ATL 3.1 – Industrialization improvements

William Piers[1],

[1] Obeo, 7 boulevard Ampère, 44481 Carquefou, France
william.piers@obeo.fr

Abstract. ATL is a standard solution for model to model transformation. Its implementation is OpenSource and hosted by Eclipse. In 2008, development and maintenance were handed over from INRIA to OBEO, in the ATL Industrialization project context. This paper is about how industrial actors and researchers collaborate in a project like ATL, what ATL 3.1 brings to the user and what will be done in the future.

Keywords: Model transformation, Eclipse, M2M, ATL, industrialization.

1 Introduction

ATL [1] is a standard solution for model to model transformation. Its implementation is OpenSource and hosted by Eclipse, in which it is recognized as the standard model to model transformation solution. In 2008, development and maintenance were handed over from INRIA [2] to OBEO [3], in the ATL Industrialization project context. Now Regular releases are scheduled, and user-oriented features have been added to the ATL tools. The ATL project is now considered as an industrial solution for model transformation.

This paper is organized as follows. Section 2 presents the ATL Industrialization project. Section 3 describes the ATL 3.1 release features. Finally the section 4 will show the planned improvements for the next releases.

2 ATL Industrialization

ATL industrialization is the result of the collaboration between a laboratory (AtlanMod [4]) and an SME (Obeo). The main scheme of such kind of collaboration, as seen by public institutions, is to ask big industrial companies for their R&D needs, then let researchers prototype them and finally build a viable commercial offer around a product which meets the need. In the case of ATL, this work took place through collaborative projects.

Obeo is involved in OpenSource since its creation, especially into the Eclipse foundation (14 Eclipse commiters are working at Obeo). Obeo is an Eclipse strategic member. ATL is recognized as a standard solution in Eclipse. From that, Obeo became in charge of improving the legacy ATL software and build a service offer around ATL (support, expertise, training). The main purpose was to make AtlanMod more focused on research themes when a company like Obeo was better qualified to enhance user-oriented features and manage maintenance.

The ATL industrialization project consists on several development axis: ergonomic, scalability, interoperability (compatibility with other MDE tools) and integration of new concepts (traceability, iterative transformations). Since Obeo has been in charge of the Industrialization project, ATL 2 and ATL 3 were released, bringing a lot of improvements: core API refactoring, performances improvements, user interface features. Obeo also provides a professional support for companies. The industrialization process also allows researchers to develop prototypes based on ATL. The objective is to finally integrate those prototypes as contributions.

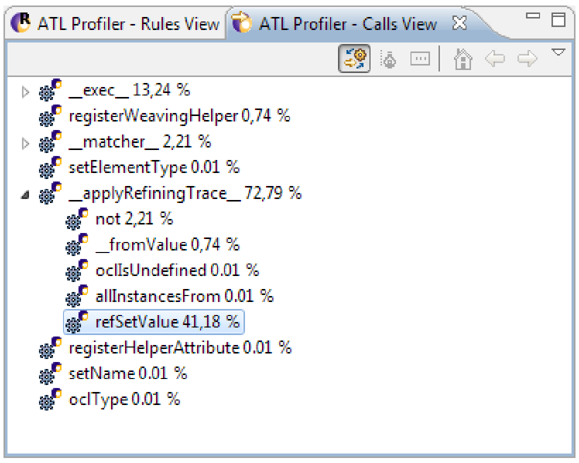
3 ATL 3.1 features

In June 2010, ATL 3.1 will be released as part of the Eclipse Helios simultaneous release. This release brings a lot of improvements, both at the user interface level and core level.

3.1 Debugging and Profiling

A lot of work has been done around the legacy debugger. Initially the ATL debugger was dedicated to the old ATL virtual machine (the core of the ATL execution process) and strongly relied on it. A refactoring was necessary, which consisted in making some parts more abstract in order to allow debugging on the newest ATL virtual machine as well as for the old one. This work achieved, it also made possible the integration of an ATL profiler, extending the same architecture as the debugger.

The ATL profiler provides a way to detect performance issues on a given transformation by a detailed analysis of the execution. An execution profile stores informations about the global cost of one method, the memory used, etc... A set of views display profiles as tables and trees, with sorting and filtering options. Execution profiles can be saved as models (.xmi files): this could be useful for a non-regression purpose for example.



```
▷ __exec__ 13,24 %
  registerWeavingHelper 0,74 %
▷ __matcher__ 2,21 %
  setElementType 0.01 %
◢ __applyRefiningTrace__ 72,79 %
    not 2,21 %
    __fromValue 0,74 %
    oclIsUndefined 0.01 %
    allInstancesFrom 0.01 %
    refSetValue 41,18 %
  registerHelperAttribute 0.01 %
  setName 0.01 %
  oclType 0.01 %
```

Fig. 1. A profiling result sample. It shows that most of the time is spent in setting values on meta-elements.

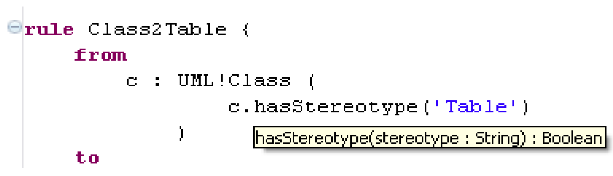
3.2 ATL file editor

Regarding the user interface, most of the improvements have been done in the ATL file editor. An advanced type-inference engine has been implemented and integrated into the existing editor to strongly extend the existing completion proposals. Now the proposals cover complex code structures, like OCL expressions, whereas previously limited to the left side of the bindings and model elements.

The types, variables, contexts computations have been added to the existing completion system which was based on an analysis of partial parsing results. The type-inference engine extends that system by visiting the parsing result and computing types for each step.

Additionally, some content assist templates have been added to the completion system, such as rules, helpers, operation calls templates.

The type engine has been reused to provide additional features, like hover information, navigation through elements declarations. Those features reuse the same type computing system as the completion processor.

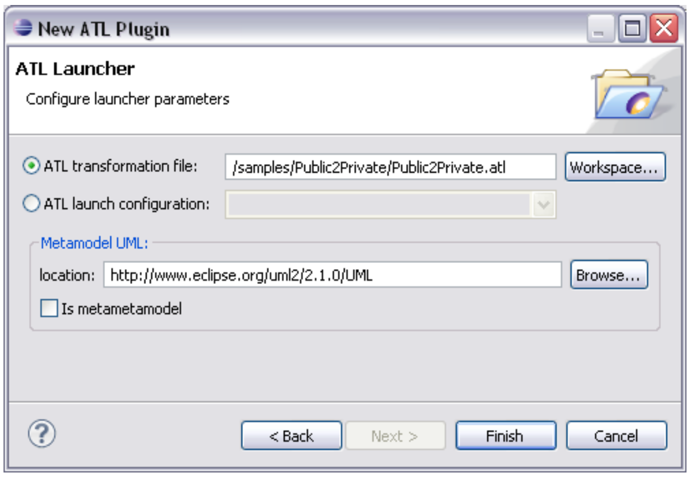


```
rule Class2Table {
    from
        c : UML!Class (
                c.hasStereotype('Table')
            )       hasStereotype(stereotype : String) : Boolean
    to
```

Fig. 2. Hover information: when the mouse stays over a variable, an attribute or in this case an helper call, it displays the type information. Here, the helper signature.

3.3 ATL plugins

ATL usability has been extended by the addition of a new wizard. This provides a way to embed an existing transformation inside of an Eclipse plugin, with an associated launching class written in Java. After that operation the ATL builder keeps synchronization between the ATL file and the Java launching class. This feature eases the integration of an ATL transformation into an application.



New ATL Plugin

ATL Launcher

Configure launcher parameters

(●) ATL transformation file: /samples/Public2Private/Public2Private.atl [Workspace...]

( ) ATL launch configuration:

Metamodel UML:

location: http://www.eclipse.org/uml2/2.1.0/UML [Browse...]

☐ Is metametamodel

< Back | Next > | Finish | Cancel

Fig. 3. The ATL plugin wizard page. The required in formations can be initialized from an ATL transformation file as from an ATL launch configuration.

4 Planned releases

ATL industrialization is a continuous project, so each future release is planned and brings bug corrections, new features and improvements. For the next releases, some objectives have been set but the roadmap is open to contributions (research, partners). At this time, for the ATL 3.2 release (scheduled June 2011), Obeo plans to work on easing and speeding up transformation development.

Graphical tools will be added on the top of existing tools to allow the graphical definition of transformation rules and the synthesis view of the matching currently achieved by a transformation. The ATL language will also be reexamined to define syntactic improvements. Finally, depending of the research teams, advanced contributions could be integrated into the main ATL code.

5 Conclusion

This paper presented current works on ATL. Many of them are provided by Obeo in the ATL industrialization context, but the door is still wide open to integrate research contributions. In the future Obeo will keep making improvements on ATL and maintenance.

The successful collaboration between Obeo, INRIA and industrial partners shows that OpenSource is an efficient and valuable way to promote an innovative technology.

References

1. ATL web site, http://www.eclipse.org/m2m/atl/
2. INRIA web site, http://www.inria.fr/
3. Obeo web site, http://www.obeo.fr/index.php?lang=en
4. AtlanMod web site, http://www.emn.fr/z-info/atlanmod/index.php/Main_Page