Diversity of MDE Toolboxes and Their Uncommon Properties

Paulo Gabriel Teixeira[1], **Bruno Gabriel Araújo Lebtag**[1], **Fábio Paulo Basso**[2],

[1]Instituto de Informática - PPGCC - Universidade Federal de Goiás (UFG)
Caixa Postal 131 – CEP 74001-970 – Goiânia – GO – Brazil

[2]UNIPAMPA - PPGES - Campus Alegrete
Av. Tiarajú, 810 - Bairro: Ibirapuitã - Alegrete - RS - CEP: 97546-550

fabiopbasso@gmail.com

{paulogabriel,brunogabriel}@inf.ufg.br

Abstract. *Model-Driven Engineering (MDE) has reached some maturity. Due to that, a high diversity of technologies and platforms have emerged to support the resolution of a range of problems and contexts in which MDEis adopted as a solution. As a consequence, when some level of reuse of those artifacts (such as model transformations, Domain-Specific Languages(DSLs) and refinement tools), difficulties are faced due to the high diversity of formats in which all those assets are specified. Since we noted this trend, we decided to search for instances in the literature that supports our hypothesis of a high degree of diversity in MDE artifacts in the state of the practice. Thus, we carried out an exploratory literature review. As a result, we summarized key studies used as input to build a search string adopted to structure a future systematic literature review. Our study contributes by classifying nine types of MDE toolboxes with uncommon properties than those usually found in MDE workbenches.*
Keywords: *Model-Driven Design, Tools, Literature Review.*

1. Introduction

Software reuse can bring benefits in terms of quality and productivity (Krueger 1992; Lucrédio et al. 2012). However, achieving large-scale reuse in the software industry is still a challenge. Model-Driven Engineering (MDE) is recognized as a remarkable methodology that fosters reuse, quality, and productivity (Lucrédio et al. 2012) by providing the so-called MDE Artifacts (France and Rumpe 2007) such as libraries, Domain-Specific Languages (DSL), and model transformations (Kelly and Tolvanen 2008). These artifacts can be adapted to be used in a diversity of contexts, leveraging the reuse degree and bringing productivity to Software Development Processes (SDPs) (Fuggetta and Nitto 2014), also by its potential of automation in code generation.

MDE has been considered a candidate to achieve large-scale reuse, in particular by initiatives such as ReMODD (France and Rumpe 2007) and GEMOC (Combemale et al. 2014). However, a higher level reuse requires not only code reuse, but a global reuse of MDE Artifacts to see impacting benefits (France and Rumpe 2007; Krueger 1992; Lucrédio et al. 2012). Thus, aforementioned contributions are devoted to classify and catalog existing contributions in the area.

To highlight existing DSL-Supporting tools and features mapped from selected primary studies, recently in (Iung et al. 2020) we presented a systematic mapping study

identifying 59 tools and analyzed their features from 230 selected papers. This was an important contribution for classification, characterizing MDE workbenches and DSL builders. However, we have observed along the years a high degree of diversity on the technologies, languages, platforms, and paradigms that have been adopted to specify and implement MDE Artifacts on academic and industrial projects, which did not appear in our previous work.

In this paper, we aim at characterizing that such diversity of artifacts can bring difficulties for future classification and cataloging, such as (i) a particular model transformation (coded in QVT, for example) can not be reused to transform a source model (specified in SysML, for example) if such model is not specified using the same DSL adopted to specify the first version of a source model for which the transformation has been written (UML, for example), even if they intend to produce the same target code and if they solve a problem for the same domain, (ii) the same aforementioned model can not be reused in distinct projects if distinct languages are adopted to specify it (SysML and UML, for example), even if the domain is the same, and (iii) a model transformation engine can not be reused if the model transformation is specified in a technology not supported for that engine. Thus, we claim that the recent initiatives for classification on the globalization of DSL and MDE Artifact repositories lack solutions concerning to the heterogeneity of MDE Artifacts, which can hamper a high-degree of reuse when adopting MDE, since the lack of uniformity in the representation of such artifacts leads to a redesign or reimplementation of solutions, increasing the costs and affecting productivity.

We found that the current knowledge about MDE Artifacts is still limited to understand issues for their large scale reuse. Thus, we conducted an exploratory ad-hoc literature review for supporting a structured protocol that will guide our next systematic mapping research. Section 2 provides a background and the methodology adopted to conduct this search. Section 3 highligths the diversity in MDE Artifacts that we have found. Section 4 discusses the main consequences from this diversity for a global reuse scenario. Finally, Section 5 presents concluding remarks.

2. Background

Different contexts (e.g., software projects, companies, processes, etc.) need the connection of different artifacts (Whittle et al. 2015). MDE Artifacts are recognized as any artifacts produced within the context of a model-driven software development process with the potential to reuse, i.e., that could be adapted to serve other purposes different from that one initially planned. MDE Artifacts include Domain-Specific Languages (DSL), models, metamodels, model transformations, transformation chains, and all sorts of artifacts that can support MDE to be adopted. MDE Artifacts can be applied in contexts that are distinct from those from which they were initially planned. Thus, these artifacts can be reusable in inter-organizational contexts, i.e., if they are properly available, they could be reused in a diversity of contexts, including different companies.

Reuse of MDE Artifacts has been treated as a prominent trend (Mussbacher et al. 2014). Some initiatives have investigated four perspectives of reuse on this topic: (i) artifacts in global repositories (such as ReMODD) (Aranega et al. 2012; Fuggetta and Nitto 2014; Couto et al. 2014; Basciani et al. 2014), (ii) support to automatic adaptation of resources (such as DSLs,

model transformations and tools) to specific contexts (Brambilla and Fraternali 2014; Monteiro et al. 2014), (iii) globalization of DSL (Combemale et al. 2015), and (iv) MDE as a Service (MDEaaS) (**?**).

The Repository for Model-Driven Development (ReMoDD) is a resource that aims to support the work of researchers and educators in the Model-Driven Development (MDD) community. Researchers and practitioners can use the repository as a vehicle for sharing exemplar models, illustrative descriptions of modeling methodologies and techniques, detailed modeling case studies, modeling success stories, and other forms of modeling experience and knowledge (Aranega et al. 2012). However, the adoption of resources available in ReMoDD requires adaptation to the context, which can be costly and can hamper the reuse.

In turn, automatic adaptation consists of a procedure to use a semantic representation of an MDE Artifact and, based on a transformation, automatically produce all the adaptations and configuration files that enable them to be reused in a specific context (Basso et al. 2013; Brambilla and Fraternali 2014; Monteiro et al. 2014). This approach is dependent on the semantic description of an MDE Artifact, which is rarely available.

The development of modern complex software-intensive systems often involves the use of multiple DSLs that capture different system aspects (Combemale et al. 2014). Supporting coordinated use of DSLs leads to what is known as the globalization of DSLs, that is, the use of multiple DSLs to support coordinated development of diverse aspects of a system (Cheng et al. 2015). To introduce MDE Artifacts in contexts, other contributions for reuse proposes DSLs for MDE Settings (Guerra et al. 2010; Vanhooff et al. 2006; Yie et al. 2012) that connect these artifacts in representation for chains, model typing, bindings, components, etc. They open the possibility of a diversity of applications, but they do not provide a means of how to integrate and interoperate distinct DSLs. MDEaaS is characterized by specialized services implemented by professionals or companies (Brambilla and Fraternali 2014; Monteiro et al. 2014) that aim to customize MDE Artifacts for introduction in target contexts. MDE Artifacts shared on a global scale are, therefore, of interest in MDEaaS.

Although advancing our understanding of tool support that promote facilities such as for upload and download of data through Software as Service (SaaS) (Basciani et al. 2014), they are all limited for artifacts represented conforms to the Eclipse Modeling Framework (EMF) (Steinberg et al. 2008). Moreover, despite the aforementioned initiatives, we believe that the high diversity of MDE technologies (not limited to EMF technologies) are an important factor that hampers the large-scale reuse of MDE Artifacts. In particular, the high diversity of technologies within the EMF project itself also brings problems to promote the reuse. Thus, we decided to investigate the literature to understand which are the main challenges associated with large-scale reuse in MDE.

This study presents results of an analyse of key papers obtained from an ad-hoc research that we intend to use as input to formulate a research protocol for a systematic literature review. We discuss our main observations and derive some findings, which are essential for formulating future research questions.

3. Diversity in MDE Artifacts

DSLs have been proposed as mechanisms that support the automation of tasks in diverse phases of software development processes. Surveyed MDE toolboxes are based on well-established concepts for MDE workbenches, DSL constructors, and toolchain utilities. Previously, MDE tools were also characterized according to system-of-Systems perspectives (Neto et al. 2014). In a recent systematic mapping study (Iung et al. 2020) we have found 59 tools devoted mainly to:

- Proposing DSLs techniques and methodologies to represent abstractions (e.g., textual and graphical, UML Profiles) from specific or general domains;
- Proposing the mapping from model representations to other specifications;
- Demonstrating how models are generated and refined in lifecycles through model-to-model transformations, as well as how models are used to generate source-code for target platforms through mappings;
- Developing model management tools, with operations for co-evolution, refactorings, and metametamodeling;
- The development of underlying frameworks for the execution of model transformations developed with more than one language and allowing compositions of model-based operations and;
- DSLs are mostly generated based on well-accepted concepts and tools for Metamodeling and constraint languages; Metamodels are generated based on Metametamodeling concepts and tools, and Metametamodels are too generated.

In the following, we present what we have found as other types of artifacts, characterized as owning uncommon properties, but that is also used in the MDE context as follows:

- System engineering artifacts: Application models and domain models serialized in different formats (XML, XMI, RDF, JSON) (Izquierdo and Cabot 2013). These models are derived from software projects, including DSLs for structured use cases and tools to extract textual data from use case documents into models.
- Software processes integrated with MDE Artifacts (Maciel et al. 2013): Models for SDPs are represented with the diverse process or workflow modeling languages (Pillat et al. 2015); Methods represented as statecharts available in the new OMG's language named Essence (Johnson et al. 2012); Model transformations and nested executions mapped for APIs with MDE Settings (Hebig et al. 2013; Mascarenhas et al. 2013).
- Tools for design/refinement and systems shared in the cloud trough the REST API (Costa et al. 2016) and OSLC (Elaasar and Neal 2013).
- Scripts and templates for code generation, including those coded in regular languages (Java, C#, C++, ...) (Horváth et al. 2006) for execution in arbitrary model transformation engines and also for running in modeling tools such as Enterprise Architect (EA), Star UML, and Visual Paradigm (Hutchinson et al. 2011).
- These scripts are also available as component libraries (Hong-min et al. 2010; Medeiros et al. 2015) and the reuse mechanisms associated include the management of cross-domain features (Sant'Anna et al. 2013).
- Ad-hoc DSLs: we consider this type of artifact any construction of a design language that is not generated automatically with EMF or by another metafacility tool. For example, A DSL is represented in an XML Schema (Neubauer et al. 2015) or developed as POJOs using Java Annotations (Basso et al. 2014).

- Database Tables and SQL Scripts (Quercini and Reynaud 2013). As strange as it sounds, a recent study presented these elements as part of MDE tool chains (Batory et al. 2013b). Besides, some metamodels and models are represented in Excel Spreadsheets (Reschenhofer and Matthes 2015).
- Adaptive MDE Artifacts. This type of representation for artifacts includes run-time adaptations for business processes (dos Santos Rocha et al. 2015), context-aware applications based on SPL concepts (Marinho et al. 2013), adaptive test cases (Basso et al. 2014) and dynamic model transformation chains (Cuadrado et al. 2014).
- Diverse documents including instructions for integration of MDE Artifacts, use case templates for developing model transformations, test plans for MDE, etc.

An eminent issue imposed by the characterized heterogeneity of constructors of MDE Artifacts is incompatibility (Neto et al. 2014), which can be handed in two perspectives: semantics (Costa et al. 2013) and syntactic (Yie et al. 2012). For example, these tools carry the semantics of applicability, which are important to select the one to be included in an automated SDP. Tools also carry syntax that, when represented with standard constructors for MDE, can be verified through computational support. Toolboxes developed with alternatives ways also support these features, but they usually are hand-coded.

4. Discussion

Batory et al. claim that in non-automated SDPs, technical stakeholders frequently turn simple tasks into complex, replicating work whose knowledge and solutions are common to many development contexts (Batory et al. 2013a). The authors refer to this problem as a dark knowledge, suggesting graph grammars (e.g., diverse models) as a solution to uncover expert know-how. Since hidden expert knowledge prejudice the software development and only a few specialists can perform some development tasks, expert knowledge should be explicit and reused.

Implicit knowledge is also a problem that difficult the specification of tasks assisted by the tool when automating SDPs, since it is dependent on tool specialists (Fuggetta and Nitto 2014; Liebel et al. 2014; Jakumeit et al. 2014). Thus, dark knowledge is also a problem to specific processes that chains many model-based tasks (Hebig and Bendraou 2014), since the know-how about the use and adaptation of these tasks in data shared in repositories/service providers are mostly descriptive. Accordingly, descriptive information cannot be processed by programs to be quickly accessed and used by software engineers in some engines for the execution of model-based operations (Etien et al. 2013). This way, it would be helpful to have access to tool specialist knowledge as a means of formal specifications, which could promote the reuse of knowledge of MDE Artifacts in diverse software process specifications.

This missing knowledge, which should be known a priori before of putting tools to work together, is a reason that difficult collaboration in MDEaaS. The absence of tool experts in the industry is one of the problems that hamper the adoption of MDE Artifacts in practice (Whittle et al. 2015). This issue is more evident in a global reuse scenario, in which shared MDE Artifacts are unknown by potential reusers. Therefore, while expert knowledge is hidden in the tool specialist mind this problem will persist, hampering initiation towards automation of SDPs with MDE.

Further works could take the papers presented in this study as input to derive a systematic literature review studies. In special, we found that current contributions are

devoted to technical data (artifacts types, data types, components, chains, and others), but descriptive data related to social and business aspects are needed to promote MDE in the large. Likewise, it is not a novelty that tacit knowledge must be converted into explicit knowledge. However, an interesting question is how to proceed in practice?

With this issue in mind, we suspect that the quality for descriptive information associated with MDE assets is essential for a collaborative and global reuse scenario in MDE, and that has been neglected by current contributions in the area. For example, an eminent problem for collaboration in MDEaaS is the lack of structured descriptive information about MDE Artifacts and Settings to helping in automated design while composing toolchains and megamodels such as: How to integrate them? How to adapt them? How to start them? What are their dependencies? Which are their pre and post conditions? All these questions should be answered considering an umbrella of investigation focusing on decision-making issues, besides the current explored technical issues.

This further literature review study is important to promote collaboration in the area. Aiming at the implementation of a KB for MDE, the state-of-art proposes a centralized repository under the umbrella of GEMOC (Combemale et al. 2015). This initiative is gaining attention in the academy, with many experts in the area contributing to specific research topics. However, as discussed in the next section through research gaps, GEMOC itself considers only the top of the iceberg of the needs in a KB for MDE.

5. Concluding Remarks

The literature of the area lacks a survey that analyzes issues for reuse considering diversity in MDE Artifacts. For example, it is important to acknowledge collaboration issues that occur when artifacts are not represented in known MDE workbenches, such as those built on the EMF umbrella. Likewise, we have seen that the state-of-the-practice is full of implementations beyond the common knowledge, but we miss a work that points out to this diversity. Thus, issues non associated with EMF-based artifacts are barely discussed in the literature and certainly affect the implementation of services to introduce MDE in target contexts: MDE as a Service (MDEaaS).

To reach the diversity of MDE Artifacts, we conducted an ad-hoc survey from literature, finding diversity in constructors for MDE Artifacts that challenges implementations for collaborative approaches for MDEaaS. To facilitate the introduction of MDE Artifacts developed by others, the literature recommends a collaborative approach through a common Knowledge Base (KB). The justification for such is that this will allow the reuse of artifacts shared on the web instead of developing them from scratch, thus needing classification of data associated with MDE Artifacts.

Currently, as far as we know, there is no knowledge base/repository to access data from these uncommon MDE Artifacts and few are acknowledged on technical requirements to make collaboration a reality. Although proposals for reuse mechanisms based on Model Repositories are alternatives, issues to apply them on a global scale have not been analyzed through a survey from the literature. For example, some model repositories have no benefit if an artifact is not constructed based on their internal structures, as those found in our ad-hoc survey. Besides, several types of representations are associated with MDE Artifacts, which limits the current proposals for the Globalization of DSLs. Thus, future work will investigate an open question: whether existing proposals are generic

enough to promote collaboration in the area?

6. Acknowledgements

Thanks to prof. Valdemar Vicente Graciano Neto by the insigths, we also thank CNPq for supporting this research under grants number 130337/2019-6 (first author). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 (second author), and to FAPERGS EDITAL FAPERGS 04/2019 - AUXÍLIO RECÉM DOUTOR-ARD.

References

[Aranega et al. 2012] Aranega, V., Etien, A., and Mosser, S. (2012). Using feature model to build model transformation chains. In *MODELS*, volume 7590 of *LNCS*, pages 562–578. Springer.

[Basciani et al. 2014] Basciani, F., Rocco, J. D., Ruscio, D. D., Salle, A. D., Iovino, L., and Pierantonio, A. (2014). Mdeforge: An extensible web-based modeling platform. CloudMDE 2014, pages 66–75.

[Basso et al. 2014] Basso, F. P., Oliveira, T. C., and Farias, K. (2014). Extending junit 4 with java annotations and reflection to test variant model transformation assets. In *29th Symposium On Applied Computing*, SAC'14, pages 1601–1608.

[Basso et al. 2013] Basso, F. P., Pillat, R. M., Oliveira, T. C., and Becker, L. B. (2013). Supporting large scale model transformation reuse. GPCE'13, pages 169–178.

[Batory et al. 2013a] Batory, D., Goncalves, R., Marker, B., and Siegmund, J. (2013a). Dark knowledge and graph grammars in automated software design. SLE'13, pages 1–18.

[Batory et al. 2013b] Batory, D., Latimer, E., and Azanza, M. (2013b). Teaching model driven engineering from a relational database perspective. MODELS'13, pages 121–137.

[Brambilla and Fraternali 2014] Brambilla, M. and Fraternali, P. (2014). Large-scale model-driven engineering of web user interaction: The webml and webratio experience. *Science of Computer Programming*, 89, Part B(0):71 – 87.

[Cheng et al. 2015] Cheng, B. H., Combemale, B., France, R. B., J'ez'equel, J.-M., and Rumpe, B. (2015). On the Globalization of Domain-Specific Languages. In *Globalizing Domain-Specific Languages*, volume 9400 of *LNCS*. Springer International Publishing.

[Combemale et al. 2015] Combemale, B., Cheng, B. H., France, R. B., Jézéquel, J.-M., and Rumpe, B. (2015). *Globalizing Domain-Specific Languages. International Dagstuhl Seminar*. Dagstuhl Castle, Germany.

[Combemale et al. 2014] Combemale, B., Deantoni, J., Baudry, B., France, R., Jézéquel, J.-M., and Gray, J. (2014). Globalizing modeling languages. *IEEE Computer, Institute of Electrical and Electronics Engineers*, 47(6):68–71.

[Costa et al. 2016] Costa, B., Pires, P. F., Delicato, F. C., and Merson, P. (2016). Evaluating REST architectures - approach, tooling and guidelines. *Journal of Systems and Software*, 112:156–180.

[Costa et al. 2013] Costa, V. O., Junior, J. M. B. O., and Murta, L. G. P. (2013). Semantic conflicts detection in model-driven engineering. In *International Conference on Software Engineering and Knowledge, 2013.*, pages 656–661.

[Couto et al. 2014] Couto, R., Ribeiro, A. N., and Campos, J. C. (2014). The modelery: A collaborative web based repository. In *ICCSA 2014*, volume 8584 of *LNCS*, pages 1–16. Springer.

[Cuadrado et al. 2014] Cuadrado, J. S., Guerra, E., and Lara, J. D. (2014). A component model for model transformations. *IEEE Transactions on Software Engineering*, 40(11):1042–1060.

[dos Santos Rocha et al. 2015] dos Santos Rocha, R., Fantinato, M., Thom, L. H., and Eler, M. M. (2015). Dynamic product line for business process management. *Business Proc. Manag. Journal*, 21(6):1224–1256.

[Elaasar and Neal 2013] Elaasar, M. and Neal, A. (2013). Integrating modeling tools in the development lifecycle with oslc: A case study. MODELS'13, pages 154–169.

[Etien et al. 2013] Etien, A., Muller, A., Legrand, T., and Paige, R. F. (2013). Localized model transformations for building large-scale transformations. *Software & Systems Modeling*, pages 1–25.

[France and Rumpe 2007] France, R. B. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. FOSE '07, pages 37–54.

[Fuggetta and Nitto 2014] Fuggetta, A. and Nitto, E. D. (2014). Software process. ICSE '14, pages 1–12.

[Guerra et al. 2010] Guerra, E., de Lara, J., Kolovos, D. S., Paige, R. F., and dos Santos, O. M. (2010). transml: A family of languages to model model transformations. MODELS 2010, pages 106–120.

[Hebig and Bendraou 2014] Hebig, R. and Bendraou, R. (2014). On the need to study the impact of model driven engineering on software processes. ICSSP 2014, pages 164–168.

[Hebig et al. 2013] Hebig, R., Giese, H., Stallmann, F., , and Seibel, A. (2013). On the complex nature of mde evolution. In *Proceedings of Model Driven Engineering Languages and Systems*, MODELS 2013, pages 436–453.

[Hong-min et al. 2010] Hong-min, R., Jin, L., and Jing-zhou, Z. (2010). Software asset repository open framework supporting customizable faceted classification. In *IEEE ICSESS*, pages 1–4.

[Horváth et al. 2006] Horváth, Á., Varró, D., and Varró, G. (2006). Automatic generation of platform-specific transformation. *Info-Communications-Technology*, LXI(7):40–45.

[Hutchinson et al. 2011] Hutchinson, J., Whittle, J., Rouncefield, M., and Kristoffersen, S. (2011). Empirical assessment of mde in industry. ICSE '11, pages 471–480.

[Iung et al. 2020] Iung, A., Carbonell, J., Marchezan, L., Rodrigues, E., Bernardino, M., Basso, F. P., and Medeiros, B. (2020). Systematic mapping study on domain-specific language development tools. *Empirical Software Engineering*, pages 1–45.

[Izquierdo and Cabot 2013] Izquierdo, J. L. C. and Cabot, J. (2013). Discovering implicit schemas in json data. In *Proceedings of the 13th International Conference on Web Engineering*, ICWE'13, pages 68–83.

[Jakumeit et al. 2014] Jakumeit, E., Buchwald, S., Wagelaar, D., Dan, L., bel Hegeds, Herrmannsdrfer, M., Horn, T., Kalnina, E., Krause, C., Lano, K., Lepper, M., Rensink, A., Rose, L., Wtzoldt, S., and Mazanek, S. (2014). A survey and comparison of transformation tools based on the transformation tool contest. *Science of Computer Programming*, 85, Part A(0):41 – 99. Special issue on Experimental Software Engineering in the Cloud(ESEiC).

[Johnson et al. 2012] Johnson, P., Ekstedt, M., and Jacobson, I. (2012). Where's the theory for software engineering? *Software, IEEE*, 29(5):96–96.

[Kelly and Tolvanen 2008] Kelly, S. and Tolvanen, J.-P. (2008). *Domain Specific Modeling: Enabling Full Code Generation*. IEEE Computer Society - John Wiley & Sons.

[Krueger 1992] Krueger, C. W. (1992). Software reuse. *ACM Comput. Surv.*, 24(2):131–183.

[Liebel et al. 2014] Liebel, G., Marko, N., Tichy, M., Leitner, A., and Hansson, J. (2014). Assessing the state-of-practice of model-based engineering in the embedded systems domain. In *Model-Driven Engineering Languages and Systems*, MODELS'14, pages 166–182.

[Lucrédio et al. 2012] Lucrédio, D., de Almeida, E. S., and Fortes, R. P. M. (2012). An investigation on the impact of MDE on software reuse. SBCARS, pages 101–110, Natal, Brazil.

[Maciel et al. 2013] Maciel, R. S. P., Gomes, R. A., Magalhães, A. P. F., da Silva, B. C., and Queiroz, J. P. B. (2013). Supporting model-driven development using a process-centered software engineering environment. *Autom. Softw. Eng.*, 20(3):427–461.

[Marinho et al. 2013] Marinho, F. G., Andrade, R. M. C., Werner, C., Viana, W., Maia, M. E. F., Rocha, L. S., Teixeira, E., Filho, J. B. F., Dantas, V. L. L., Lima, F., and de Aguiar, S. B. (2013). Mobiline: A nested software product line for the domain of mobile and context-aware applications. *Sci. Comput. Program.*, 78(12):2381–2398.

[Mascarenhas et al. 2013] Mascarenhas, A. F. M., Andrade, A., and Maciel, R. P. (2013). Mtp: Model transformation profile. In *Software Components, Architectures and Reuse (SBCARS), 2013 VII Brazilian Symposium on*, pages 109–118.

[Medeiros et al. 2015] Medeiros, A. L., Cavalcante, E., Batista, T., and Silva, E. (2015). Archspl-mdd: An adl-based model-driven strategy for automatic variability management. In *SBCARS*, pages 120–129, Belo Horizonte, Minas Gerais, Brazil.

[Monteiro et al. 2014] Monteiro, R., Assumpcao Pinel, R., Zimbrao, G., and Moreira de Souza, J. (2014). The mdarte experience: Organizational aspects acquired from a successful partnership between government and academia using model-driven development. In *MODELSWARD*, pages 575–586.

[Mussbacher et al. 2014] Mussbacher, G., Amyot, D., Breu, R., Bruel, J.-M., Cheng, B. H., Collet, P., Combemale, B., France, R. B., Heldal, R., Hill, J., Kienzle, J., Schöttle, M., Steimann, F., Stikkolorum, D., and Whittle, J. (2014). The relevance of model-driven engineering thirty years from now. In *Model-Driven Engineering Languages and Systems*, pages 183–200.

[Neto et al. 2014] Neto, V. V. G., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. In *ECSAW August 25 - 29, Vienna, Austria*, ECSAW'14, pages –.

[Neubauer et al. 2015] Neubauer, P., Bergmayr, A., Mayerhofer, T., Troya, J., and v, M. (2015). Xmltext: From xml schema to xtext. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering*, SLE 2015, pages 71–76.

[Pillat et al. 2015] Pillat, R. M., Oliveira, T. C., Alencar, P. S., and Cowan, D. D. (2015). BPMNt: A BPMN extension for specifying software process tailoring. *Information and Software Technology*, 57(0):95 – 115.

[Quercini and Reynaud 2013] Quercini, G. and Reynaud, C. (2013). Entity discovery and

annotation in tables. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 693–704.

[Reschenhofer and Matthes 2015] Reschenhofer, T. and Matthes, F. (2015). An empirical study on spreadsheet shortcomings from an information systems perspective. In *Business Information Systems - 18th International Conference, BIS 2015, Poznań, Poland, June 24-26, 2015, Proceedings*, pages 50–61.

[Sant'Anna et al. 2013] Sant'Anna, C., Garcia, A., Batista, T. V., and Rashid, A. (2013). Mastering crosscutting architectural decisions with aspects. *Softw., Pract. Exper.*, 43(3):305–332.

[Steinberg et al. 2008] Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2008). *EMF: Eclipse Modeling Framework (2nd Ed.)*. Addison-Wesley Professional.

[Vanhooff et al. 2006] Vanhooff, B., Baelen, S. V., Hovsepyan, A., Joosen, W., and Berbers, Y. (2006). Towards a transformation chain modeling language. SAMOS'06, pages 39–48.

[Whittle et al. 2015] Whittle, J., Hutchinson, J., Rouncefield, M., Håkan, B., and Rogardt, H. (2015). A taxonomy of tool-related issues affecting the adoption of model-driven engineering. *Software & Systems Modeling*, pages 1–19.

[Yie et al. 2012] Yie, A., Casallas, R., Deridder, D., and Wagelaar, D. (2012). Realizing model transformation chain interoperability. *Software & Systems Modeling*, 11(1):55–75.