On Developing Open Source MDE Tools: our Eclipse Stories and Lessons Learned

Hugo Bruneliere and Jordi Cabot

AtlanMod Team (Inria, Mines Nantes, LINA)
Ecole des Mines de Nantes, 4 rue Alfred Kastler, 44307 Nantes, France
{hugo.bruneliere,jordi.cabot}inria.fr
http://www.emn.fr/z-info/atlanmod

Abstract. *Tool development has always been a fundamental activity of Software Engineering. Nowadays, open source is changing the way this is done in many organizations. Traditional ways of doing things are progressively enhanced or even sometimes replaced by new organizational schemes, benefiting as much as possible from the properties of open source (OS). This is especially true in innovative areas such as Model Driven Engineering (MDE) in which new tools are constantly created, developed and disseminated, many of them coming from research teams. This poses some hard questions: What is the actual impact of OS in terms of tool development? How to best take advantage of OS communities? And what are the opportunities for research teams in this context? Capitalizing on experiences in developing MDE OS tools on top of the Eclipse platform and its license model, we try to give some insights on these questions in this paper.*

Keywords: MDE, Tool, Development, Industrialization, Open Source, Eclipse, Lessons Learned

1 Introduction

The world is full of open source prototypes or tools that are globally ignored, even sometimes by their own creator(s) that do not have the time/resources or simply the wish to continue supporting them. They may have been developed as proofs-of-concepts to experiment on innovative ideas (and publish the corresponding research papers) or as formal deliverables within collaborative projects, and forgotten not so long after. This finding is particularly true in the context of relatively new or emerging fields, such as Model Driven Engineering (MDE).

We think we can all agree this is an unfortunate situation. The benefits of transferring such pieces of technology from innovation labs to the real (software) industry being numerous and well-known [4], it may seem strange that many of the available open source (MDE) tools are not more largely integrated and used by practitioners. The main argument generally mentioned is their poor maturity level according to the industry criteria. In fact, except for a few pure "innovators" [14], technical innovation is only one of the factors that influence

company decisions regarding the adoption of new solutions. Many other aspects such as usability, robustness, performance, documentation, long-term support, pricing or general policies are usually regarded as (much) more important.

One of the main reasons for this lack of maturity in many cases is that creators too often attempt to simply release their prototype/tool as open source software hoping the "community" will jump in and naturally take care of carrying out the main development, maintenance and upgrade tasks. In practice, doing this alone appears to be most of the time insufficient. Many open source projects quickly get out of funding or finally fail to attract any real interest from community members, developers or potential users [10]. Therefore, we believe more elaborated and structured development and industrialization processes should be strongly encouraged.

In this paper we present our past and present experiences (more or less successful depending on the cases) in initiating and developing/promoting open source MDE tools, the main issues we have faced (until today), as well as some of the interesting lessons learned in the process. Note that most of our experience concerns the well-known Eclipse open source community and its Eclipse Public License (EPL) [18].

The paper is structured as followed. Section 2 first introduces our "free" experiments, i.e. prototypes that have been developed based on individual initiatives in the team without any particular external support, and how they have evolved with time. Then section 3 presents another set of prototypes, developed in a more structured and "funded" context (i.e. industrial and/or research collaborative projects), and the impact it has on them. Section 4 reports on our main success stories and how the corresponding tools become sustainable thanks to the "industrialization triangle" we have been able to set up in those cases. Finally Section 5 summarizes the main lessons learned notably regarding the use of open source (and related communities such as Eclipse) as a good medium (if used correctly) to develop and industrialize such MDE prototypes/tools, before Section 6 concludes the paper.

2 The Free Way: Developing tools on our own

Within the context of our research team, some model-based prototypes/tools have been initiated and developed mostly on an individual basis, meaning that they were mainly the outcome of personal initiatives not directly supported by any particular funded project or collaboration (either industrial- or research-based). For these tools, results have been very contrasted. While one of these tools ends up being later on really successful (but mostly because we changed its development model, see the case of ATL in Section 4), others had a much more limited impact or have been simply "given" to the community some months after their launching. The next paragraphs introduce examples that did manage to raise some real interest but never actually made it as widely used tools or commercial solutions.

EMF2CSP [8] is a tool providing support for the automated verification of EMF models (i.e. UML or Ecore ones), notably checking several correctness properties such as satisfiability or OCL constraints validity. This tool is the continuation of *UML2CSP* and has been developed in the context of a PhD thesis. In both cases, the design and developments were realized outside of any particular project or collaboration. While both tools are highly cited in the research community on model verification, the tool and approach themselves have raised a relatively low interest at the industrial level.

Another example is the EMF-Rest tool [1] that has been developed within the context of a Master thesis. It provides basic support for generating RESTful APIs from EMF models, allowing these models to be used on the Web via JSON objects. Started as a simple prototype, the first obtained results were very promising in terms of new capabilities and potential applications. Also, we received encouraging feedback from the industry while introducing the tool at an Eclipse community event. However the creator of the tool left our team and, without other available resources at that time, we decided to ask the community for help in developing if further. This tool is currently still available as open source, including its full source code, documentation, examples, etc. Unfortunately, nobody has really strongly committed so far to continue its development.

Based on these examples (and others we cannot detail here due to space limitations, e.g. the Collaboro tool [6] for the collaborative development of DSLs), the main benefit of this way of developing tools is obviously the complete liberty that you have to explore different problems, experiment with several conceptual and technical solutions, etc. Not directly depending on any particular funded project or collaboration gives a lot of autonomy in terms of architectural decisions, technological choices and of course (research) directions to be taken. However, this freedom comes with a usually high price. The most important limitation relates to the available resources and the degree of commitment you can expect from the involved people. This is particularly true when the creators of such tools have temporal positions, and therefore frequently leave the team or just change topics. As another side effect, working somehow isolated can considerably reduce the potential visibility of the work as well as the relevance of the obtained results.

3 The Funded Way: Developing tools as part of collaborative projects

As most of research teams, our activity is largely funded by collaborative actions or projects with other academics and industrials. In this context, there is room for thinking on MDE innovations and creating corresponding model-based prototypes that meet (some of) the objectives or requirements of the projects. Once again, the results obtained with these tools have been very varied. While one of these tools is actually a main success for us (cf. the case of MoDisco in Section 4), others are still in development and have no guarantee of being sustainable

in a more or less near future. The next paragraphs present examples of such prototypes.

Neo4EMF [2] is a tool linking EMF with the Neo4j graph database management system. The main objective of this tool is to offer Neo4j-specific EMF code generation facilities that allow EMF models to be loaded, queried and stored in graph databases, benefiting from fast storage and on-demand loading/unloading capabilities. It has been started as part of a French collaborative project (*ITM Factory*) and is also developed in the context of a PhD thesis directly funded by an European collaborative project (*MONDO*). This work has already resulted in a scientific publication and been presented to Eclipse community events, receiving a generally positive feedback in both cases. The *MONDO* project is still going on today, but so far there is still no well-defined path/vision for the tool's future after the project ends.

A similar story can be told for the EMF Views tool [16]. EMF Views is a solution for building so-called model "views" by linking several (potentially large and heterogeneous) EMF models. Relying on a virtualization mechanism at both metamodel and model levels, such views can then be considered and handled (read-only) as regular EMF models by users or other EMF-based tools. The current tool is an extended version of the Virtual EMF prototype as originally created within the *CESAR* European project. The developments have then been continued as part of the *TEAP* French project, and are still going on today. However, in this particular case, the funding will end in the coming months and decisions will have to be taken regarding the next steps for this tool. A possible option is the reuse and extension of the tool in the context of another French project (*MoNoGe*) started more recently, which will postpone hard decisions on the tool's future for a couple of more years.

Compared to the "free" way presented earlier, the main benefit of the funded approach is the security in terms of allocated resources and time: funds are explicitly distributed and guaranteed at least during the whole duration of the project or collaboration. Another important advantage is the collaborative aspect, notably if involving one or several industrial partners. This allows ensuring more easily that the addressed problems or targeted applications correspond to real needs from the industry. However, some limitations can be sometimes encountered. As an example the environment can be quite constrained by the (industrial) partners' requirements, e.g. in terms of challenges to be tackled, technical basis to be used, legal or administrative issues. Moreover, exploitable results generally have to be proposed at the end, making the expectations (much) higher for the developed solutions. This can be somehow contradictory with the research activity which implies more uncertain results by nature.

4 The Sustainable Way: Developing tools in an industrialization triangle

Being a research team, it is very difficult (and not really our goal) to lead the process of making our tools industrial/commercial-ready (i.e. mature enough in terms of tests, documentation, interface, etc. to be used as is within companies).

Next section proposes a possible industrialization schema that could enable research teams end up with mature tools while focusing on the things they do best (research and innovation).

4.1 A General Industrialization (Business) Model

Our model can be represented by a virtuous triangle (see Figure 1) with the following actors as vertices:

- research labs (innovation providers),
- end-users / communities (e.g. big companies),
- SMEs (technology providers).

In this schema, the SME has a very important role to play as taking care of actually "industrializing" the prototype proposed by the research lab according to the needs from the end-users. Of course, there also exists models of direct collaboration between a research lab and a big company [13] but we do believe the introduction of such an SME in the process can bring additional benefits (e.g. allowing them to delegate some tasks). From the SME side, the main challenge resides in identifying and setting a valuable business model [12], notably considering open source as a key element [11]. However, both the research labs and the partner companies (i.e. the users) are also concerned and can actually support the SMEs in this (e.g. by participating actively to dissemination actions).

During all that process, the tool remains available in open source. In our experience but also others [3], it largely facilitates the communication and agreements between all partners (e.g. via open bug tracking systems, source repositories, emailing lists, etc.) while ensuring sufficient individual benefits for all of them. Moreover, the fact of working openly helps accelerating the creation and progressive growth of a relevant user base, even when the protoype/tool is still under active development.

The relationship between the actors in the triangle can be described as a six-step process that can be iterated over several times for a same tool, possibly involving different and/or new partners in the loop:

- **Problem description from the end-users.** End-users are providers of real problems to solve in their industrial context. In this sense, the industrialization triangle forces a problem-driven research approach where the user base must be large enough to represent a potential market sufficiently attractive for a SME.
- **Evaluation of research challenges.** The lab filters the list of problems, according to the state-of-the-art in the domain, and identifies those possibly implying relevant research questions.

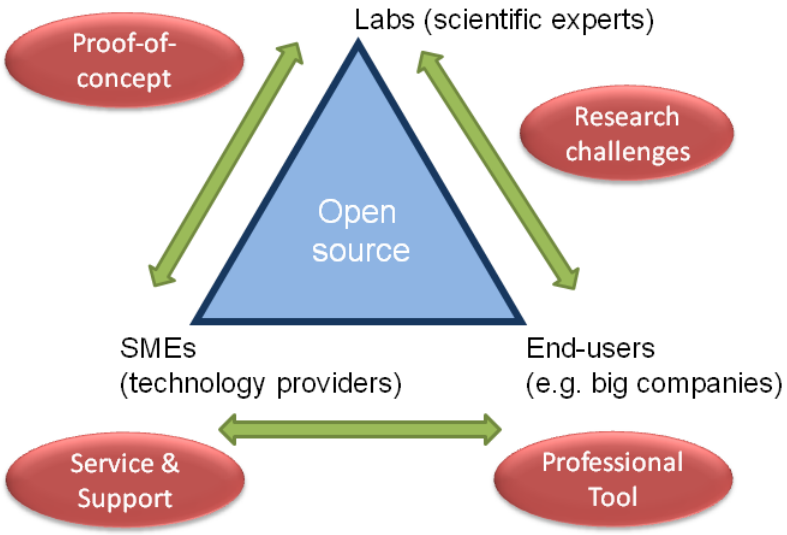

Fig. 1. A virtuous industrialization triangle

- **Research work.** The lab performs the activities required to solve the selected problems and builds proof-of-concept(s) validating the research results. At this point, obtained results are mature enough to be published but the prototype implementation is not yet ready to be used in an industrial environment.
- **Identification of a suitable SME.** Once the real potential of the tool is validated, the research lab looks for a technology provider. For instance, it can be a SME which has already collaborated with the research lab in the past (in the context of a funded research project or collaboration).
- **Industrialization of the research tool.** The lab and the SME collaborate to transform the prototype into a commercial tool. The SME takes over traditional software development and maintenance tasks, in exchange for gaining visibility in the community and proposing specialized services around the tool, while the lab provides its scientific expertise.
- **Official release of the industrialized version.** The final tool is made freely available to the community. Thus, all partners can extend the tool (e.g., the lab to implement some advanced features only relevant for its research context, the SME to provide a commercial adaptation to a specific customer) which will be adopted or not by the rest of the users.

4.2 Two Success Stories and Some Feedback

We have been able to successfully apply this triangle twice. We briefly describe these two experiences and, based on them, summarize the main advantages/drawbacks of this model.

The most successful tools of our team, not only in terms of research publications but also in terms of industrial applications and acceptance, are ATL (a tool dedicated to model transformation) [9] and MoDisco (a framework for implementing model driven reverse engineering solutions) [5]. We initiated these two tools as part of European research projects (respectively *MOTOR/CARROLL* and *MODELPLEX*), cf. other examples from Section 3. To be more accurate, the real genesis of ATL was even a bit before within the context of a Master thesis based on the exploratory work from a previous PhD thesis, cf. other examples from Section 2. Quickly after their inception, both tools were proposed to and integrated by the Eclipse Foundation as part of the Eclipse "Modeling" project, under the open source Eclipse Public License (EPL) (cf. next Section 5). This largely contributed to the development of significant user communities. Some big companies were already experimenting with the tools, but the lack of industrial support at the time was preventing them to deploy the developed solutions in their actual production environments. In each case a local technology/solution provider joined the project, respectively Obeo and Mia-Software (Sodifrance).

In the case of ATL, Obeo agreed to be in charge of the maintenance of the stable version and the general improvement of the tooling (e.g. by integrating some of the innovations we proposed). In the case of MoDisco, Mia-Software has been taking care of maintaining the existing integration framework and developing new high-quality components for improving the overall solution. In exchange, both companies have complemented their product suites, as well as extended their service offer by selling dedicated support and training when needed. Moreover they have also gained visibility, hence creating new opportunities for business and further participation in collaborative projects. The two tools progressively grew up reaching a sufficient maturity level to officially become parts of the yearly Eclipse Simultaneous Releases. In parallel, active user communities have been developed and effectively contributed to a larger dissemination of the projects and their results.

Compared to the two previously described ways of developing tools, this sustainable triangle offers many advantages. Among them, we can notably cite the fact of working on real industrial challenges to be solved, having an easier access to real testing scenarios (from end users), benefiting from highly qualified professional and technical support, gaining more visibility from the community (dissemination in open source), etc. However a limitation of this approach is its difficulty to be set, and more particularly the complexity to find the right technology provider (e.g. an innovative SME). Also, such an approach requires a clean structuring and concrete applicable results which are not always easy to obtain in a research environment.

5 Some (other) lessons learnt

There are many factors to have in mind when selecting a proper development strategy promoting a Software prototype to a tool. For instance, from a research perspective, the culture of entrepreneurship of the host organization [15] or other

parameters such as the internal policy/management and external ecosystem [7] are some general aspects to be considered. Beyond choosing the right development strategy for you (among the main models presented above and possible combinations of them), there are several elements that we believe research teams should pay particularly attention to when developing open source tools expected to become widely adopted as industrial/commercial solutions at some point:

- **Using the right open source license**. Open source is a common factor between the three strategies for tool development we have presented. Independently from the chosen (business) model, we believe relying on an open source license is a must as it simplifies a lot all common actions between the partners, notably concerning legal aspects (e.g. intellectual property) or results exploitation and dissemination (especially for the research team). However, all open source licenses and related communities do not offer the same opportunities. When selecting an open source license, we recommend choosing one that allows commercial adaptations and redistributions. Thus, in our cases, the use of the Eclipse Public License (EPL) [18] creates a win-win situation for both academics and industrials.
- **Integrating a widely recognized community**. Open source is not enough per se to attract new users as many open source projects are half-dead, of relatively poor quality or not really adopted. Instead, being part of a lively ecosystem and having an official project in a recognized community (e.g. Eclipse) gives a lot more visibility. It helps attract both collaborators (that would like to see their name linked to that community) and users (that will perceive it as a guarantee of high-quality). We recommend identifying such a community in its domain to really benefit from the visibility and interactions with its members. However this is a bidirectional effort: the research team itself also needs to invest on the community (e.g. in our case we attend the Eclipse conferences or use other Eclipse projects beyond the ones we develop).
- **Following structured development processes**. Building a real tool requires a well-defined development process (milestones, bug tracking, version control, tests, coding guidelines, etc.). We were not following all these best practices at the beginning, but the growing complexity and increasing number of users encouraged us to adopt them. Being part of a community with its open procedures (e.g. the Eclipse Development Process [17]) really helps in this matter. In addition, planned release cycles (e.g. yearly Eclipse Simultaneous Releases) force the project teams to release and update tool versions (and related documentation) on a regular basis. But deploying such a process can be quite heavy and so cannot be supported by a research group alone (e.g. which may not have the required experience), making the partnership with a company even more important.
- **Relying on a reference framework**. For a tool to be stable and reliable, it must be built on solid ground. Thus reusing/extending an already well-established and recognized base framework is a guarantee of a certain quality level, and also helps targeting a more widespread audience. In the context

of our tools, we naturally decided to use EMF as the reference open source modeling framework in the Eclipse community. However, while capitalizing on the numerous benefits brought by such a reference framework, we also inherit from its drawbacks that should not be underestimated. For instance, EMF has some scalability issues when dealing with very big models and these are interesting open spaces for research and experiments. As an answer to this, we do collaborate with one of our partner SMEs (namely Mia-Software) to propose solutions improving the current situation (cf. Neo4EMF as introduced in section 3).

– **Being supported and appreciated by the research team's host lab for the transfer/industrialization effort** It is necessary to benefit from dedicated resources/structures, offered by the hosting institution, in order to help the research team during such an industrialization process. Indeed, this process requires significant (and non-core research) additional effort and knowledge in legal, financial, logistic or commercial aspects, which are not competences always found in research. As an example in our case, Inria and Mines Nantes are providing support to their different research teams via dedicated "Innovation / Technology Transfer" entities. However, despite of this, the required extra effort is not always rewarded at its real value by the hosting research institution, highlighting the more global problem of current evaluation criteria in research organizations. Consequently (mostly due to resource limitation) many research teams voluntarily ignore this part of what should also be part of their normal working activity.

6 Conclusion

There is no silver bullet when it comes to create, develop and promote successful MDE tools. All along the paper, we have presented three different (and sometimes complementary) possible approaches applied (more or less successfully) in various cases depending on a given context or area of application. All of them have in common the fact that open source is considered as a key element in the process, more particularly when it comes to development, dissemination and business purposes.

From a public research perspective, and independently from the finally selected model, the main challenge is finding the right balance between the fundamental nature of research activity and the expected (and evaluated) results in terms of scientific publications, innovative conceptual solutions, corresponding prototypes, etc. In the short-term, spending a lot of effort on such tool development may seem counterproductive compared to the more immediate results that can be obtained if focusing only on publishing scientific papers. However, in the medium- or long-term, a successful open source tool may be one of the biggest assets a research team may produce, which is particularly true in an engineering domain such as Software Engineering. Later on, this can notably translate in many benefits for the team like getting more interesting contacts, collaboration opportunities (projects/contracts) and so potential available resources for continuously exploring different research lines.

Acknowledgments. We would like to thank all the past and present AtlanMod team members as well as collaborators from partner companies that have been working on these various open source projects/initiatives within the last years.

References

1. Alvarez C., Canovas, J., Cabot, J., Bruneliere, H.: EMF Rest: EMF models as REST APIs. In: EclipseCon Europe 2013 - Modeling Symposium, Germany, October 2013.
2. Benelallam, A., Gomez, A., Sunye, G., Tisi, M., Launay,D.: Neo4EMF, a scalable persistence layer for EMF models. In: Proceedings of ECMFA 2014. UK, 2014.
3. Bordeleau, F.: Open Source Modeling: The Key Importance of the Community and the Impact on Business Models. In: EclipseCon France 2014, June 19, 2014.
4. Bozeman, B.: Technology Transfer and Public Policy: a Review of Research and Theory. In: Research Policy, vol. 29, issues 4-5, pp. 627-655. Elsevier (2000).
5. Bruneliere, H., Cabot, J., Dupe, G., Madiot, F.: MoDisco: a Model Driven Reverse Engineering Framework. In: Information and Software Technology, vol. 56, issue 8, doi: http://dx.doi.org/10.1016/j.infsof.2014.04.007, pp. 1012-1032. Elsevier (2014).
6. Canovas Izquierdo, J.L., Cabot, J.: Community-driven language development. In: Proceedings of MISE Workshop at ICSE 2012, pp. 29-35, Switzerland. IEEE (2012).
7. Friedman, J., Silberman, J.: University Technology Transfer: Do Incentives, Management, and Location Matter? In: Journal of Technology Transfer, vol. 28, num. 1, pp. 17-30. Springer (2003)
8. Gonzalez, C.A., Buttner, F., Clariso, R., Cabot, J.: EMFtoCSP: A tool for the lightweight verification of EMF models. In: Proceedings of FormSERA 2012, pp. 44-50. Zurich, Switzerland, IEEE (2012).
9. Jouault, F., Allilaire, F., Bezivin, J., Kurtev, I.: ATL: a Model Transformation Tool. In: Science of Computer Programming, vol. 72, issues 1-2, doi: http://dx.doi.org/10.1016/j.scico.2007.08.002, pp. 31-39. Elsevier (2008).
10. Krishnamurthy, S.: Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects. In: First Monday, vol. 7, num. 6. (2002).
11. Lindman, J., Rossi, M., Puustell, A.: Matching Open Source Software Licenses with Corresponding Business Models. In: IEEE Software, vol. 28, num. 4, pp. 31-35. IEEE (2011).
12. Popp, K.M.: Software Industry Business Models. IEEE Software, vol 28, num. 4, pp. 26-30. IEEE (2011).
13. Sandberg, A., Pareto, L., ARTS, T.: Agile Collaborative Research: Action Principles for Industry-Academia Collaboration. In: IEEE Software, vol. 28, num. 4, pp. 74-83. IEEE (2011).
14. Rogers, E. M.: Diffusion of Innovations (5th Edition). The Free Press (2003).
15. Wright, M., Birley, S., Mosey, S.: Entrepreneurship and University Technology Transfer. In: Journal of Technology Transfer, vol. 29, num. 3-4, pp. 235-246. Springer (2004)
16. Villa Calle, J.D., Bruneliere, H.: EMF Views: Dealing with several interrelated EMF models. In: EclipseCon North America 2014 - Modeling Symposium, San Francisco, California, U.S.A., March 19, 2014.
17. Eclipse Development Process, `http://www.eclipse.org/projects/dev_process/development_process.php`
18. Eclipse Public License (EPL), `http://www.eclipse.org/legal/epl-v10.html`