MeTAGeM-Trace User Manual

Álvaro Jiménez, Juan M. Vara, Verónica A. Bollati and Esperanza Marcos

Kybele Research Group, University Rey Juan Carlos, 28933 MADRID, Spain

---

This manual shows how to use `MeTAGeM-Trace` to develop the `Families2-Persons`[1] transformation This scenario is used in order to avoid adding accidental complexity. A number of more complex scenarios (apart from the one used here) can be downloaded from the afore-mentioned Web site.

1. Set-up

We are going to use the source files provided with the ATL `Families2-Persons` basic example as source and target (meta)models. Therefore, following tasks must be completed before start the development of the transformation:

1. Download the source files of the ATL basic example[2] to your local drive. Create a new project: `File -> New -> General -> Project`.
2. Import the project into your workspace: `File -> Import -> Existing projects into workspace -> Select archive file (radio button) -> Finish`.
3. Name the project as `Families2persons`.

2. Defining the `MeTAGeM` transformation model

The process starts by defining a `MeTAGeMDSL` model that collects (at metamodel level) the relationships that must hold between source and target models. `MeTAGeM-Trace` provides a wizard to collect the data needed for the creation of such model:

- `File -> New -> Other -> MeTAGeM-Trace -> Metagem model`. See Figure 1(a).
- Next, you have to set the source and target metamodels of the transformation (recall that several source and target metamodels are allowed).
- To that end, click the `Add model` button in order to give a (local) name and state which is the source file of each metamodel involved in the transformation. See Figure 1(b) and (c).

[1] http://www.eclipse.org/m2m/atl/basicExamples_Patterns/
[2] http://www.eclipse.org/m2m/atl/atlTransformations/Families2Persons/Families2Persons.zip

- As a result, an empty `Metagem` model is created. See Figure 2.

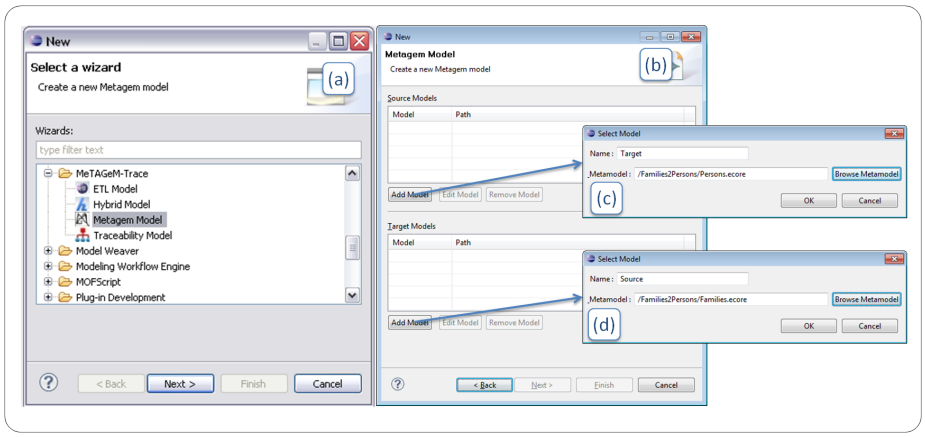

Figure 1: Metagem model creation wizard

As shown in the figure, the model is automatically opened in a multi-panel editor that bundles three different panels to show separately the source models, the high-level transformation metamodel (`Metagem` model) and the target metamodels. If there are several source or target metamodels, they are be co-located vertically in their corresponding panel.

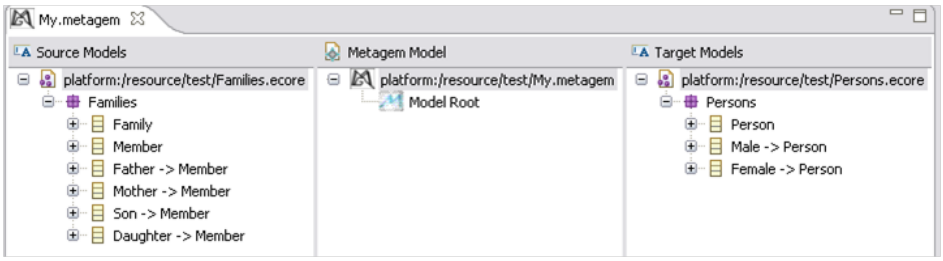

Figure 2: Empty `Metagem` model

Next, the different relationships between source and target metamodels have to be explicitly defined in the model. To that end, right-click on the root object of the (`Metagem`) model (`Model root`) `-> New child ->` Select the type of relationship to create (Figure 3). This time, we create a `One-To-One` relationship.

Once the relationship object has been added, the source and target elements of such relationship has to be set. To that end, drag elements from source and target models and drop them on the transformation model. Note that the editor automatically limits the number of elements that can be referenced by a given relationship depending on the nature of the relationship. For instance, after

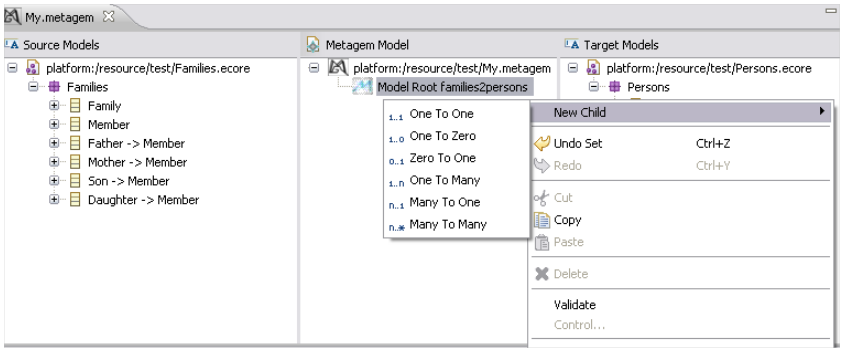

Figure 3: Adding relationship objects to `Metagem` model

setting a source and target element in a One-To-One relationship, drag & drop functionality over such relationship object is automatically disabled.

Back to the example, Figure 4 shows that the source element has already been set (`Families!Father`) whereas `Persons!Person` is been set as the target object for the `One-To-One` relationship previously added.

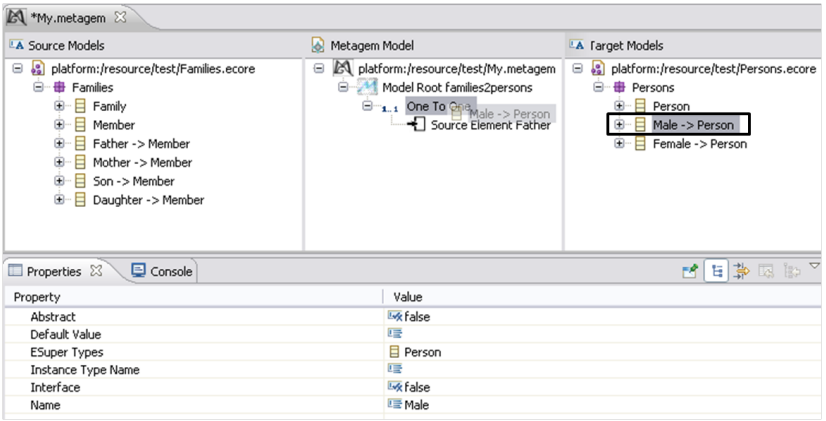

Figure 4: Setting the target element of a One-To-One relationship

Likewise, nested relationships can be defined in order to relate the properties of source and target classes. For instance, Figure 5 shows that the `One-To-One` relationship between `Families!Male` and `Persons!Person` contains another `One-To-One` relationship that binds their `fistName` and `fullName` properties

Proceeding this way, the final `Families2Persons Metagem` model is shown in Figure 6.

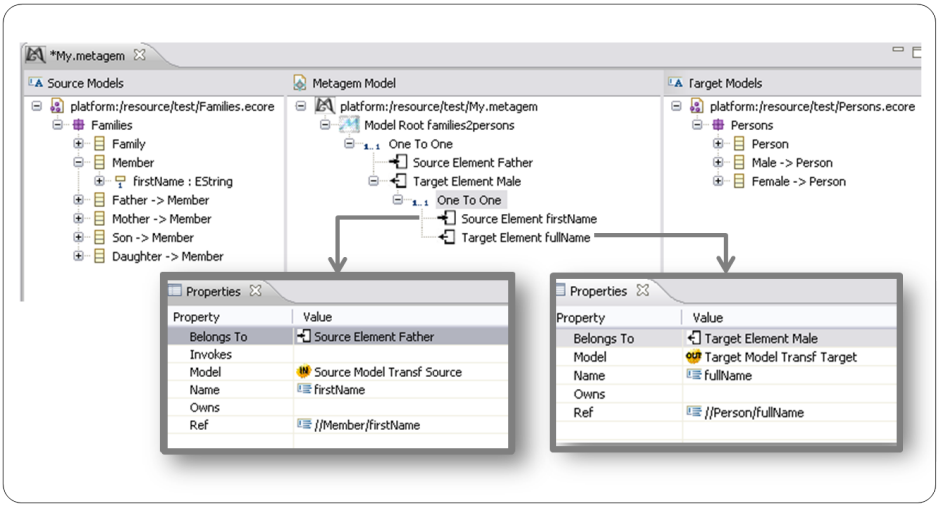

Figure 5: Adding nested relationships to a `Metagem` model

3. Producing the Hybrid transformation model

Next step of the process is to map the high-level specification of the transformation (`Metagem` transformation model) into a transformation model for the Hybrid approach:

- Right-click on the `Metagem` model in the project explorer.
- Run As -> MeTAGeM->Hybrid. See Figure 7(a).
- Confirm or modify the suggested name and location for the Hybrid transformation model that will be produced. See Figure 7(b).

Note that implicit model-checking is done before the `MeTAGeM2Hybrid` transformation is run. As a result, the Hybrid transformation model of Figure 8 is obtained. Every relationship defined in the Metagem model has given rise to, at least, a (Mapping) `Rule` object and a `Trace Rule` object in the Hybrid model.

Nevertheless, some refinement is frequently needed in order to add imperative constructions to the transformation (recall that we target the Hybrid approach). This use to take the form of auxiliary functions used to return some value after perform some computation. For instance, shows the creation of the `getFatherName()` operation for `Family!Father` objects.

- Right-click on the `root` element.
- New Child -> Operation. See Figure 9(a).
- Set the `name`, the `return type` and the `context` (object type for which the operation will work) of the `operation`. See Figure 9(b).

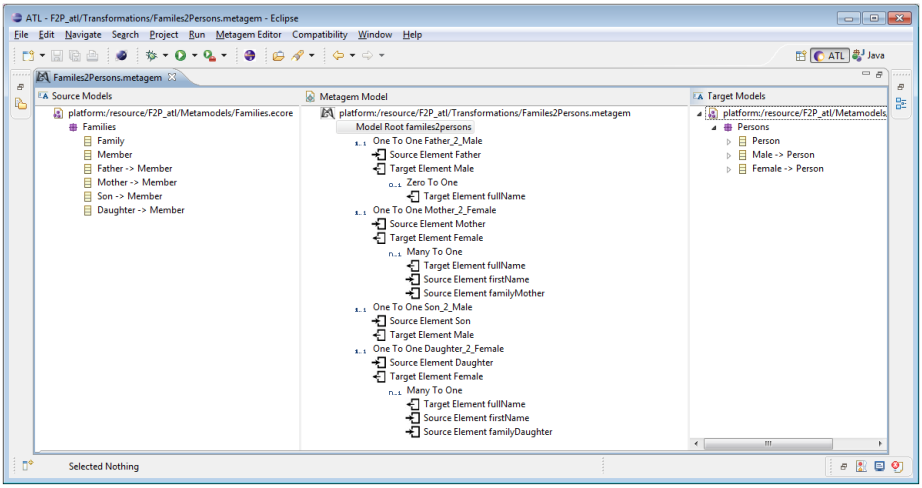

Figure 6: Complete `Families2Persons Metagem` model

- You might set the body of the operation as well using the properties view. See Fig. 9(c).
- Finally, if needed, arguments can also be defined. See Figure 9(d).

Some bindings might need also refining. If the binding is immediate, e.g. `a <- b` there is no need to worry. It could be set directly in the Metagem model. However, if the binding is not immediate, i.e. some computation is needed before assigning the value, it has to be manually modeled in the Hybrid transformation model (note that this refinement could be delayed and done over the ATL or ETL model that will be later generated).

Back to the example, the value of the `Persons!Person.firstName` property is returned by the `getFatherName()` operation previously defined. Therefore, it has to be explicitly modeled. This is shown in Figure10

- The `Father_2_Male` contains a binding object to set the value of the `Person.firstName` property.
- Hence, the `LeftPattern` of such binding points to the `Person.firstName` property.
- What rests is to set the `RightPattern` of the binding to point to the `getFatherName()` operation.
- This way, once the transformation is run, the value returned by the operation will be assigned to the property.

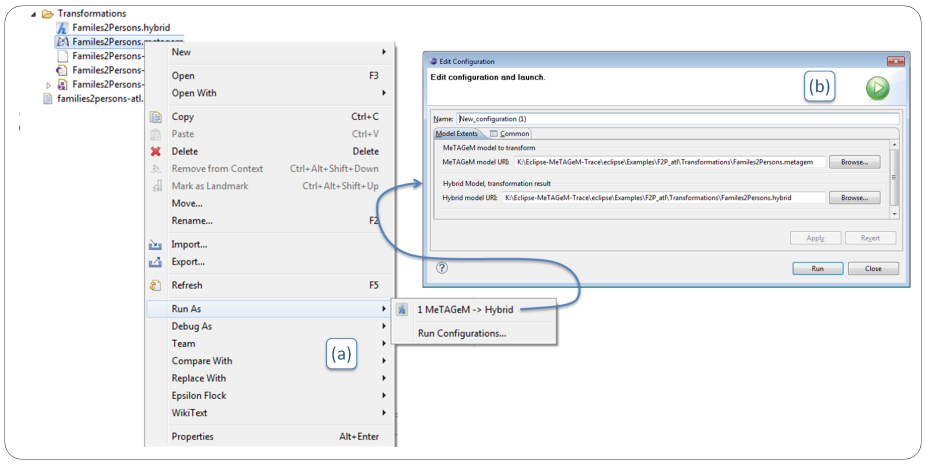

Figure 7: Running the `MeTAGeM2Hybrid` transformation

Another type of binding that needs refinement is the one that entails two or more source elements. Note that a binding is always related to just one target element since it is defined to set the value for such object. However, several source objects might be needed to compute such value. For instance, let suppose that we are not using an operation to get the value for the `Persons!Person.fullName` property.

- Such value is obtained from concatenating the value of the `Families!Member.firstName` property and the `Families!Member.lastName` property.
- Therefore, the binding object that sets the value for the `Persons!Person.fullName` property has to reference the two mentioned source properties.
- The `belongsTo` reference of the objects representing such properties in the transformation model has to be manually set to the corresponding objects.
- The result is shown in Figure 11

It is worth noting that the `belongsTo` reference is automatically resolved when the binding references to just one source object, .

As shown before, the Hybrid transformation model considers the modeling of two types of traceability relationships. `TraceRule` objects represent traces that will be derived from the execution of a mapping rule (a kind of top-level trace) whereas `TraceBinding` objects represent the traces that will be generated by the execution of a binding. These objects are automatically generated from the `Metagem` transformation model. However, the user may be interested in defining new traceability relationships. To that end, you should proceed as follows.

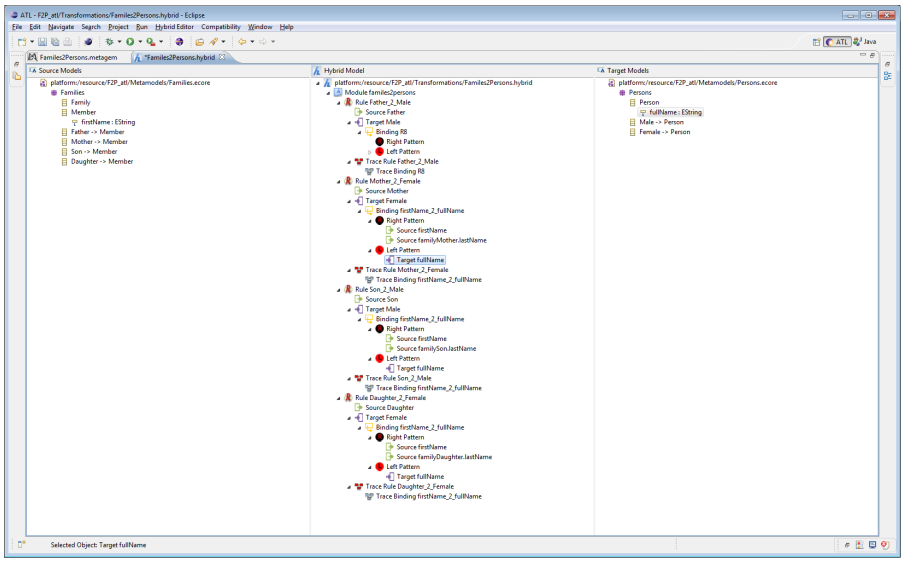

Figure 8: `Families2Persons` Hybrid transformation model

- First of all, a new Rule object has to be added since the existing ones were produced by the `Metagem2Hybrid` transformation. So they already contain a TraceRule object and the tool does not allow to create more TraceRule objects inside them. Note that this is the right behavior since traces can be n-ary.
- Right-click on the `root element -> New Child -> Rule`. See Figure 12(a).
- Set the source and target elements of the rule (e.g. `Families!Member` and `Persons!Person`). See Figure 12(b).
- Right-click on the `root element -> New Child -> TraceRule`. See Figure 12(c).
- Optionally, set the name of the `TraceRule` object. Note that the tool set automatically its source and target elements by looking at the containing `Rule`. See Figure 12(e).

`TraceBinding` objects are created similarly. The main difference is that the user must specify to which binding does the `TraceBinding` belong to since a rule usually contains several bindings.

4. Producing the ATL/ETL transformation model

Next step of the MDD development of model transformations supported by `MeTAGeM-Trace` consists of mapping the Hybrid transformation model into an

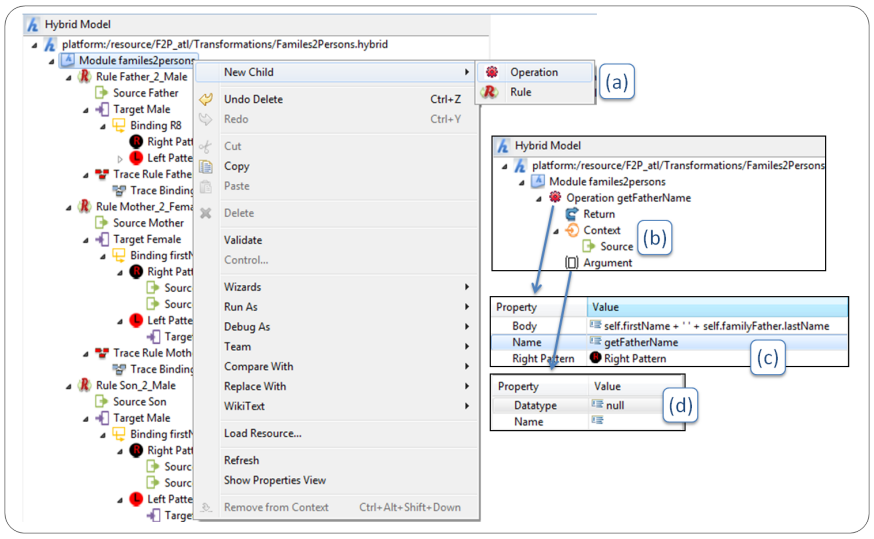

Figure 9: Adding `getFatherName()` operation to the Hybrid transformation model

ATL or ETL transformation model, depending on the targeted language. To that end, `MeTAGeM-Trace` bundles a pair of launchers

- Right-click on the Hybrid transformation model and select the targeted language. See 13.
- Confirm or modify the suggested name and location for the ATL/ETL transformation model that will be produced.
- The corresponding ATL or ETL transformation model is automatically obtained. See 13 a) and b) respectively.
- As it happened with the `Metagem2Hibrid` transformation, model checking is implicitly invoked before the selected transformation (either `Hybrid2ATL` or `Hybrid2ETL`) is executed.

5. Generating the source-code

Last step of the process is to obtain the source-code that implements the modeled transformation. To that end, `MeTAGeM-Trace` we should proceed as with the previous automated steps:

- Right-click on the ATL/ETL transformation model.
- Select the targeted language, either ATL or ETL (see Figure 14):
  - `Extract ATL model to ATL file (MeTAGeM-Trace).`

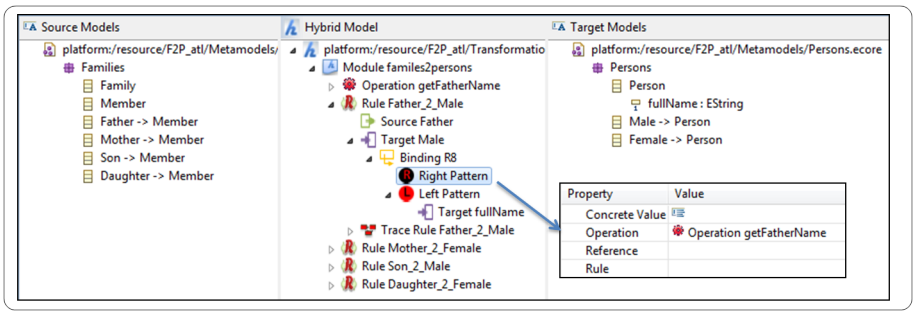

Figure 10: Setting the `RightPattern` of a binding to the value returned by the `getFatherName()` operation

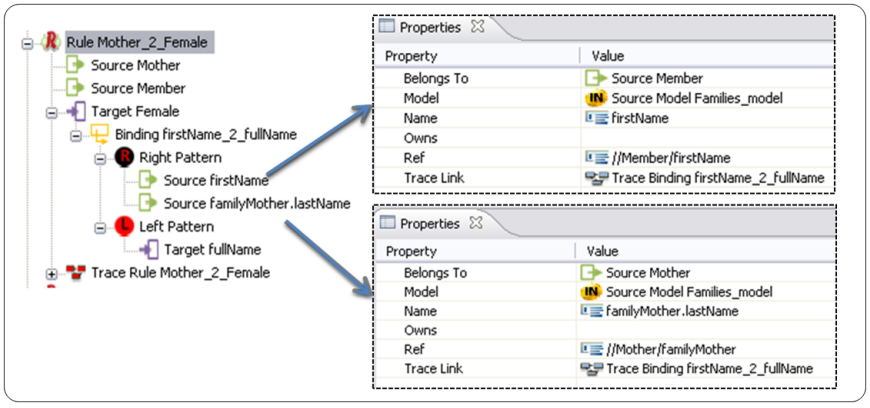

Figure 11: Refining multiple source bindings: setting the value of the `belongsTo` reference

– `Extract ETL model to ETL file (MeTAGeM-Trace).`

Note that the source-code generated will probably need from manual refinement. As mentioned before, code for model navigation is usually needed, as well as some re-naming of variables. Nevertheless, we have empirically assessed that more than 80% of the source-code of the transformation is generated by `MeTAGeM-Trace`, improving drastically development times. In this sense, it is wirth noting that `MeTAGeM-Trace` is oriented to help model transformation developers, i.e. it is not a tool intended for novice developers although we have discovered that it contributes to reduce the learning curve of model transformation development.

6. Generating trace models

Recall that apart from providing a tool to develop model transformations, the main advantage of `MeTAGeM-Trace` is that produced transformations are be able to generate, not only the corresponding target models, but also a trace

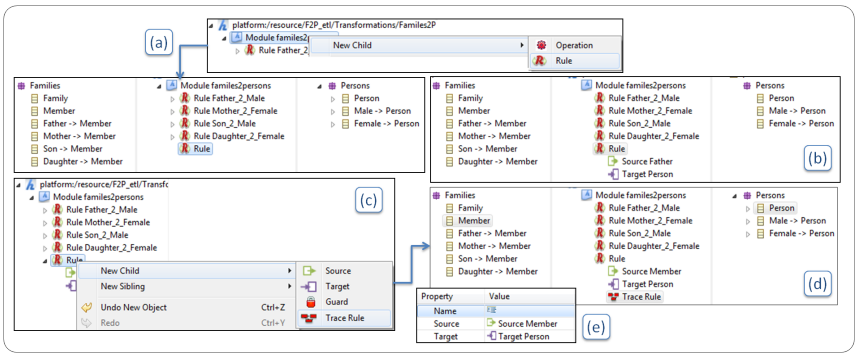

Figure 12: Creating a `TraceRule` object

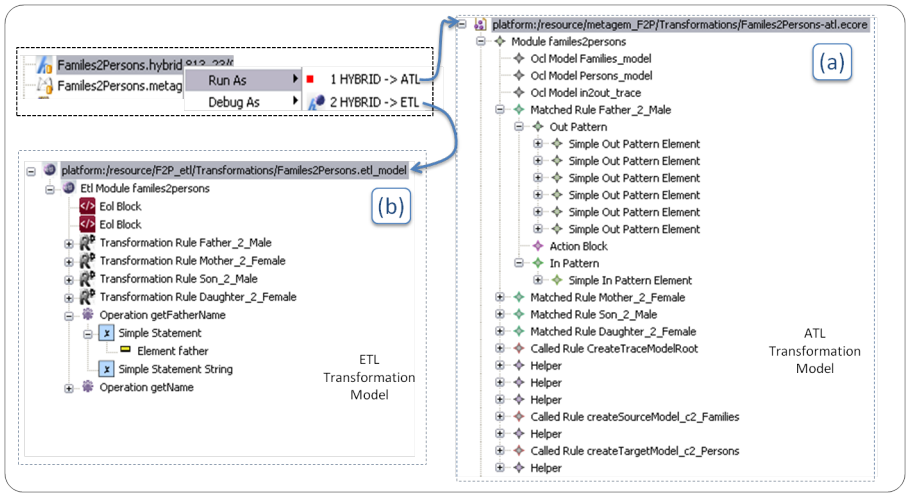

Figure 13: Running the (a)`Hybrid2ATL` and (b)`Hybrid2ETL` transformations

model. Therefore, the last section of this manual is devoted to show such functionality in action by running the `Families2Persons` transformation.

Note that running the transformation is an issue related with the corresponding targeted language. You can look the corresponding manual for more information on how to run ATL and ETL transformations. The only difference is that the transformation will produce an "extra" model in this case .

- Right-click over the source file of the transformation (either ATL or ETL, though we will focus on the ATL one here).
- Run-as -> Run configurations
- Select the source, target and trace models metamodels. See Figure 15: note that the Traces metamodel is already available in the EMF registry.

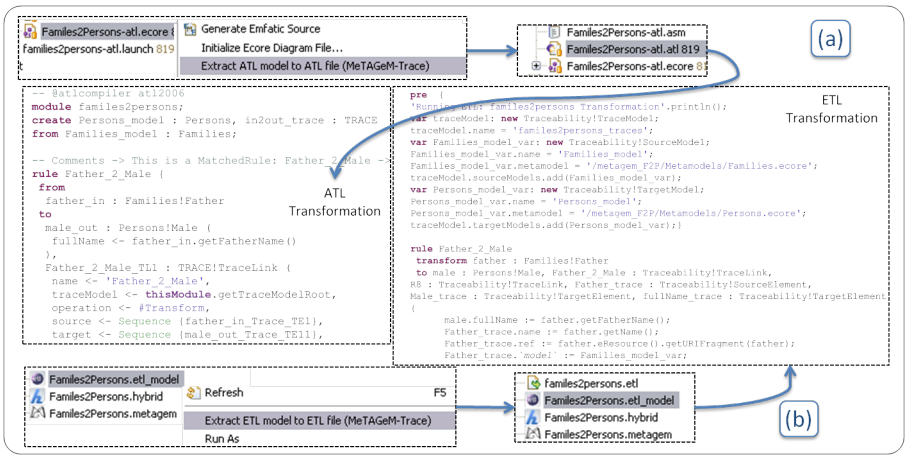

Figure 14: Generating ATL(a) or ETL(b) source-code

- Run the transformation using the above-described configuration.

As a result, apart from the target model, a trace model like the one shown in 16 is generated. If you open the model with the multi-panel editor bundled in MeTAGeM-Trace you will notice that:

- When a trace-link object is selected in the trace model, corresponding source and target objects are automatically highlighted in the left and right panel. For instance, the picture shows that the `Mother_2_Female` trace link has been selected. Consequently, the source and target objects referenced by such trace (`Marge` and `MargeSimpson`) are automatically highlighted. automatically highlighted in the corresponding models.

- Likewise, if a source or target object is selected in the left or right panel, the trace links referencing it are automatically highlighted in the central panel.

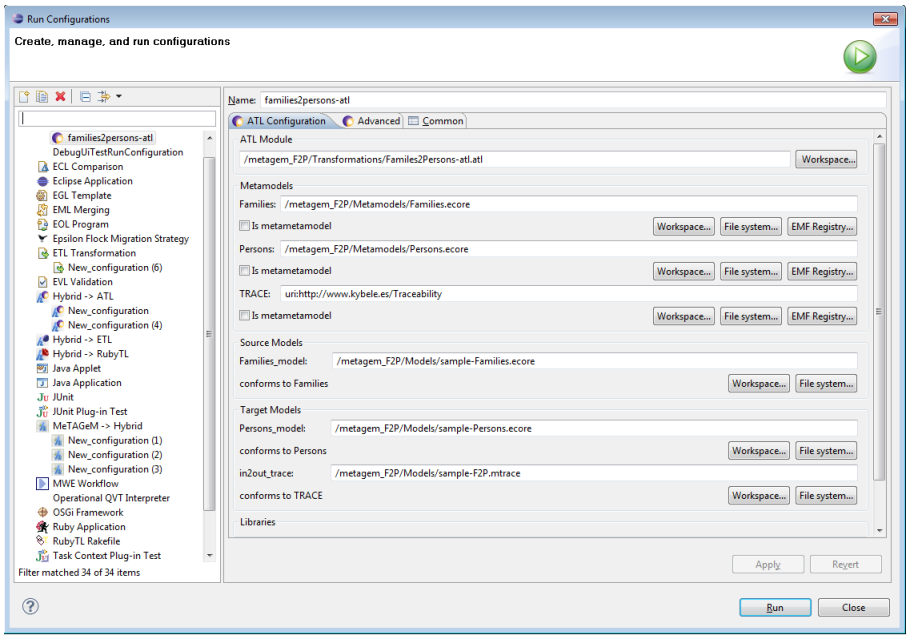

Figure 15: Running the ATL transformation

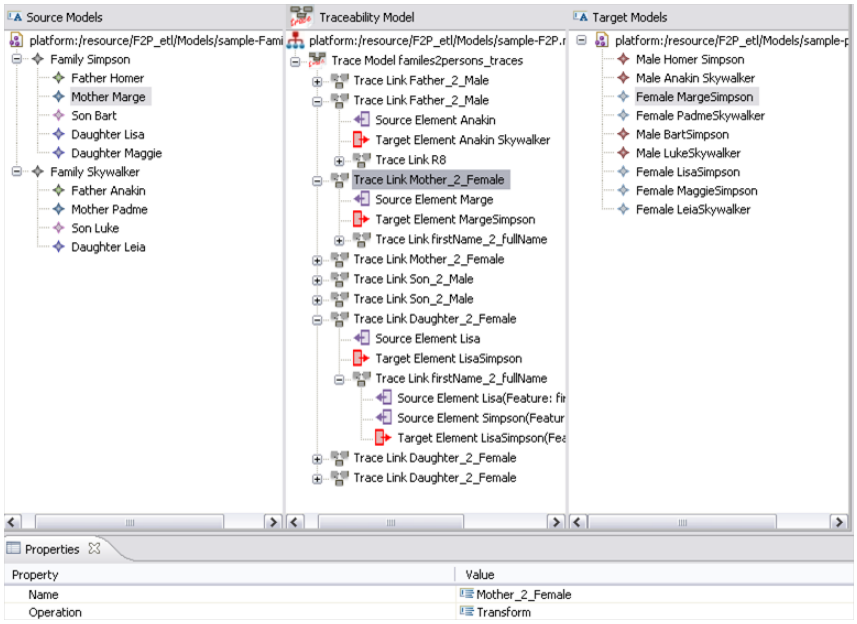

Figure 16: `Families2Persons` trace model