The *QVT* standard is the most accepted solution for model transformations, but there is a patent lack of consensus about its implementation. It is a vast and complex standard that, although being a few years old, has plenty of inconsistencies and issues to be solved before making it a solid reality. The knowledge obtained through hours of investigation over the standard by the different vendors is barely shared, while the standard still remains unclear in several aspects of the specification in terms of implementation. This is directly related to the absence of a reference implementation, and it makes implementing the standard anything far from simple or straightforward. Having different programming paradigms within the same standard, or the direct dependency with another complex language like *OCL* are just some of the difficulties that all vendors have to face when dealing with *QVT* implementation.

EMF seems to be an almost unanimous choice related to metamodeling frameworks. Apart from that, each vendor seem to tackle the execution in their very own way. There are currently different solutions, ending up in different execution engines, having at one's disposal code interpreters, compilers that generate the equivalent *java* transformation code, or a virtual machine approach, where the source language is implemented via a low level transformation language.

A language-specific execution engine may have some drawbacks. The extensibility of this approach is questionable and relies strongly on the design and flexibility of the architecture. Possible drastic changes of the standard in the future (a fact not that unfeasible for a still young field as the one concerning this proposal) would make harder the continuity of certain projects. Furthermore, the lack of reuse would difficult the maintenance of those solutions. Future emerging languages would potentially be out of the scope of those execution engines, evidencing the fragility of the approach.

In our opinion, the solution should focus on the fact that the standard might evolve and new languages could emerge. Flexibility and reuse become requirements that any transformation engine should fulfill, and they seem to be an intrinsic part of virtual machine-based approaches. Certain solutions, as *The Atlas Transformation Language Virtual Machine* (*ATL-VM*) by *The Atlas Group*, or *Atomic Transformation Code* (*ATC*) by *Open Canarias*, have shown to be capable of facing efficiently implementation of multiple high level transformation languages through an intermediate-layer approach. The advantages of electing such intermediate infrastructure are discussed in the "*Making a case for supporting a byte-code model transformation approach*" [1] submission.

Using this approach, *Open Canarias* has reduced *QVT* execution [2] to a few repeatable steps. Starting from an instance of the concrete syntax of the language, a parser generates a model of its abstract syntax. This model is used as input for an *ATC* transformation, capable of generating an executable *ATC* model which contains all the semantics of the original transformation. The resulting model is suitable to be executed under the *ATC* transformation engine.

The adoption of a virtual machine-based transformation language is being very fruitful at *Open Canarias*, allowing us to face the implementation and finally give support to *QVT*-related languages such as *OM*, *Core* and *OCL* reusing much of the same infrastructure. Moreover, our approach has shown the capability to tackle with minimal effort changes on *QVT* languages and the power granted to implement future transformation languages.

[1] *http://www.modelset.es/publications/2008/ProposingVMStandard.pdf*
[2] *http://www.modelset.es/downloads/QVTExecution.jpg*