative Layer

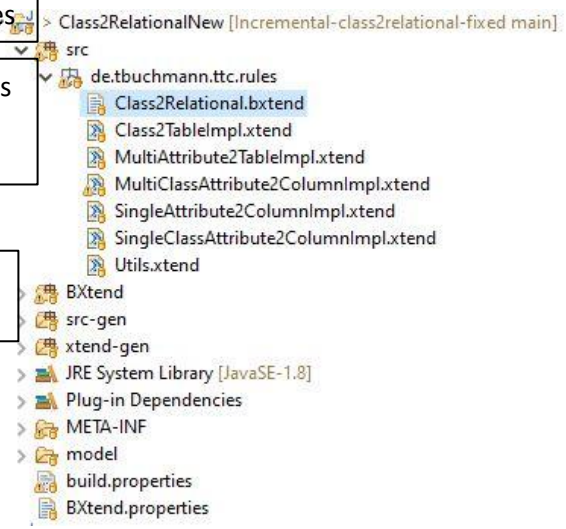
```
1  sourcemodel "Class"
2  targetmodel "Relational"
3
4  // Transformation
5  rule DataType2Type
6      src DataType dt;
7      trg Type t;
8
9      dt.name --> t.name;
10
11 rule SingleAttribute2Column
12     src Attribute att | filter;
13     trg Column col;
14
15     att.name --> col.name;
16     {att.type : DataType2Type} --> {col.type : DataType2Type};
17
18 rule MultiAttribute2Table
19     src Attribute att | filter;
20     trg Table tbl;
21
22     att.name att.owner --> tbl.name;
23     att.name att.type att.owner --> tbl.col;
24
25 rule SingleClassAttribute2Column
26     src Attribute att | filter;
27     trg Column col;
28
29     att.name att.type --> col.name;
30     att.name att.type --> col.type;
31
32 rule MultiClassAttribute2Column
33     src Attribute att | filter;
34     trg Table t;
35         Column id | creation;
36         Column fk | creation;
37
38     att.name att.owner att.type --> t.name;
39     att.name att.owner --> id.name;
40     att.name att.owner --> fk.name;
41
42 rule Class2Table
43     src Class clz;
44     trg Table tbl | creation;
45
46     clz.name --> tbl.name;
47     {clz.attr : SingleAttribute2Column, SingleClassAttribute2Column, MultiAttribute2Table} --> tbl.col;
48
```

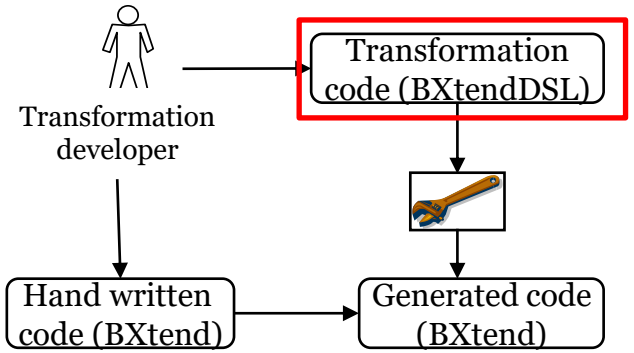
define source and target metamodels

Transformation of DataTypes and Types

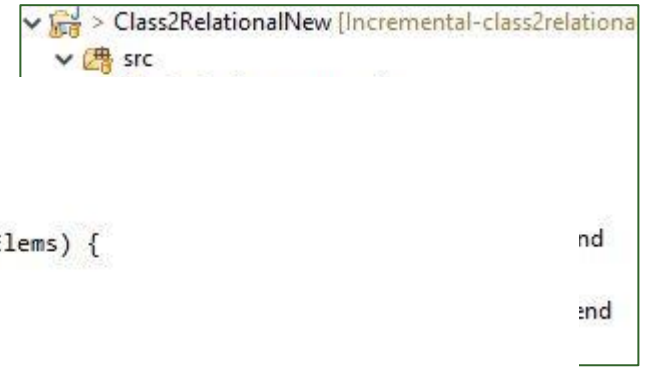
Modifiers for creating stubs on the imperative layer

Mappings of attributes / references





BXtendDSL Solution: Imperative Layer




```
class SingleAttribute2ColumnImpl extends SingleAttribute2Column {
    new(Cla
        sup
    }

    // Mode
    overrid
        !(a
    }
}
```

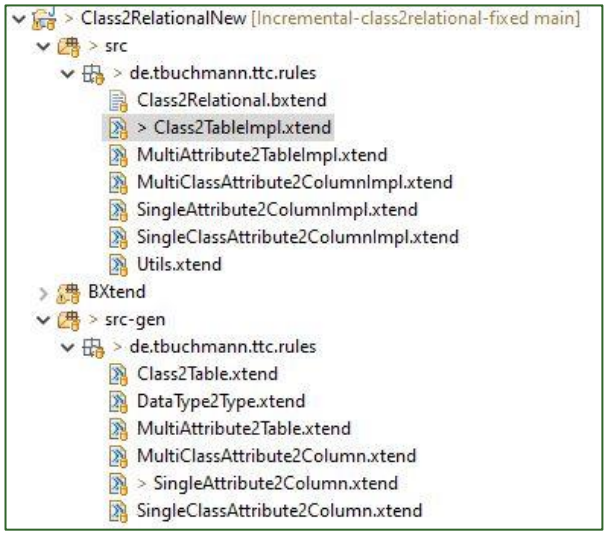
```
abstract class SingleAttribute2Column extends Elem2Elem {
    new(Class2Relational trafo) {
        super("SingleAttribute2Column", trafo)
    }

    override CorrModelDelta sourceToTarget(Set<EObject> _detachedCorrElems) {
        this.createdElems = new ArrayList<EObject>()
        this.spareElems = new ArrayList<EObject>()
        this.detachedCorrElems = _detachedCorrElems

        val _matches = new ArrayList<Source>()
        for (att : sourceModel.allContents.filter(typeof(atl.research.class_.Attribute)).filter[filterAtt(it)]
                .toIterable()
        ) {
            _matches += new Source(att)
        }

        for (_match : _matches) {
            val att = _match.att

            val _corr = wrap(att).updateOrCreateCorrSrc()
            val _colType = new CorrElemType("Column", false)
            val _trg = _corr.getOrCreateTrg(_colType)
            val col = unwrap(_trg.get(0) as SingleElem) as atl.research.relational_.Column

            col.setName(att.getName())
            att.getType()?.corr?.assertRuleId("DataType2Type")
            col.setType(if (att.getType() !== null && att.getType().corr.ruleId == "DataType2Type") {
                unwrap(att.getType().corr.target.get(0) as SingleElem) as atl.research.relational_.Type
            })
        }

        return new CorrModelDelta(this.createdElems, this.spareElems, this.detachedCorrElems)
    }
```

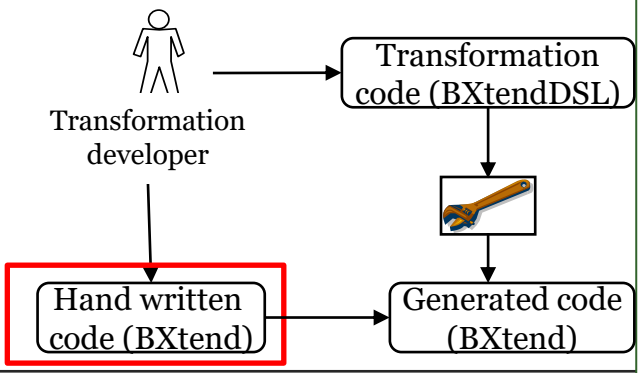
Transformation developer

Hand written code (BXtend)

(BXtend)

BXtendDSL Solution: Imperative Layer






```
override protected colFrom(List<Column> attSinCol, List<Column> attSinCol_2, List<Table> attMulTbl,
    Table parent
) {
    // Helper 3
    val columnsList = newArrayList
    // Helper 10
    if (!parent.col.empty) {
        var key = parent.col.get(0)
        columnsList += key
    }

    // Transformation 4
    for (Column c : attSinCol) {
        // Tracing 11
        var obj = unwrap(c.corr.source.get(0) as SingleElem) as Attribute
        // Transformation 14
        if (obj.type !== null) {
            columnsList += c
        }
        else {
            c.owner = null
            EcoreUtil.delete(c, true)
        }
    }

    // Transformation 4
    for (Column c : attSinCol_2) {
        // Tracing 11
        var obj = unwrap(c.corr.source.get(0) as SingleElem) as Attribute
        // Transformation 14
        if (obj.type !== null) {
            columnsList += c
        }
        else {
            EcoreUtil.delete(c, true)
        }
    }
    // Transformation 4
    for (Table t : attMulTbl) {
        // Tracing 11
        var obj = unwrap(t.corr.source.get(0) as SingleElem) as Attribute
        // Transformation 8
        if (obj.type === null)
            EcoreUtil.delete(t, true);
    }
    // Transformation 3
    new Type4col(columnsList)
}
```

Evaluation

- Qualitative Analysis

Test	Correctness	Completeness
Correctness1	ok	expected1.xmi
Correctness2	ok	no match
Correctness3	ok	expected1.xmi
Correctness4	ok	expected1.xmi
Correctness5	ok	no match
Correctness6	ok	no match
Correctness7	error	no match
Correctness8	ok	expected1.xmi
Correctness9	ok	no match
Correctness10	ok	expected2.xmi
Correctness11	ok	no match
Correctness12	ok	no match
Correctness13	ok	no match
Correctness Couple	ok	no expected
Correctness Full	ok	no expected
Scale 1	error	no expected
Scale 200	ok	no expected
Scale 2000	ok	no expected

Evaluation

- Quantitative Analysis: LOC Metrics

Metric	BXtendDSL Declarative	BXtendDSL Imperative	Total
LOC	42	131	173
#Words	113	388	501
#Characters	866	3344	4210

Conclusion

- We used BXtendDSL – a DSL for specifying bidirectional and incremental model transformations
- The case only required to specify the forward direction
- The declarative language needs an extension to allow navigation to container elements
  - and a way to loosen strict checking of applied rules when assigning previously transformed elements to references
- Find the solution on (fixed, in order to run with the provided framework)
  - https://github.com/tbuchmann/Incremental-class2relational-fixed

Thank you!

Any Questions?