Evaluation of Domain-Specific Languages in Practice

Proposal for a Habilitation at the Technische Universität München

Dr. Levi Lúcio

January 9, 2019

In this document we introduce our research topic of interest for pursuing a habilitation at theTechnische Universität München.

1. Document Structure

We will start by providing some background about the topic of Domain-Specific Languages in section 2. We then go in section 3 on to discuss the state of the art of the domain and to highlight the trends in the domain. In section 4 we introduce our topic of interest and in section 5 we lay out the context under which the research will take place.

2. Background

Computer languages are very often domain-specific, in the sense that they specialize in describing specifications, operational or not, of certain artifacts. Domain-Specific Languages (DSLs) are often presented in opposition ro General-Purpose Languages (GPLs) which are all-purpose generic programming languages such as Java or C. The premise of DSL construction is that such languages should explicitly address a domain of interest by making the concepts of the domain first-class citizens [25]. This can imply a trade-off in generality, meaning some DSLs may not be Turing-complete as such computational power may not be required in the domain of interest. The advantages put forward by the proponents of the DLS approach are increased productivity for language users and reduced maintenance costs of DSL programs. Additionally, DSLs are expected to lower the barrier of computer language usage to non-experts, thus allowing users from other domains to more easily specify the artifacts and computations they require [31].

DSLs have a rich history of contributions in different areas of knowledge. Notable examples of DSLs that left permanent marks in their domains are *lex* and *yacc* [20] for compiler construction, HTML [7] for the world-wide web, VHDL [3] for hardware specification, Excel for spreadsheets, Latex [27] for typesetting, SQL [8] for database querying or MATLAB [14] for technical computing.

During the past decade a number of DSL workbenches saw the light of day. The Eclipse Modelling Framework (EMF) had significant impact in the academia and became one of the most popular frameworks for DSL construction. More recently, the MPS workbench from Jetbrains has delivered a powerful DSL construction workbench with professional support and possibilities to design attractive GUIs that provide a very interesting complement to the notion of domain specificity. Professional DSL workbenches such as MetaEdit+ have repeatedly made strong cases for the industrial adoption of DSLs and have gathered a niche market in the domain.

During the past few years, DSLs have attracted considerable interest from the industry. The paradigm speaks directly to engineers who wish to build their systems ground-up and making as much use as possible of domain knowledge that is typically hard-earned. In this context, clients often regard DSLs as key-in-hand solutions. They encompass simplified means to express domain-specific computations while abstracting from accidental complexity linked to the software or the hardware running underneath.

Companies such as itemis, PROTOS or MetaCase have successfully developed business models around DSLs. They leverage their knowledge of modelling and DSL workbenches to deliver to customers key-in-hand software solutions. Such solutions have been developed for disparate domains such as the automotive, avionics, power tools, health, biology, among many others, and help either software developers or final users in achieving their tasks.

Despite these successes, the potential and real impact of DSLs in industry is still ill understood. A recent compelling report from Tolvanen and Kelly [42] states that while their company specialized in DSLs, MetaCase, can affirm with certainty that the gains in productivity of their clients range from 500% to 1000%. In the same article, they claim academic research in the domain thus not been able to validate this in general. They speculate this is due on the one hand to the quality of their tool MetaEdit+ and their industrial experience, and on the other hand to the poor quality of the academic tools for supporting DSL construction.

Anecdotally, at the PAINS workshop at MODELS 2018, an interesting discussion raged between a high-profile DSL proponent and a top-level BMW manager. While the DSL proponent insisted that the (MPS-based) technology was ready and could serve as a "silver bullet" of sorts, the BMW manager replied that the attempts of using DSLs at his company where "hit-and-miss" and that even when DSLs did prove successful, it was not understood why. A question from that same manager that was particularly thought-provoking was: "how do I build and abstraction and know that it is a good one?". A more general criticism to the DSL approach in general was: "quality standards are missing, how do I to judge or trust your processes of building DSLs while making decisions that will benefit my company?".

3. State of the Art

In 2000, a landmark white paper from the company MetaCase claimed that, using their commercial tool MetaEdit+, they developed DSLs that increased the productivity of the company 10-fold [32]. Such claims spawned a large amount of interest both in the industry and academia for DSLs.

Following such interest, authors such as Spinellis or Völter pursued the idea of proposing patterns for DSLs construction [38, 45], much in the light of what had been very successfully done by the "gang of four" for design patterns [13]. Such research culminated in the very well-cited work of Mernik, Heering and Sloane in 2005 [31] which builds on the work of Spinellis and aims at providing the DSL developer with the understanding of where and when design decisions matter in DSL development. Also, reports on industrial best practices such as the work from Wile [46] started to emerge at this time.

The raise in interest also brought upon a number of DSL workbenches both open-source and commercial such as the Eclipse Modelling Framework (EMF) for Eclipse [9] or Microsoft's DSL tools [33], GME [2] or AToM3 [1]. In particular, EMF was heavily adopted by academia for pursuing research on DSLs and together with GMF [10] soon became the platform of choice for academic DSL development.

While the enthusiasm around DSLs continued strong throughout the 00s and work on how to develop good languages kept on being published [44], other authors began to question the state of practice in general. In particular Kelly from MetaCase raises in [23] the point that DSL development often takes a standpoint of self sufficiency, ignoring pitfalls that mostly have to do with taking the domain for which the language is developed or its users too lightly. This was in the same year corroborated by work from Gabriel *et al.* [12] who reviewed a significant amount of literature related to the evaluation of DSLs. The authors claim that, while the benefits of DSLs in terms of increased productivity in well-defined domains were often put forward in the literature, little to no evidence to back up such claims was offered.

Having recognized the fact that DSLs were critically missing supporting evaluation that would back up the promises from the early days, authors such as Kolovos and Paige [25] proposed sets of characteristics that "good" DSLs should exhibit. Examples of such generic characteristics are conformity to the domain of choice, supportability by tools, integrability with other languages, longevity, simplicity, quality, scalability or usability. Several studies from the late 00's and beginning of the 10's [25] have concentrated on proposing means to quantitatively assess such and other characteristics [37, 16, 5, 21]. A notable study dating from this period from Kosar *et al.* [26] provides empirical evidence that programs written using DSLs are more easily comprehensible by programmers than programs written using GPLs, one of the original claims of the DSL community.

During this same period several contemporary DSL workbenches made their first appearance: the Meta Programming System (MPS)[1] from JetBrains [19], Sirius used by Obeo as a core technology [36], Xtext from itemis AG [17], among others [24]. Benefiting from the previous decade of development and professional software development teams, these tools exhibit a level of maturity that allows them to be used in industrial settings with success. For instance Sirius has been used to develop DSLs for the aerospace, transportation or energy industries [35]. MPS [19] has been used to develop commercial tools for branches such as software development, health, financial companies or automotive, among others. Xtext [17] is also used by large players such as Google, SAP or Bosch.

While the above seems to indicate that DSLs have successfully permeated the industry, authors from recent surveys remain very critical of the actual state of the art. Mernik [30] published a systematic mapping study in 2017 where he shows that the *maintenance* and *validation* phases of DSL development are grossly under-studied in the literature when compared to the *domain analysis*, *design* and *implementation* phases (see figure 1).

Equally, Tolvanen and Kelly who have been active DSL community since the late 90's have recently published a summary of their experiences with their commencial tool MetaEdit+ [41]. There, they state that although they can consistently claim a 5 to 10 times improvement in productivity for their customers, the DSL community has not been able to do the same in general. Note that this does not contradict the paragraph above on adoption by the industry, as: 1) adoption does not mean increased productivity, and 2) it is to be expected that if productivity increases do exist, corporate secrecy would be used to avoid losing competitive advantages. Tolvanen and Kelly do mention in [41] that the reason why the academic community has failed so far to prove or disprove most of the fundamental claims for advantages of DSLs is the poor quality of the overall tooling as well as DSLs used and developed by academic research. This notion is shared by us.

Nonetheless, modern DSL workbenches have provided the means for experimenting with the construction of DSLs in a systematic manner. Based on the work of Freire *et al.* [11] on formalizing software engineering experiments, Häser and his colleagues have proposed in [15] a framework in MPS that allow defining and conducting experiments for validating DSLs. Very recently work has started to emerge on concrete frameworks for the evaluation of usability during DSL development [4] and DSL maintenance [40] –

[1] Note that while the MPS project exists since 2003, its mainstream appearance dates to a decade later.

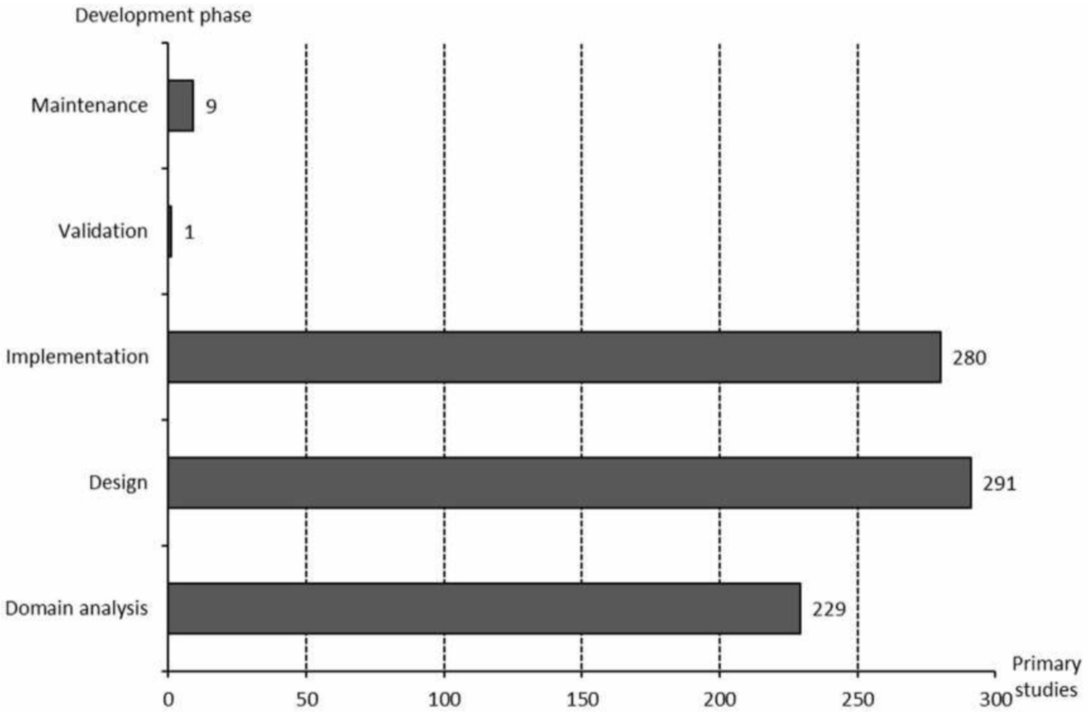

Figure 1: Distribution of 810 scientific contributions by phases of DSL construction (extracted from [30])

responding to the issues identified by Mernik [30] illustrated in figure 1.

Summary From the state of the art above, we can identify the following tends:

- For almost two decades professional DSL workbench builders have consistently reported case studies of application of DSLs to industrial problems, some of those use cases being highly successful. They do nonetheless mention that obtaining the data for validating the DSL after shipment is hard [43].
- During that same period academia as struggled to provide evidence that the promises of DSLs in terms of increased productivity hold. Only recently, partly due to the arrival of mature DSL workbenches produced by professional software, is actual quantification of properties of DSLs such as e.g. usability becoming possible. Experimental validation of DSLs based on established theory is still under-explored [30].
- The DSL workbenches developed by academia are, thus far, insufficient to scientifically validate the premises of the DSL-based software development. This is due partly to the poor quality of the tooling produced

by academia [41] and partly to very low access to real industrial use cases. Most of the surveys we describe above use as study material other academic papers, leading to starvation of real information "from the trenches".

- With the notable exceptions of the work of Tolvanen and Kelly and Vlter mentioned above in this section, the bridges between the academia and industry are brittle in the DSL domain.

4. Proposed Topic of Research

From the state of the art in section 3 it is clear to us that a substantial gap exists between academic research in DSLs and their usage in practice. For this reason, we formulate our main research question as:

How can we identify, measure and use characteristics of DSLs that lead to their sustained and sustainable adoption by the industry?

Such a research naturally leads to a number of sub-research questions, namely:

- What are the characteristics of a DSL that can be quantitatively assessed in order improve the chances of a real increase in productivity by the industry?
- What tooling is adequate to do such quantitative assessment?
- How can those characteristics be summarized in a way that is convincing for software-agnostic staff (domain experts or management), leading to the qualification of the DSL development process?

Regarding the tooling used to perform the quantitative assessment, we are firmly convinced that areas such as formal methods and machine learning have an important role to play and we have collected evidence of this through our past work (see section 5). We thus formulate a corollary research questions as: how can formal methods and machine learning help in the qualification and production phases of a DSL?

Note that our mid-to-longterm goal is to inseminate the academic community with the practical experiences and struggles of making DSLs used practice, leading to the theoretical establishment of the domain.

5. Context for Development of the Research

We have significant experience in the design and implementation of DSLs for areas such as software testing [28], model transformations [6], verification [34, 22], process management [29] and software refinement [39]. Such languages were also developed over a variety of DSL workbenches, for instance the Eclipse EMP framework, MPS or AToM3. Some of the DSLs were developed for academic purposes, while other were or are being developed with industrial partners such as Diehl Aerospace, Airbus and Rolls-Royce.

The role that fortiss itself plays and will play on this research is not to be understated: at fortiss we have built and currently have the kind of access to projects and industrial and academic partners that occurs very seldom in academia.

Not to be understated also is the existing body of research on the topic we are proposing. Although the gap between academia and industry is clear and has been repeatedly stated in the literature we have reviewed in section 3, exciting results on validating DSLs such as the ones recently presented by Lara *et al.* [18], Freire *et al.* [11], Haser *et al.* [15] or Barisic *et al.* [5] are starting points for this research. Also, systematic work by Mernik [31, 26, 30] on assessing the state of the art provides an excellent starting point for this research, as well as the clear-cut reports from the industry from Kelly and Tolvanen [42, 23, 37, 24, 41, 43] and Völter [44].

!!!something on FM and ML here!!!

In the annexes of this document we provide detailed information about the context in which our research will be developed:

- Synergies with groups at fortiss (appendix A)
- Completed and running projects at fortiss (appendix B)
- Network (appendix C)
- Objectives for the first year of research (appendix D)

Appendix A Synergies with groups at fortiss

Domain-Specific Languages are a cross-cutting theme at fortiss. All competence fields at fortiss deal with DSLs in one form or the other, even or despite doing it unknowingly.

- The obvious synergy of our research topic is with the Model-Based Software engineering (MbSE) group: MbSE has a model-centric view on systems and software engineering and builds languages using model-driven techniques. Its main current theme is design-space exploration and the modelling of embedded systems, but other themes such as requirements engineering, formal methods or security are also being pursued. DSLs can be of great assistance to MbSE by bringing systematic notions of language construction and evaluation to the development table. A collaboration that is already ongoing is the exploration of machine learning (in the context of the MAGNET project) to deliver support to users of AF3.
- Common work is also ongoing with the Autonomous Systems (AS) group, in the form of the FaktorBUILD project proposal. Some of the work of AS relies on probabilistic programming to model situations where incomplete knowledge must be reliably used to make decisions in real-time. Here, we help in constructing DSLs at an adequate level of abstraction such that the IDE provided to users of factor graphs can leverage good abstractions, static analyses and formal methods to increase the productivity of factor graph programmers.
- The collaboration with the Industry 4.0 (i4) group has been so far very successful, having lead to the implementation of a tool providing DSLs to express industrial capabilities. Such capabilities, or skills, can be subsequently be matched to automatically automatically synthesize controllers for industrial machines. Upcoming work for 2019 in the area will aim at connecting the completed DSL-based tool to AutomationML and 4Diac in order to connect the work with outside formats while providing simulation capabilities.

Appendix B Completed and Running Projects at fortiss

- IETS3
  - Consortium: fortiss, itemis, ZF, Diehl aerospace
  - Running time, funding and personnel: 2 years / 300K Euro / 3 people
- CBMD (as project leader)

  - Consortium: fortiss, PROTOS, SQMi, University of Augsburg
  - Running time, funding and personnel: 2 years / 190K Euro / 2 people
- MAGIC (as project leader)
  - Consortium: fortiss, University of Montral
  - Running time and funding: 2 years / 10K Euro + 40K Euro (Eigenforshungsgeld) / 3 people
- MAGNET (as project leader)
  - Consortium: fortiss
  - Running time, funding and personnel: 6 months / 70K Euro / 8 people Eigenforshungsgeld
- ARTEMIS (as project leader)
  - Consortium: fortiss, Airbus
  - Running time, funding and personnel: 6 months / 75KEuro / 2 people (Levi + HiWi)
- BaSys4.0 (as software developer for the “industrial skills” theme)

Appendix C Network

Here I will describe the main currently active research and industrial connections (others exist that may be reactivated at need):

Inside fortiss:

- HCE (Yuanting Liu and team, project MAGNET)
- SD (Tahira Iqbal and Parisa Elahidoost, project MAGNET and requirements engineering)
- i4 (project MAGIC, networking with University of Montreal)
- AS (Dhiraj Gulati and Vincent Aravantinos, on BaSys and Factor-BUILD)

Academic:

- University of Montréal, Canada (project MAGIC)
- University of Namur, Belgium (tutorial and paper on machine learning and formal verification)
- University of Antwerp, Belgium (proposal for H2020)
- LMU, Germany (with Prof. Dirk Beyer in the context of the Faktor-BUILD proposal)

- University of Lübeck, Germany (with Prof. Philipp Rostalski in the context of the FactorGraph proposal)
- TU Wien (with Manuel Wimmer in the context of model transformations)

Industrial:

- PROTOS (KMU) (in the context of the CBMD project)
- Rolls-Royce (in the context of the EARS-related work)
- Siemens (in the context of the FaktorBUILD proposal)
- Festo (in the context of the BaSys project)
- ABB (in the context of the BaSys project)

Appendix D Objectives for the first year of research

- Establish a set of criteria for the quality of DSLs in practice. In particular, we are interested in understanding which measurable criteria can be used to facilitate the adoption of DSLs in the industry.
- Evaluate the usage of machine learning in the context of requirements engineering and in general as a means to aid in the construction and operation of good and reliable DSLs.
- Establish an ongoing collaboration with Prof. Dirk Beyer from the LMU in Munich. Prof. Beyer is an expert in formal methods and is part of the consortium for the FaktorBUILD project. He is, in particular, very enthusiastic about applying his CPAchecker C model checker to examples coming from Airbus Defense in the context of the ARTEMIS project. Publishing with Prof. Beyer will also be pursued.
- Evaluate the prototype developed for the MAGNET project in the real-world scenario of tutorials of AF3. Calibrate the suggestions provided by the machine learning algorithm in function of such an evaluation.
- Write one or more articles on the results of the MAGNET project.
- Evaluate the usage of machine learning in the context of requirements engineering and in general as a means to aid in the construction and operation of good and reliable DSLs.
- Establish a set of criteria for the quality of DSLs in practice. In particular, I am interested in understanding which measurable criteria can be used to facilitate the adoption of DSLs in the industry.
- Write one or more articles on the results of the skills (Fähigkeiten) work-package of the BaSys project.

- Continue research on the topic of Process-Aware model driven development environments to be implemented in AF3. Complete the ongoing journal paper on process-aware model-driven development environments.
- Help Sudeep Kanav in establishing his PhD research topic by publishing results on compositional model checking at top venues. Continue Sudeep's scientific training.
- Provide Tatiana Chuprina with the right tools such that she can finish her thesis proposal in the area of requirements engineering.

D.1 Technical:

- Build a prototype tool in MPS to generate LTL from EARS requirements and model-check C code written for those requirements.
- Build a release-ready prototype of a recommender system for AutoFOCUS3 that can be effectively used in tutorials on the tool given to industry or academia.
- Complete the development of the skill-matching and controller synthesis prototype for BaSys. The prototype will semi-automatically generate a controller for a robotic arm in 4Diac. A demonstrator of the complete chain from skill definition down to robot arm movement will be built based on virtual robotic arm simulator provided by Festo AG.
- Integrate compositional model checking in the eTrice tool, such that it can be used in production by PROTOS.

References

[1] AToM3: A Tool for Multi-Formalism and Meta-Modelling. http://atom3.cs.mcgill.ca/.

[2] GME: Generic Modeling Environment. http://w3.isis.vanderbilt.edu/projects/gme/.

[3] Peter J. Ashenden. *The designer's guide to VHDL, 2nd Edition*. The Morgan Kaufmann series in systems on silicon. Kaufmann, 2002.

[4] Ankica Barisic, Vasco Amaral, and Miguel Goulão. Usability driven DSL development with USE-ME. *Computer Languages, Systems & Structures*, 51:118–157, 2018.

[5] Ankica Barišić, Pedro Monteiro, Vasco Amaral, Miguel Goulão, and Miguel Monteiro. Patterns for evaluating usability of domain-specific languages. In *Proceedings of the 19th Conference on Pattern Languages of Programs*, PLoP '12, pages 14:1–14:34, USA, 2012. The Hillside Group.

[6] Bruno Barroca, Levi Lucio, Vasco Amaral, Roberto Félix, and Vasco Sousa. Dsltrans: A turing incomplete transformation language. In *Software Language Engineering - Third International Conference, SLE 2010, Eindhoven, The Netherlands, October 12-13, 2010, Revised Selected Papers*, pages 296–305, 2010.

[7] Tim Berners-Lee, Roy T. Fielding, and Henrik Frystyk Nielsen. Hypertext transfer protocol - HTTP/1.0. *RFC*, 1945:1–60, 1996.

[8] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.

[9] Eclipse Foundation. Eclipse Modeling Framework. https://www.eclipse.org/modeling/emf/.

[10] Eclipse Foundation. GMF: Graphic Modeling Framework. https://www.eclipse.org/modeling/gmp/.

[11] Marília Aranha Freire, Uirá Kulesza, Eduardo Aranha, Andreas Jedlitschka, Edmilson Campos Neto, Silvia Teresita Acuña, and Marta Gómez. An empirical study to evaluate a domain specific language for formalizing software engineering experiments. In *The 26th International Conference on Software Engineering and Knowledge Engineering, Hyatt Regency, Vancouver, BC, Canada, July 1-3, 2013.*, pages 250–255, 2014.

[12] Pedro Gabriel, Miguel Goulão, and Vasco Amaral. Do software languages engineers evaluate their languages? *CoRR*, abs/1109.6794, 2011.

[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[14] Amos Gilat. *MATLAB: An Introduction with Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2007.

[15] Florian Häser, Michael Felderer, and Ruth Breu. An integrated tool environment for experimentation in domain specific language engineering. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, EASE '16, pages 20:1–20:5, New York, NY, USA, 2016. ACM.

[16] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Domain-specific languages in practice: A user study on the success factors. In Andy Schürr and Bran Selic, editors, *Model Driven Engineering Languages and Systems*, pages 423–437, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[17] itemis AG and TypeFox. Sirius. https://www.eclipse.org/Xtext/.

[18] Javier Luis Cánovas Izquierdo, Jordi Cabot, Jesús J. López-Fernández, Jesús Sánchez Cuadrado, Esther Guerra, and Juan de Lara. Engaging end-users in the collaborative development of domain-specific modelling

languages. In Yuhua Luo, editor, *Cooperative Design, Visualization, and Engineering*, pages 101–110, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[19] JetBrains. Meta Programming System. https://www.jetbrains.com/mps/.

[20] Stephen C. Johnson. Language development tools on the unix system. *IEEE Computer*, 13(8):16–21, 1980.

[21] Gökhan Kahraman and Semih Bilgen. A framework for qualitative assessment of domain-specific languages. *Software & Systems Modeling*, 14(4):1505–1526, Oct 2015.

[22] Sudeep Kanav, Levi Lucio, Christian Hilden, and Thomas Schütz. Design and Runtime Verification Side-by-Side in eTrice. Submitted to NFM 2019.

[23] S. Kelly and R. Pohjonen. Worst practices for domain-specific modeling. *IEEE Software*, 26(4):22–29, July 2009.

[24] Steven Kelly. Empirical comparison of language workbenches. In *Proceedings of the 2013 ACM Workshop on Domain-specific Modeling*, DSM '13, pages 33–38, New York, NY, USA, 2013. ACM.

[25] Dimitrios S. Kolovos, Richard F. Paige, Tim Kelly, and Fiona A. C. Polack. Requirements for domain-specific languages. 2006.

[26] Tomaž Kosar, Marjan Mernik, and Jeffrey C. Carver. Program comprehension of domain-specific and general-purpose languages: comparison using a family of experiments. *Empirical Software Engineering*, 17(3):276–304, Jun 2012.

[27] Leslie Lamport. *Latex: A Document Preparation System*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[28] Levi Lucio. *SATEL A test intention language for object-oriented specifications of reactive systems*. PhD thesis, 05/28 2009. ID: unige:2498.

[29] Levi Lúcio, Saad bin Abid, Salman Rahman, Vincent Aravantinos, Ralf Kuestner, and Eduard Harwardt. Process-aware model-driven development environments. In *Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA, September, 17, 2017.*, pages 405–411, 2017.

[30] Marjan Mernik. Domain-specific languages: A systematic mapping study. In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann

Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM 2017: Theory and Practice of Computer Science*, pages 464–472, Cham, 2017. Springer International Publishing.

[31] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, 2005.

[32] MetaCase. MetaEdit+ revolutionized the way Nokia develops mobile phone software.

[33] Microsoft. Microsoft DSL Tools. https://tinyurl.com/yb934lvz.

[34] Bentley James Oakes, Javier Troya, Levi Lucio, and Manuel Wimmer. Fully verifying transformation contracts for declarative ATL. In *18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS 2015, Ottawa, ON, Canada, September 30 - October 2, 2015*, pages 256–265, 2015.

[35] Obeo. Obeo – Sirius webpage. https://www.obeodesigner.com/en/product/sirius.

[36] Obeo. Sirius. https://www.obeo.fr/en/products/eclipse-sirius.

[37] Juha Kärnä Polar Electro Professorintie, Juha. Karna, and Steven Kelly MetaCase Ylistönmäentie. Evaluating the use of domain-specific modeling in practice. 2009.

[38] Diomidis Spinellis. Notable design patterns for domain-specific languages. *Journal of Systems and Software*, 56(1):91–99, 2001.

[39] Eugene Syriani, Vasco Sousa, and Levi Lúcio. Structure and behavior preserving statecharts refinements. *Sci. Comput. Program.*, 170:45–79, 2019.

[40] J. Thanhofer-Pilisch, A. Lang, M. Vierhauser, and R. Rabiser. A systematic mapping study on dsl evolution. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 149–156, Aug 2017.

[41] J. Tolvanen and S. Kelly. Model-driven development challenges and solutions: Experiences with domain-specific modelling in industry. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 711–719, Feb 2016.

[42] Juha-Pekka Tolvanen and Steven Kelly. Model-driven development challenges and solutions - experiences with domain-specific modelling in industry. In *MODELSWARD 2016 - Proceedings of the 4rd International Conference on Model-Driven Engineering and Software Development, Rome, Italy, 19-21 February, 2016.*, pages 711–719, 2016.

[43] Juha-Pekka Tolvanen and Steven Kelly. Effort used to create domain-specific modeling languages. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and*

Systems, MODELS 2018, Copenhagen, Denmark, October 14-19, 2018, pages 235–244, 2018.

[44] Markus Völter. Best practices for dsls and model- driven development. *Journal of Object Technologies*, 8(6), 2009.

[45] Markus Völter and Jorn Bettin. Patterns for model-driven software-development. In *Proceedings of the 9th European Conference on Pattern Languages of Programms (EuroPLoP '2004), Irsee, Germany, July 7-11, 2004.*, pages 525–560, 2004.

[46] David S. Wile. Lessons learned from real DSL experiments. In *36th Hawaii International Conference on System Sciences (HICSS-36 2003), CD-ROM / Abstracts Proceedings, January 6-9, 2003, Big Island, HI, USA*, page 325, 2003.