Tutorial of the ATL transformation language
http://github.com/jesusc/atl-tutorial

Part II

ANATLYZER

jesus.sanchez.cuadrado@gmail.com
@sanchezcuadrado
http://sanchezcuadrado.es

Motivation

- Writing a model(-to-model) transformation is a complex task
  1. You must handle every possible input configuration
  2. You must ensure the target model is syntactically correct
  3. The mapping itself must be semantically correct

Motivation

- Moreover:
  - The reliability of any MDE process depends on the correctness of its transformations
  - The same transformation will be used many times to generate many models, even in different projects (errors percolate every project!)

Motivation

- There are also accidental details due to the transformation language.
- In ATL:
  - It is dynamically typed
  - There is no formal semantics
  - Design decisions may not be optimal

Motivation

- Consider this copy rule
  - Is it right?

```
rule class2class {
  from s : UML!Class
  to   t : UML!Class (
     -- Is there anything missing here?
  )
}
```

AnATLyzer

- A static analyser for ATL model transformations
- Static analysis
  - Detect problems before executing the transformation
  - Goal:
    - Be precise: few false positives
    - Be complete: few false negatives

What can anATLyzer do for you?

- AnATLyzer detects more that 50 types of problems
  - Navigation & typing problems
  - Rule problems
  - Transformation integrity problems
  - Style problems

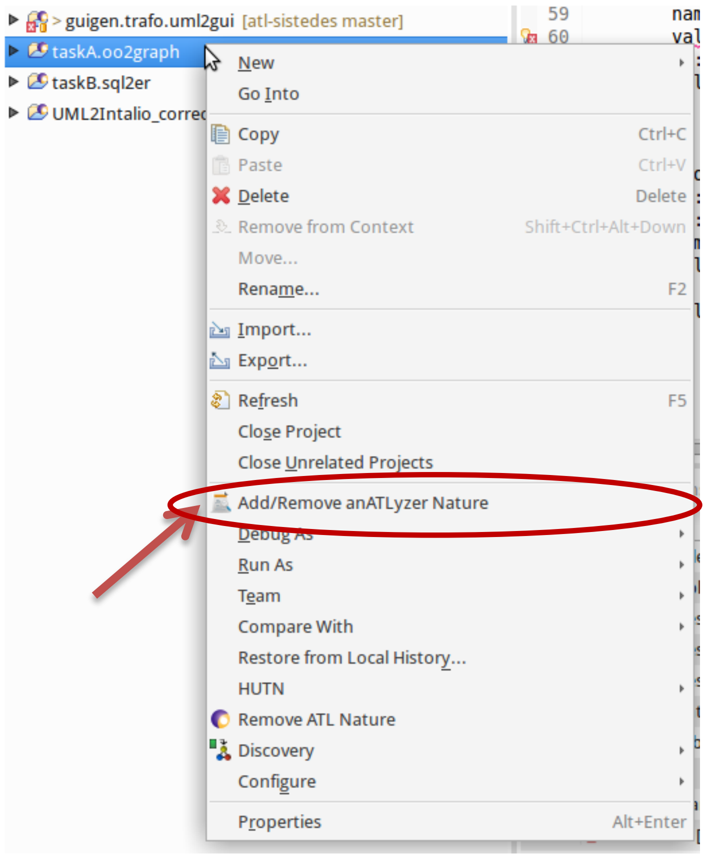
AnATLyzer

Using AnATLyzer

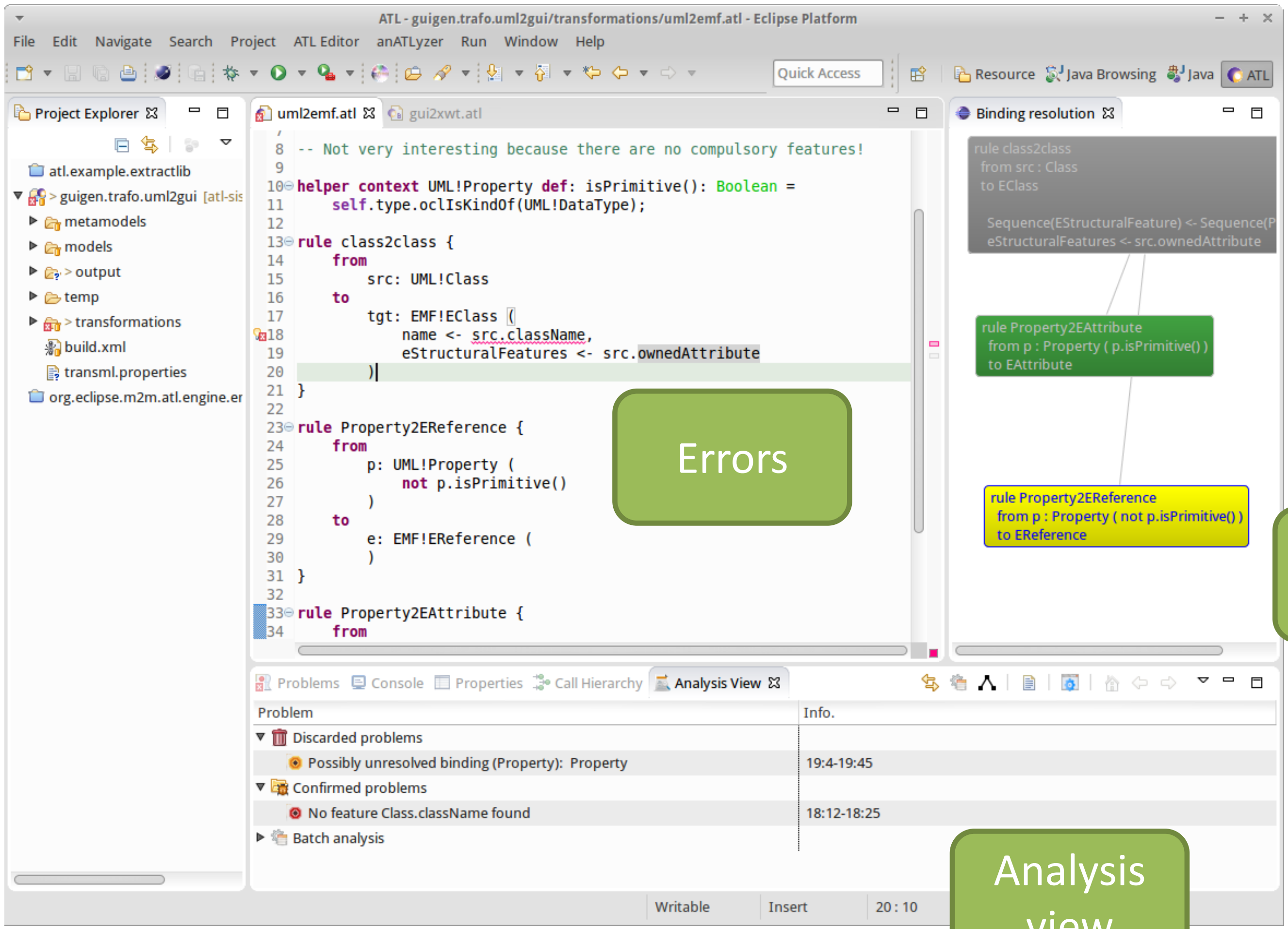
cd2gui_errors.atl

Setting up the project

- Right-click on an ATL project
- Select “Add/Remove” anATLyzer feature
- All transformations in the project will automatically be analysed



User interface


ATL - guigen.trafo.uml2gui/transformations/uml2emf.atl - Eclipse Platform
File Edit Navigate Search Project ATL Editor anATLyzer Run Window Help
Quick Access
Resource Java Browsing Java ATL
Project Explorer
atl.example.extractlib
> guigen.trafo.uml2gui [atl-sis
metamodels
models
> output
temp
> transformations
build.xml
transml.properties
org.eclipse.m2m.atl.engine.er
uml2emf.atl
gui2xwt.atl
Binding resolution
Problems Console Properties Call Hierarchy Analysis View
Writable Insert 20 : 10

```
-- Not very interesting because there are no compulsory features!

helper context UML!Property def: isPrimitive(): Boolean =
    self.type.oclIsKindOf(UML!DataType);

rule class2class {
    from
        src: UML!Class
    to
        tgt: EMF!EClass (
            name <- src.className,
            eStructuralFeatures <- src.ownedAttribute
        )
}

rule Property2EReference {
    from
        p: UML!Property (
            not p.isPrimitive()
        )
    to
        e: EMF!EReference (
        )
}

rule Property2EAttribute {
    from
```

rule class2class
from src : Class
to EClass

Sequence(EStructuralFeature) <- Sequence(P
eStructuralFeatures <- src.ownedAttribute

rule Property2EAttribute
from p : Property (p.isPrimitive())
to EAttribute

rule Property2EReference
from p : Property (not p.isPrimitive())
to EReference

Problem	Info.
Discarded problems	
Possibly unresolved binding (Property): Property	19:4-19:45
Confirmed problems	
No feature Class.className found	18:12-18:25
Batch analysis	

Errors

Visualization

Analysis view

The analysis view

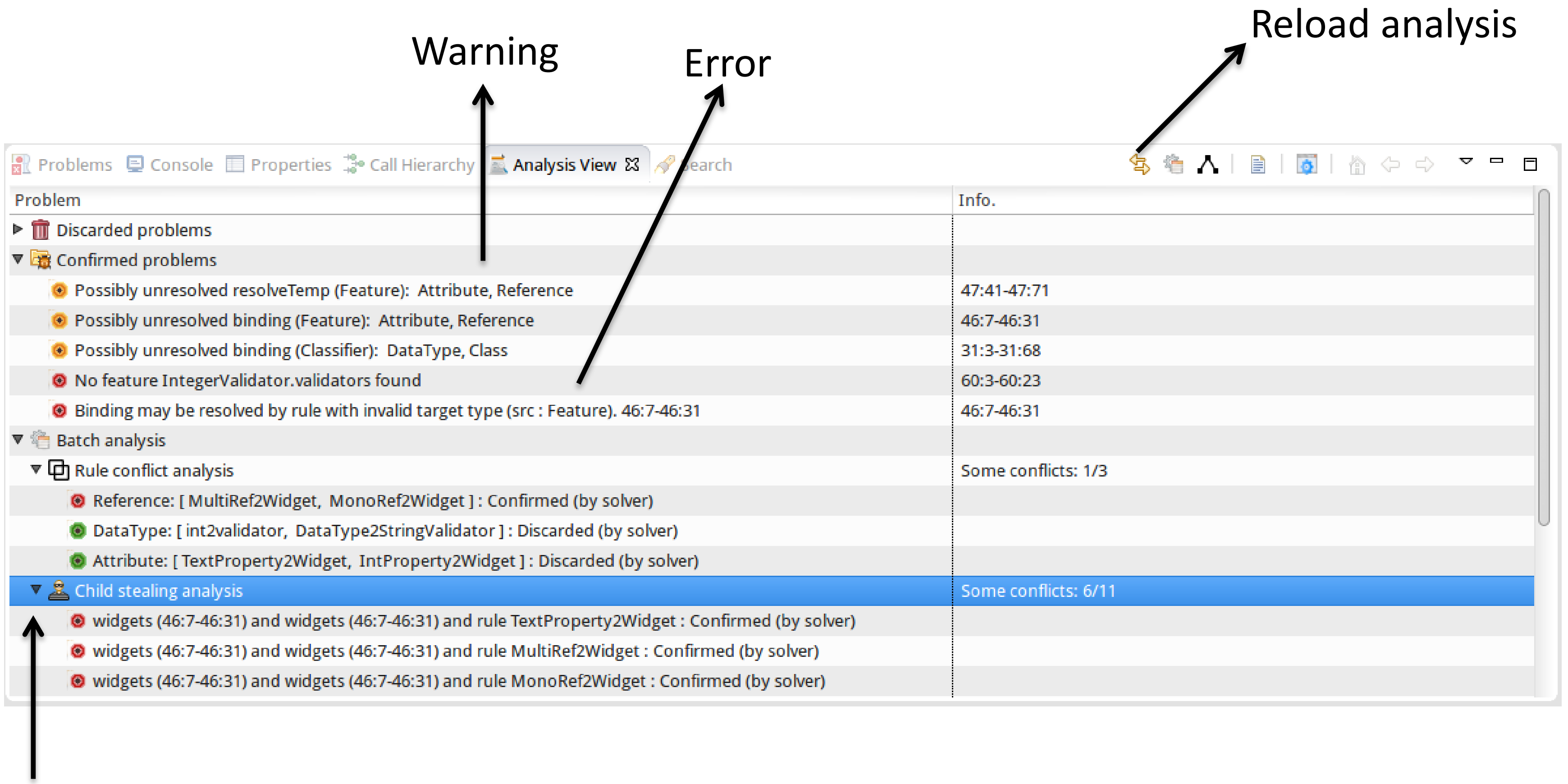
- Show list of detected problems
- Provide access to batch analysis
  - Rule conflict
  - Child stealing (experimental!)

- Show the view
  - Window -> Show view -> Other ...
  - AnATLyzer -> Analysis View

The analysis View

- Confirmed
  - It is a true error. Should be fixed somehow.
  - Try some quick fix! CTRL + 1
- Discarded
  - We used model finding to ensure it is not an error
  - Can be ignored
- Unknown
  - It is a smell but we cannot check if it is an error.
- Running
  - Errors which are currently being processed
  - Most of the time you could not see this.
- Time out
  - If it takes to long to confirm the problem

The analysis view



Problem	Info.
Discarded problems	
Confirmed problems	
Possibly unresolved resolveTemp (Feature): Attribute, Reference	47:41-47:71
Possibly unresolved binding (Feature): Attribute, Reference	46:7-46:31
Possibly unresolved binding (Classifier): DataType, Class	31:3-31:68
No feature IntegerValidator.validators found	60:3-60:23
Binding may be resolved by rule with invalid target type (src : Feature). 46:7-46:31	46:7-46:31
Batch analysis	
Rule conflict analysis	Some conflicts: 1/3
Reference: [MultiRef2Widget, MonoRef2Widget] : Confirmed (by solver)	
DataType: [int2validator, DataType2StringValidator] : Discarded (by solver)	
Attribute: [TextProperty2Widget, IntProperty2Widget] : Discarded (by solver)	
Child stealing analysis	Some conflicts: 6/11
widgets (46:7-46:31) and widgets (46:7-46:31) and rule TextProperty2Widget : Confirmed (by solver)	
widgets (46:7-46:31) and widgets (46:7-46:31) and rule MultiRef2Widget : Confirmed (by solver)	
widgets (46:7-46:31) and widgets (46:7-46:31) and rule MonoRef2Widget : Confirmed (by solver)	

Double-click on “Rule conflict analysis” or “Child stealing” to execute

Keyboard shortcuts

- CTRL + 1
  - Over an error, show quick fix
  - Over a normal statement, show quick assist

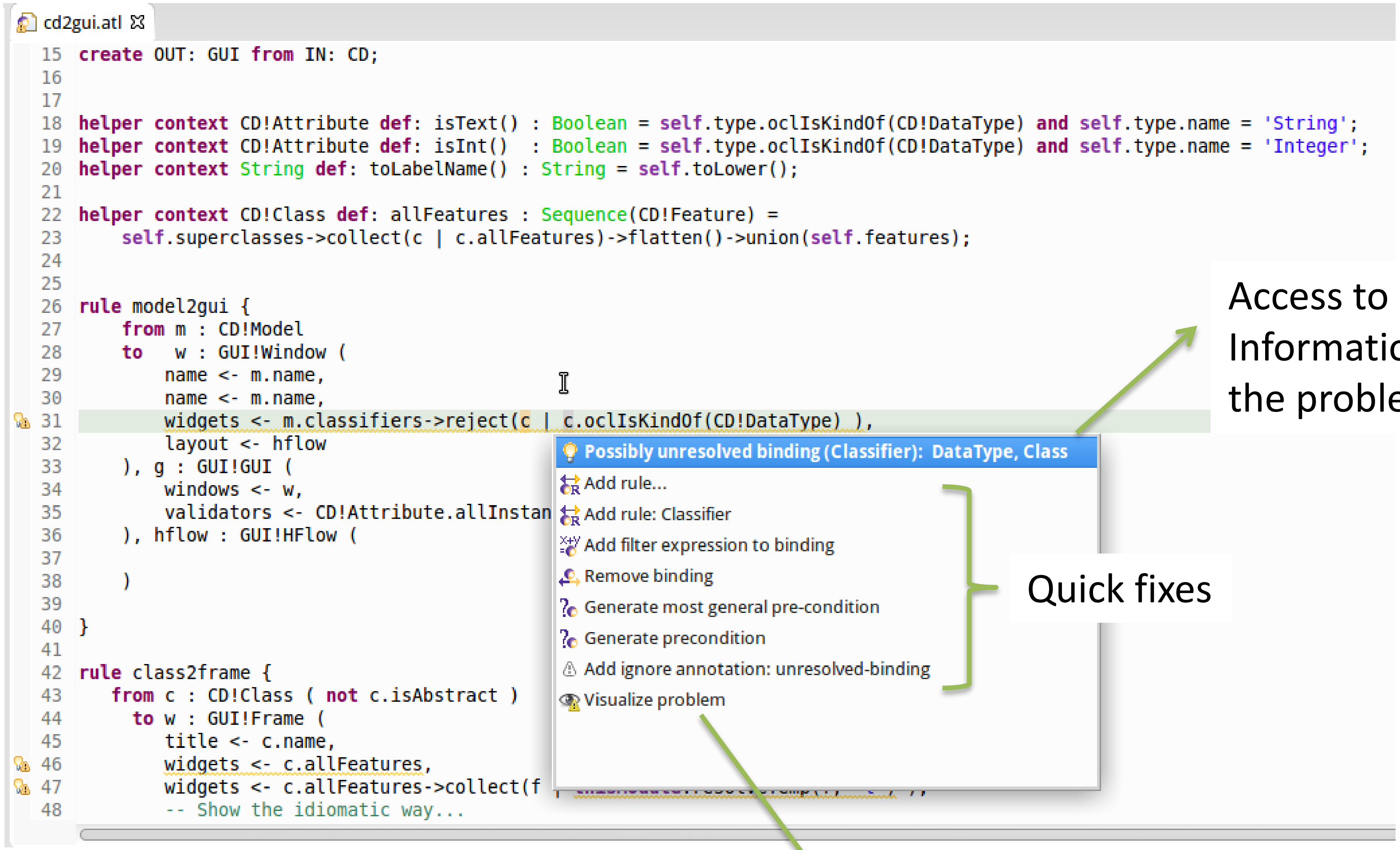
- Be ready to use CTRL-Z to undo...

- CTRL + S to save and re-analyse
  - The analysis is mostly incremental

Keyboard shortcuts (Inherited from ATL Editor)

- Auto-complete
  - CTRL+SPACE
  - Not completely precise
- Go to definition (e.g., helper, definition)
  - CTRL + Click
  - F3 with the keyboard
- Comment / Uncomment
  - CTRL+SHIFT+C

Quick fixes



cd2gui.atl

```
create OUT: GUI from IN: CD;


helper context CD!Attribute def: isText() : Boolean = self.type.oclIsKindOf(CD!DataType) and self.type.name = 'String';
helper context CD!Attribute def: isInt()  : Boolean = self.type.oclIsKindOf(CD!DataType) and self.type.name = 'Integer';
helper context String def: toLabelName() : String = self.toLower();

helper context CD!Class def: allFeatures : Sequence(CD!Feature) =
    self.superclasses->collect(c | c.allFeatures)->flatten()->union(self.features);


rule model2gui {
    from m : CD!Model
    to   w : GUI!Window (
        name <- m.name,
        name <- m.name,
        widgets <- m.classifiers->reject(c | c.oclIsKindOf(CD!DataType) ),
        layout <- hflow
    ), g : GUI!GUI (
        windows <- w,
        validators <- CD!Attribute.allInstan
    ), hflow : GUI!HFlow (

    )

}

rule class2frame {
   from c : CD!Class ( not c.isAbstract )
    to w : GUI!Frame (
        title <- c.name,
        widgets <- c.allFeatures,
        widgets <- c.allFeatures->collect(f | [illegible]
        -- Show the idiomatic way...
```

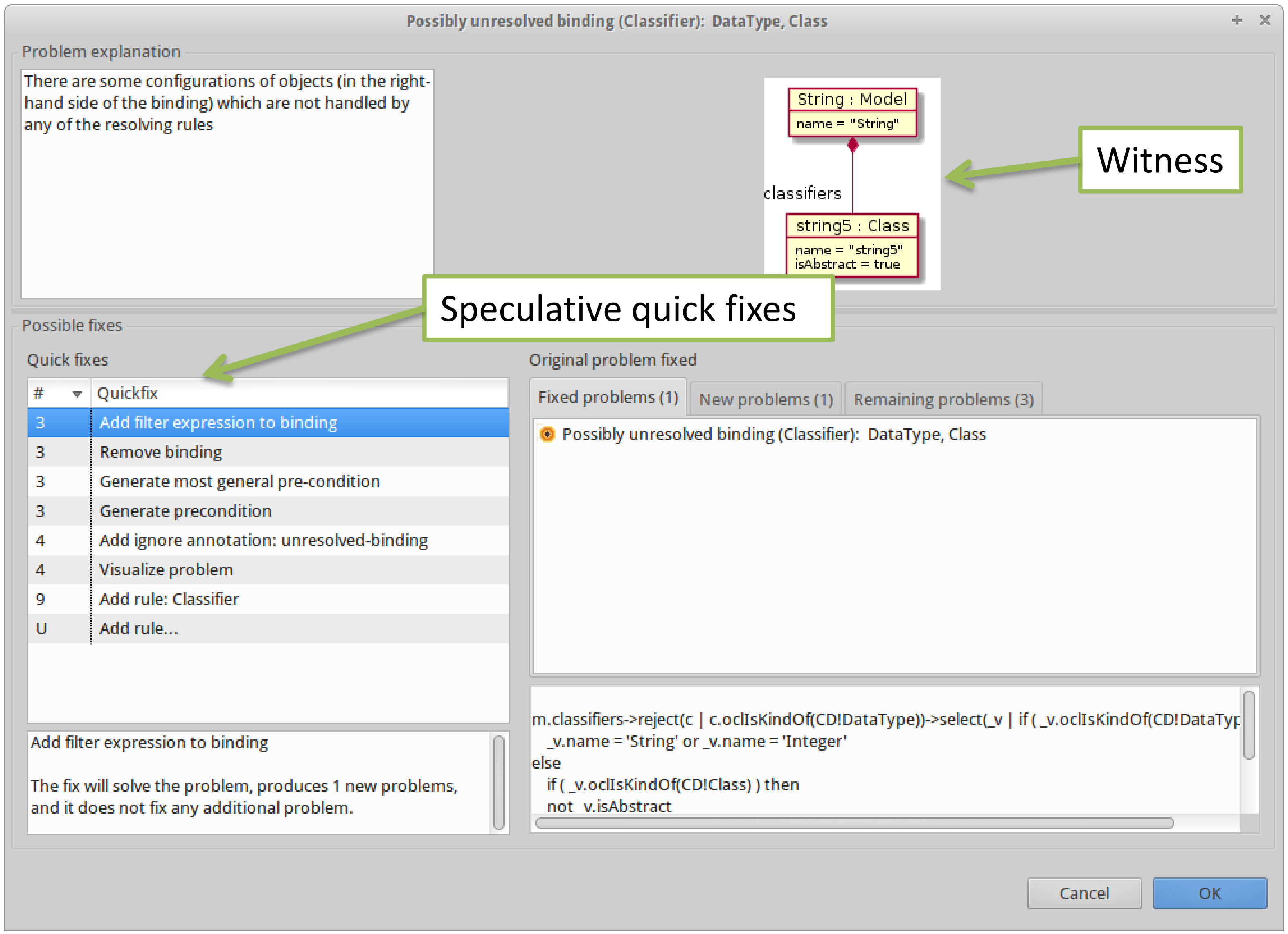
Possibly unresolved binding (Classifier): DataType, Class
- Add rule...
- Add rule: Classifier
- Add filter expression to binding
- Remove binding
- Generate most general pre-condition
- Generate precondition
- Add ignore annotation: unresolved-binding
- Visualize problem

Access to detailed Information about the problem

Quick fixes

Visualization quick access

Problem information



Problem explanation

There are some configurations of objects (in the right-hand side of the binding) which are not handled by any of the resolving rules

Possible fixes

Quick fixes

#	Quickfix
3	Add filter expression to binding
3	Remove binding
3	Generate most general pre-condition
3	Generate precondition
4	Add ignore annotation: unresolved-binding
4	Visualize problem
9	Add rule: Classifier
U	Add rule...

Add filter expression to binding

The fix will solve the problem, produces 1 new problems, and it does not fix any additional problem.

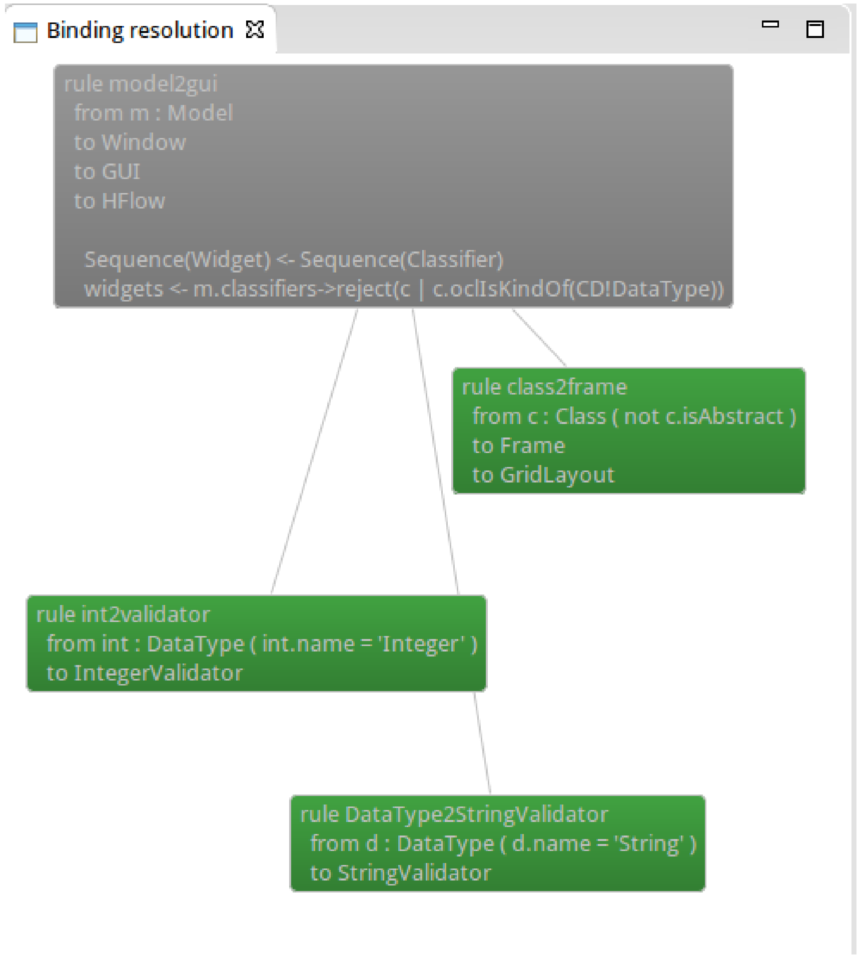
Original problem fixed

Fixed problems (1) | New problems (1) | Remaining problems (3)

Possibly unresolved binding (Classifier): DataType, Class

```
m.classifiers->reject(c | c.oclIsKindOf(CD!DataType))->select(_v | if ( _v.oclIsKindOf(CD!DataTyp
  _v.name = 'String' or _v.name = 'Integer'
else
  if ( _v.oclIsKindOf(CD!Class) ) then
  not  v.isAbstract
```

Visualization


Binding resolution

```
rule model2gui
 from m : Model
 to Window
 to GUI
 to HFlow

  Sequence(Widget) <- Sequence(Classifier)
  widgets <- m.classifiers->reject(c | c.oclIsKindOf(CD!DataType))
```

```
rule class2frame
 from c : Class ( not c.isAbstract )
 to Frame
 to GridLayout
```

```
rule int2validator
 from int : DataType ( int.name = 'Integer' )
 to IntegerValidator
```

```
rule DataType2StringValidator
 from d : DataType ( d.name = 'String' )
 to StringValidator
```

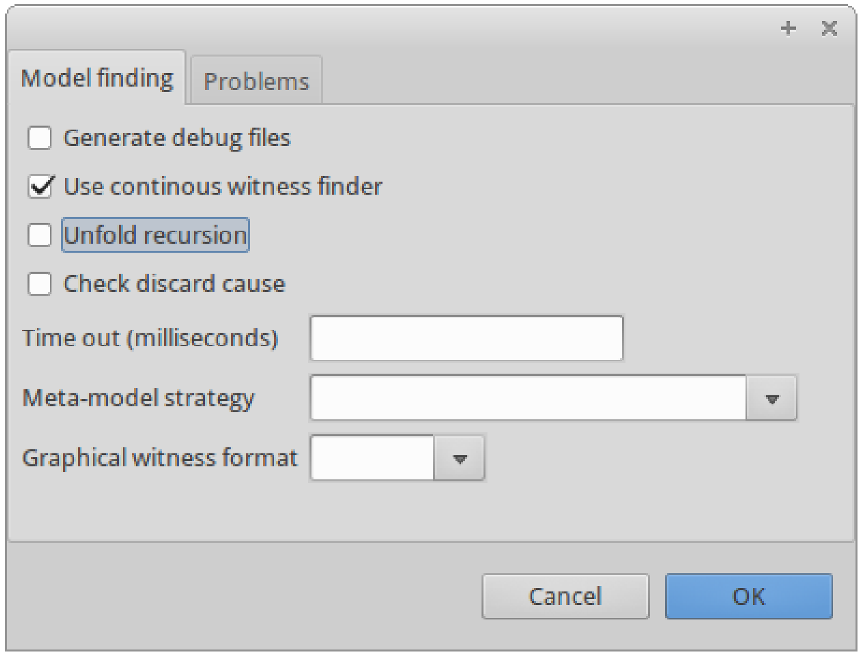
Visualization

- Available as quick assist for bindings and also as quick fix for binding errors
- Currently visualization does not use constraint solving to prune, you get all "possible" resolutions
  - In the previous example: `int2validator` and `DataType2StringValidator` could be pruned from the visualization

Configuration

- Right-click on the ATL file
  - anATLyzer -> Configure anATLyzer

Configuration

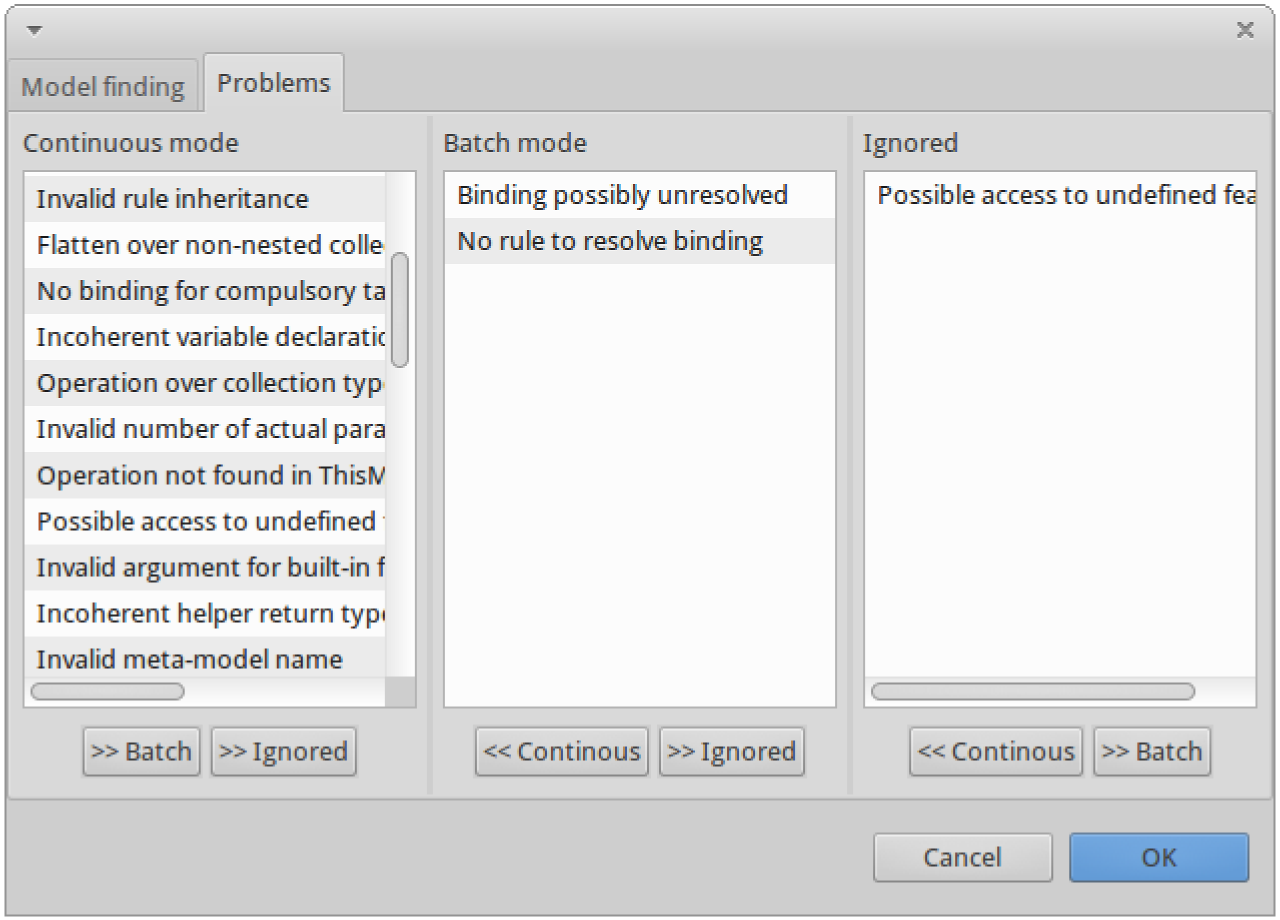


- [ ] Generate debug files
- [✓] Use continous witness finder
- [ ] Unfold recursion
- [ ] Check discard cause

Time out (milliseconds)
Meta-model strategy
Graphical witness format

- Continous mode
  - Recommended
  - Untick to execute model finder on demand
- Unfold recursion
  - Experimental support for recursive helpers
- Check discard cause
  - Errors can be discarded due to meta-model issues
- Time out

Configuration



Continuous mode

- Invalid rule inheritance
- Flatten over non-nested colle
- No binding for compulsory ta
- Incoherent variable declaratic
- Operation over collection typ
- Invalid number of actual para
- Operation not found in ThisM
- Possible access to undefined
- Invalid argument for built-in f
- Incoherent helper return typ
- Invalid meta-model name

Batch mode

- Binding possibly unresolved
- No rule to resolve binding

Ignored

- Possible access to undefined fea

Technical information

- Installation
  - Requirements:
    - Java 8
    - ATL 3.x
    - UML2 plug-in, for UML support (optional)
    - Zest 1.5 , for visualization support (optional)
    - Tested on Eclipse Luna and Mars
  - Web site:
    - http://miso.es/tools/anATLyzer.html
  - Update site:
    - http://sanchezcuadrado.es/projects/anatlyzer/sites/anatlyzer.updatesite/
  - Source code available at Github:
    - https://github.com/jesusc/anatlyzer

AnATLyzer

Types of problems

Types of problems

- AnATLyzer detects more than 50 types of problems
- Classification:
  - Typing and navigation
    - Typing w.r.t. meta-models and use of OCL
  - Transformation integrity
    - Checks related to the transformation structure
  - Target meta-model conformance
    - Does the output model conforms to the target meta-model?
  - Transformation rules
    - Issues related to (matched) rule usage

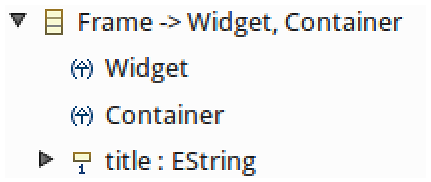
Description	Phase	Precision	Severity
Typing (with respect to source/target meta-model and helper definitions)			
Invalid meta-model name	typing	static	error-load
Invalid meta-class name	typing	static	error-load
Invalid enum literal	typing	static	error-load
Feature not found	typing	static	runtime-error
Feature not found in union type	typing	static	runtime-error
Feature found in subtype	typing	sometimes-solver	runtime-error
Operation not found	typing	static	runtime-error
Operation found in subtype	typing	sometimes-solver	runtime-error
Attribute not found in thisModule	typing	static	runtime-error
Operation not found in thisModule	typing	static	runtime-error
Object without container	typing	static	runtime-error
Incoherent variable declaration	typing	static	warning-style
Incoherent helper return type	typing	static	warning-style
Invalid number of actual parameters	typing	static	runtime-error
Invalid actual parameter type	typing	static	warning-behaviour
Navigation			
Collection operation not found	typing	static	runtime-error
Collection operation over no collection ("→" vs. ".")	typing	static	warning-style
Operation over collection type ("." vs. "→")	typing	static	warning-style
Feature access in collection	typing	static	runtime-error
Iterator over empty collection	typing	static	warning-behaviour
Feature access over possibly undefined receptor	typing	sometimes-solver	runtime-error
Feature access over possibly undefined receptor via empty collection	typing	always-solver	runtime-error
Flatten over non-nested collection	typing	static	warning-perf
Foreach statement expected collection	typing	static	runtime-error
Wrong iterator body type	typing	static	runtime-error
Change select-first for any	typing	static	warning-perf
Iterator over no collection type	typing	static	runtime-error
Invalid argument for built-in function	typing	static	runtime-error
Invalid operand	typing	static	runtime-error
Invalid operator	typing	static	runtime-error
Transformation integrity constraints			
Invalid rule inheritance	typing	static	runtime-error
Matched rule without output pattern	typing	static	runtime-error
Matched rule with non-boolean filter	typing	static	runtime-error
Abstract class instantiation	typing	static	runtime-error
Read access to target model	typing	static	warning-behaviour
Lazy rule with filter	typing	static	warning-behaviour
Target meta-model conformance			
No binding for compulsory target feature	analysis	static	error-target
Binding resolved by rule with invalid target	analysis	sometimes-solver	error-target
Collection assigned to mono-valued binding	analysis	static	error-target
Incompatible primitive value for primitive binding	analysis	static	error-target
Model element assigned to primitive binding	analysis	static	error-target
Primitive value assigned to object binding	analysis	static	error-target
Invalid assignment in imperative binding	typing	static	runtime-error
Transformation rules			
No rule to resolve binding	analysis	static	warning-behaviour
Binding possibly unresolved	analysis	always-solver	warning-behaviour
No rule to resolve a resolveTemp operation	typing	static	warning-behaviour
ResolveTemp possibly unresolved	analysis	always-solver	warning-behaviour
Undefined output pattern in resolveTemp operation	typing	static	runtime-error
Rule conflict	analysis (separate)	sometimes-solver	runtime-error

Typing and navigation

- OCL expressions should be typed against the source meta-model
- AnATLyzer detects problems like:
  - Invalid references to classes and features
  - Invalid iteration expressions
  - Invalid variable declarations
  - “Null pointer exceptions”
  - “Feature found in subtype”

Target conformance problems

- No binding for compulsory feature



Frame -> Widget, Container
Widget
Container
title : EString

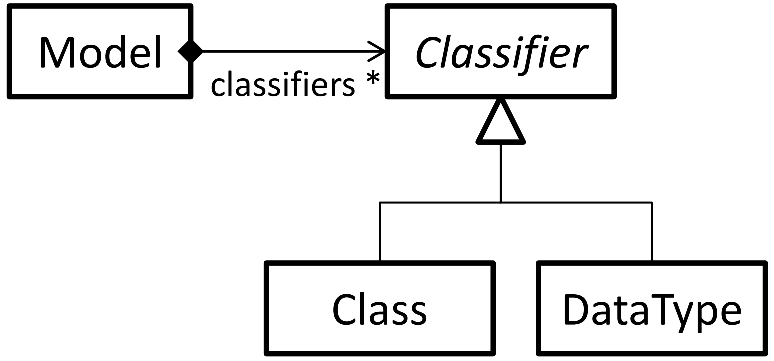
```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
       widgets <- c.features
  )
}
```

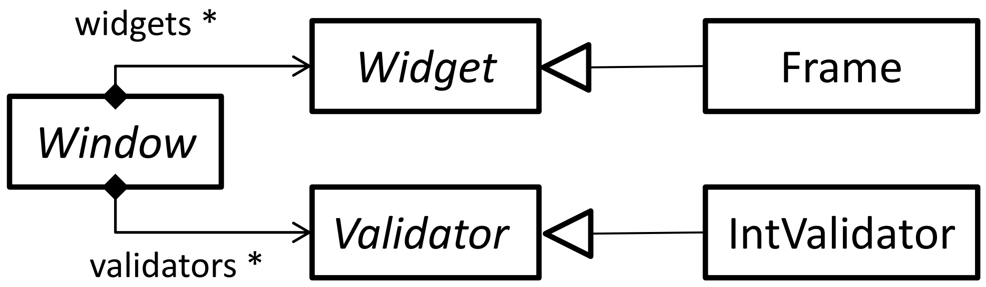
  - Feature `title` is compulsory, but the rule is not setting it.
  - Will cause problems in other transformations relying on the existence of a value for `title`.

Target conformance problems

- Binding resolved by rule with invalid target
  - Difficult to detect
  - Typically occur when there are different structures and inheritance is involved
  - Also, one needs to be careful when a rule has several target patterns
    - Only the first one is assigned

Target conformance problems





```
rule model2gui {
  from m : CD!Model
  to   w : GUI!Window (
         name <- m.name,
         widgets <- m.classifiers
  )
}
```

```
rule class2frame {
 from c : CD!Class ( not c.isAbstract )
 to w : GUI!Frame  ( … )
}
```

```
rule  int2validator {
 from d : CD!DataType (d.name = ‘Integer’)
 to w : GUI!Validator  ( … )
}
```

Transformation integrity

- ATL code which is syntactically correct but leads to unexpected behaviour.
- Example
  - Are filters in lazy rules allowed?

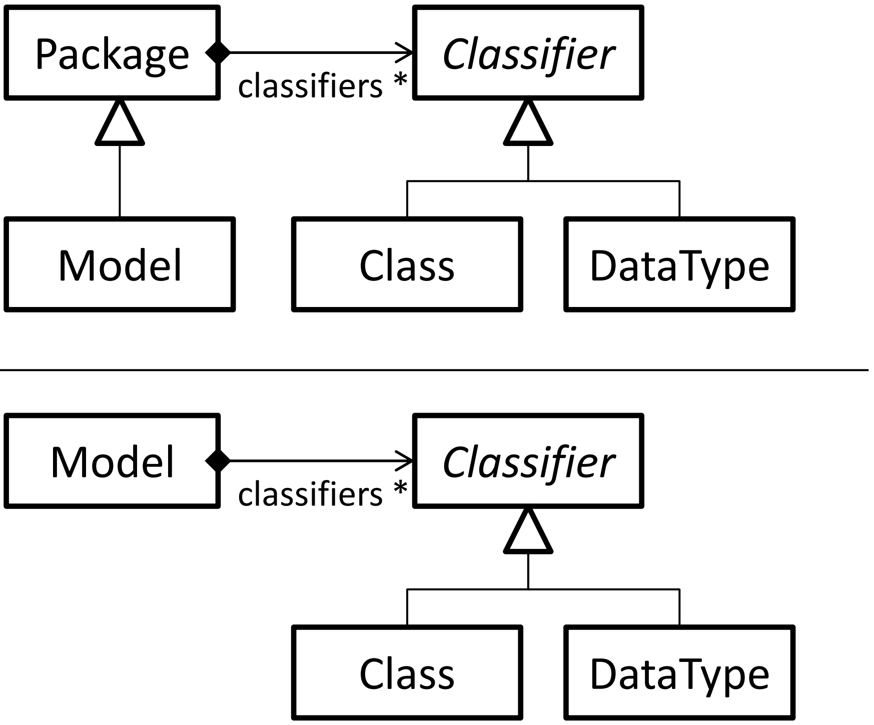
```
lazy rule attribute2text {
  from f : CD!Feature ( f.oclIsKindOf(CD!Attribute) )
  to   t : GUI!Text
}
```

- The lazy rule will be executed regardless of the filter.
  - This does not apply it the lazy rule inherits from an abstract lazy rule.

uml2cd_rule_conflict.atl

Transformation rules problems

- Rule conflict
  - Two matched rules should not match the same source element



```
rule model2model {
  from m1: UML!Model
  to   m2: CD!Model
}

rule package2model {
  from p: UML!Package
  to   m: CD!Model
}
```

Solution #1. Make model2model inherit package2model
Solution #2. Add filter p.oclIsType(UML!Package)

Transformation rules problems

- Unresolved binding
  - What happens when there is no rule to resolve an element appearing in the right part of a binding?
  - Example:

```
rule model2gui {
  from m: CD!Model
    to  w: GUI!Window (
     widgets <- m.classifiers->
                  select(c | c.oclIsKindOf(CD!Class))
```

  - If you have a rule with a filter to discard abstract classes, you get

Cannot set feature widgets to value [org.eclipse.emf.ecore.impl.DynamicEObjectImpl@4a12c7a0 (eClass: org.eclipse.emf.ecore.impl.EClassImpl@54087d0d (name: Frame) (instanceClassName: null) (abstract: false, interface: false)), org.eclipse.emf.ecore.impl.DynamicEObjectImpl@632e536 (eClass: org.eclipse.emf.ecore.impl.EClassImpl@789537ef (name: Class) (instanceClassName: null) (abstract: false, interface: false)), org.eclipse.emf.ecore.impl.DynamicEObjectImpl@f99ae63 (eClass: org.eclipse.emf.ecore.impl.EClassImpl@54087d0d (name: Frame) (instanceClassName: null) (abstract: false, interface: false)), org.eclipse.emf.ecore.impl.DynamicEObjectImpl@6704dd1e (eClass: org.eclipse.emf.ecore.impl.EClassImpl@54087d0d (name: Frame) (instanceClassName: null) (abstract: false, interface: false))], inter-model references are forbidden. Configure launching options to allow them.

Transformation rules problems

- Unresolved binding
  - Should be treated appropriately
  - It is a smell of incompleteness in the transformation
    - Not all cases are covered
  - If the cases don't need to be considered:
    - Filter the right-hand side of the binding
    - Write a pre-condition
    - Ignore (but documenting)

Pre-conditions

- Useful to document the conditions under which the transformation actually works
- Used by anATLyzer to filter out problems
  - Need to be written formally
- AnATLyzer:
  - Support as module annotations
  - Used to check problems

```
-- @pre CD!DataType.allInstances()->forAll(c |
--    c.name = 'Integer' or c.name = ‘String' or c.name = 'Date' )
--
```

↑
Leave an empty line comment as separator

Preconditions

```
-- @pre UML!Classifier.allInstances()->forAll(c |
--     c.oclIsTypeOf(UML!Class) or c.oclIsTypeOf(UML!DataType))

module "uml2cd preconditions";
create OUT: CD from IN: UML;

rule Model2Model {
   from m : UML!Model
     to w : CD!Model (
        name <- m.name,
        classifiers <- m.ownedType->select(c | c.oclIsKindOf(UML!Classifier))
   )
}

rule Class2Class {
   from m : UML!Class
     to w : CD!Class ( name <- m.name )
}

rule DataType2DataType {
   from m : UML!DataType
     to w : CD!DataType ( name <- m.name )
}
```

Annotations

- Ignore annotations
  - They are used to remove problems of a certain type in a rule or helper
  - Easy access via a quick fix
  - Examples:
    - **-- @ignore unresolved-binding**
    - **-- @ignore no-binding-compulsory-feature**

Annotations

- Force return type
  - To prefer declared type over inferred
  - Type inference is typically precise, but false positives may arise
  - **-- @force-declared-return-type**

```
-- @force-declared-return-type
helper context UML!Element def: getContainingModel() : UML!Model =
        if self.refImmediateComposite().oclIsTypeOf(UML!Model) then
                self.refImmediateComposite()
        else
                self.refImmediateComposite().getContainingModel()
        endif;
```

Special operations

- `oclAsType`
  - ATL does not have a downcasting operation
  - If you implement this dummy operation:

```
helper context OclAny def: oclAsType(t : OclType) : OclAny = self;
```

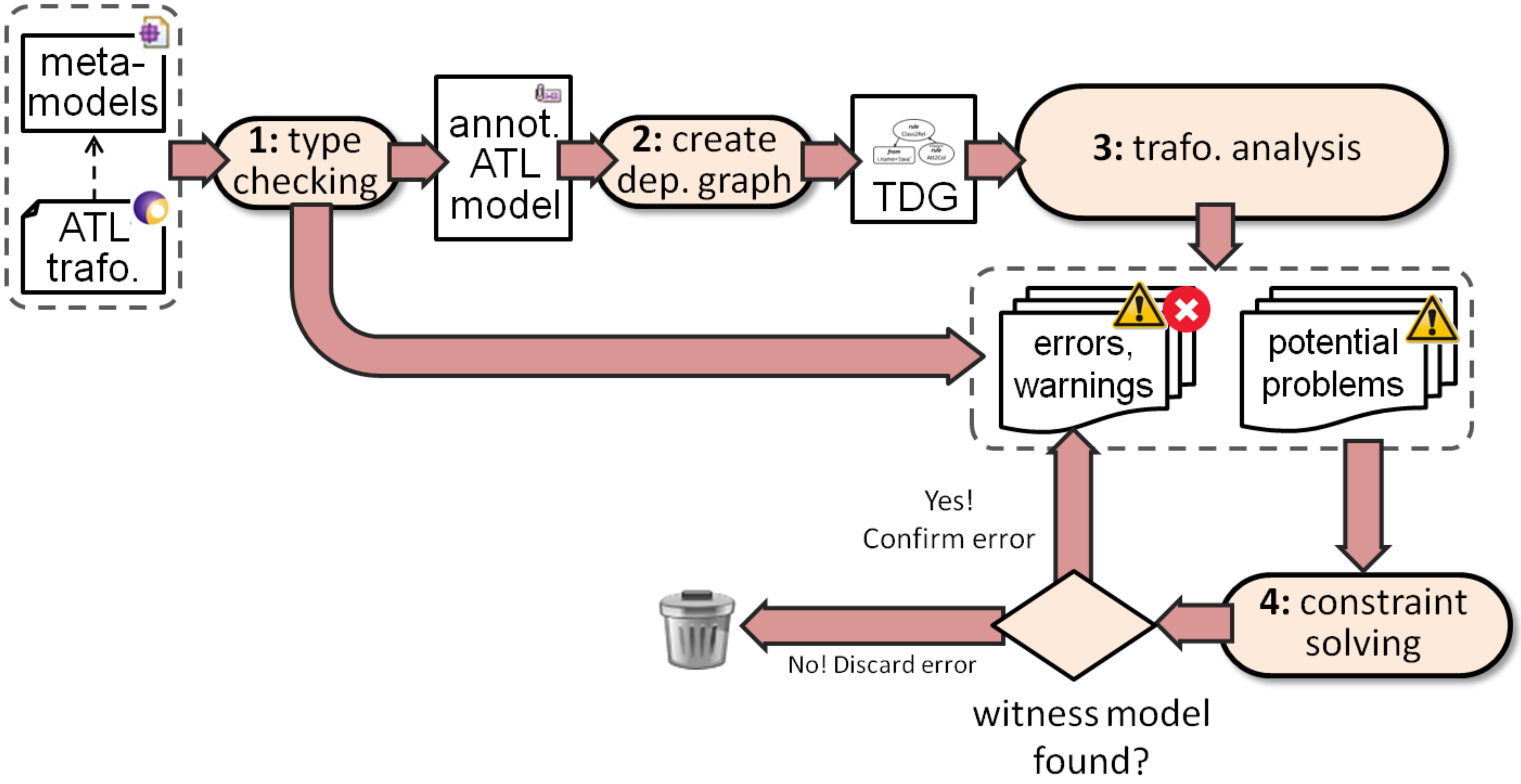
  - AnATLyzer recognizes to avoid so many nested ifs.

- `fail_(str : message)`
  - OclUndefined.fail_("Pattern match error")
  - To indicate an impossible path in your code

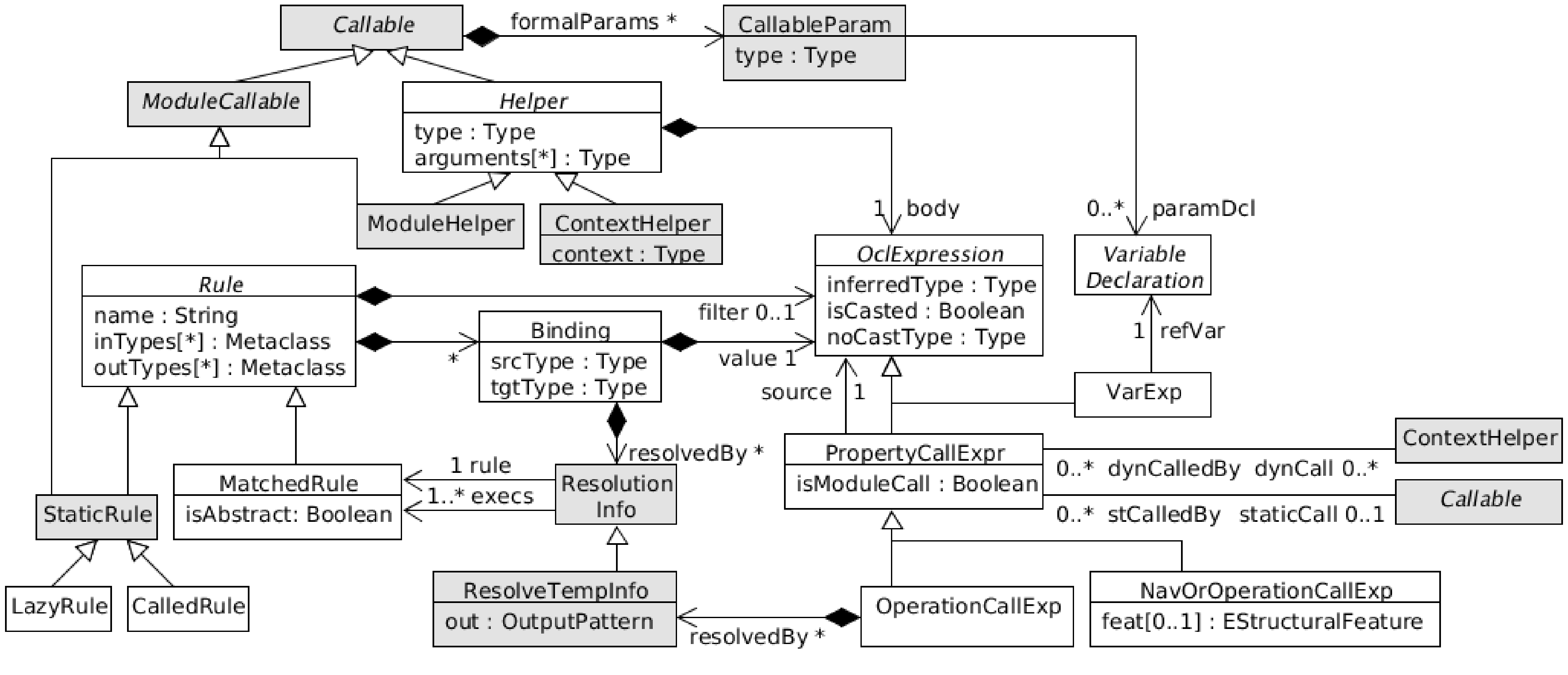
AnATLyzer

Implementation details

How does it works?



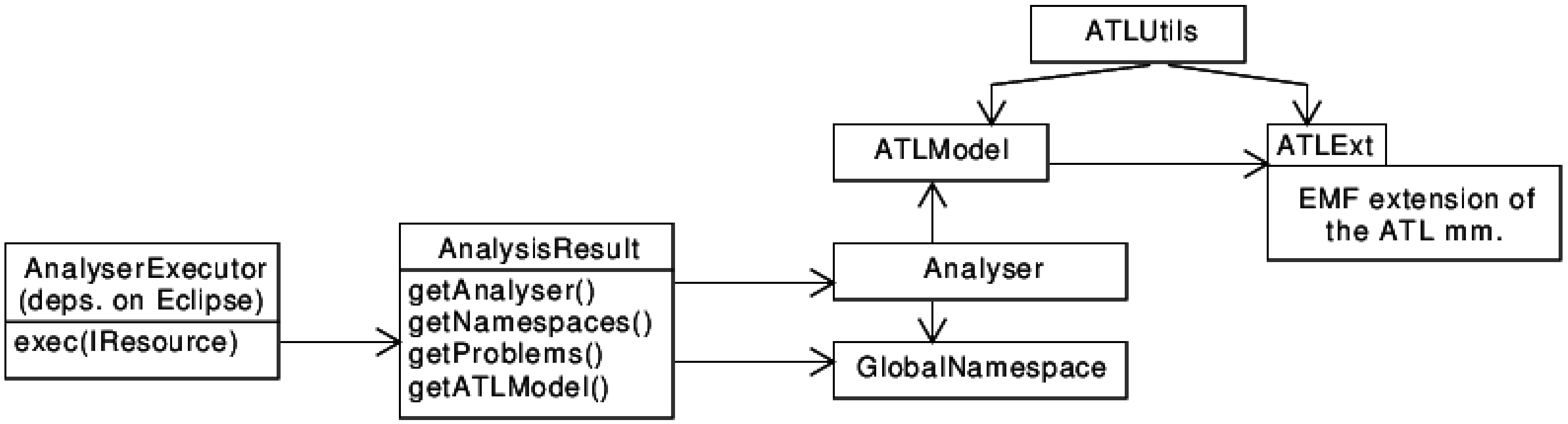
Transformation dependency graph



API

- Executing the analyser
- Access to:
  - The anATLyzer ATL's abstract syntax
  - Launching the model finder
- Implementing new analysis
- Implementing quick fixes
- Contributing actions

API


ATLUtils
ATLModel
ATLExt
EMF extension of the ATL mm.
AnalyserExecutor (deps. on Eclipse)
exec(IResource)
AnalysisResult
getAnalyser()
getNamespaces()
getProblems()
getATLModel()
Analyser
GlobalNamespace

Limitations

- Many!
  - Including fixing bugs
- Cannot re-analyse dependent transformations or changes in the meta-model
  - Lack of standard mega-model

Limitations

- Typing
  - Type inference for (mutually) recursive helpers may lead to false positives sometimes
- Mapping to USE Validator
  - We have good coverage but we have to work on e.g., Map and Tuple support

References

- *Uncovering Errors in ATL Model Transformations Using Static Analysis and Constraint Solving*. Jesús Sánchez Cuadrado, Esther Guerra, Juan de Lara ISSRE 2014: 34-44
- *Quick fixing ATL transformations with speculative analysis*. Jesús Sánchez Cuadrado, Esther Guerra, Juan de Lara. Software and Systems Modeling, 2016 (Springer), *In press*.

(Available at http://miso.es)

More information

- If you need more information because:
  - You want to use it
  - You have found a bug
  - You want to collaborate

- Send me an email: jesus.sanchez.cuadrado@gmail.com