*an object structure* $os \in [\![cd]\!]^{cw}$ *with an object* $o \in os.obj$ *such that* $c \in o.types$*. CD* $cd$ *is finitely-satisfiable iff every class* $c \in cd.classes$ *is finitely-satisfiable.*

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

### 4.4. Open-World Semantics

We consider an expansion-based open-world semantics for CDs (Nachmann et al. 2022). This not only ensures consistency of type-inheritance within object structures, but also more accurately reflects the notion of underspecification.

**Definition 4.6** (Open-World Semantics of Class Diagrams)**.** *An object structure* $os$ *is an open-world instance of a Class Diagram* $cd$ *iff there exists an expansion* $cd^x$ *of* $cd$ *(see Def. 4.8) with* $os \in [\![cd^x]\!]^{cw}$*. The open-world semantics of* $cd$*, denoted as* $[\![cd]\!]^{ow}$*, is the set of all open-world instances.*

The notion of open-world refinement is then defined analogous to its closed-world counterpart:

**Definition 4.7** (Open-World Refinement)**.** *We say that a Class Diagram* $A$ *is a refinement of a Class Diagram* $B$ *under the open-world assumption iff* $[\![A]\!]^{ow} \subseteq [\![B]\!]^{ow}$

Intuitively, a CD-expansion preserves all existing classes, associations and extends relations, but it permits additions.

**Definition 4.8** (Expansion)**.** *A Class Diagram* $cd^x$ *is an expansion of a Class Diagram* $cd$ *iff*

1. $cd.classes \subseteq cd^x.classes$,
2. $cd.abstract \subseteq cd^x.abstract$,
3. $cd.assoc \subseteq cd^x.assoc$
4. $cd.bidir \subseteq cd^x.bidir$
5. $\forall c_1, c_2 \in cd.classes : c_1 \prec_{cd} c_2 \implies c_1 \prec_{cd^x} c_2$,
6. $\forall a \in cd.assoc : cd.cardL(a) = cd^x.cardL(a) \wedge cd.cardR(a) = cd^x.cardR(a)$

Referring back to our motivating example in Sect. 3, adding `OtherSystem` as a subclass to `System` in CD $v3$ constitutes an expansion of CD $v3$. The object structure *witness* from Figure 4 is an instance of this expansion in the closed world according to Def. 4.3, and thus it is an open-world instance of $v3$. On the other hand, when considering CD, $v2$, the link (`tw0`, `original`, `sys0`) violates the semantics of the association (`Digital`, `twins`, `original`, `Machine`) as `Machine` $\notin$ `sys0`$.types$ (see Def. 4.3). This might not be immediately clear when looking at Figure 4, as super-types are usually not depicted in ODs. To avoid ambiguity, our implementation uses the stereotype «$instanceof = \ldots$» in diff-witnesses produced by the open-world operator to list all types an object instantiates.

### 4.5. Overlapping Associations

Overlapping associations are permitted by the abstract syntax we defined in Sect. 4.1. Concerning our previous semantics definitions, we distinguish between refining/super-associations and conflicting association.

**Definition 4.9** (Refining and Super-Association)**.** *An association* $a' = (c_1', r_1', r_2, c_2')$ *refines another association* $a := (c_1, r_1, r_2, c_2)$ *with respect to a class diagram* $cd$ *iff (1)* $c_1 \prec_{cd}^* c_1', c_2 \prec_{cd}^* c_2'$ *and (2)* $a, a' \in cd.bidir \implies r_1 = r_1'$*. In this case, we also refer to* $a'$ *as a refining association and to* $a$ *as a super association of* $a$*.*

Refining association are a valid concept with regards to set-theory: an association restricts the set of objects that can be accessed via the role name, a refining association further restricts this set for a subclass. However, for implementation purposes, this type of overriding is unfortunately not supported by most object-oriented programming languages (*e.g.,* Java).

Note special cases of *redundant* super-associations whose constraints are implied by their refining associations and *conflicting* associations as defined in Def. 4.10.

**Definition 4.10** (Conflicting Associations)**.** *An association* $a = (c_1, r_1, r_2, c_2)$ *conflicts with another association* $a' = (c_1', r_1', r_2, c_2')$ *with respect to a class diagram* $cd$ *iff* $c_1 \prec_{cd}^* c_1'$ *and either (1)* $c_2 \nprec_{cd}^* c_2'$ *or (2)* $a, a' \in cd.bidir$ *and* $r_1 \neq r_1'$*. We say that a class diagram* $cd$ *is conflict-free, if it contains no conflicting associations.*

Unlike refining associations, conflicting associations cannot be instantiated as they induce contradicting constraints on the formal semantics of a CD.

## 5. Alloy-Based Semantic Differencing

Alloy is a textual modeling language based on relational first-order logic (Jackson, Daniel 2006). Models are defined in Alloy modules and consist of signature declarations, fields, predicates, and facts. The Alloy Analyzer can be used to find instances of signatures that satisfy predicates and facts within a specified search-space.

`CDDiff` translates two CDs into an Alloy module in order to find diff-witnesses using the Alloy Analyzer (Maoz et al. 2011b; Kautz et al. 2017). The semantics of each CD are expressed by a predicate in the Alloy module. When computing a diff-witness the predicate of the first CD has to hold, while the predicate of the second CD must not. In the translation of (Kautz et al. 2017), an abstract signature `Obj` is used to represent objects in the semantic domain of a CD. Each class in the input-CDs is translated into a signature that extends `Obj`. Role names and attribute names in the input-CDs are translated to signatures extending an abstract signature `FName` (field name). The signature `Obj` contains the function `get` that given a field name, returns a set of objects, values and enumeration-values. Predicates restrict the `get`-function such that it accurately encodes the semantics of associations and attributes.

Note that for executing `CDDiff`, a diff-size has to be specified that limits the number of instances for each top-level signature (*e.g.,* `Obj`). If not specified by the user, a heuristic based on the number of classes and associations in the input-CDs is used.

In order to accommodate open-world semantic differencing, we add an abstract signature `Type`, as shown in Listing 1, that allows us to specify and track the types instantiated by an object. The `Type` signature is extended by a singleton signature for each class in the input-CDs. Furthermore, an additional predicate is generated that allows specifying the reflexive transitive hull of

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

the extends relation via the `super`-attribute for each input-CD, respectively. For the translation to ODs, we utilize the stereotype «*instanceof* = ...» to depict all types of an object.

For example, the class `DataModel` in the CD $v3$ (Figure 3) would be translated into a sub-signature of `Obj`. A corresponding `lone` sub-signature of `Type`, with the name `Type_DataModel`, would also be generated, containing both itself and `Type_Model` in field `Type_DataModel.super`. A full example-module `cwdiff_DT3_DT2_module.als` is included in the `doc`-folder of the GitHub-project.

```
// object signature now includes a type-attribute
abstract sig Obj {
  get : FName -> {Obj + Val + EnumVal},
  type: Type }

// introduce new abstract sig Type
abstract sig Type {
  super: set Type,
  inst : set Obj}

// define the super-types for a set of objects
pred ObjTypes[obj: set Obj, types: set Type]{
  all o:obj| o.type.super = types}

// specify the instances of a type
fact InstancesOfTypes {
  all t: Type |
    t.inst = {o:Obj | t in o.type.super}
}
```

**Listing 1** Modifications to the generic part of the Alloy Module to enable multi-instance CD-semantics.

The complete, modified translation from CDs to `Alloy` with signature `Type` (e.g., for closed-world semantic differencing) is is implemented in class `CD2AlloyGenerator`[1].

## 6. Solution I: Open-World Semantic Differencing with Alloy

In order to realize open-world semantic differencing in Alloy, we reduce the problem to a more restrictive, finite version of itself. This is possible since elements that are absent from both input-CDs cannot by themselves induce a semantic difference, hence we need, for the most part, only consider elements that are modelled in either CD. As such, we only use existing class and role names and permit Alloy to add new elements to the set of associations and the extends-relation. To ensure completeness, we add a new non-abstract subclass to each abstract class in the CD as well as an additional dummy class to each input-CD beforehand. This allows the Alloy Analyzer to consider all relevant expansions of the input-CDs and thus detect any semantic differences in the open world.

Note that the added non-abstract sub-classes allow indirect instantiation of abstract classes without additional constraints. Moreover, the added dummy class represents an arbitrary new class previously not present in either input-CD and is needed for certain edge cases, *e.g.,* consider the situation in Figure 5 where adding the class `Dummy` to CD $P1$ and reusing the role-name `knows` from $P2$ allows the operator to detect the semantic difference from $P1$ to $P2$ exemplified by the OD $PDiff$. In $PDiff$, we find that `Person bob knows d0`, which is invalid according to $P2$, as `b0` is not an instance of `Person`.

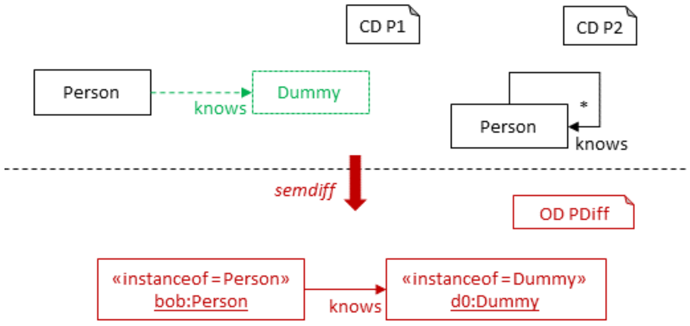

**Figure 5** Example for semantic difference induced by existing association and the added dummy-class.

The correctness and soundness for this approach follow directly from the correctness and soundness of the reduction-based approach introduced in Sect. 7.

In order to implement the Alloy-based approach, we extend the class `CD2AlloyGenerator` via the subclass `OpenWorldGenerator`[1] and override some of the existing translation-procedures. First and foremost, we no longer restrict the `get`-function to associations (and attributes) defined in the respective CD, nor do we restrict objects to instances of classes in the first input-CDs. The inheritance-relation is also left underspecified. Instead, we now need to ensure anti-symmetry, reflexivity and transitivity of any inheritance-relation, as well as consistent use of field names across all objects of the same type (see Listing 2). A full example-module `owdiff_DT3_DT2_module.als` is included in the `doc`-folder of the GitHub-project.

```
// ensures inheritance remains anti-symmetric
fact NoCyclicInheritance {
  all t1: Type | all t2: Type |
   {t2 in t1.super} && {t1 in t2.super} => {t1 = t2}
}

// reflexivity and transitivity of inheritance
fact ReflexiveTransitiveInheritance {
  all t1: Type | t1 in t1.super
  all t1: Type | all t2: Type |
    {t2 in t1.super} => {t2.super in t1.super}
}

// consistent use of field names for all objects
// of the same type
fact ConsistencyOfGet {
  all src: Obj | all q : FName |
    some src.get[q]  => {
    {src.get[q] in EnumVal and
      {one e:Enum |
        ObjAttrib[src.type.inst,q,e.values]}} or
    {src.get[q] in Val and
      {one v:Val | ObjAttrib[src.type.inst,q,v]}} or
    {src.get[q] in Obj and
```

---

[1] https://github.com/MontiCore/cd4analysis/tree/develop/src/cddiff/java/de/monticore/cddiff/cd2alloy/generator

```
      {some target : Type | all o : src.type.inst |
        o.get[q] in target.inst}}}
}
```

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

**Listing 2** Additional type-related facts for open-world semantic differencing.

## 7. Solution II: Open-to-Closed-World Reduction

Our second approach for open-world semantic differencing reduces the analysis to closed-world semantic differencing via a transformation of the input-CDs. The transformation relies on the existence of a common abstract super-class $Object$. We specify the transformation as follows:

**Transformation** Let $traf : CD \times CD \to CD \times CD$ such that for $traf(A, B)$:

1. Every class exclusive to $B$ is added to $A$ as a non-abstract class without extends-relations.
2. We set $c \prec_A Object$ for every class $c$ in $A$ without extends-relation.
3. We add a new non-abstract subclass $c_{sub}$ to every abstract class $c$ in $A$.
4. For each association $(c_1, r_1, r_2, c_2)$ in $B$, we add an association $(c_1, r_1, r_2, Object)$ with underspecified cardinalities to $A$, unless this conflicts with existing associations.
5. Every class exclusive to $A$ is added to $B$ as a non-abstract class without extends relations.
6. For each class in $B$ and each of its super-classes in $A$ we add an extends-relation, unless this causes inheritance cycles. Afterwards we remove redundant extends-relations.
7. We add each associations from $A$ to $B$ but leave cardinalities underspecified, unless this conflicts with existing associations.

The purpose of this transformation is to (1) prevent the existence of closed-world witnesses that are not also open-world witnesses by copying elements from $A$ to $B$ without causing syntactical errors and (2) add elements to $A$ that induce a closed-world witness iff a semantic difference exists under an open-world assumption. The latter is achieved by, *e.g.*, adding the exclusive classes from $B$ to $A$ without extends relation in order to induce type-differences as well as introducing super-associations of associations exclusive to $B$ to $A$ that target the common abstract super-class. Moreover, the addition of a non-abstract subclass to each abstract class in $A$ allows their indirect instantiation without additional constraints.

Consider the CDs $A$ and $B$ depicted in Figure 6 as a concrete example: The transformation adds the class `Task` from $B$ to $A$ to allow its instantiation via the closed-world operator. The common abstract super-class `Object` and its non-abstract subclass $ObjSub$ are added, as well. Moreover, an association from `Employee` to `Object` with role name `todo` is also added to $A$. This association allows the closed-world operator to create corresponding links between objects of type `Employee` and objects of any other type. To prevent incorrect witnesses, the elements of the resulting CD $A'$ that are missing in CD $B$ are copied. This includes the classes `Object`, `ObjSub` and `Manager`, as well as corresponding associations and extends-relation. The OD $todoDiff$ presents a potential diff-witness found by the closed-world operator after transforming the input CDs. Note that the `Task accounting` is not a `todo` for any `Employee`. Instead, we find that the `Employee bob` has a `todo` that is not a `Task`. Both of these circumstances are in violation of the semantic constraints implied by the association in the CD $B$. As such, we find that $todoDiff \in [\![A]\!]^{ow} \setminus [\![B]\!]^{ow}$.

We now demonstrate that this transformation in combination with a closed-world differencing operator is sufficient to detect open-world semantic differences in conflict-free finitely-satisfiable CDs.

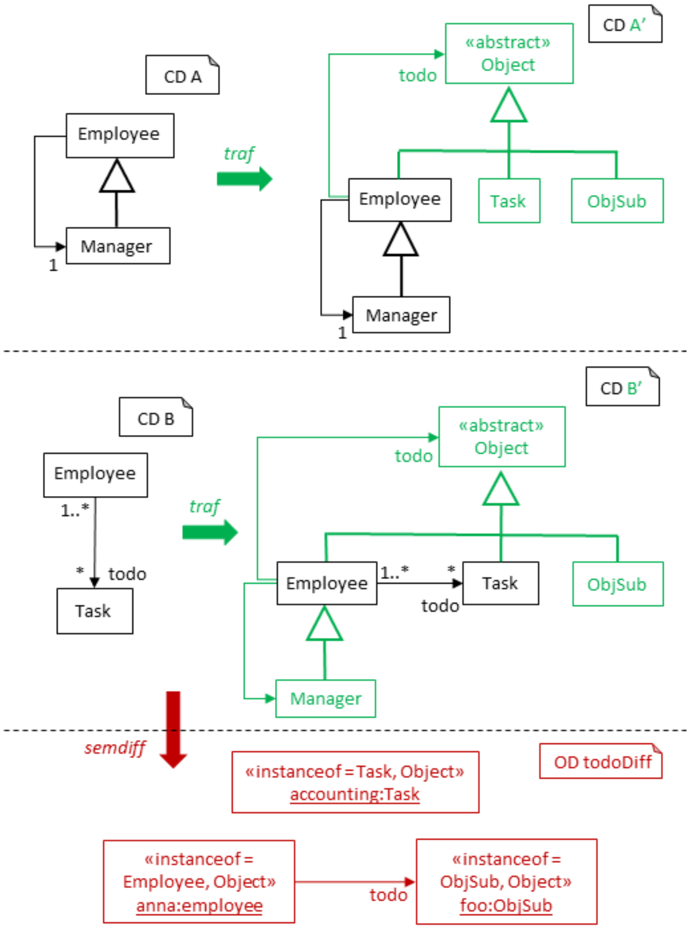

**Figure 6** Example for reduction-based diff.

In the following let $A$ and $B$ be arbitrary conflict-free finitely-satisfiable CDs and let $(A', B') := traf(A, B)$. We first show that finite satisfiability of the first input-CD $A$ is preserved by the transformation.

**Lemma 7.1.** *$A'$ is conflict-free and finitely-satisfiable.*

*Proof.* By construction $A'$ is conflict free. Since all associations in $A'$ that are not present in $A$ have no cardinality constraints, it follows that for every object structure $os \in [\![A]\!]^{cw}$, we find that $os \in [\![A']\!]^{cw}$ holds, as well. Thus, any class that is both in $A$

and $A'$ is finitely-satisfiable in $A'$. Furthermore, any class in $A'$ that is not present in $A$, extends at most 1 class. We conclude that these classes must also be finitely-satisfiable in $A'$. ☐

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

Next we proof that the reduction is sound, *i.e.,* that any witness produced by the closed-world semantic differencing operator for the transformed input-CDs is a witness for a semantic difference under the open-world assumption.

**Theorem 7.2** (Soundness). *If* $os \in [\![A']\!]^{cw} \backslash [\![B']\!]^{cw}$ *then* $os \in [\![A]\!]^{ow} \backslash [\![B]\!]^{ow}$.

*Proof.* Since $A'$ is by construction an expansion of $A$, it follows that for every $os \in [\![A']\!]^{cw}$, we also have $os \in [\![A]\!]^{ow}$. Next, we proof by contradiction that for any $os \in [\![B]\!]^{ow} \cap [\![A']\!]^{cw}$, we also have $os \in [\![B']\!]^{cw}$. Assume this is not the case. Then there exists an object structure $os \in [\![B]\!]^{ow} \cap [\![A']\!]^{cw}$ such that $os \notin [\![B']\!]^{cw}$. First, we conclude that there exists an expansion $B^x$ of $B$ such that $os \in [\![B^x]\!]^{cw} \cap [\![A']\!]^{cw}$. Second, it follows that $os$ violates at least one condition outlined in Def. 4.3 with regards to $B'$. We now show that this cannot be the case:

1. $A'$ and $B'$ contain the same classes.
2. Since $os \in [\![B^x]\!]^{cw} \cap [\![A']\!]^{cw}$, for all $o \in os.obj$ and all $c \in A'.classes$, it holds that $o.type \prec^*_{A'} c$
   $\iff c \in o.types$
   $\iff o.type \prec^*_{B^x} c$.
   Moreover, since $\prec^*_{B^x}$ induces no inheritance-cycle, it then follows from the transformation $traf$ that
   $o.type \prec^*_{A'} c \iff o.type \prec^*_{B'} c$.
3. For every $(o_1, r_2, o_2) \in os.links$ there exists an association $a = (c_1, r_1, r_2, c_2) \in A'.assoc$ with $c_1 \in o_1.types$ and $c_2 \in o_2.types$ that does not conflict with any association in $B$. Consequently, $B'$ then also contains this association.

4./5. Every association in $B'$ that is not also present in $B$ has no cardinality constraints and is present in $A'$, instead.

6. Every bidirectional association in $B'$ is also a bidirectional association in $B$.

☐

Now all that is left is to demonstrate completeness of the reduction, *i.e.,* under the assumption that the closed-world semantic differencing operator finds a diff-witness if and only if a semantic difference in the closed world, the transformation ensures that an open-world diff-witness is found if a semantic difference exists in the open world.

By $\hat{cd}$ we denote a minimal extension of CD $cd$ to include super-class $Object$ (if not already included).

**Theorem 7.3** (Completeness). *If* $[\![A']\!]^{cw} \backslash [\![B']\!]^{cw} = \emptyset$ *then* $[\![\hat{A}]\!]^{ow} \backslash [\![\hat{B}]\!]^{ow} = \emptyset$.

*Proof.* This follows from Lemmas 7.4 and 7.5. ☐

**Lemma 7.4.** *If* $[\![A']\!]^{cw} \subseteq [\![B']\!]^{cw}$, *then* $[\![A']\!]^{ow} \subseteq [\![B']\!]^{ow}$.

*Proof.* Assume that $[\![A']\!]^{cw} \backslash [\![B']\!]^{cw} = \emptyset$. Since $A'$ and $B'$ share the same classes and $A'$ is finitely-satisfiable, $B'$ must also be finitely-satisfiable. It follows that $B'.abstract \subseteq A'.abstract$. This also means that $c_1 \prec_{A'} c_2 \iff c_1 \prec_{B'} c_2$, i.e. both CDs must have exactly the same inheritance hierarchy. Furthermore, every association in $A'$ must also be in $B'$. The same must hold for bidirectional associations, as well. Otherwise there would be an association in $A'$ that is in conflict with an association in $B'$. Since $A'$ is conflict-free, we would then be able to find an $os \in [\![A']\!]^{cw} \backslash [\![B']\!]^{cw}$. This, however, contradicts our assumption. By the same argument, it must follow that any non-redundant $b \in \hat{B}.assoc$ is also in $\hat{A}.assoc$. Additionally, for every $b = (c_1, r_1, r_2, c_2) \in B'.assoc$, there is a sub-set of associations

$$S = \{(c'_1, r'_1, r_2, c'_2) \in A'.assoc : c_1 \prec^*_{B'} c'_1\}$$

such that $\bigcap_{a \in S} A'.cardL(a) \subseteq B'.cardL(b)$ as well as $\bigcap_{a \in S} A'.cardR(a) \subseteq B'.cardR(b)$.

Since any abstract class $c \in A'.abstract$ has a unique subclass, we conclude that any addition done equally to both $A'$ and $B'$ will have no effect on the semantic difference, and thus for every expansion $(A')^x$ of $A'$ and every $os \in (A')^x$, we can find an expansion $(B')^x$ of $B'$ such that $os \in (B')^x$, as well. ☐

**Lemma 7.5.** *If* $[\![A']\!]^{ow} \backslash [\![B']\!]^{ow} = \emptyset$, *then* $[\![\hat{A}]\!]^{ow} \backslash [\![\hat{B}]\!]^{ow} = \emptyset$

*Proof.* Since $B'$ is an expansion of $\hat{B}$ and any expansion of $B'$ is also an expansion of $\hat{B}$, it follows that for all $os_b \in [\![B']\!]^{ow}$, we have $os_b \in [\![\hat{B}]\!]^{ow}$, as well. Now consider a $diff \in [\![\hat{A}]\!]^{ow} \backslash [\![\hat{B}]\!]^{ow}$. Without loss of generality, let $c_{sub} \notin o.types$ for all $o \in diff.obj$, then $diff \in [\![A']\!]^{ow} \backslash [\![\hat{B}]\!]^{ow}$ ☐

Note that, however, in the implementation of the closed-world operator `CDDiff` (Maoz et al. 2011b) all object models are bounded by a finite and typically small scope, which consequently also bounds the completeness of the open-world operator in practise.

### Extension: Variability Open-/Closed-World Differencing

We have extended our transformation with a variability mechanism allowing the combination of open- and closed-world semantics interpretation of elements in both input-CDs. The extension is configured via UML/P stereotypes. Individual classes can be marked as «*complete*», then no super-classes and outgoing associations can be added. Furthermore, the CD itself can be marked as «*complete*», as well. This prevents any additions to the set of classes, the set of associations and the extends relation. In this manner, we are able to select an input-CD or specific elements of the CD that should be considered under a closed-world assumption. This is useful when comparing a predecessor version of a CD to a *final version* that should not be expanded, or alternatively if specific class declarations should not be expanded from a predecessor to a successor version, because it might constitute a breaking change with respect to a current implementation or violate existing requirement specification.

For example, consider the CD $v3$ depicted in Figure 3: If we consider this CD as our *final version* and mark it as «*complete*», then no additions to the extends relation can be made. As such, any instance of the abstract class `System` must also be and instance of `Machine` (note that the class `OtherSystem` is not part of the CD at this point). Consequently, we have refined the

CD $v2$ depicted in Figure 2. This would otherwise not be the case as explained in Sect. 3.

In order to preserve completeness of the differencing operator, we extended the transformation to add outgoing dummy associations to classes in $A$ that were only marked as «*complete*» in $B$. This necessarily will induce a diff-witness.

[RPS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

## 8. Implementation and Evaluation

The implementation of our open-world semantic differencing approaches have been integrated into the `CDDiff`-tool which is part of the larger `CD4Analysis`-project developed at the Chair of Software Engineering at RWTH Aachen University. The project is publicly available at `https://github.com/MontiCore/cd4analysis`. `CDDiff` can be used as Java-library via the public methods that are provided by the `CDDiff` class or as a CLI-Tool via the `MCCD.jar`.

### 8.1. Running CDDiff for Open-World semantics

Recall the CDs from our motivating example in Sect. 3. Our implementation of the semantic differencing operator operates on the textual CD notation of `CD4Analysis` (Schindler 2012) with `cd`-files as input. We provide the files `DigitalTwin1.cd`, `DigitalTwin2.cd` and `DigitalTwin3.cd` in the `doc` folder of the `CD4Analysis` project.

Executing the `MCCD.jar` with the the following CLI-command, we can compute the semantic difference of `DigitalTwin2.cd` and `DigitalTwin1.cd`.

```
java -jar MCCD.jar -i DigitalTwin2.cd --semdiff \
  DigitalTwin1.cd --open-world
```

As expected, no diff-witnesses are produced as `DigitalTwin2.cd` is a refinement of `DigitalTwin1.cd` under the open-world assumption. The output message reads: `************* No diff-witnesses **************`.

The option `--open-world` uses the reduction-based approach introduced in Sect. 7 by default. Users may also specify the approach by choosing `reduction-based` or `alloy-based` as an argument.

Next we want to check whether `DigitalTwin3.cd` is a refinement of `DigitalTwin2.cd`. In order to reduce the potential size of diff-witnesses and thereby the search-space of the diff-operation, we may use the option `--diffsize`. If this option is not used, the maximum size of Alloy-solutions is set to twice the number of classes (and interfaces) in the input-CDs. We may also output the diff-witness as an `od`-file by specifying an output-directory with `-o`. More configuration options are documented in the `README` file. A link for downloading the `MCCD.jar` is also provided there.

```
java -jar MCCD.jar -i DigitalTwin3.cd --semdiff \
  DigitalTwin2.cd --open-world reduction-based \
  --diffsize 5 -o target/DT
```

A witness produced by the reduction-based implementation when executing this command is given in Listing 3. In addition to the witness, the implementation also outputs the transformed CDs for inspection.

```
objectdiagram witness_0 {

  <<instanceof="Object, SystemSub4Diff, System">>
  SystemSub4Diff0:SystemSub4Diff;

  <<instanceof="DigitalTwin, Object">>
  DigitalTwin0:DigitalTwin;

  <<instanceof="DigitalShadow, Object">>
  DigitalShadow0:DigitalShadow;

  <<instanceof="DataTrace, Object">>
  DataTrace0:DataTrace ;

  <<instanceof="DataModel, Model, Object">>
  DataModel0:DataModel ;

  link DigitalTwin0 -> (models) DataModel0;
  link DigitalTwin0 -> (original) SystemSub4Diff0;
  link DigitalTwin0 -> (shadows) DigitalShadow0;
  link DigitalShadow0 -> (traces) DataTrace0;
  link DataTrace0 (traces)<->(of) SystemSub4Diff0;
}
```

**Listing 3** A reduction-based diff-witness in the open-world semantics of `DigitalTwin2.cd` but not in the open-world semantics of the `DigitalTwin3.cd`

A witness produces by the Alloy-based implementation when executing the same command with the argument `alloy-based` instead of `reduction-based` is shown in Listing 4.

```
objectdiagram witness_0 {

  <<instanceof="DigitalShadow, DataTrace, Dummy4Diff
    , SystemSub4Diff, System">>
  SystemSub4Diff0:SystemSub4Diff;

  <<instanceof="ProcessModel, ModelSub4Diff,
    DigitalTwin, Dummy4Diff, Model">>
  ModelSub4Diff0:ModelSub4Diff;

  link SystemSub4Diff0 -> (original) SystemSub4Diff0;
  link SystemSub4Diff0 -> (original) ModelSub4Diff0;
  link SystemSub4Diff0 -> (of) SystemSub4Diff0;
  link SystemSub4Diff0 -> (traces) SystemSub4Diff0;
  link ModelSub4Diff0 -> (models) ModelSub4Diff0;
  link ModelSub4Diff0 -> (original) SystemSub4Diff0;
  link ModelSub4Diff0 -> (of) SystemSub4Diff0;
  link ModelSub4Diff0 -> (of) ModelSub4Diff0;
  link ModelSub4Diff0 -> (shadows) SystemSub4Diff0;
}
```

**Listing 4** An alloy-based diff-witness in the open-world semantics of `DigitalTwin2.cd` but not in the open-world semantics of the `DigitalTwin3.cd`

Unlike the witness from Listing 3, the one produced by the `alloy-based` implementation and shown in Listing 4 contains only two objects. These two objects instantiate multiple classes that were originally not related by inheritance in the input-CDs. This is a consequence of the Alloy Analyzer choosing to expand the extends-relation. Similarly, the role names of associations in the input-CDs have been reused for additional associations that are instantiated by links in the diff-witness, *e.g.,* the role name `original` is used to target the object `ModelSub4Diff` despite it not being an instance of type `System`. These additions may be considered to hinder browsing the diff-witness for relevant information on the semantic difference between the two input-CDs. In comparison, the diff-witness in Listing 3, which was

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

produced by the reduction-based implementation, contains no superfluous instances. We assume that, in general, the likelihood of superfluous instances is reduced for the reduction-based approach as the transformation employed by the reduction is deterministic and the closed-world operation that follows is not permitted to re-use role names or expand the extends-relation.

Before we conclude this sub-section, let us demonstrate a potential use of the stereotype *«complete»*. If we add `«complete»` at the start of line 3 in `DigitalTwin3.cd` and repeat the analysis, then no diff-witness is found. This is because now no additions to the extends-relation are permitted in the first input-CD and thus any instance of type `System` must also be an instance of type `Machine`.

## 8.2. Evaluation

We are interested in two research questions:

- RQ1: How does the size and readability of computed witnesses differ between the encoding and the translation approach?
- RQ2: How does the performance differ between the encoding and the translation approach?

First, our translations employ different approaches to adding classes and relations between classes for computing open-world semantics. These might have an impact on the size and readability of computed diff witnesses.

Second, we believe that a performance evaluation is worthwhile as our two implementations of an open-world CDDiff operator are quite different in the techniques employed. We compare the performance of the operators in terms of ratios of their running times. Note that a comparison to the closed-world CDDiff from (Maoz et al. 2011b) would not be very meaningful, as open-world and closed-world semantic differencing are quite different problems.

Finally, we present our validation of both implementations in Sect. 8.3 and discuss threats to validity in Sect. 8.4.

We performed all experiments on an 11th Gen Intel Core i7-1185G7 CPU, 3.0 GHz, with 32 GB RAM, running Windows 10. Our implementation of the CDDiff operator uses the latest stable release of Alloy 6.0 with SAT4J as a SAT solver in its default configuration.

### 8.2.1. Corpus of CDs

To evaluate our work, we have collected 25 pairs of consecutive versions of CDs (38 individual CDs) from different sources[2]. These sources are:

- 7 pairs of (11 individual) CDs from (Maoz et al. 2011b),
- 5 pairs of (10 individual) CDs manually created as part of this work to validate and evaluate general performance and edge cases, and
- 5 pairs of (10 individual) CDs of the CD4Analysis project[3]

Minor, syntactic modification had to be made to the latter two sets of CDs due to recent changes in the `cd4analysis` project.

### 8.2.2. RQ1: Size and Readability

It is not straight forward how to evaluate the size and readability of generated diff-witnesses. A user-study is not in the scope of this work and we, instead, propose objective measures that may be taken as proxies for the readability of diff-witnesses. First, we consider the diff-witnesses with many objects and many links to be more difficult to read than those with fewer. Second, we consider CDs with object with low numbers of types per object to be easier to read than those with many types, i.e., more instances of objects in complex inheritance hierarchies.

Note that the numbers of objects and types per diff-witness are constrained by the diff-size parameter as an upper bound, which is not the case for the number of links. The latter is only constrained by the elements of the CDs. For our experiments we use diff-size = 5 as a reasonable default parameter of `CDDiff`. Also note that the diff-witnesses produced by our implementation are not necessarily unique, as the implementation relies on Alloy, which gives no guarantees in this regard.

For the translation-based open-world semantics, computed diff-witnesses across all CDs of our corpus have an average of 1.750 objects, 4.517 links, and objects are of 2.754 types on average. For the Alloy-based open-world semantics, computed diff-witnesses have an average of 2.133 objects, 2.033 links, and objects are of 2.195 types on average. The direct comparison between the two approaches shows that the Alloy-based approach produces diff-witnesses with less objects and more links, as well as more types per object. We presume that this is (1) because the Alloy Analyzer can arbitrarily expand the extends-relation and (2) because role names can be reused for new associations. The results are witnesses that are less readable.

### 8.2.3. RQ2: Performance

We first tested the performance of both approaches using 5 pairs of synthetic CDs of increasing size. Each of these CDs was constructed to contain an equal number of classes and associations, as well as a proportional number of syntactic differences. Each diff operation was performed with two CDs of equal size ranging from 5 to 25 classes (in steps of 5). Table 1 (Alloy-based) and Table 2 (reduction-based) show the number of classes and associations of both input-CDs, as well as the corresponding running time in seconds for each open-world approach and various diff-sizes (ds, an upper limit on the size of diff-witnesses). Note that because of the Alloy Analyzer's non-determinism the reported running times may vary on different machines.

In order to better compare the two approaches regarding their performance we computed the running-time-ratios of Alloy-based to reduction-based diff and displayed them in Table 3. Note that while the running times of the reduction-based approach start out slightly worse than for the Alloy-based approach, as the diff-size increases, the former overtakes the latter in terms of performance. Moreover, it appears that running-times-ratio of Alloy-based diff to reduction-based diff worsens as the input-CDs increase in size. This seems to indicate a greater running time complexity of the Alloy-based approach.

---

[2] All CDs are available at: `https://github.com/MontiCore/cd4analysis/tree/develop/src/cddifftest/resources/validation`.

[3] These are all CDs with at least two revisions from `https://github.com/MontiCore/cd4analysis/tree/develop/doc` at commit `5f48096`.

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

| CD size | ds=3 | ds=5 | ds=10 | ds=15 |
|---|---|---|---|---|
| 5 | 0.847s | 1.031s | 1.969s | 3.155s |
| 10 | 0.516s | 1.018s | 3.068s | 9.411s |
| 15 | 1.120s | 2.755s | 12.091s | 3.1360s |
| 20 | 2.148s | 5.833s | 23.240s | 72.002s |
| 25 | 3.313s | 8.210s | 38.890s | 142.491s |

**Table 1** Running times of CDDiff for the Alloy-based open-world approach for increasing diff-sizes ds from 3 to 15 and increasing sizes of CDs from 5 to 25 classes

| CD size | ds=3 | ds=5 | ds=10 | ds=15 |
|---|---|---|---|---|
| 5 | 0.704s | 0.837s | 1.005s | 2.084s |
| 10 | 0.866s | 1.203s | 2.186s | 3.530s |
| 15 | 1.659s | 3.037s | 4.713s | 11.599s |
| 20 | 2.550s | 3.314s | 7.851s | 17.140s |
| 25 | 3.684s | 4.759s | 14.083s | 27.199s |

**Table 2** Running times of CDDiff for the reduction-based open-world approach for increasing diff-sizes ds from 3 to 15 and increasing sizes of CDs from 5 to 25 classes

We continued testing the performance of both approaches using the CDs from (Maoz et al. 2011b). More, specifically we performed the diff operations by comparing the successor version of each diagram to its direct predecessor. We display the running-time-ratios in Table 4. Here, we found the same trends as before: increases in diff-size and size of the CD led to a better running-times-ratio for the reduction-based approach.

Finally, we evaluated the performance of both approaches on CDs from the `CD4Analysis`-project. We compared two versions of each CD. The running-time-ratios are displayed in Table 5 and confirm the trend observed before.

| CD size | ds=3 | ds=5 | ds=10 | ds=15 |
|---|---|---|---|---|
| 5 | 1.203 | 1.231 | 1.959 | 1.514 |
| 10 | 0.596 | 0.846 | 1.403 | 2.666 |
| 15 | 0.675 | 0.907 | 2.565 | 2.704 |
| 20 | 0.842 | 1.760 | 2.960 | 4.201 |
| 25 | 0.899 | 1.723 | 2.761 | 5.239 |
| mean | 0.843 | 1.293 | 2.330 | 3.265 |

**Table 3** Ratio of running times between the Alloy-based and the reduction-based analyses reported in Table 1 and Table 2.

| CD pair | ds=3 | ds=5 | ds=10 | ds=15 |
|---|---|---|---|---|
| DE | 3.644 | 4.327 | 3.398 | 2.865 |
| EA | 0.636 | 0.821 | 1.149 | 1.633 |
| EMT | 0.524 | 0.457 | 1.132 | 0.981 |
| Library v2/v1 | 1.428 | 2.477 | 3.383 | 3.123 |
| Library v3/v2 | 1.301 | 1.764 | 1.440 | 1.311 |
| Library v4/v3 | 1.148 | 2.539 | 3.736 | 4.363 |
| Library v5/v4 | 2.047 | 1.78 | 1.475 | 1.676 |
| mean | 1.290 | 1.804 | 2.234 | 2.291 |

**Table 4** Ratios of running times for the alloy-based and the reduction-based CDDiff analysis on CDs from (Maoz et al. 2011b) and different diff-sizes ds ranging from 3 to 15.

| CD pair | ds=3 | ds=5 | ds=10 | ds=15 |
|---|---|---|---|---|
| Management | 1.708 | 1.627 | 2.114 | 2.402 |
| MyCompany | 0.972 | 1.281 | 1.565 | 1.425 |
| MyExample | 0.695 | 0.554 | 1.461 | 2.094 |
| MyLife | 1.611 | 1.621 | 1.824 | 2.411 |
| Teaching | 0.623 | 0.870 | 1.608 | 1.923 |
| mean | 1.121 | 1.191 | 1.714 | 2.051 |

**Table 5** Ratios of running times for the alloy-based and the reduction-based CDDiff analysis on the `CD4Analysis` CDs and different diff-sizes ds ranging from 3 to 15.

### 8.3. Validation

To ensure correct implementation of the extensions presented in Sect. 6 and Sect. 7, we have validated our work on many CD pairs, including the ones listed as the corpus of our evaluation. In particular, we have performed the following automated checks for diff-sizes 3, 5 and 10:

- Each CD is semantically equivalent to itself, *i.e.,* no diff-witnesses are produced (automatically determined).
- Each CD is a refinement of the empty CD, *i.e.,* no diff-witnesses are produced (automatically determined).
- Both implementations of open-world semantic differencing always agree on the existence and absence of diff-witnesses (automatically determined).
- No diff-witnesses are produced when comparing a refining successor version of a CD to its predecessor (determined by manual inspection).
- Diff-witnesses are produced when comparing a non-refining successor version of a CD to its predecessor (determined by manual inspection).
- All diff-witnesses produced are sound (automatically determined, see below).

To verify the soundness of the produced witnesses, we devel-

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

oped the `OD2CDMatcher`[4], a tool which can determine whether an object-structure in the form of an OD is in the semantics of a CD. Note that this tool relies on an alternative implementation and is independent of the encoding of CD semantics in Alloy used by the CDDiff operators. `OD2CDMatcher` can operate under both the closed-world and open-world assumption. As such, we not only verified the soundness of the witnesses with respect to the open-world semantics of the original input-CDs for both approaches, but for the reduction-based approach, we also checked the closed-world semantics of the transformed input-CDs. In this manner, we were able to identify and correct several issues in our implementation.

### 8.4. Threats To Validity

First, the UML CD standardization (Object Management Group 2017) only describes the semantics of CDs using natural language and gives no formal definition of semantics. Our approach is built upon previous definitions of CD semantics, but other modelers might have a different interpretation. Similarly, other modelers may disagree with us permitting overlapping associations in CDs or our handling of underspecified associations. Overlapping associations are easily handled by a model analyzer such as Alloy, as they simply constitute a conjunction of logical constraints. They allow refinement of associations for sub-classes consistent with set logic and permitting them simplifies our implementation. Another assumption is the implicit existence of the super-type *Object*, on which the validity of our reduction-based approach depends. To mitigate the threat of ambiguity in semantics of CDs, we base our work on previously published implementations (Maoz et al. 2011b; Kautz et al. 2017) and make all new implementations publicly available.

Second, regarding our evaluation, we cannot guarantee that the legibility of diff-witnesses and running time of operations for larger CDs and greater diff-size will always be improved for the reduction-based approach compared to the Alloy-based approach. This was simply the case for the examples in this paper and further evaluation might be necessary to make generalized claims. To mitigate the threat of a biased selection we have included previously published CDs in our evaluation that were not created specifically for open-world semantic differencing.

Third, our experiments use a single SAT solver in the default configuration as provided by Alloy. It is well-known that different SAT solver may lead to different performance results (Wang et al. 2019). To mitigate this threat of varying performance results, we have selected SAT4J for all experiments as the default and reportedly most stable SAT solver bundled with Alloy.

Finally, the answer to RQ1 relies on the diff-witnesses produced by Alloy. These diff-witnesses are not unique and may be different on different machines and software configurations. To mitigate the threat of different possible solutions, we have calculated averages over multiple pairs of CDs, we have captured the diff-witnesses produced on our machine, and we make all data available for inspection and comparison.

## 9. Conclusion and Outlook

In this paper, we outlined two approaches for open-world semantic differencing for UML/P CDs to assist change management in the early design phases of MDD. Both approaches extend the operator `CDDiff` introduced in (Maoz et al. 2011b). Furthermore, we demonstrated that the problem of open-world semantic differencing can be reduced to closed-world semantic differencing by comparing appropriate CD-expansions of the input-CDs, a fact that both approaches exploit: The first approach uses the Alloy Analyzer to implicitly construct and consider these expansions, while the second approach directly transforms the input-CDs based on a deterministic transformation algorithm. Our evaluation found that the transformation-based approach yields more readable diff-witnesses in shorter time.

We extended our approach to enable modelers to specify individual input-CDs or elements of a CD that should be considered under the closed-world assumption when computing an open-world difference: Marking a class with the stereotype «*complete*» forbids additional super-classes and additional outgoing associations. Marking the whole CD with the stereotype «*complete*» forbids additions to the set of classes, set of associations, and the inheritance relations of classes.

As future work, additional stereotypes could be introduced to allow for more semantic variability, *e.g.,* a class might be marked as «*sealed*» to forbid additional sub-classes. In the future, we also intend to conduct a case-study on the usefulness of automatic refinement checking and semantic differencing with `CDDiff` when designing object-oriented software.

### Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 250902306

## References

Alanen, M., & Porres, I. (2003). Difference and Union of Models. In *«uml» 2003 - the unified modeling language. modeling languages and applications.*

Balaban, M., & Maraee, A. (2013). Finite Satisfiability of UML Class Diagrams with Constrained Class Hierarchy. *ACM Trans. Softw. Eng. Methodol.*, *22*(3).

Bruneliere, H., García, J., Desfray, P., Khelladi, D. E., Hebig, R., Bendraou, R., & Cabot, J. (2015). On lightweight metamodel extension to support modeling tools agility. In *Ecmfa@staf 2015* (Vol. 9153, pp. 62–74). Springer.

Butting, A., Kautz, O., Rumpe, B., & Wortmann, A. (2017, April). Semantic Differencing for Message-Driven Component & Connector Architectures. In *International conference on software architecture (icsa'17)* (p. 145-154). IEEE.

Drave, I., Eikermann, R., Kautz, O., & Rumpe, B. (2019, February). Semantic Differencing of Statecharts for Object-oriented Systems. In S. Hammoudi, L. F. Pires, & B. Selić (Eds.), *Proceedings of the 7th international conference on model-driven engineering and software development (modelsward'19)* (p. 274-282). SciTePress.

Drave, I., Kautz, O., Michael, J., & Rumpe, B. (2019, September). Semantic Evolution Analysis of Feature Models. In

---

[4] The tool is included in the `CD4Analysis`-repository on GitHub at: https://github.com/MontiCore/cd4analysis

[RRS23] J. O. Ringert, B. Rumpe, M. Stachon:
On Implementing Open World Semantic Differencing for Class Diagrams.
In: Journal of Object Technology,
Volume 22(2), pp. 2:1-14, Jul. 2023.

T. Berger et al. (Eds.), *International systems and software product line conference (splc'19)* (p. 245-255). ACM.

Fahrenberg, U., Acher, M., Legay, A., & Wasowski, A. (2014). Sound merging and differencing for class diagrams. In S. Gnesi & A. Rensink (Eds.), *Fundamental approaches to software engineering* (pp. 63–78). Berlin, Heidelberg: Springer Berlin Heidelberg.

Guy, C., Combemale, B., Derrien, S., Steel, J., & Jézéquel, J. (2012). On model subtyping. In *ECMFA 2012* (Vol. 7349, pp. 400–415). Springer.

Harel, D., & Rumpe, B. (2004, October). Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer*, *37*(10), 64-72.

Herrmann, C., Krahn, H., Rumpe, B., Schindler, M., & Völkel, S. (2007). An Algebraic View on the Semantics of Model Composition. In *Conference on model driven architecture - foundations and applications (ecmda-fa'07)* (p. 99-113). Springer, Germany.

Jackson, Daniel. (2006). *Software abstractions: Logic, language, and analysis*. The MIT Press.

Jácome-Guerrero, S. P., & de Lara, J. (2018). Controlling meta-model extensibility in model-driven engineering. *IEEE Access*, *6*, 19923–19939.

Jiang, Y., Shao, W., Zhang, L., Ma, Z., Meng, X., & Ma, H. (2004). On the classification of uml's meta model extension mechanism. In *UML 2004* (Vol. 3273, pp. 54–68). Springer.

Kautz, O. (2021). *Model Analyses Based on Semantic Differencing and Automatic Model Repair*. Shaker Verlag.

Kautz, O., Maoz, S., Ringert, J. O., & Rumpe, B. (2017, July). *CD2Alloy: A Translation of Class Diagrams to Alloy* (Technical Report No. AIB-2017-06). RWTH Aachen University.

Kautz, O., & Rumpe, B. (2018a, October). On Computing Instructions to Repair Failed Model Refinements. In *Conference on model driven engineering languages and systems (models'18)* (p. 289-299). ACM.

Kautz, O., & Rumpe, B. (2018b, October). Semantic Differencing of Activity Diagrams by a Translation into Finite Automata. In *Proceedings of models 2018. workshop me.*

Kehrer, T., Kelter, U., & Taentzer, G. (2011). A Rule-Based Approach to the Semantic Lifting of Model Differences in the Context of Model Versioning. In *International conference on automated software engineering (ase'11).*

Küster, J. M., Gerth, C., & Engels, G. (2009). Dependent and Conflicting Change Operations of Process Models. In *Model driven architecture - foundations and applications.*

Küster, J. M., Gerth, C., Förster, A., & Engels, G. (2008). Detecting and Resolving Process Model Differences in the Absence of a Change Log. In *Business process management.*

Langer, P., Mayerhofer, T., & Kappel, G. (2014). Semantic model differencing utilizing behavioral semantics specifications. In J. Dingel, W. Schulte, I. Ramos, S. Abrahão, & E. Insfran (Eds.), *Model-driven engineering languages and systems* (pp. 116–132). Cham: Springer International Publishing.

Maoz, S., Ringert, J. O., & Rumpe, B. (2011a). ADDiff: Semantic Differencing for Activity Diagrams. In *Conference on foundations of software engineering (esec/fse '11)* (p. 179-189). ACM.

Maoz, S., Ringert, J. O., & Rumpe, B. (2011b). CDDiff: Semantic Differencing for Class Diagrams. In M. Mezini (Ed.), *Ecoop 2011 - object-oriented programming* (p. 230-254). Springer Berlin Heidelberg.

Maoz, S., Ringert, J. O., & Rumpe, B. (2011c, July). *An Operational Semantics for Activity Diagrams using SMV* (Technical Report No. AIB-2011-07). Aachen, Germany: RWTH Aachen University.

Nachmann, I., Rumpe, B., Stachon, M., & Sebastian, S. (2022, June). Open-World Loose Semantics of Class Diagrams as Basis for Semantic Differences. In *Modellierung 2022* (p. 111-127). Gesellschaft für Informatik.

Object Management Group. (2017). *OMG Unified Modeling Language (OMG UML), v2.5.1.*

Reiter, R. (1978). On Closed World Data Bases. In *Logic and data bases.*

Rumpe, B. (2011). *Modellierung mit UML, 2te Auflage*. Springer Berlin.

Schindler, M. (2012). *Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P*. Shaker Verlag.

Steel, J., & Jézéquel, J. (2007). On model typing. *Softw. Syst. Model.*, *6*(4), 401–413.

T. Kehrer, U. Kelter, and G. Taentzer. (2013). Consistency-Preserving Edit Scripts in Model Versioning. In *International conference on automated software engineering (ase).*

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

Thüm, T., D., & Kästner, C. (2009). Reasoning about Edits to Feature Models. In *Proceedings of the 31st international conference on software engineering.*

Wang, W., Wang, K., Zhang, M., & Khurshid, S. (2019). Learning to optimize the alloy analyzer. In *ICST 2019* (pp. 228–239). IEEE.

## About the authors

**Jan Oliver Ringert** is Professor of Software Enginering at Bauhaus-University Weimar, Germany. His research focuses on the analysis of software models and their evolution. You can contact the author at jan.ringert@uni-weimar.de.

**Bernhard Rumpe** is Professor at RWTH Aachen University, Germany and head of the Chair of Software Engineering. His main interests are rigorous and practical software and system development methods based on adequate modeling techniques. This includes agile development methods as well as modelengineering based on UML/SysML-like notations and domain specific languages. You can contact the author at rumpe@se-rwth.de or visit www.se-rwth.de.

**Max Stachon** is a research assistant and Ph.D. candidate at the Chair of Software Engineering at RWTH Aachen University. His research focuses on model semantics, refinement and difference analysis. You can contact the author at stachon@se-rwth.de or visit www.se-rwth.de.