Part III

ANATLYZER API & EXTENSIONS

AnATLyzer usage scenarios

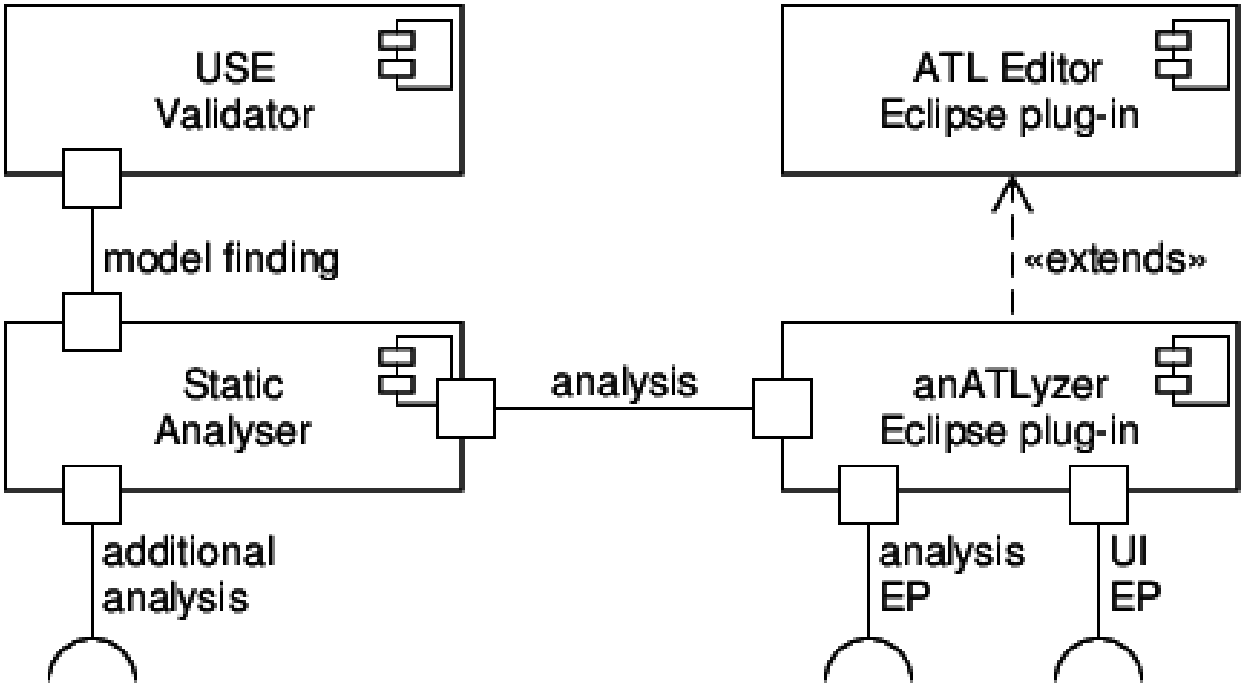
- IDE to develop ATL transformations
  - End-user perspective

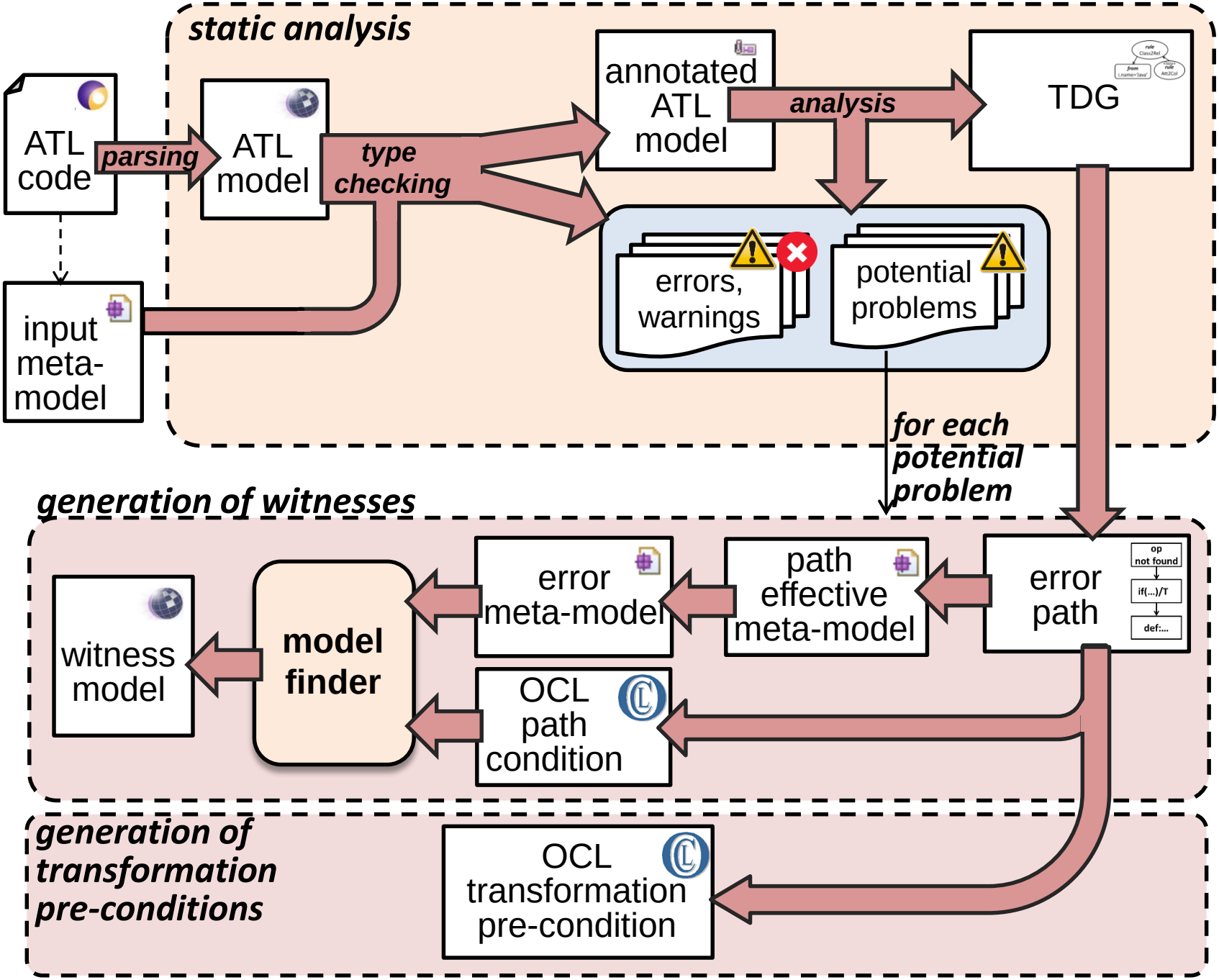
- Develop other tools for ATL
  - ATL tooling developer
- Run experiments on ATL transformations
  - Useful for researchers in MT

Learn how to use AnATLyzer programmatically

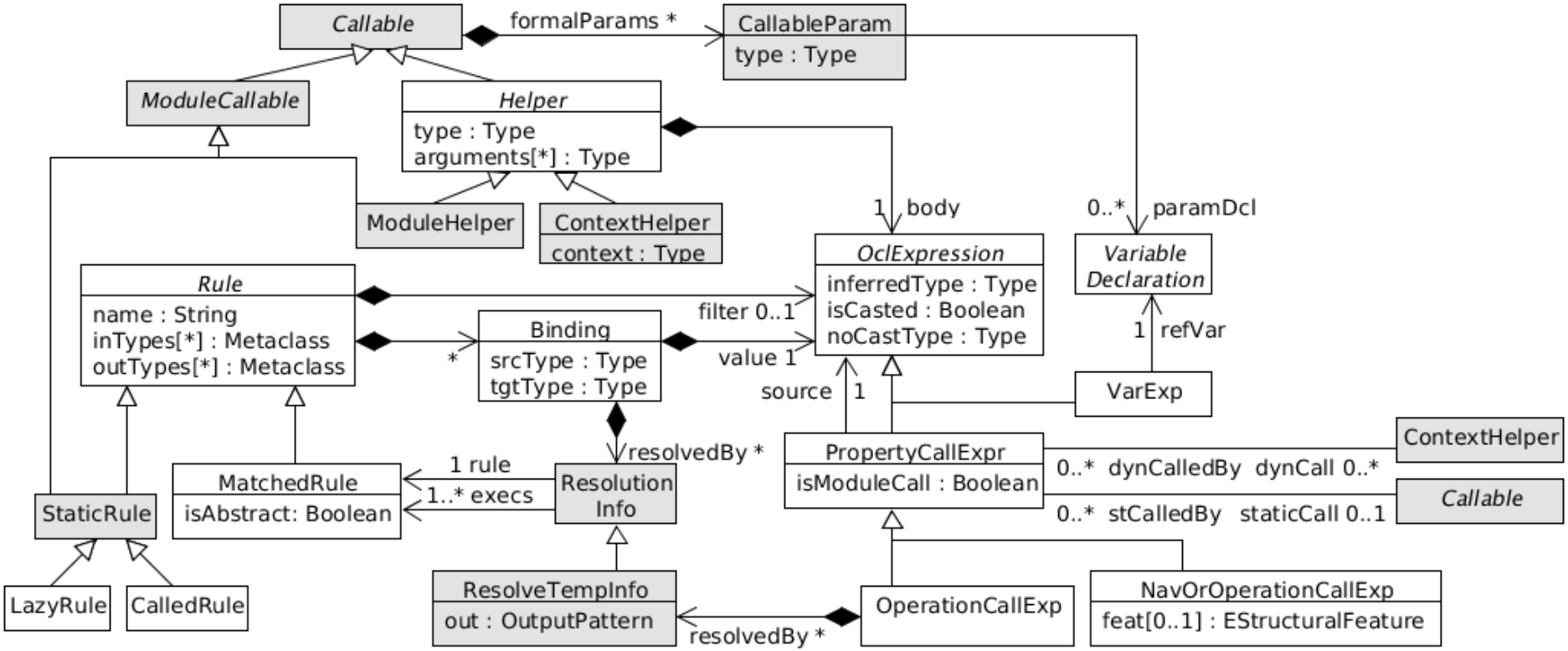
Architecture

Architecture



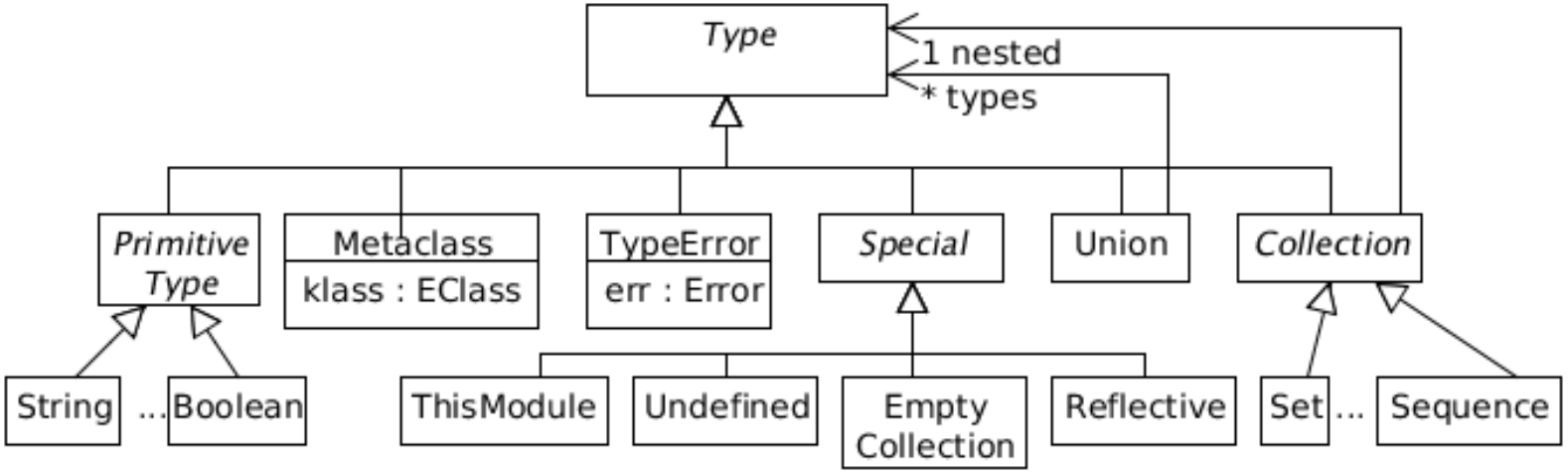

static analysis
ATL code
parsing
ATL model
type checking
annotated ATL model
analysis
TDG
input meta-model
errors, warnings
potential problems
for each potential problem
generation of witnesses
witness model
model finder
error meta-model
path effective meta-model
error path
OCL path condition
generation of transformation pre-conditions
OCL transformation pre-condition

Extended ATL meta-model



Available in: anatlyzer.atl.typing/model/generated/merged.ecore

Types meta-model



Available in: anatlyzer.atl.typing/model/types.ecore

Eclipse extension points

- Plug-in anatlyzer.atl.editor
  - anatlyzer.atl.editor.additionalanalysis
  - anatlyzer.atl.editor.quickfix

  - anatlyzer.atl.editor.quickassist
  - anatlyzer.atl.editor.problemexplanation
  - anatlyzer.atl.editor.views.additionalactions
  - anatlyzer.atl.editor.witnessvisualizer
  - anatlyzer.atl.editor.witnessfinder

Executing AnATLyzer programmatically

Executing the analyser

- Make use of the AnATLyzer infrastructure
  - Profit from the TDG
    - E.g., to implement optimized access to model repository for a specific transformation
  - Build generators from ATL to other technologies
    - E.g., a new ATL compiler for another platform
    - E.g., compile a some verifier backend
  - Run experiments over transformations in batch mode
    - E.g., compare how many errors are made by students

Project: models17.tutorial.standalone

ExampleAnalysis.java

AnATLyzer in standalone mode

```
final File atlTrafoFile = new File("examples/families2persons.atl");
final String[] names      = new String[] { "Persons", "Family" };
final String[] metamodels = new String[] { "examples/Persons.ecore",
                                           "examples/Families.ecore" };
// Configure the ATL resource
Resource atlTrafo = AtlLoader.load(atlTrafoFile.getAbsolutePath());
AnalysisLoader loader = AnalysisLoader.fromResource(atlTrafo, metamodels, names);

// Launch the type checking and the static analysis
AnalysisResult result = loader.analyse();

// For each detected problem, check if the constraint solver needs to be run
// to confirm the problem
for (Problem problem : result.getProblems()) {
  if ( AnalyserUtils.isWitnessRequred(problem) ) {
    StandaloneUSEWitnessFinder.confirmOrDiscard(problem, result);
  }

  if ( AnalyserUtils.isConfirmed(problem) ) {
    String desc  = AnalyserUtils.getProblemDescription(problem);
    String pdesc = problem.getDescription().replaceAll("\n", " ");
    String location    = (problem instanceof LocalProblem) ?
                         ((LocalProblem) problem).getLocation() : "no-location";

    System.out.println(des+ " "+atlTrafoFile.getName() + " " + location + " " + pdesc);
  }
}
```

ExampleMetrics.java

Using the extended ATL model
Example: Using the TDG

```
// Launch the type checking and the static analysis
AnalysisResult result = loader.analyse();

// Access the extended ATL model
ATLModel extATLModel = result.getATLModel();

// Query model declarations
List<ModelInfo> modelDeclarations = ATLUtils.getModelInfo(extATLModel);

for (ModelInfo declaration : modelDeclarations) {
  System.out.println("Signature: " + declaration.getModelName() + " : " +
                                      declaration.getMetamodelName());
  System.out.println("Is input: " + declaration.isInput());
  if ( declaration.hasMetamodelInfo() ) {
     System.out.println(declaration.getURIorPath());
     System.out.println("isURI: " + declaration.isURI());
  }
}

// Compute metrics
computeMetrics(extATLModel);
```

ExampleMetrics.java

Using the extended ATL model Example: Using TDG

```
public void computeMetrics(ATLModel model) {
    int numMatchedRules = 0;
    int numHelpers = 0;
    int numRuleLinks = 0;

    Module module = model.getModule();

    for (ModuleElement elem : module.getElements()) {
       if ( elem instanceof MatchedRule ) {
          numMatchedRules++;
          MatchedRule r = (MatchedRule) elem;
          for (OutPatternElement ope : r.getOutPattern().getElements()) {
             for (Binding binding : ope.getBindings()) {
                numRuleLinks += binding.getResolvedBy().size();
             }
          }
       } else if ( elem instanceof Helper ) {
          numHelpers++;
       }
    }

    System.out.println("Num. matched rules: " + numMatchedRules);
    System.out.println("Num. rule links: " + numRuleLinks);
    System.out.println("Num. helpers: " + numHelpers);
  }
```

ExampleMetricsWithVisitor.java

Using the extended ATL model Example: The TDG and the visitor

```
public class MetricsWithVisitor extends AbstractVisitor {
  int numMatchedRules = 0;
  int numHelpers  = 0;
  int numRuleLinks = 0;

  public void computeMetrics(ATLModel model) {
    // Start the depth-first traversal
    startVisiting(model.getRoot());

    System.out.println("Num. matched rules: " + numMatchedRules);
    System.out.println("Num. rule links: " + numRuleLinks);
    System.out.println("Num. helpers: " + numHelpers);
  }

  @Override
  public void inMatchedRule(MatchedRule self) { numMatchedRules++; }
  @Override
  public void inStaticHelper(StaticHelper self) { numHelpers++; }
  @Override
  public void inContextHelper(ContextHelper self) { numHelpers++; }
  @Override
  public void inBinding(Binding self) { numRuleLinks += self.getResolvedBy().size(); }

}
```

AnATLyzer within Eclipse

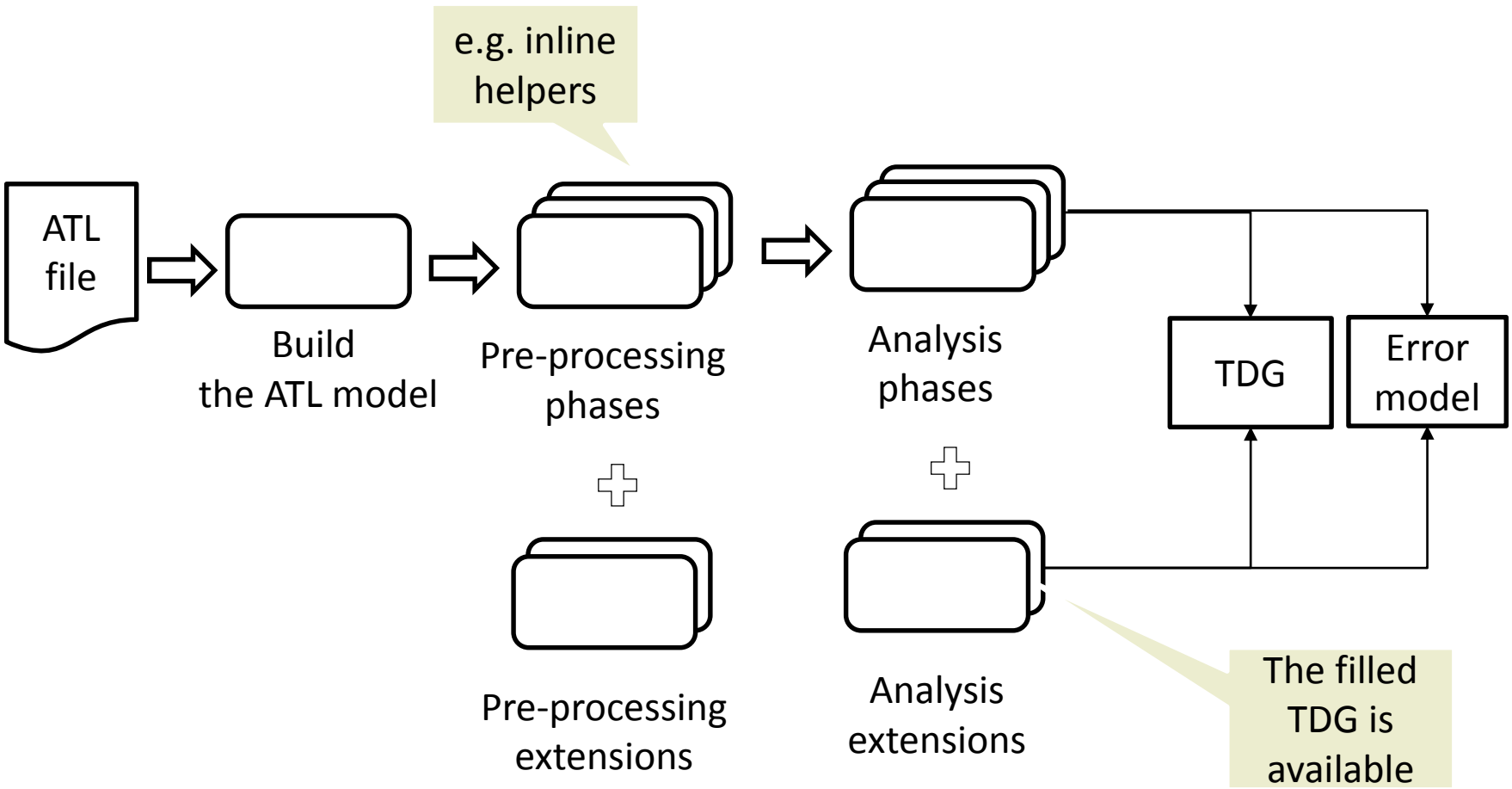
```
public class ExecuteAnalysisHandler extends AbstractHandler {

  public Object execute(ExecutionEvent event) throws ExecutionException {
    IEditorPart editor = HandlerUtil.getActiveEditor(event);
    if ( editor instanceof AtlEditorExt ) {
       IFile file = (IFile) ((AtlEditorExt) editor).getUnderlyingResource();

       // Query the index in case the file has already been analysed
       AnalysisResult r = AnalysisIndex.getInstance().getAnalysis(file);
       if( r == null ) {
          try {
             // Otherwise, execute the analysis
             // This is not executing the model finder for "witness-required" problems
             r = new AnalyserExecutor().exec(file);
          } catch (Exception e) { e.printStackTrace(); }
       }

       // Do something with the analysis
       MessageDialog.openInformation(null, "Analysis available!",
           "Problems: " + r.getLocalProblems().size());
    }
    return null;
  }
}
```

Implementing new analysis

AnATLyzer extensions

- AnATLyzer analysis are structured in a set of phases
- An extension may add a new phase
  - Before the regular analysis
  - After the regular analysis
    - In this stage the TDG is already filled with the results of the analysis

AnATLyzer extensions



Example

- UML-specific analysis
  - Feature “type” has cardinality “optional”
  - If UML is used as target model, it is likely that we want to make sure that “type” is initialized
  - Build an analysis that mark a problem if a rule is not setting a type feature

Example

- Use an extension point to install the analysis:
  - `anatlyzer.atl.editor.additionalanalysis`
  - The AnalysisProvider checks the applicability conditions and instantiate the new analysis

```
public class UmlAnalysis implements AnalysisProvider {
  @Override
  public List<AnalyserExtension> getExtensions(ATLModel m, GlobalNamespace ns) {
    List<ModelInfo> declarations = ATLUtils.getModelInfo(m);
    boolean isUmlMetamodel = declarations.stream().anyMatch(d ->
            d.getURIorPath().startsWith("http://www.eclipse.org/uml2/"));

    if ( isUmlMetamodel ) {
       return Collections.singletonList( new UmlAnalysisExtension() );
    }

    return Collections.emptyList();
  }
}
```

```java
public class UmlAnalysisExtension extends AbstractVisitor implements AnalyserExtension {
  private ATLModel model;

  @Override
  public boolean isPreparationTask() { return false; }

  @Override
  public void perform(ATLModel model, GlobalNamespace ns, Unit root) {
    this.model = model;
    startVisiting(root);
  }

  @Override
  public void inSimpleOutPatternElement(SimpleOutPatternElement self) {
    Metaclass m = (Metaclass) self.getInferredType();
    EClass    c = m.getKlass();
    if ( c != null && UMLPackage.Literals.TYPED_ELEMENT.isSuperTypeOf(c) ) {
       // Check if there is a bindings setting the "type" property
       if ( self.getBindings().stream().noneMatch(b ->
               b.getWrittenFeature() == UMLPackage.Literals.TYPED_ELEMENT__TYPE ) ) {

           model.getErrors().signalGenericProblem("Property 'type' is not set",
                                                  "type-not-set", self);
       }
    }
  }
}
```

Implementing quick fixes

Example

- UML-specific quick fix
  - Solution to the problem of assigning “superClass”
  - We will learn a bit about how to query the extended ATL model
  - It requires extensive creation of ATL model elements
  - We will change an ATL abstract syntax in-place

Example

```
-- Transforms Java classes (e.g., obtained with MoDISCO)
-- into UML classes and sets the inheritance links
module java2uml;
create CD : UML from CODE: JAVA;

helper context JAVA!ClassDeclaration
         def : getSuperClass : JAVA!ClassDeclaration = ... ;

rule class2class {
  from s1 : JAVA!ClassDeclaration
    to t1 : UML!Class (
        name <- s1.name,
        superClass <- s1.getSuperClass
    )
}
```

Example

```
rule class2class {
  from s1 : JAVA!ClassDeclaration
    to t1 : UML!Class (
        name <- s1.name,
        -- Binding changed to create a Generalization object
        generalization <-
            thisModule.createGeneralization(s1, s1.getSuperClass)
    )
}

-- Lazy rule automatically generated
lazy rule createGeneralization {
   from c : JAVA!ClassDeclaration, c1 : JAVA!ClassDeclaration
     to g : UML!Generalization (
        specific <- c,
        general <- c1
     )
}
```

Implementing a quick fix

- An extension point is provided
  - anatlyzer.atl.editor.quickfix
  - Two options:
    - quickfix: single quick fix
    - quickfix set: a group of related quick fixes

- Many examples are available in the anATLyzer source code, under project found in: `plugins/anatlyzer.atl.editor.quickfix`

Implementing a quick fix

- Example:
  - Quick fix for *“assignment to readonly feature”*
  - Create new plug-in project
    - Open MANIFEST.MF
    - Add anatlyzer.atl.editor.quickfix to “Dependencies”
  - Go to “Extensions” tab
    - Click on “Add”
    - Find and select **“anatlyzer.atl.editor.quickfix”**
    - It implicitly imports other required plugins like “anatlyzer.atl.editor” and “anatlyzer.atl.typing”

Implementing a quick fix

- Example:
  - Add the extension
    - Right-click on “New” -> “Quick fix”
    - In the “Apply” field specify the class that will handle the quick fix life cycle.
  - E.g., `anatlyzer.uml.quickfix.AssignmentToReadonlyFeature_CreateLazyRule`
  - The class must implement AtlProblemQuickfix
    - Alternatively, extend AbstractAtlQuickfix which already does much of the work.

Important methods

- AbstractAtlQuickfix
  - It has two phases: isApplicable and apply/getQuickfixApplication
- Main query methods (require a marker if invoked from isApplicable!)
  - getProblem
  - getProblematicElement
  - getATLModel
  - getAnalysisResult

Example

```
public class AssignmentToReadonlyFeature_CreateLazyRule
    extends AbstractAtlQuickfix  {

  @Override
  public boolean isApplicable(IMarker marker) throws CoreException {
     if ( getProblem(marker) instanceof AssignmentToReadonlyFeature ) {
       Binding b = (Binding) getProblematicElement(marker);
       return b.getWrittenFeature() ==
                 UMLPackage.Literals.CLASS__SUPER_CLASS &&
              b.getPropertyName().equals("superClass");
     }
     return false;
  }

  @Override
  public void apply(IDocument document) {
     try {
        QuickfixApplication qfa = getQuickfixApplication();
        new InDocumentSerializer(qfa, document).serialize();
     } catch (CoreException e) { }
  }
```

Example

The rest of the code is too large,
let’s use the editor

Implementing a quick fix

- Why?
  - Contribute new quick fixes to anATLyzer
  - Create quick fixes specific to your meta-models
    - Take advantage of domain information
    - Useful if you write transformations for the same meta-model many times
- Main difficulty:
  - You need to be familiar with anATLyzer problems
  - Described with a meta-model (errors.ecore)
  - See also ErrorModel, its elements
    - Or observe what is highlighted in the IDE to know which would be the abstract syntax element pointed by the problem

Available in: anatlyzer.atl.typing/model/errors.ecore