Assessing the Use of Eclipse MDE Technologies in Open-Source Software Projects

Dimitrios S. Kolovos[1], Nicholas Matragkas[2], Ioannis Korkontzelos[3], Sophia Ananiadou[3], and Richard Paige[1]

[1]Department of Computer Science, University of York, UK
{dimitris.kolovos,richard.paige}@york.ac.uk
[2]Department of Computer Science, University of Hull, UK
n.matragkas@hull.ac.uk
[3]National Centre for Text Mining, School of Computer Science, University of Manchester, UK
{firstname.lastname}@manchester.ac.uk

Abstract. We assess the use of several Eclipse-based Model-Driven Engineering technologies (e.g. EMF, GMF, Xtext, Sirius, ATL, QVTO, Epsilon) in open-source software development projects hosted on GitHub. We discuss our data collection and analysis methodology, and present a number of findings related to the extent to which such technologies appear to be used in open-source projects on GitHub. Our study has revealed, among others, a significant number of GitHub repositories that make use of such technologies and a substantial community of increasingly active MDE-literate developers.

1 Introduction

Several studies have assessed the use of Model-Driven Engineering (MDE) technologies in a range of industry sectors. However, to our knowledge no study has been conducted on the use of such technologies in open-source projects. In contrast to existing studies which are predominantly based on interviews and questionnaires, in this study we extract data directly from open-source GitHub repositories, in order to assess the degree to which a number of different technologies are used *in the wild*. This study focuses on Eclipse-based modelling technologies, including Eclipse Foundation-hosted technologies (e.g. Ecore, ATL, IncQuery, QVTo, Epsilon), as well as technologies hosted outside the Eclipse Foundation – which however provide deep integration with the Eclipse Modeling Framework (`http://www.eclipse.org/emf`) – such as Kermeta and EMFText. The questions that we are interested in answering through this study include the following:

- Which of these technologies demonstrate significant adoption in open-source projects?
- Is the activity related to these technologies increasing or decreasing over time?

– What is the size of the community of GitHub developers who are familiar with these technologies?

The rest of the paper is organised as follows. Section 2 discusses the process we followed to collect the required data from GitHub and Section 3 presents the analysis we performed on this data and the results we obtained. Section 4 then discusses threats to the validity of the obtained results, Section 5 presents an overview of related work, and Section 6 concludes the paper and provides directions for further research on this topic.

2 Data Collection

For this study we have chosen to focus on publicly accessible GitHub repositories due to the dominant role of GitHub in the open-source software development world and the comprehensive API it provides for extracting information of interest. We have also chosen to focus on Eclipse-based MDE technologies due to the fact that the Eclipse Modeling Project (`http://eclipse.org/modeling/`) arguably fosters the most active open-source MDE community. This section outlines the steps of the process we followed to collect data of interest from GitHub.

2.1 Identifying Technologies of Interest

The first step of the process was to identify specific Eclipse-based technologies that we would include in our study. The following technologies were selected for inclusion.

– Ecore: The metamodelling language of the Eclipse Modelling Framework;
– Acceleo (`http://www.eclipse.org/acceleo/`), Xpand (`http://eclipse.org/modeling/m2t/?project=xpand`), JET (`http://eclipse.org/modeling/m2t/?project=jet`): Model-to-text transformation languages that provide deep integration with EMF;
– ATL (`http://eclipse.org/atl/`): Hybrid model-to-model transformation language that provides support for EMF models;
– Xtext (`http://www.eclipse.org/Xtext/`), EMFText (`http://www.emftext.org`): Frameworks for specifying textual concrete syntaxes for Ecore metamodels;
– Emfatic (`http://eclipse.org/emfatic/`): Textual concrete syntax for Ecore;
– OCL[1]: Declarative language for specifying constraints on Ecore metamodels;
– QVTo (`http://www.eclipse.org/mmt/?project=qvto`): Imperative model-to-model transformation language that provides support for EMF models;
– IncQuery (`http://www.eclipse.org/incquery/`): Incremental pattern matching language for EMF models;
– Epsilon (`http://www.eclipse.org/epsilon/`): Family of model management languages and tools that provides support for EMF models;
– Kermeta (`http://www.kermeta.org/`): Object-oriented model management language for EMF;

[1] `http://www.eclipse.org/modeling/mdt/ocl` and `http://www.dresden-ocl.org`

– Henshin (http://www.eclipse.org/henshin/): EMF-based graph transformation language.

Projects that were initially considered but eventually left out of the study after a preliminary evaluation phase included VIATRA2 (http://wiki.eclipse.org/VIATRA2) (very few search results on GitHub, replaced by a new version which is currently under heavy development), CDO (http://eclipse.org/cdo/) (very few results outside of the tool's own repository), QVTd (currently, only an incomplete prototype is provided by the respective Eclipse project (https://projects.eclipse.org/projects/modeling.mmt.qvtd), and UML/Papyrus (UML models can be used in both MDE and non-MDE projects - i.e. exclusively for documentation and communication purposes). Including a wider range of both Eclipse-based (e.g. Spoofax (http://strategoxt.org/Spoofax)) and non-Eclipse based (e.g. MetaEdit+ (http://www.metacase.com/), JetBrains MPS (http://www.jetbrains.com/mps/)) technologies and tools would clearly be an interesting direction for future work.

2.2 Mining GitHub for Files of Interest

Having identified the technologies of interest, the next step was to use GitHub's search capabilities to find files strongly related to each technology (e.g. files with a *.atl* extension for ATL). GitHub's search interface allows searching for files with specific extensions (across all public GitHub repositories), however it also requires a search term (that must appear in the content of the retrieved files) to also be included in the query. After manual experimentation we identified the extensions and search terms in Table 1, which provided the largest number of relevant results for each technology. We also manually sampled the returned results to ensure that no irrelevant files were returned. This was the

Table 1: File extensions and search terms per technology

Techn.	**Ext.**	**Search Term**	**Techn.**	**Ext.**	**Search Term**
ATL	atl	rule	QVTo	qvto	transformation
Emfatic	emf	class	Acceleo	mtl	template
EGL*	egl	var	IncQuery	eiq	pattern
Eugenia*	ecore	"gmf.diagram"	GMF	gmfgraph	figure
EOL*	eol	var	Xtext	xtext	grammar
ETL*	etl	transform	Ecore	ecore	EClass
EVL*	evl	context	OCL	ocl	context
Sirius	odesign	node	Henshin	henshin	rule
MOFScript	m2t	texttransformation	Kermeta	kmt	class
Xcore	xcore	class	JET	javajet	jet
EMFText	cs	syntaxdef	Xpand	xpt	for

case for all technologies except EGL and Acceleo. Files with a *.egl* extension that contain the search term *var* can be either Epsilon Generation Language

model-to-text transformation templates or IBM's Enterprise Generation Language (`http://www.eclipse.org/edt/`) programs. The latter are not of interest to this study and were filtered out by also requiring that the returned snippet contains the *[%* characters, which are used by Epsilon's EGL to denote dynamic template sections. Similarly, files with a *.mtl* extension containing the search term *template* can be Acceleo templates or Blender 3D *material* files. The latter are of no interest to this study and had to be filtered out in a similar fashion.

It is worth noting that GitHub's search interface returns the number of files that meet the query's conditions but only the details (i.e. repository, path in the repository, snippet) of up to 1,000 files returned by the query (for example, the query for Ecore files (`http://github.com/search?q=EClass+extension:ecore&type=Code`), returns 15,038 files but the search interface allows only for the details of 1,000 of them to be retrieved). To retrieve details for additional files of interest, we collected all repositories returned by a first round of queries, and we queried these repositories again for all file extensions of interest. The number of search results and the number of actual files that we managed to retrieve the details of are displayed in Table 2. This endeavour revealed that GitHub's search interface

Table 2: Number of search results and retrieved files per technology

Technology	**Search Results**	**Retrieved Files**	**Technology**	**Search Results**	**Retrieved Files**
Ecore	15038	9629	QVTo	1247	1396
Xpand	7005	4974	GMF	1138	1100
Acceleo	6091	3158	Emfatic	525	510
JET	5323	3682	Sirius	428	503
OCL	3131	2418	EMFText	375	365
Xtext	2272	1605	Kermeta	339	343
ATL	1404	1481	Xcore	305	313
Epsilon	1352	1765	IncQuery	187	192
Henshin	1281	1142	MOFScript	34	34

is not particularly reliable. For example, after the dataset expansion phase we have collected details for 1481 ATL files, 77 more than the number returned by GitHub's search interface in our first search (1404)). Similar discrepancies have been observed for other technologies including Sirius, IncQuery, Kermeta, and Epsilon – however the scale of these discrepancies is not prohibitive for the extraction of useful insights. There are also technologies where we were able to collect the details of a smaller number of files than the number returned by GitHub's search interface – however, this is expected due to GitHub's 1000-results-per-query limitation discussed above.

2.3 Collecting Commits

Having collected the details (i.e. repository, path and snippet) of files of interest, the next step was to use GitHub's API to collect additional information about

the commits in which each of these files was involved since its creation. Each "commit" API response contains information about the *committer* and the *author* of the commit (Git distinguishes between the author of a commit – i.e. the person that did the job – and the person that performed the commit in the repository). For each of these roles, the response contains the real name and email address of the person, (optionally) their GitHub ID, and the commit date. Since not all commits have associated GitHub IDs (in fact, from the 74K commits we collected in this study, more than 31K are not associated with GitHub IDs), in the remainder of this work, we identify developers using their email addresses (as opposed to their GitHub ID).

2.4 Unification

The last step of the process involved unifying the collected information in the form of a model that conforms to the Ecore metamodel illustrated in Figure 1. In its current state, the model contains data about 1,928 repositories, 34,442 files and 74,403 commits. We have made the model and its metamodel available under `http://github.com/kolovos/datasets/tree/master/github-mde` so that the results presented in the next section can be independently reproduced and verified.

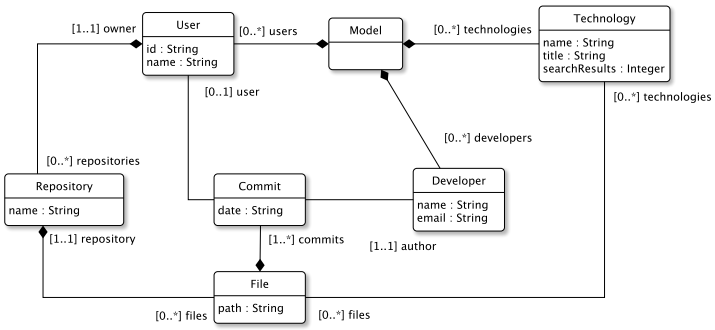

Fig. 1: The Unification Metamodel

3 Data Analysis and Results

Having collected and unified the data of interest in the form of a model conforming to the Ecore metamodel of Figure 1, our next step was to analyse the data and obtain insights related to the use of the MDE technologies involved in our study, both individually and as a whole.

3.1 Number of Relevant Files per Technology

We first report on the number of relevant files found across GitHub. Figure 2 provides an overview of the obtained measurements. We can observe the dominance of Ecore which accounts for more than twice the files compared to the next

technology (Xpand). We also observe that files related to model-to-text transformation languages (Xpand, Acceleo and JET) appear much more frequently compared to files related to model-to-model transformation (ATL, QVTo, Henshin, Kermeta) and model query (OCL, IncQuery) languages.

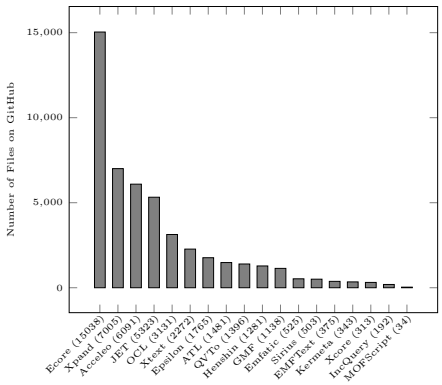

Fig. 2: Number of files per technology

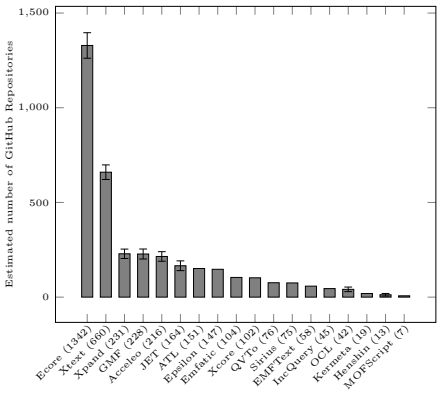

Fig. 3: Number of repositories per technology

3.2 Number of Repositories per Technology

Conceivably, a small number of repositories that contain a large number of files related to a particular technology could lead to misleading interpretations of the chart in Figure 2. Therefore, in Figure 3 we present the estimated number of repositories in which files related to each technology appear.

To compute these estimates, we accept that the files related to each technology for which we have been able to retrieve details comprise a representative sample of the entire population of files related to this technology. Consequently, we can assume that the ratio of repositories over files in the sample generalises over the entire population. To estimate the error in this computation, we used binomial proportion confidence intervals.

We modelled the occurrence of a new, previously unseen repository in each of the files in the sample as a Bernoulli trial, since files are independent as far the repository that they belong to is concerned. We computed normal approximation (Wald), exact binomial (Clopper-Pearson), Wilson score, adjusted Wald (Agresti-Coull) and Jeffreys intervals for 90%, 95% and 99% confidence levels. We observed that in all cases, normal approximation, exact binomial, Wilson score and Jeffreys intervals coincide or exhibit a difference equal or less than 1%. Figure 3 displays the normal (Wald) approximation intervals for 95% confidence level.

While Ecore continues to dominate the chart, there are a few interesting differences. First, Xtext and GMF (6th and 11th in Figure 2), climb up to 2nd

and 4th places in this chart. This is expected as model-to-text transformations typically consist of many files (templates), while graphical and textual syntaxes are often limited to one file per language. In the same spirit, while model-to-text transformation languages still appear to be more popular compared to M2M languages, the gap is substantially smaller in this chart.

We also observe that OCL and Henshin (5th and 9th in Figure 2), slip down to the 15th and 17th position in this chart. This suggests that there are a few repositories that contain a large number of OCL constraints and Henshin transformations. Further analysis revealed that for OCL, two repositories (*dresden-ocl/dresdenocl* and *regressiontesttool/rtt*) account for nearly 50% of the OCL files on GitHub. Similarly, a single repository (*CoWolf/CoWolf*) accounts for more than 80% of all Henshin files on GitHub.

3.3 Number of MDE-literate Developers

Overall, we have identified 2,195 developers (developers with unique email addresses) who are responsible for at least one commit related to files of interest across all technologies included in this study. The breakdown for each technology is presented in Figure 4. Compared to Figure 3, we don't observe any major changes in the relative order of technologies. Estimates and confidence intervals were computed in the same way used in Figure 3.

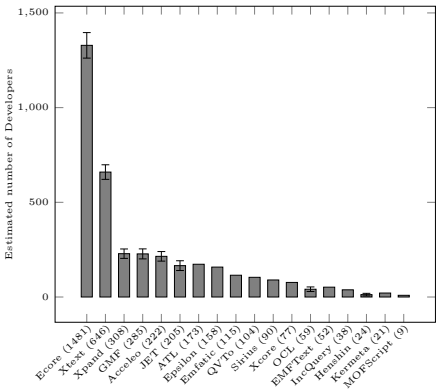

Fig. 4: Estimated number of developers per technology

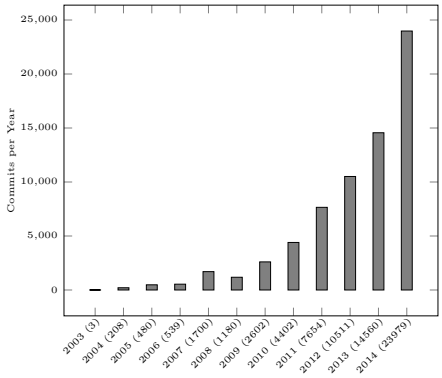

Fig. 5: Number of commits per year across all technologies

3.4 Activity per Year

Figure 5 illustrates the number of commits that involve files of interest (across all technologies) over time. We observe a steep increase, from 4.4K commits in 2010 to 24K commits in 2014[2]. This can be attributed to different extents to the

[2] We intentionally excluded results for 2015 as at the time of writing this paper we are only a few months into the year.

increasing popularity of Git and GitHub and to an increase of the use of Eclipse MDE technologies in open-source software development over the last few years.

3.5 Last-updated Files per Year

On a related measurement, Figure 6 illustrates the number of files that were last updated on different years (as with the previous chart, we omit data for the current year). In this figure we observe that more than 40% of the files of interest can be considered as *active* as they have been updated at least once over the last year.

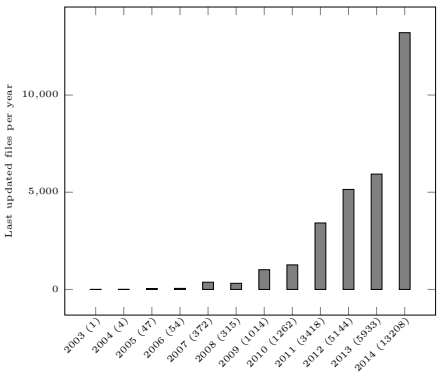

Fig. 6: Number of files last updated on each year

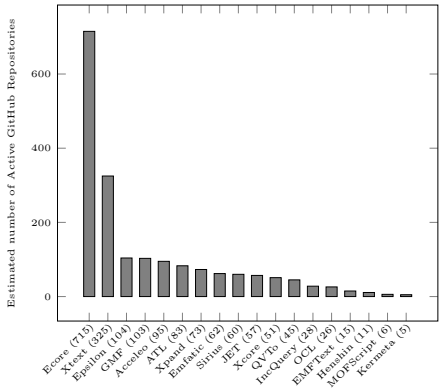

Fig. 7: Estimated number of active repositories per technology

4 Threats to Validity

The following threats to the validity of the results reported in this paper have been identified:

- The results reported largely rely on the accuracy of the number of results reported by GitHub's interface in response to search queries. As previously discussed, there is evidence to suggest indications that GitHub's search interface does not return all matching files for a query. In the most extreme case, while according to the GitHub search interface, 1352 Epsilon-related files exist on GitHub, our extended search has revealed 1765 files. Smaller deviations of similar nature were identified for ATL (1404/1481), QVTo (1247/1396) and Sirius (428/503). Substantial deviations can have significant ripple effects in dependent metrics such as the number of (active) repositories per technology, the number of commits etc.
- Although we extensively sampled the results returned by GitHub's search interface to avoid including irrelevant files in our unified model, it is conceivable that a (small) number of irrelevant files may have still found their way in, thus skewing the results obtained in favour of specific technologies.

5 Related Work

We are not aware of any previous work that evaluates the use of MD* practices and technologies in open-source software developed *in the wild*; existing studies utilise techniques such as surveys, interviews, and case studies to assess MD* adoption, maturity, usage patterns, etc, in industrial settings.

More particularly, in [1,2] the authors perform an industry-wide survey in which they surveyed 450 MDE practitioners and carried out in-depth interviews with 22 more. The findings of this study suggest that MDE is widely used by developers in the industry. The main motivation for using MDE though is not code generation of entire systems, as originally thought, but generation of key parts of a system, often using domain-specific modeling languages developed specifically for a given purpose. This study does not focus on specific MDE technologies, but rather on the generic use of MDE principles in industrial settings.

In [3] the authors performed an empirical study to investigate the state of practice of MDE, and to identify factors that are considered important for its adoption. The subjects of this study are developers from four different companies. The authors use different techniques such as interviews and surveys. The main finding of this study is that tool maturity is the most important factor for adopting MDE. Similarly to the survey in [1], the authors do not focus their analysis on any particular MDE technologies or tools, but they provide a general assessment of MDE.

Likewise, Whittle et al. [4] investigate what is the impact of tools on MDE adoption, and places tooling in an organisational context. To achieve this, the authors use a set of 20 in-depth interviews from two organisations. The main contribution of this work is a taxonomy of tool-related considerations, which can be used to reflect on existing MDE tooling.

The aforementioned empirical investigations focus mainly on the use of MDE in general, and they do not focus on particular technologies or tools. Moreover, these studies focus on the adoption of MDE in industrial settings but they do not consider its adoption in open-source software projects.

In a broader context, in [5] a survey methodology is used to study the adoption of programming languages. To this end, the authors analyse a dataset consisting of 200,000 SourceForge projects, and 590,000 projects tracked by Ohloh. The results of this work indicated that language adoption follows a power law. Moreover, it is found that the community of a language plays a more important role for language adoption than language features such as performance, and reliability.

Finally, in [6] 100,000 GitHub projects are analysed in order to assess the popularity, impact, and interoperability of programming languages. The authors find that earlier popular languages like C are still widely used, while the advent of web development has made relevant languages such as JavaScript and Ruby pervasive.

6 Conclusions and Further Work

This paper reported on the process and the findings of a study that involved mining GitHub to assess the use of Eclipse-based MDE technologies in open-source software development projects. The key findings of our study follow:

- We have identified a substantial number of GitHub repositories (1,928) which make use of the technologies included in this study;
- We have identified a substantial community of 2,195 developers who are responsible for at least one commit related to these technologies;
- We have observed an increasing number of commits on files related to these technologies over the last decade;
- We have observed that model-to-text transformation languages (XPand, Acceleo and JET) appear to be more widely used than model-to-model transformation languages (ATL, QVTo, Henshin);
- We have identified that technologies that are led by industrial organisations (Ecore, Xtext, Xpand, GMF, Acceleo and JET) are more widely used compared to technologies predominately developed in academia (Epsilon, ATL, Kermeta, Henshin).

In future iterations of this work, we plan to extend the selection of participating technologies to include languages, frameworks and tools beyond the bounds of the Eclipse Modelling Framework (e.g. commercial UML tools, Simulink, Microsoft's T4, MetaEdit+, JetBrains MPS, MediniQVT, Modelmorf, USE).

Acknowledgments This research was part supported by the EU, through the OSSMETER FP7 STREP project (grant #318736).

References

1. John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. Empirical assessment of mde in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 471–480. ACM, 2011.
2. Jon Whittle, John Hutchinson, and Mark Rouncefield. The state of practice in model-driven engineering. *Software, IEEE*, 31(3):79–85, 2014.
3. Parastoo Mohagheghi, Wasif Gilani, Alin Stefanescu, and Miguel A Fernandez. An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. *Empirical Software Engineering*, 18(1):89–116, 2013.
4. Jon Whittle, John Hutchinson, Mark Rouncefield, Håkan Burden, and Rogardt Heldal. Industrial adoption of model-driven engineering: are the tools really the problem? In *Model-Driven Engineering Languages and Systems*, pages 1–17. Springer, 2013.
5. Leo A Meyerovich and Ariel S Rabkin. Empirical analysis of programming language adoption. In *ACM SIGPLAN Notices*, volume 48, pages 1–18. ACM, 2013.
6. Tegawendé F Bissyandé, Ferdian Thung, David Lo, Lingxiao Jiang, and Laurent Réveillere. Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 303–312. IEEE, 2013.