Consistency Rules (CR)

CR1A: All classes in a UML model need traces to the corresponding JAVA program.

```
context UML Class inv:
self.javaRef.isDefined() and self.javaRef.traces.size() > 0
```

CR1B: All classes in a JAVA program need traces to the corresponding UML model.

```
context Java Class inv:
self.umlRef.isDefined() and self.umlRef.traces.size() > 0
```

CR2A: All properties in a UML model need traces to the corresponding JAVA fields.

```
context UML Property inv:
self.javaRef.isDefined() and self.javaRef.traces.size() > 0
```

CR2B: All fields in a JAVA program need traces to the corresponding UML property.

```
context Java Field inv:
self.umlRef.isDefined() and self.umlRef.traces.size() > 0
```

CR3A: All operations in a UML model need traces to the corresponding JAVA methods.

```
context UML Operation inv:
self.javaRef.isDefined() and self.javaRef.traces.size() > 0
```

CR3B: All methods in a JAVA program need traces to the corresponding UML operations.

```
context Java Method inv:
self.umlRef.isDefined() and self.umlRef.traces.size() > 0
```

CR4A: All classes in a UML model need to be in the corresponding JAVA program.

```
context UML Class inv:
self.javaRef.traces-> exists(javaClass : <JavaClass>| javaClass.name = self.name)
```

CR4B: All classes in a JAVA program need to be in the corresponding UML model.

```
context Java Class inv:
self.umlRef.traces-> exists(umlClass : <UMLClass> | umlClass.name = self.name)
```

CR5A: All fields in a UML class need to be in the corresponding JAVA class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass>| javaClass.fields ->
     forAll(f | self.attribute -> exists(a | f.name = a.name)))
```

CR5B: All fields in a JAVA class need to be in the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass> | umlClass.attribute ->
     forAll(a | self.fields -> exists(f | f.name = a.name)))
```

CR6A: All fields in a UML class need to be in the corresponding JAVA class or super class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass>| javaClass.allFields ->
    forAll(a | self.allAttributes -> exists(f | f.name = a.name)))
```

CR6B: All fields in a JAVA class or superclass need to be in the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass> | umlClass.attribute ->    forAll
    (a | self.allFields -> exists(f | f.name = a.name)))
```

CR7A: The type of a UML attribute needs to match the type of the corresponding JAVA field.

```
context UML Property inv:
self.javaRef.traces -> exists(javaField :<JavaField> | javaField.type.type.name =
    self.type.name)
```

CR7B: The type of a JAVA field needs to match the primitive type of the corresponding UML attribute.

```
context Java Field inv:
self.umlRef.traces -> exists(umlAttribute : <UMLProperty> | umlAttribute.type.name =
     self.type.type.name)
```

CR8A: All operations in a UML class need to be in the corresponding JAVA class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass> | self.ownedOperation ->
    forAll(umlOperation : <UMLOperation>|  javaClass.methods-> exists(javaMethod :
    <UMLMethod>   | javaMethod.name = umlOperation.name)))
```

CR8B: All methods in a JAVA class need to be in the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass>   | self.methods -> forAll(
    javaMethod : <UMLMethod> |  umlClass.ownedOperation-> exists(umlOperation : <
    UMLOperation> | javaMethod.name = umlOperation.name)))
```

CR9A: The return type of a UML operation needs to match the return type of the corresponding Java Method.

```
context UML Operation inv:
self.javaRef.traces -> exists(javaMethod : <UMLMethod> | self.type.name.toLower() =
    javaMethod.returnType.type.qualifiedName)
```

CR9B: The return type of a JAVA method needs to match the return type of the corresponding UML operation.

```
context Java Method inv:
self.umlRef.traces ->  exists(umlOperation : <UMLOperation>| not umlOperation.type.
    isDefined() or umlOperation.type.name.toLower() = self.returnType.type.
    qualifiedName)
```

CR10A: All parameters in a UML operation need to be in the corresponding JAVA method.

```
context UML Operation inv:
self.javaRef.traces ->  exists(javaMethod : <UMLMethod> | (not javaMethod.returnType
    .isDefined()) or javaMethod.returnType.type.qualifiedName = self.type.name.
    toLower())
```

CR10B: All parameters in a Java method need to be in the corresponding UML operation.

```
context Java Method inv:
self.umlRef.traces -> exists(umlOperation : <UMLOperation> | not umlOperation.type.
    isDefined() or umlOperation.type.name.toLower() = self.returnType.type.
    qualifiedName)
```

CR11A: A generalization of a UML class need to be represented by the direct super class in the corresponding JAVA class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass>| self.generalization -> forAll
    (umlGeneralization : <UMLGeneralization> | umlGeneralization.target ->  exists
    (umlGeneralizationTarget : <UMLClass> | javaClass.superClass.isDefined() and
    umlGeneralizationTarget.name = javaClass.superClass.type.name)))
```

CR11B: A super class of a Java class need to be represented by the direct generalization in the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass> | umlClass.generalization ->
    forAll(umlGeneralization : <UMLGeneralization> | umlGeneralization.target ->
    exists(umlGeneralizationTarget : <UMLClass> | self.superClass.isDefined() and
    umlGeneralizationTarget.name = self.superClass.type.name)))
```

CR12A: All interfaces defined by a UML class need to be implemented by the corresponding JAVA class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass>| self.interfaceRealization ->
    forAll(umlInterface : <UMLInterfaceRealization> | javaClass.interfaces ->
    exists(javaInterface : <JavaClassReference> | umlInterface.contract.name =
    javaInterface.name)))
```

CR12B: All interfaces implemented by a JAVA class need to be defined by the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass> | umlClass.interfaceRealization
    -> forAll(umlInterface : <UMLInterfaceRealization> | self.interfaces ->  exists
    (javaInterface : <JavaClassReference> | umlInterface.contract.name =
    javaInterface.name)))
```

CR13A: All compositions of a UML class need to be represented by the fields in the corresponding JAVA class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass>| self.attribute -> select(
    umlRelationShip1 : <UMLProperty> | umlRelationShip1.aggregation.name = '
    composite') ->  forAll(umlRelationShip : <UMLProperty> | javaClass.fields ->
    exists(javaField : <JavaField>  | (javaField.type.type.name = umlRelationShip.
    type.name))))
```

CR13B: All compositions of a Java class need to be represented by the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass> | umlClass.attribute -> select(
    umlRelationShip1 : <UMLProperty> | umlRelationShip1.aggregation.name = '
    composite') ->  forAll(umlRelationShip : <UMLProperty> | self.fields ->  exists
    (javaField : <JavaField>  | (javaField.type.type.name = umlRelationShip.type.
    name))))
```

CR14A: All aggregations of a UML class need to be represented by the fields in the corresponding JAVA class.

```
context UML Class inv:
self.javaRef.traces -> forAll(javaClass : <JavaClass>| self.attribute -> select(
    umlRelationShip1 : <UMLProperty> | umlRelationShip1.aggregation.name = 'shared
    ') ->  forAll(umlRelationShip : <UMLProperty> | javaClass.fields ->  exists(
    javaField : <JavaField>  | (javaField.type.type.name = umlRelationShip.type.
    name))))
```

CR14B: All aggregations of a Java class need to be represented by the corresponding UML class.

```
context Java Class inv:
self.umlRef.traces -> forAll(umlClass : <UMLClass> | umlClass.attribute -> select(
    umlRelationShip1 : <UMLProperty> | umlRelationShip1.aggregation.name = 'shared
    ') -> forAll(umlRelationShip : <UMLProperty> | self.fields -> exists(
    javaField : <JavaField> | (javaField.type.type.name = umlRelationShip.type.
    name))))
```