Part I

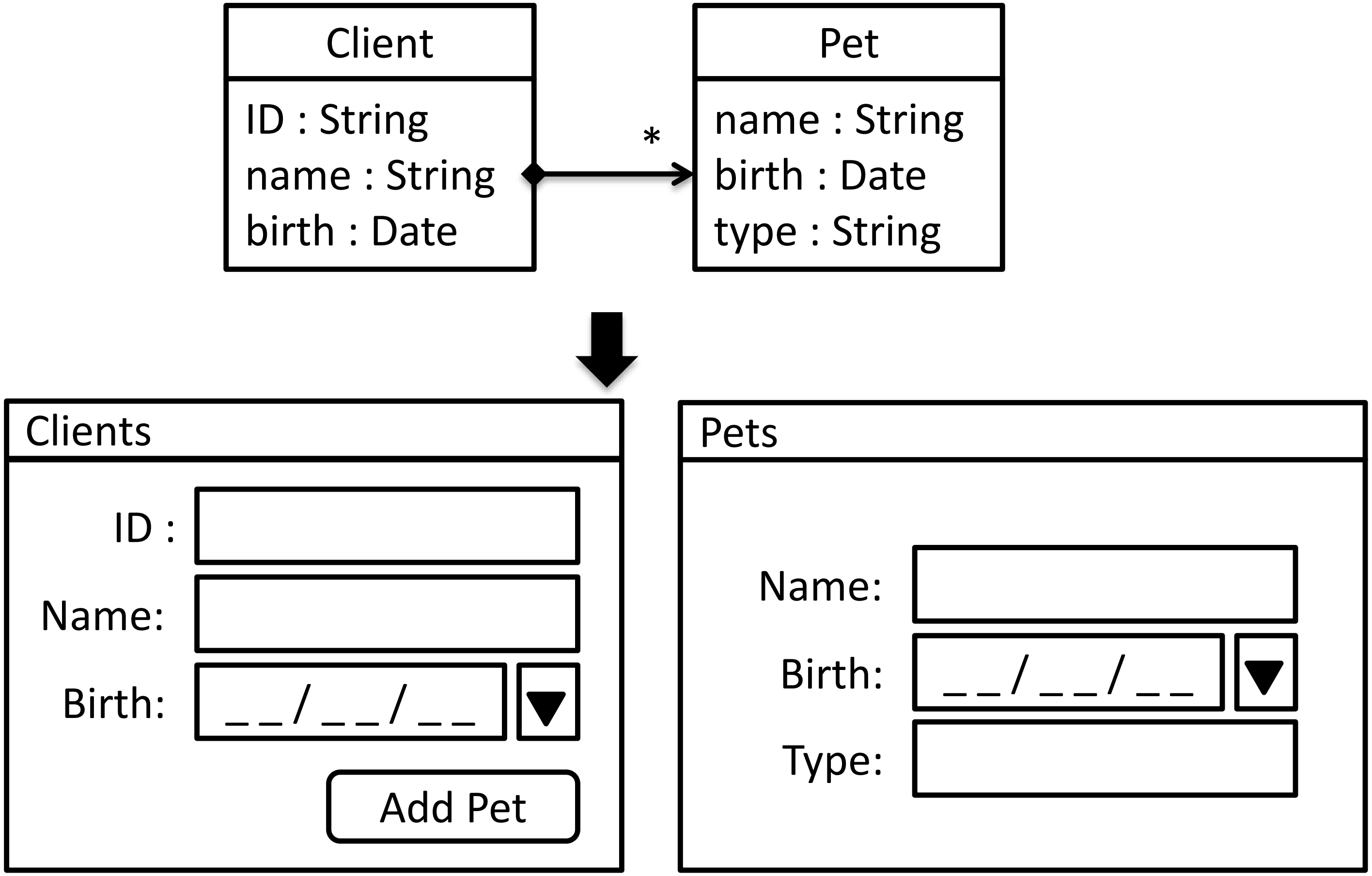
INTRODUCTION TO ATL

ATL Language

- ATL: ATLAS Transformation Language
  - Mature transformation infrastructure (>10 years)
  - Widely used language
  - https://eclipse.org/atl/
- ATL characteristics
  - Designed for model-to-model transformations
  - Source models are read-only
  - Target models are write-only
  - Rule-based + implicit reference resolution
  - Model navigation in OCL
  - Limited imperative constructs

Transformation example

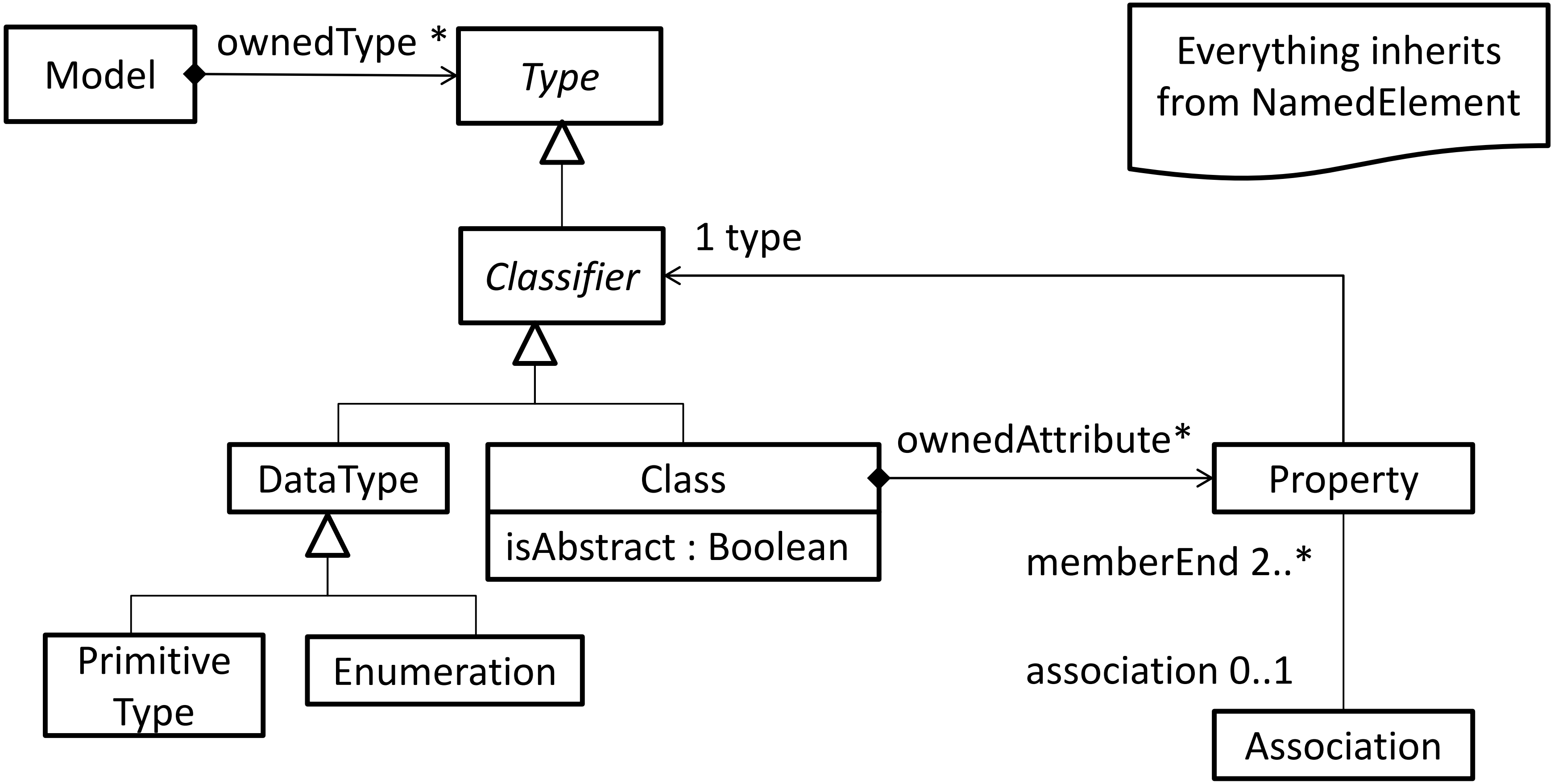
- UML class diagram to GUI



Transformation example

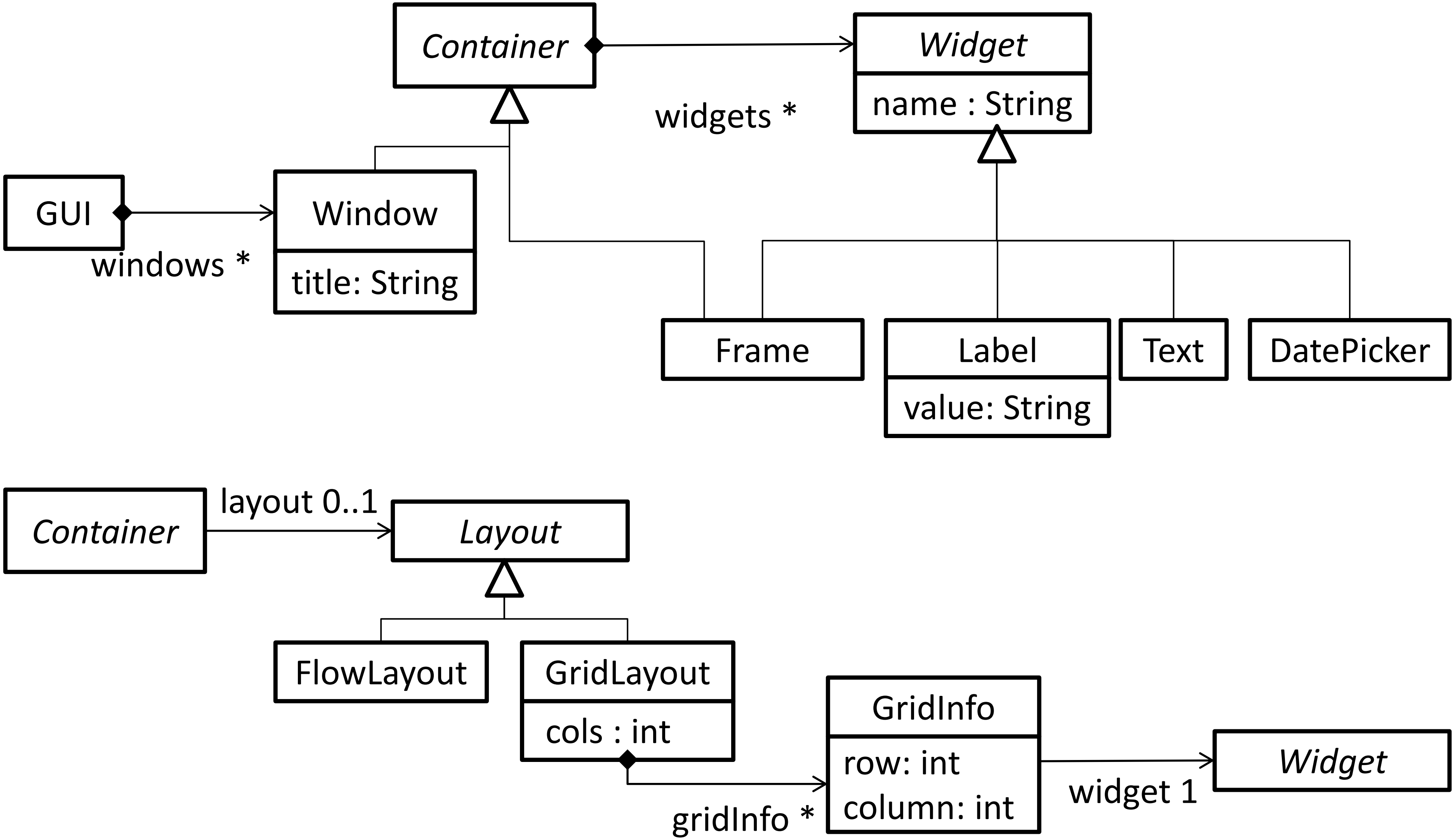
- Mapping at the high-level
  - Model would be a GUI
  - Class would be Frame or a Window
  - Each property a UI element
    - String properties would be text widgets
    - Date properties would be a date picker
    - References can be converted to buttons, etc.

UML class diagram meta-model



Everything inherits from NamedElement

GUI meta-model



uml2gui_intro_01.atl

ATL transformation

```
module “uml2gui";
create OUT: GUI from IN: CD;

helper context CD!Property def : isText() : Boolean =
  self.type.name = ‘String’;


rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
     title <- c.name,
     widgets <- c.ownedAttribute
  )
}

rule property2text {
  from p : CD!Property ( p.isText() )
  to t : GUI!Text
}
```

Module declaration

Helper

In pattern

Bindings

Out pattern

Matched rule

Introduction to ATL

Basic constructs

Module definition

- Name
  - No need to coincide with the file name
    - (need to be the same for EMFTVM)
  - Dots not allowed. Several words, with “ “

```
-- @atlcompiler atl2006
-- @nsURI CD=http://www.eclipse.org/uml2/5.0.0/UML
-- @path GUI=/models17.tutorial.uml2gui/metamodels/gui.ecore

module “uml2gui”;
create OUT : GUI from IN : CD;
```

Module definition

- Meta-model references
  - Not compulsory, but recommended
  - Enables auto-completion (+ anATLyzer)
  - @nsURI for registered meta-models
  - @path for workspace files

```
-- @atlcompiler atl2006
-- @nsURI CD=http://www.eclipse.org/uml2/5.0.0/UML
-- @path GUI=/models17.tutorial.uml2gui/metamodels/gui.ecore

module “uml2gui”;
create OUT : GUI from IN : CD;
```

Module definition

- Compiler directive

    -- @atlcompiler atl2004

    -- @atlcompiler atl2006

    -- @atlcompiler atl2010

    -- @atlcompiler emftvm

```
-- @atlcompiler atl2006
-- @nsURI CD=http://www.eclipse.org/uml2/5.0.0/UML
-- @path GUI=/models17.tutorial.uml2gui/metamodels/gui.ecore

module “uml2gui”;
create OUT : GUI from IN : CD;
```

Rules

- Matched rule
- Lazy rule
- Unique lazy rule

In this part

- Called rule
- Entry point rule
- Endpoint rule

Not covered

Matched rules

- Structure
  - Input pattern (from)
    - Optional filter/guard
  - Output pattern (to)
    - Contains bindings (<-)
  - Imperative block (do)
    - Optional. Discouraged.

- Behaviour
  - Executed implicitly, at the top level
  - Target elements created automatically
  - Target features initialized with *bindings*

```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
     title <- c.name,
     widgets <- c.ownedAttribute
  )
  do { ... }
}
```

Bindings

- Structure
  - Left part
    - Target feature
  - Right part
    - OCL expression

Primitive binding

```
    title <- c.name,
```

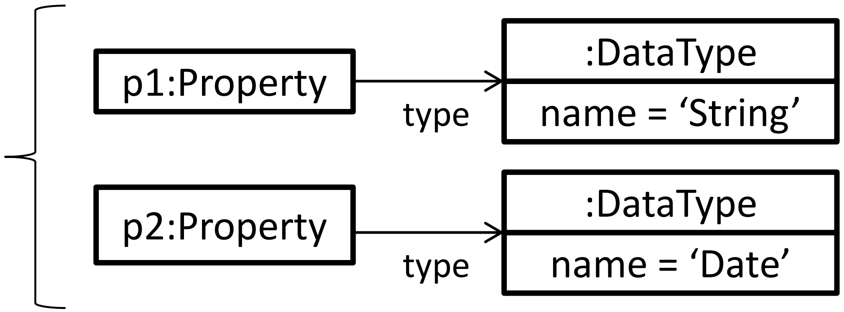
Object binding

```
    widgets <- c.ownedAttribute
```

- Behaviour
  - Right part is flattened
  - Primitive bindings
    - Left is primitive type
    - Right is primitive value
    - Direct assignment
  - Object bindings
    - Left type is meta-class
    - Right value is object

Binding resolution

```
rule class2frame {
  from c : CD!Class
  to   f : GUI!Frame (
   widgets <- c.ownedAttribute
  )
}
```



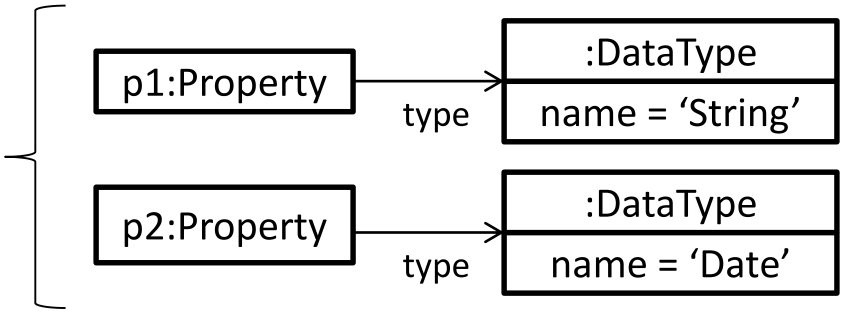
p1 matched by the input pattern?

```
rule property2text {
 from p : CD!Property ( p.isText() )
 to   t : GUI!Text
}
```

Object instantiated
Assigned to feature

Binding resolution

```
rule class2frame {
  from c : CD!Class
  to   f : GUI!Frame (
   widgets <- c.ownedAttribute
  )
}
```



p2 matched by the input pattern?

```
rule property2int {
  from p : CD!Property ( p.isDate() )
  to   t : GUI!DatePicker
}
```

Object instantiated
Assigned to feature

uml2gui_intro_02.atl

Resolving elements explicitly

- Problem: We want to attach a label to each widget.
  - Solution: add an additional *out pattern element*

```
rule property2text {
  from p : CD!Property ( p.isText() )
  to   t : GUI!Text ( ... ),
       l : GUI!Label ( ... )
}
```

Resolving elements explicitly

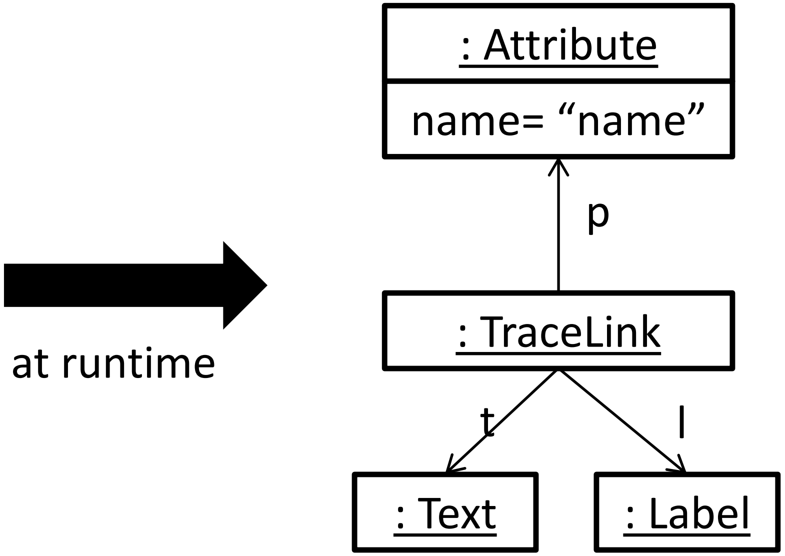
- Next problem, we need to link the label to its container
  - Remember, ATL only resolves the first element
  - Solution: `resolveTemp`

- **`thisModule`**`.resolveTemp(obj, ‘varName’)`
  - Performs the trace lookup for `obj` explicitly
  - Retrieves the element created with the output pattern element whose variable name is `‘varName’`

Resolving elements explicitly

```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
    title <- c.name,
    widgets <- c.ownedAttribute,
    widgets <- c.ownedAttribute->collect(a | thisModule.resolveTemp(a, ‘l’))
  )
}


rule property2text {
  from p : CD!Property ( p.isText() )
  to   t : GUI!Text ( ... ),
       l : GUI!Label ( ... )
}
```

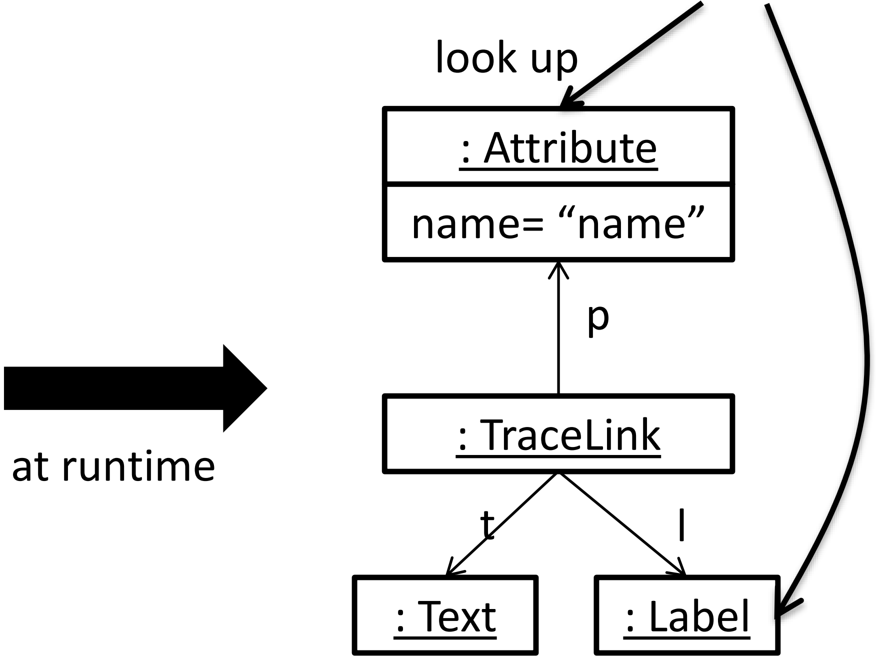


at runtime

Resolving elements explicitly

```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
    title <- c.name,
    widgets <- c.ownedAttribute,
    widgets <- c.ownedAttribute->collect(a | thisModule.resolveTemp(a, ‘l’))
  )
}


rule property2text {
  from p : CD!Property ( p.isText() )
  to   t : GUI!Text ( ... ),
       l : GUI!Label ( ... )
}
```



Model navigation – OCL

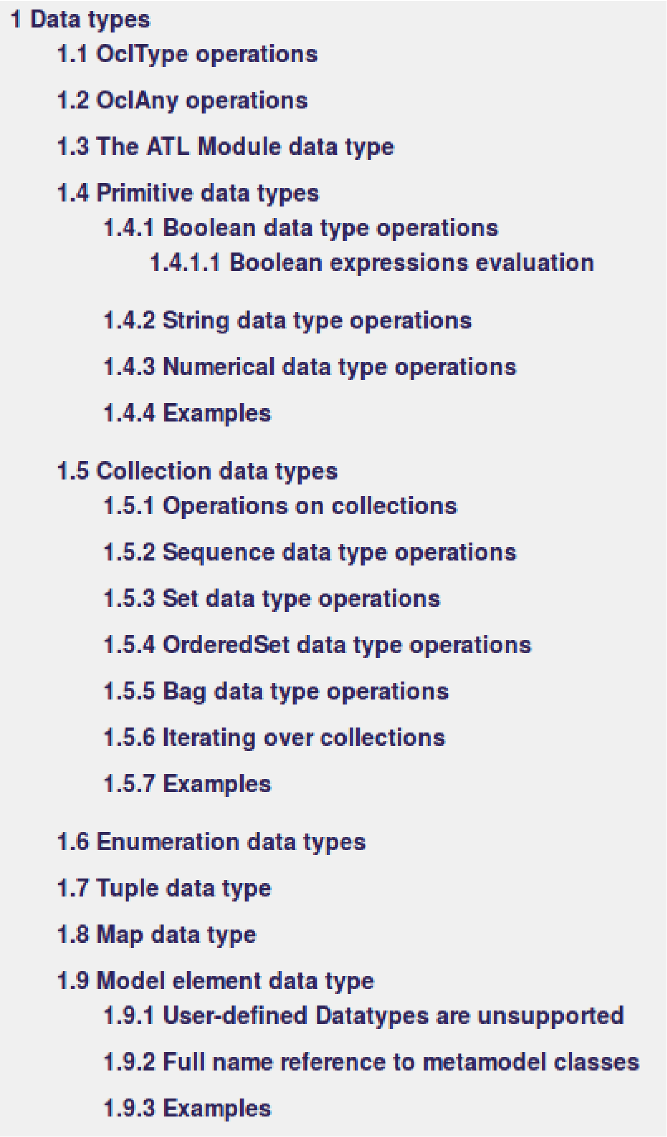
- ATL implements its own variant
  - Somewhat out of date with respect to newer versions
    - e.g., lack of closure operation
  - OCL is statically typed, ATL/OCL is not!
  - Model elements named with syntax `MM!Type`

OCL

- Example: get all attributes of type String in a class diagram

```
aModel.classifiers->
   select(c | c.oclIsKindOf(CD!Class))->
   collect(c | c.features->select(f | f.oclIsKindOf(CD!Attribute) )->
   flatten()->
   select(a | if a.type.oclIsUndefined() then
                  a.type.name = ‘String’
              else
                  false
              endif)
```

OCL



- Details about the supported operations in the ATL guide
- https://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language

Helpers

- “Methods” attached to (meta-model) types at runtime
- Two types
  - Module helpers
  - Context helpers
- Two modes
  - Operation
  - Attribute

Helpers

- **Module helpers**
  - Global helpers
  - Methods attached to “this transformation module”

```
thisModule.propsByName(‘age’)

helper def: propsByName(name : String) : Set(CD!Property) =
  CD!Attribute.allInstances()->select(p | p.name = name);
```

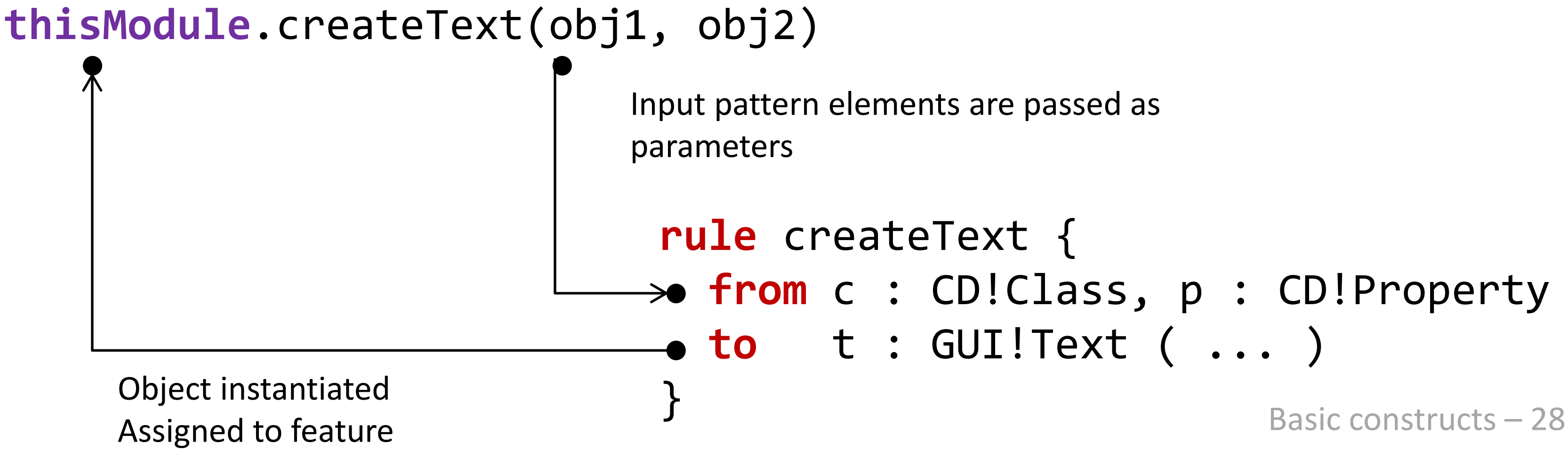
- **Context helpers**
  - Methods attached at runtime to a meta-class

```
aClass.hasProperty(‘age’)

helper context CD!Class def: hasProperty(name : String): Boolean =
  self.ownedAttribute->exists(p | p.name = name);
```

Lazy rules

- Rules which are explicitly invoked
  - – Same structure as matched rules
  - – No trace links are generated
- Can be invoked many times over the same source element



```
thisModule.createText(obj1, obj2)
```

Input pattern elements are passed as parameters

```
rule createText {
  from c : CD!Class, p : CD!Property
  to   t : GUI!Text ( ... )
}
```

Object instantiated
Assigned to feature

Lazy rules

```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
     title <- c.name,
     widgets <- c.ownedAttributes->collect(f |
        if f.isText()      then thisModule.property2text(f)
        else if f.isDate() then thisModule. property2date(f)
        else                    OclUndefined endif endif
     )
  )
}
```

You need to “pattern match” explicitly unless you use rule inheritance

```
lazy rule property2text {
  from p : CD!Property
  to   t : GUI!Text
}
```

```
lazy rule property2date {
  from p : CD!Property
  to   t : GUI!DatePicker
}
```

Unique lazy rules

- Similar to lazy rules, but they keep trace links
  - Useful if a matched rule is subordinated to the execution of others
  - Required if the target element of a lazy rule must be reused
    - For example, the previous modification did not considered the layout...

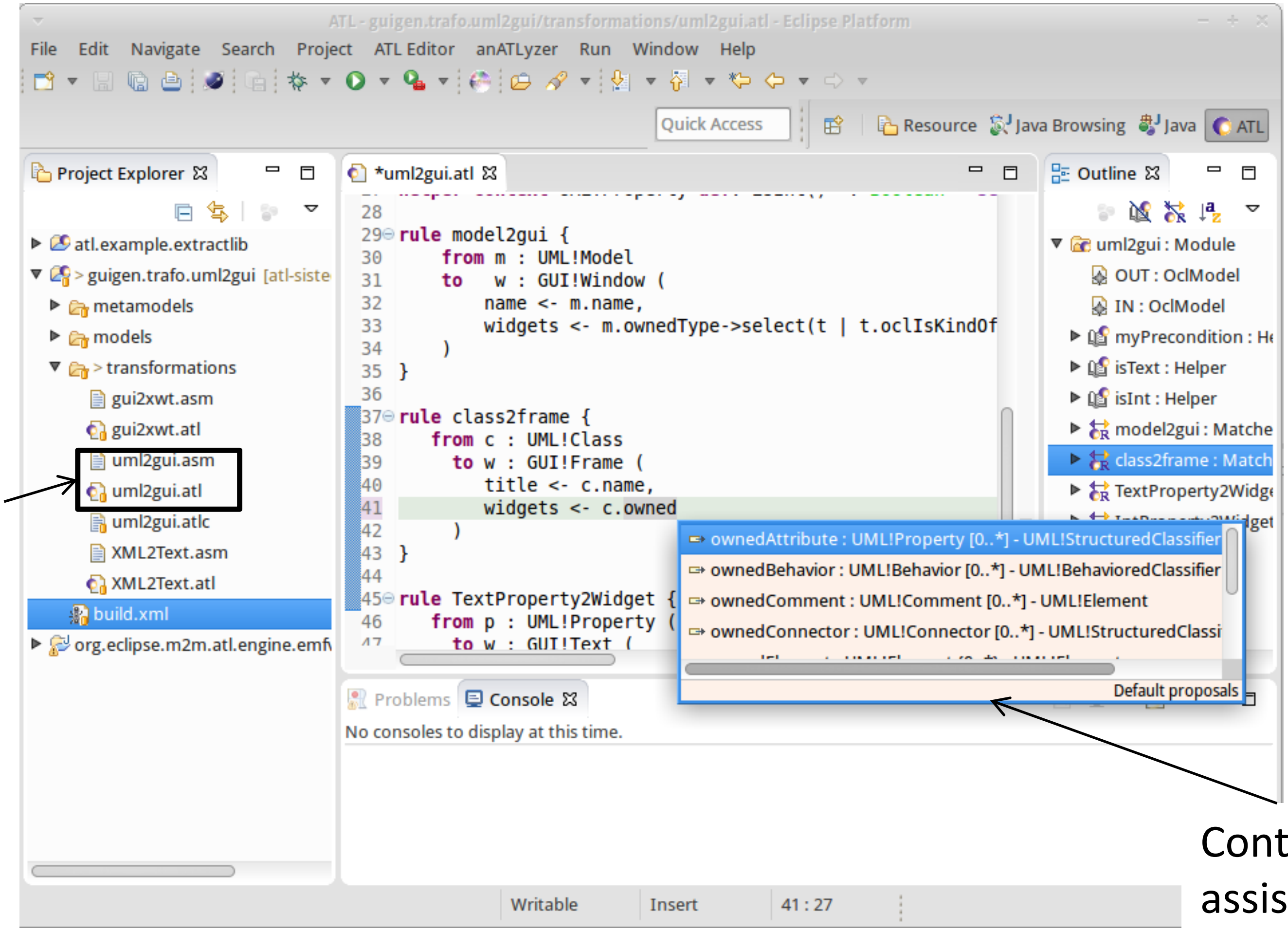
Introduction to ATL

Tooling

ATL Plug-in

- Features
  - ATL perspective
    - Register meta-model button
  - Editor with syntax highlighting
  - Automatic compilation
  - Autocompletion + Code templates
    - CTRL + SPACE
  - Outline view
  - ATL Console
  - Launching transformations

ATL Editor



```
rule model2gui {
    from m : UML!Model
    to   w : GUI!Window (
        name <- m.name,
        widgets <- m.ownedType->select(t | t.oclIsKindOf
    )
}

rule class2frame {
   from c : UML!Class
     to w : GUI!Frame (
        title <- c.name,
        widgets <- c.owned
     )
}

rule TextProperty2Widget {
   from p : UML!Property (
     to w : GUI!Text (
```

- ownedAttribute : UML!Property [0..*] - UML!StructuredClassifier
- ownedBehavior : UML!Behavior [0..*] - UML!BehavioredClassifier
- ownedComment : UML!Comment [0..*] - UML!Element
- ownedConnector : UML!Connector [0..*] - UML!StructuredClassi

Default proposals

No consoles to display at this time.

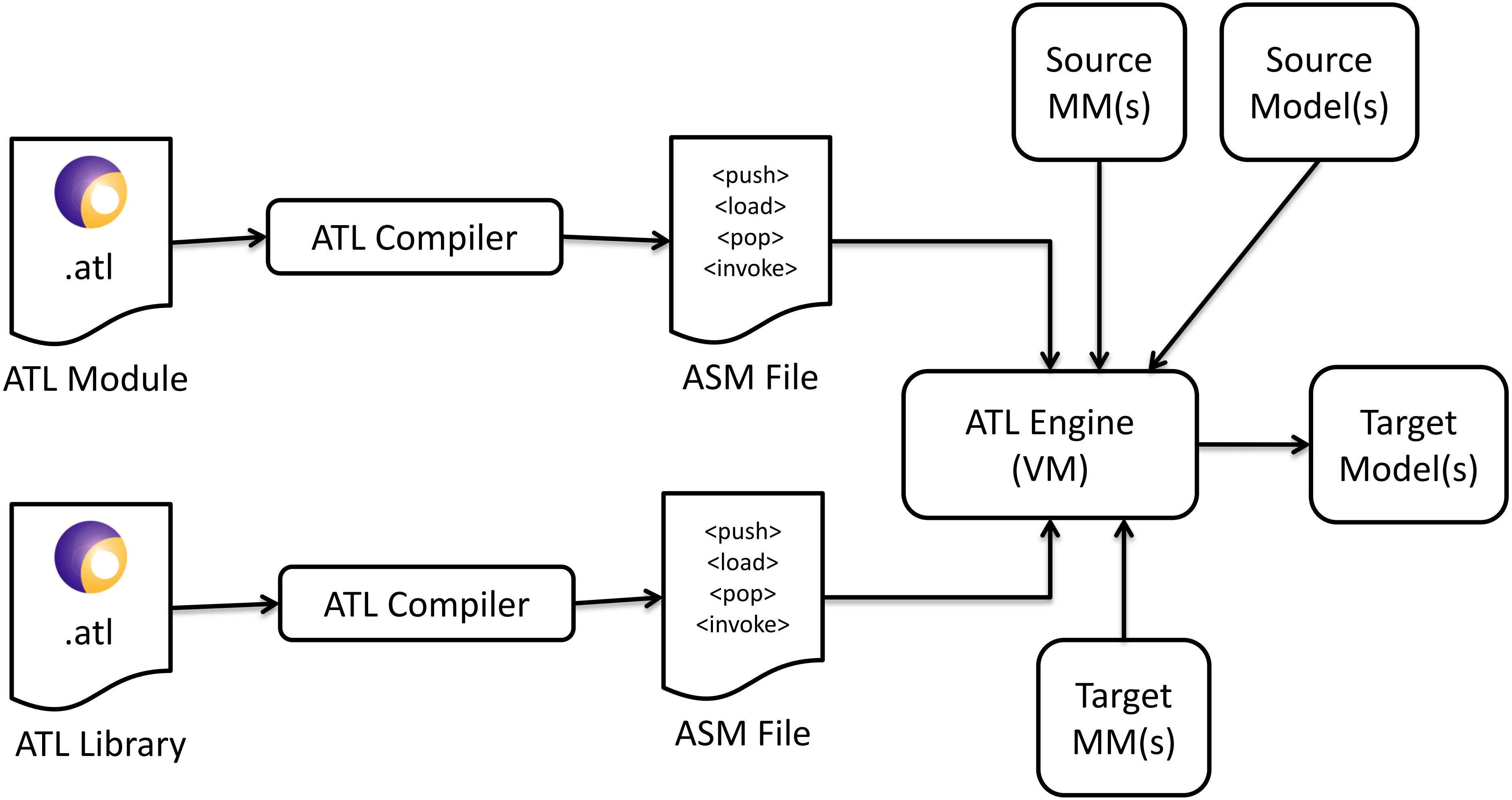
Automated compilation

Content assist

Syntax error highlighting

Dedicated launcher

Compilation & Execution



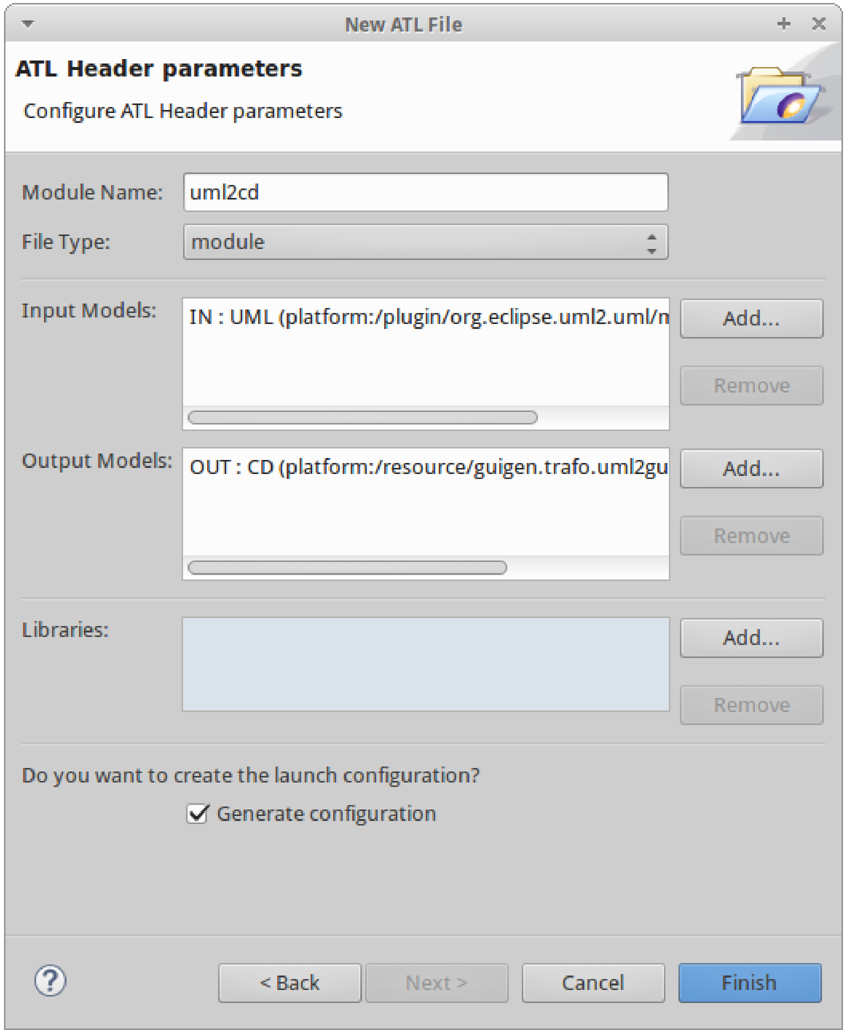
Project structure

- File -> Project .. -> ATL Project
  - The projects are created with no structure
  - Possible structure

```
myProject
  + launching
  + metamodels
  + models
  + output
  + transformations
```

New ATL transformation

- File -> New ... -> ATL File



ATL Header parameters

Configure ATL Header parameters

Module Name: uml2cd

File Type: module

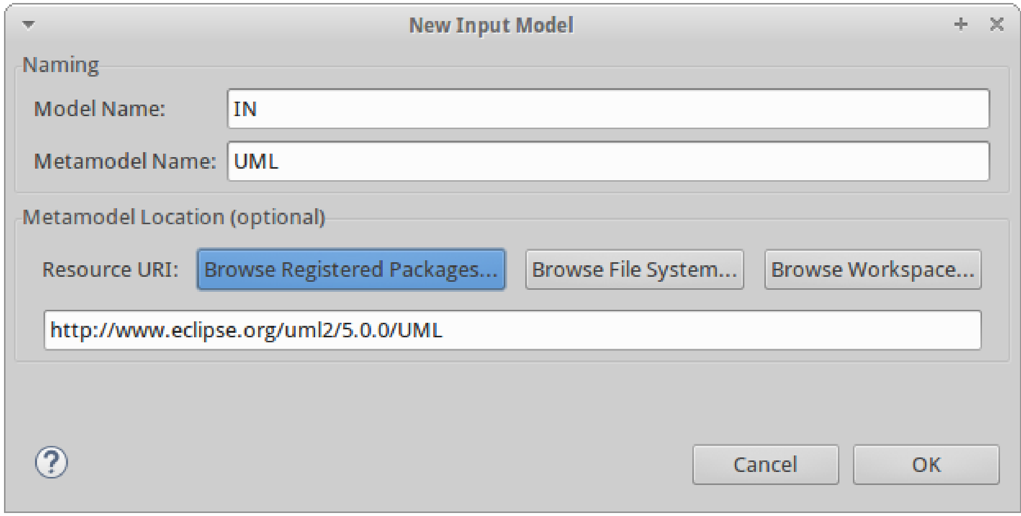
Input Models: IN : UML (platform:/plugin/org.eclipse.uml2.uml/n

Output Models: OUT : CD (platform:/resource/guigen.trafo.uml2gu

Libraries:

Do you want to create the launch configuration?

☑ Generate configuration



Naming

Model Name: IN

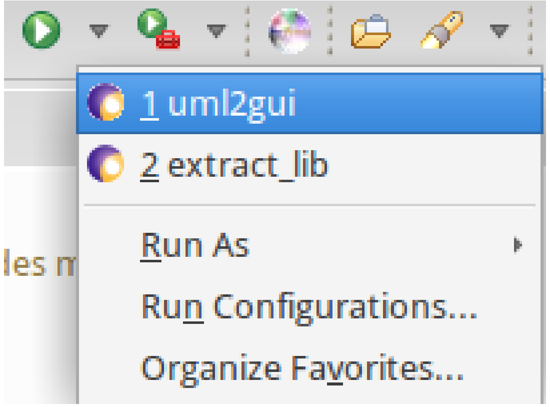
Metamodel Name: UML

Metamodel Location (optional)

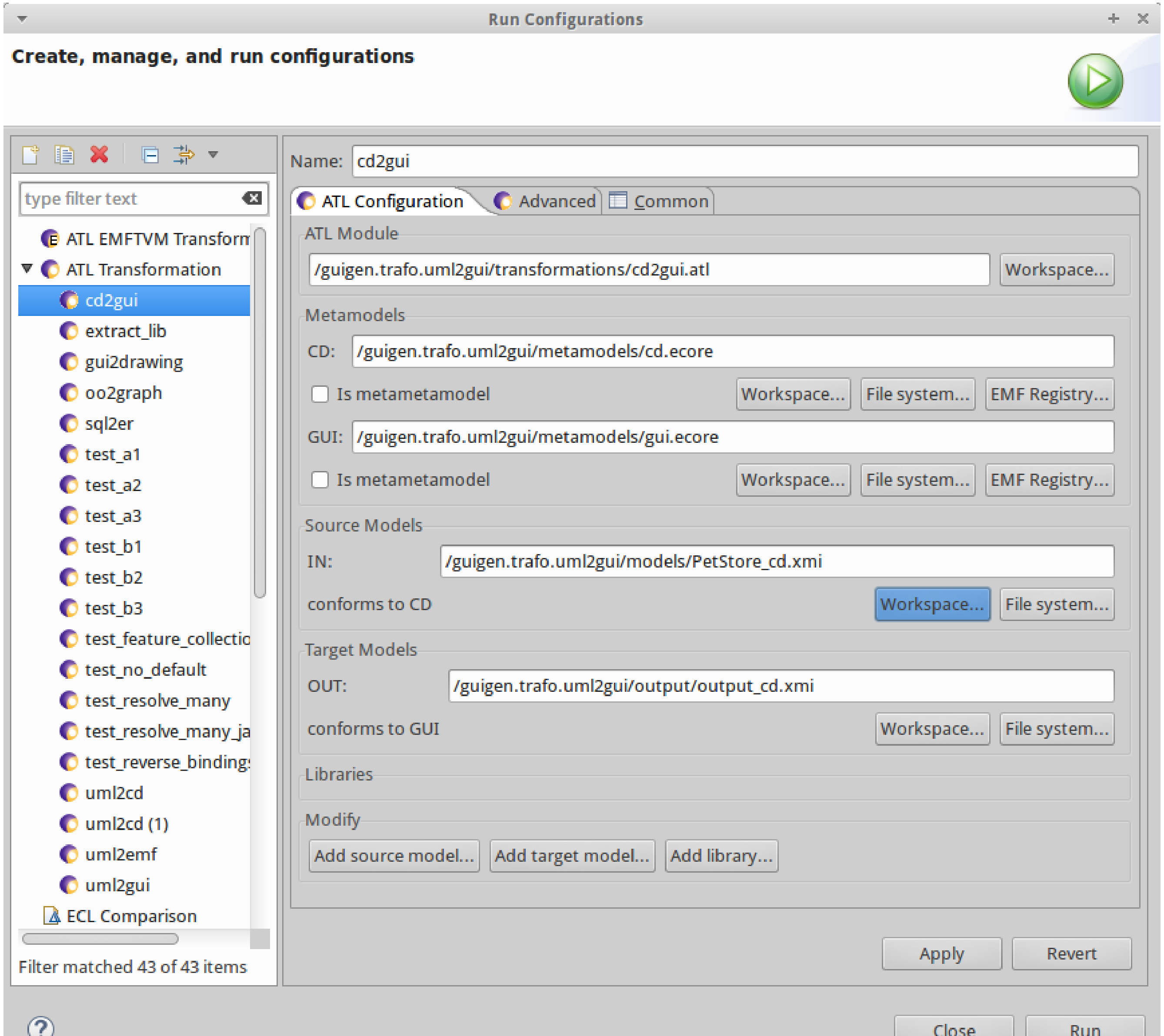
Resource URI: http://www.eclipse.org/uml2/5.0.0/UML

Launching

- Dedicated launcher
  - Based on Eclipse infrastructure
  - Accessible via the “play button”
- Right-click on the ATL file
  - Run as… -> ATL Transformation
  - Meta-model information is automatically filled in if you have the proper annotations



Launching



Run Configurations

Create, manage, and run configurations

Name: cd2gui

ATL Module

/guigen.trafo.uml2gui/transformations/cd2gui.atl

Metamodels

CD: /guigen.trafo.uml2gui/metamodels/cd.ecore

☐ Is metametamodel

GUI: /guigen.trafo.uml2gui/metamodels/gui.ecore

☐ Is metametamodel

Source Models

IN: /guigen.trafo.uml2gui/models/PetStore_cd.xmi

conforms to CD

Target Models

OUT: /guigen.trafo.uml2gui/output/output_cd.xmi

conforms to GUI

Libraries

Modify

Filter matched 43 of 43 items

Launching

- Opening the output model
  - Not that easy…
- XMI files does not include schemaLocation information
- Registering meta-models is a must
  - The ATL perspective must be active to have access to the register meta-model button
  - Right-click on the target meta-model file
    - Register meta-model