Automated Analyses of Model-Driven Artifacts

Obtaining Insights Into Industrial Application of MDE

Josh G. M. Mengerink,
Alexander Serebrenik
Eindhoven University of Technology
Eindhoven, The Netherlands
j.g.m.mengerink,a.serebrenik@tue.nl

Ramon R. H. Schiffelers*
ASML
Veldhoven, The Netherlands
ramon.schiffelers@asml.com

Mark G. J. van den Brand
Eindhoven University of Technology
Eindhoven, The Netherlands
m.g.j.v.d.brand@tue.nl

ABSTRACT

Over the past years, there has been an increase in the application of model driven engineering in industry. Similar to traditional software engineering, understanding how technologies are actually used in practice is essential for developing good tooling, and decision making processes. Unfortunately, obtaining and analyzing empirical data in a model-driven context is still tedious and time consuming, introducing large lead-times. In this paper we present a framework for the automated extraction, analysis, and visualization of data and metrics on model-driven artifacts. We subsequently present various examples of how the framework was successfully applied in a large industrial setting to answer a plethora of different questions with respect to decision making and tool development.

CCS CONCEPTS

• **Software and its engineering → Model-driven software engineering**; *Software evolution*; *Maintaining software*;

KEYWORDS

Model Driven Engineering, Mining Software Repositories, Model Comparison, Metrics, Software Evolution

ACM Reference format:
Josh G. M. Mengerink, Alexander Serebrenik, Ramon R. H. Schiffelers, and Mark G. J. van den Brand. 2017. Automated Analyses of Model-Driven Artifacts. In *Proceedings of IWSM Mensura, Gothenburg, Sweden, October 2017 (Gothenburg'17),* 5 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Model-Driven Engineering (MDE) promises to increase productivity and quality in software development [21]. These promises are a reason for adoption of MDE in various industries such as automotive [17], lithography [31], and health-care [32].

However, MDE also has downsides: besides the heavily researched topic of maintenance in MDE [10, 11, 16, 20, 29], Hutchinson *et al.* [18] have performed an empirical study showing that there are several factors hampering adoption of MDE in industry. The authors state that "a lack of knowledge of exactly how MDE is used in industry" gives rise to "a danger that resources may be wasted and that software tools will fail to develop appropriately" [18].

Consider for instance the object constraint language (OCL) [34], the OMG [3] standard for defining constraints on objects. The OCL is a rich and expressive language. As such, developing tooling that supports OCL entails a significant effort. Indeed such claims are supported by the study of Hutchinson *et al.* [18]: "43% [of respondents] think that MDE tools are too expensive (24% disagree)". Thus, in order to reduce the development cost of tools, and increase their chance at adoption, tools have to be tailored to practical applications. To effectuate this for OCL, in previous work, we have empirically studied the practical usage of OCL on GitHub [27]. This study suggests that only 10 OCL constructs make up 97.8% of OCL expressions in practice. Such results are invaluable when constructing tooling, as tool implementations can be limited to practically occurring constructs. Furthermore, analytical tractability can be tailored towards those cases that occur most frequently in practice.

Unfortunately, obtaining the empirical data necessary for such empirical studies is often tedious. In this paper we present the EMF (Meta)Model Analysis (EMMA) tool, an analysis framework that allows users to: (1) easily gather MDE-related artifacts from version control systems; (2) preprocess the data using extensive querying and filtering options; (3) easily add new metrics to existing analyses tools already in EMMA; (4) customize data selection, filtering, and aggregation using SQL, and (5) automatically visualize analyses outputs using the integrated dashboard. Integrating all aforementioned features, EMMA greatly increases the lead-time on obtaining insights in MDE-related artifacts. The obtained insights subsequently allows for faster and better decision making.

The remainder of this paper is structured as follows: In Section 2 we describe EMMA and its components. In Section 3 we report on industrial case studies, and discuss them in Section 4. Finally, we present future work in Sections 5 and conclude in Section 6.

2 EMMA

The EMF (Meta)Model Analysis Tool (EMMA), consists of several components which provide operations on various sources of data. The general process flow of EMMA is illustrated in Figure 1: EMMA starts with extracting MDE-related files of interest (VCS miner) and storing them in the local intermediate data structure. Then EMMA performs several analyses such as computation of metrics and differences between models, and stores the results in the database. Finally, EMMA includes the data explorer allowing the user to visualize and inspect their data.

*Also with Eindhoven University of Technology.

Gothenburg'17, October 2017, Gothenburg, Sweden

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

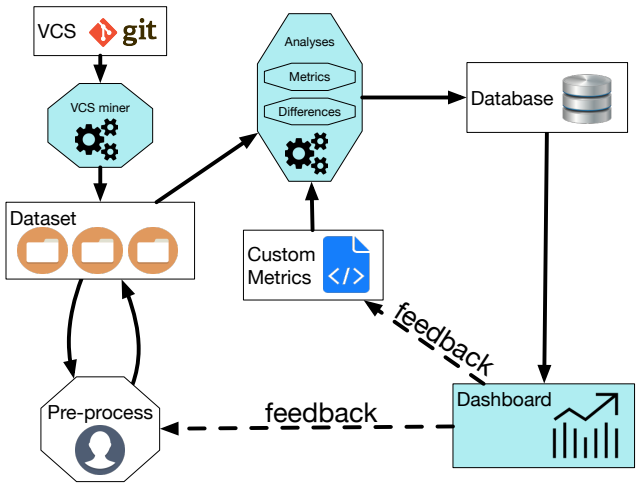

Figure 1: An overview of the EMMA work flow. EMMA's main components have been highlighted in blue.

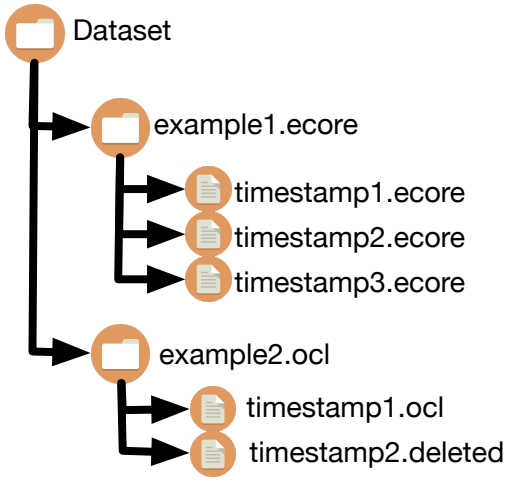

Figure 2: File structure that is used for persisting mined files onto disk. This structure allows reconstruction of "snapshots" of the repository at particular points in time. [27]

2.1 VCS Miner

The VCS Miner component of EMMA systematically mines files defined to be of interests from version-control system (VCS) repositories. For example, in case of OCL, one could mine all files ending in `.ocl` and `.ecore` (cf. [15, 27]). The mined files are then persisted locally, using the intermediate data structure.

2.2 Intermediate Data Structure

The files obtained are persisted onto disk as illustrated in Figure 2.

On top of the file structure, EMMA offers an API for interacting with this file-structure programatically (*e.g.,* query and filter). This API is defined by means of a metamodel, as illustrated in Figure 3.

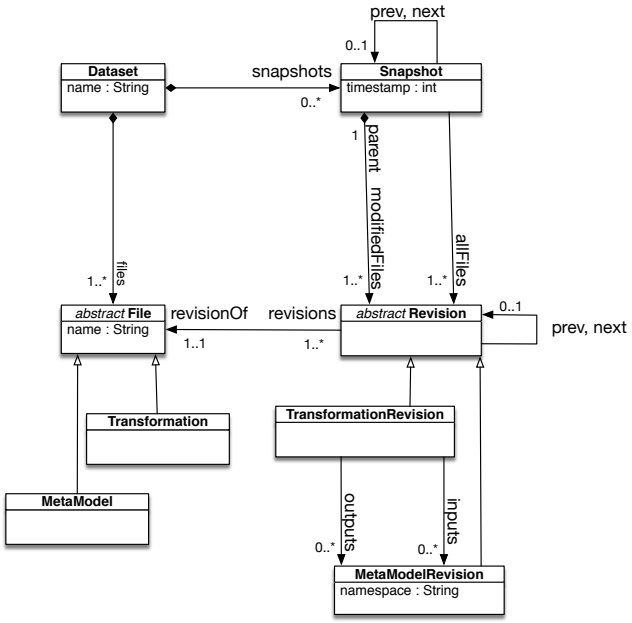

Figure 3: The intermediate datastructure metamodel.

The core concept is, as the name suggests, the `Dataset`. A `Dataset` consists of various `Files`, which have consecutive versions over time (called `Revisions` of that file). Each `Revision` corresponds to a file on the disk, obtained using the VCS miner.

Various file-types are supported, for brevity here, only metamodel (*i.e.,* Ecore [1]) and transformation (*e.g.,* QVTo [4, 12]) are presented. Other file-types include models and constraints (*e.g.,* OCL). Furthermore, as a convenience, the `Snapshot` gathers all `Revisions` that exist at a particular point in the `Snapshot.allFiles` relation. In essence, every commit in the VCS corresponds to a `Snapshot`. The `Revisions` that were actually modified in that commit are also present in the `Snapshot.modifiedFiles` relation.

2.3 Analysis

After preprocessing, the user can analyze their data. At present EMMA supports two types of analyses: Metrics (Section 2.3.1) and Differencing (Section 2.3.2).

2.3.1 Metrics. The first analysis EMMA is able to perform is metrics calculation. In addition to the default metrics defined in EMMA, a user can introduce custom metrics. A user is required to specify for what object type their metric is defined (*e.g.,* Graph, Node) as well as how the metric is calculated on that object type (*e.g.,* graph.nodes.size). An example of two such specifications, vertex out degree and number of nodes in the graph, is presented in Listing 1. Such "custom metrics" are similar to features offered in many well-known software analyses/visualization frameworks such as Moose/Roassal [5], or M3 [7].

EMMA iterates over every object in every artifact of the dataset and calculates the defined metrics on objects of the defined types. An example fragment of data yielded by the metrics component

Listing 1: An example of metrics on a graph DSL

```
@Metric(name="outdegree")
public int outgoing(Vertex v) {
        return v.edges;
}

@Metric(name="numNodes")
public int outgoing(Graph g) {
        return g.nodes.size();
}
```

Table 1: An example of metrics computed by EMMA

File_id	Object_id	Type	outDegree	inDegree	numNodes
1	1	Node	10	10	
2	2	Node	10	5	
3	3	Graph			15
...					

Table 2: An example of differences computed by EMMA

Change	ParentType	Feature	ValueType	Value
ADD	AndExp	right	BooleanLiteralExp	TRUE
CHANGE	AndExp	left	BooleanLiteralExp	FALSE
...				

of EMMA can be found in Table 1. All mined data is subsequently persisted into an SQL-database for further analyses.

2.3.2 Differences. In order to analyze changes over time, we have incorporated EMFCompare [2] in EMMA. EMFCompare allows for a meta-model independent way of computing differences between artifacts. For instance, in Table 2 we show a few example differences computed by EMFCompare [2, 25].

2.3.3 Database. All results of the analyses performed (*e.g.,* tables in Sections 2.3.1 and 2.3.2) are persisted in a database. This provides power users with with data selection, filtering, and grouping. However, for the most common tasks, EMMA provides a user-friendly data explorer and visualizer, as described in Section 2.4.

2.4 Data Explorer

Visualizing metrics and other data on a dashboard is common practice in many classical software-engineering metrics packages such as Grimoire [14], SECONDA [28] and dashboards as offered by Bitergia. Also in MDE, visualization dashboards are being constructed, such as the one for metamodel clustering on MDEForge [6].

EMMA also includes a web-frontend that allows users to explore and visualize their data. The primary data-processing is performed using pivot tables [19]. This allows users to filter, group, and aggregate their data. The resulting data can then be visualized using *e.g.,* bar charts and scatter plots. Examples of visualizations produced using EMMA can be found in Figures 4 and 5. Figure 4 shows the evolution of the number of different elements, such as the number

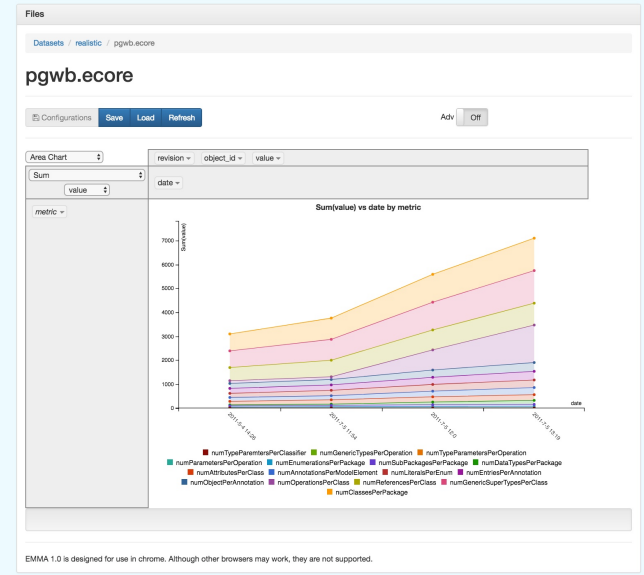

Figure 4: Evolution of size and structure of a metamodel.

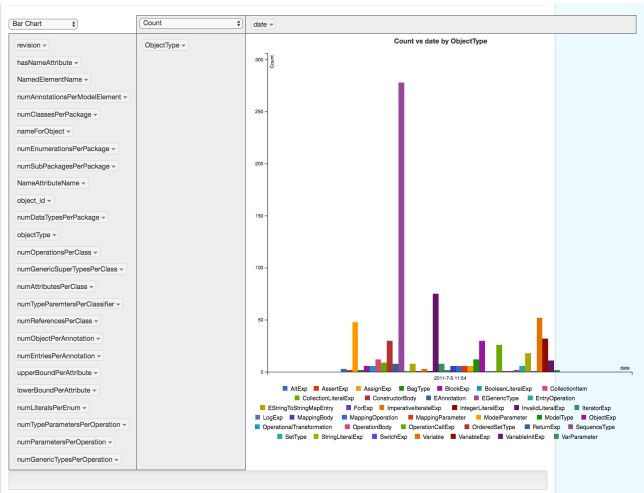

Figure 5: Frequency of language constructs used in a model-to-model transformation at a single point in time.

of classes per package and the number of references per class, of a DSL called PGWB. Figure 5 shows the frequency of language constructs used in a model-to-model transformation.

3 INDUSTRIAL APPLICATIONS

Our research takes place at ASML, provider of lithography equipment for the semiconductor industry. At ASML, MDE is used to allow engineers to model systems in terms of their domains. Such "design models" can subsequently be transformed to dedicated analysis formalisms using model-to-model transformations. An example of such application is the CARM [31] ecosystem of DSLs. The CARM ecosystem consists of twenty-two DSLs and a hundred model-to-model transformations with a revision history of over six years.

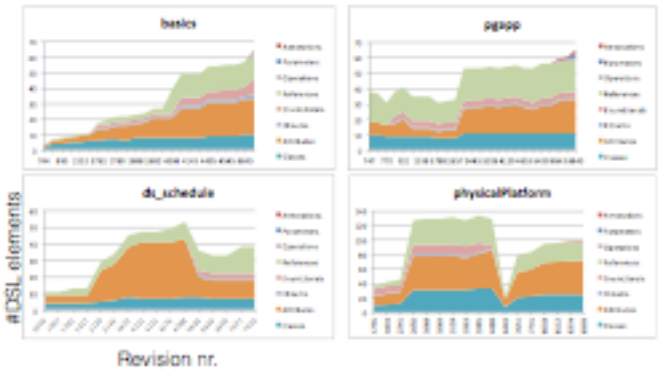

Figure 6: Size and structure of CARM [31] DSLs over time, some plots taken from previous works [23, 33].

We have deployed EMMA at ASML, and used it to facilitate various studies. These studies have all contributed to gaining insights, tailoring of tools, or improving decision-making processes. Throughout this section we briefly describe a number of these studies, explain how EMMA contributed in the analyses, and highlight the conclusions drawn.

3.1 Lehman's Laws

The first question we discuss, relates to the evolution of DSLs. In traditional software engineering, there are the well-known "Lehman's Laws of Evolution" [8, 13, 22]. In particular, the law of increasing complexity states that "as an E-type system evolves, its complexity increases unless work is done to maintain or reduce it.".

At ASML, the question arose whether evolution of their DSLs adheres to the law of increasing complexity. There, using EMMA we performed a case study on the CARM ecosystem [31].

We defined metrics to calculate the number of occurrences of each datatype in DSL definitions, *e.g.,* "number of classes per package", "number of attributes per class". By plotting these metrics, as illustrated in Figure 6, we observe increasing trends in various DSLs, and indeed find that there are specific moments in which DSLs are refactored to reduce complexity and increase maintainability.

Using this knowledge, we show that Lehman's law of increasing complexity holds for the DSLs at ASML, and that DSL maintenance is a topic that should not be taken lightly. As a result, more research into DSL evolution has been commissioned by ASML. Some of the research results obtained are detailed in the coming sections.

3.2 Nature of Metamodel Evolution

To support MDE at ASML, research is being conducted into the automated co-evolution of metamodels and models. In order to gain insight into how metamodels actually evolve, the differencing module of EMMA was used to gain insight into the most frequently occurring kinds of changes. We performed an exploratory case study [30] on the CARM ecosystem of DSLs [31].

In CARM, we compared subsequent versions of every DSL in the main branch of the git repository. We chose to only study the main branch, as subsequent versions resemble finished products.

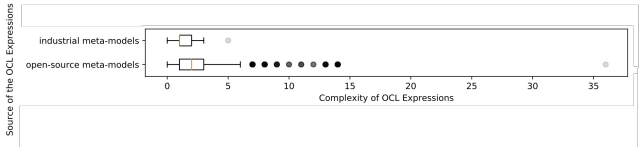

Figure 7: Boxplot of complexity for open-source and industrial metamodels, taken from [26]

The results of our case study [25], were that only 70 of 176 possible types of evolution actually occurred in practice. This knowledge allowed us to effectively upgrade existing tooling in order to automate model co-evolution [33].

3.3 OCL Usage

The DSLs employed at ASML are highly complex and, as such, metamodels alone are not sufficiently expressive to accurately model the language. To address this limitation ASML uses OCL. We study the differences in complexity between industrial and open-source OCL code. Indeed, if open source OCL code and industrial OCL code are not that different, then open source OCL code can be used as a proxy for industrial OCL code enabling further studies and further tools. Conclusions of those studies can be then transferred to and the tools can be applied to the industrial OCL code.

For this purpose, we use the large-scale dataset of open source files containing OCL obtained and studied in previous work [27]. For the industrial OCL code, we used EMMA to create a similar dataset of industrial files containing OCL, derived from CARM [31].

Subsequently, the Metrics module in EMMA was used to compute complexity metrics [26] on both datasets. Using the data-explorer, insights into the differences were gained, as illustrated in Figure 7. Using the provided boxplots and appropriate statistical hypothesis testing, we concluded that while industrial OCL code appears to be less complex than open-source OCL code, this difference is not statistically significant, suggesting that the open source OCL code and the industrial OCL code are not that different [24].

4 DISCUSSION

In Section 3 we reported on a series of industrial studies conducted using EMMA. Preliminary conclusion of these studies is that the results obtained with EMMA are encouraging: ASML engineers are using EMMA to gain insight into the MDE artifacts they are responsible for.

To encourage replication of our studies and further development of EMMA, we make it publicly available on GitLab[1].

5 FUTURE WORK

We envision various directions of future work.

First, we consider extending EMMA to incorporate mining data from additional data sources. For instance, similar to MetricsGrimoire [14], we would like to incorporate data from such sources as bug trackers, code review repositories and mailing lists. Indeed, these data sources can provide complementary insights in communication and collaboration between software engineers creating DSL

[1] gitlab.com:PHD-MDCE/MetricsMiner, gitlab.com:PHD-MDCE/EMMA-Frontend

metamodels, on the one hand, and software engineers in charge of models and model transformations dependent on those metamodels, on the other hand.

Next, at present EMMA does not offer support for concrete syntax definitions (*e.g.,* XText [9]). We would like to extend EMMA to incorporate even more file types, such as concrete and graphical syntax definitions.

While the results produced by EMMA are already being used by the ASML engineers, we plan to evaluate usability of EMMA in the industrial context by means of several user studies. Furthermore, so far EMMA has only be applied to the open source data and to commercial data of ASML. Application of EMMA to MDE artifacts of different companies or from different industries might necessitate adaptations to EMMA.

6 CONCLUSIONS

In this paper we have presented EMMA, the EMF (Meta)Model Analysis tool. EMMA is an analysis framework and data exploration dashboard. EMMA allows industrial users and researchers alike to gain easy insight into their MDE artifacts, allowing for more effective and accurate decision making.

Furthermore, we have presented several case studies, in which we have applied EMMA in a large-scale industrial setting. There, we have found that EMMA allows us to gain easy insights into many facets of the MDE artifacts present.

REFERENCES

[1] Ecore. lhttp://www.eclipse.org/modeling/emf/. (????). Accessed: 2016-7-20.

[2] 2015. EMF Compare. https://www.eclipse.org/emf/compare/. (2015). Accessed: 2015-04-07.

[3] 2015. OMG. http://www.omg.org. (2015). Accessed: 2017-07-03.

[4] 2015. QVTo. http://www.eclipse.org/mmt/?project=qvto. (2015). Accessed: 2015-04-07.

[5] Vanessa Pena Araya, Alexandre Bergel, Damien Cassou, Stéphane Ducasse, and Jannik Laval. 2013. Agile visualization with Roassal. *Deep Into Pharo* (2013), 209–239.

[6] Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Amleto Di Salle, Ludovico Iovino, and Alfonso Pierantonio. 2014. MDEForge: an Extensible Web-Based Modeling Platform. In *Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, CloudMDE@MoDELS 2014, Valencia, Spain, September 30, 2014. (CEUR Workshop Proceedings)*, Richard F. Paige, Jordi Cabot, Marco Brambilla, Louis M. Rose, and James H. Hill (Eds.), Vol. 1242. CEUR-WS.org, 66–75. http://ceur-ws.org/Vol-1242/paper10.pdf

[7] Bas Basten, Mark Hills, Paul Klint, Davy Landman, Ashim Shahi, Michael J Steindorfer, and Jurgen J Vinju. 2015. M3: A general model for code analytics in Rascal. In *Software Analytics (SWAN), 2015 IEEE 1st International Workshop on*. IEEE, 25–28.

[8] John Businge, Alexander Serebrenik, and Mark G. J. van den Brand. 2010. An empirical study of the evolution of Eclipse third-party plug-ins. In *IWPSE-EVOL*, Andrea Capiluppi, Anthony Cleve, and Naouel Moha (Eds.). ACM, 63–72.

[9] Moritz Eysholdt and Heiko Behrens. 2010. Xtext: Implement Your Language Faster Than the Quick and Dirty Way. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion (OOPSLA '10)*. ACM, New York, NY, USA, 307–309. https://doi.org/10.1145/1869542.1869625

[10] Moritz Eysholdt, Sören Frey, and Wilhelm Hasselbring. 2009. EMF Ecore Based Meta Model Evolution and Model Co-Evolution. *Softwaretechnik-Trends* 29, 2 (2009). http://pi.informatik.uni-siegen.de/stt/29_2/01_Fachgruppenberichte/SRE/09-frey.pdf

[11] Jokin García, Oscar Diaz, and Maider Azanza. 2013. Model Transformation Co-evolution: A Semi-automatic Approach. In *SLE*. LNCS, Vol. 7745. Springer, 144–163.

[12] Christine M. Gerpheide, Ramon R. H. Schiffelers, and Alexander Serebrenik. 2016. Assessing and improving quality of QVTo model transformations. *Software Quality Journal* 24, 3 (2016), 797–834. https://doi.org/10.1007/s11219-015-9280-8

[13] Michael W. Godfrey and Daniel M. German. 2014. On the evolution of Lehman's Laws. *Journal of Software: Evolution and Process* 26, 7 (2014), 613–619. https://doi.org/10.1002/smr.1636

[14] Jesús M. González-Barahona, Gregorio Robles, and Daniel Izquierdo-Cortazar. 2015. The MetricsGrimoire Database Collection. In *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, Massimiliano Di Penta, Martin Pinzger, and Romain Robbes (Eds.). IEEE Computer Society, 478–481. https://doi.org/10.1109/MSR.2015.68

[15] Regina Hebig, Truong Ho-Quang, Michel R. V. Chaudron, Gregorio Robles, and Miguel Angel Fernández. 2016. The quest for open source projects that use UML: mining GitHub. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016*, Benoit Baudry and Benoît Combemale (Eds.). ACM, 173–183. https://doi.org/10.1145/2976767

[16] R. Hebig, D. E. Khelladi, and R. Bendraou. 2017. Approaches to Co-Evolution of Metamodels and Models: A Survey. *IEEE Transactions on Software Engineering* 43, 5 (May 2017), 396–414.

[17] Markus Herrmannsdörfer, Sebastian Benz, and Elmar Juergens. 2008. Automatability of Coupled Evolution of Metamodels and Models in Practice. In *MoDELS*. Springer, 645–659.

[18] John Edward Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. 2011. Empirical assessment of MDE in industry. In *ICSE*. 471–480.

[19] Bill Jelen and Michael Alexander. 2006. *Pivot Table Data Crunching for Microsoft(R) Office Excel(R) 2007 (Business Solutions)*. Que Corp., Indianapolis, IN, USA.

[20] Djamel Eddine Khelladi, Regina Hebig, Reda Bendraou, Jacques Robin, and Marie-Pierre Gervais. 2016. *Metamodel and Constraints Co-evolution: A Semi Automatic Maintenance of OCL Constraints*. Springer, 333–349.

[21] Anneke G Kleppe, Jos B Warmer, and Wim Bast. 2003. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional.

[22] M. M. Lehman. 1984. On Understanding Laws, Evolution, and Conservation in the Large-program Life Cycle. *J. Syst. Softw.* 1 (Sept. 1984), 213–221. https://doi.org/10.1016/0164-1212(79)90022-0

[23] Josh Mengerink, Ramon RH Schiffelers, Alexander Serebrenik, and Mark van den Brand. 2016. DSL/Model Co-Evolution in Industrial EMF-Based MDSE Ecosystems.. In *ME@ MODELS*. 2–7.

[24] Josh G. M. Mengerink, Jeroen Noten, Ramon R. H. Schiffelers, Mark G. J. van den Brand, and Alexander Serebrenik. 2017. A Case of Industrial vs. Open-Source OCL: Not So Different After All. In *20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*.

[25] J. G. M. Mengerink, Alexander Serebrenik, Ramon R. H. Schiffelers, and M. G. J. van den Brand. 2016. A Complete Operator Library for DSL Evolution Specification. In *2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, Raleigh, NC, USA, October 2-7, 2016*. IEEE Computer Society, 144–154. https://doi.org/10.1109/ICSME.2016.32

[26] Jeroen Noten. 2017. *Usage Analysis of the Object Constraint Language in Model Driven Engineering*. Master's thesis. Eindhoven University of Technology, the Netherlands.

[27] Jeroen Noten, Josh G. M. Mengerink, and Alexander Serebrenik. 2017. A Data Set of OCL Expressions on GitHub. In *14th Working Conference on Mining Software Repositories*.

[28] Javier Pérez, Romuald Deshayes, Mathieu Goeminne, and Tom Mens. 2012. SECONDA: Software Ecosystem Analysis Dashboard. In *16th European Conference on Software Maintenance and Reengineering, CSMR 2012, Szeged, Hungary, March 27-30, 2012*, Tom Mens, Anthony Cleve, and Rudolf Ferenc (Eds.). IEEE Computer Society, 527–530. https://doi.org/10.1109/CSMR.2012.69

[29] Louis M. Rose, Richard F. Paige, Dimitrios S. Kolovos, and Fiona A. C. Polack. 2009. An analysis of approaches to model migration. In *MoDSE-MCCM*. 6–15.

[30] Per Runeson and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2008), 131.

[31] Ramon R. H. Schiffelers, Wilbert Alberts, and J. P. M. Voeten. 2012. Model-based Specification, Analysis and Synthesis of Servo Controllers for Lithoscanners. In *6th International Workshop on Multi-Paradigm Modeling*. ACM, 55–60.

[32] Pieter Van Gorp, Irene Vanderfeesten, Willem Dalinghaus, Josh Mengerink, Bram van der Sanden, and Pieter Kubben. 2012. Towards generic MDE support for extracting purpose-specific healthcare models from annotated, unstructured texts. In *International Symposium on Foundations of Health Informatics Engineering and Systems*. Springer Berlin Heidelberg, 213–221.

[33] Y. Vissers, J. G. M. Mengerink, Ramon R. H. Schiffelers, Alexander Serebrenik, and Michel A. Reniers. 2016. Maintenance of specification models in industry using Edapt. In *2016 Forum on Specification and Design Languages, FDL 2016, Bremen, Germany, September 14-16, 2016*, Rolf Drechsler and Robert Wille (Eds.). IEEE, 1–6. https://doi.org/10.1109/FDL.2016.7880374

[34] Jos Warmer and Anneke Kleppe. 2003. *The Object Constraint Language: Getting Your Models Ready for MDA* (2 ed.). Addison-Wesley.