Group Proposal

Dr. Levi Lúcio

January 8, 2019

Abstract

We propose the creation of new nachwuchs-Kompetezfeld at fortiss in the area of Domain Specific Languages (DSLs).

1 Name

Intelligent Domain-Specific Systems (IDS2)

2 Mission

Increasingly build better DSLs to tackle industrial problems by: 1) applying them to implement and support technology transfer projects at fortiss, and 2) evaluating their practical impact.

3 Need

Computer languages are pervasive in software and systems engineering, to the point that software engineers become unaware that the choice of the language set to use plays a major role in projects' success.

The IDS2 group aims at making DSLs a primary theme of concern at fortiss. Many projects implemented at fortiss have a strong language development component. As as such the group will be flexible in the range of projects it will acquire and develop. The cross-section between IDS2 and other groups at fortiss is wide: learning to design better languages will empower ourselves and other scientists at fortiss to better transfer their results into industry. Additionally, DSLs are a powerful tool to add robustness to fortiss' contributions in software engineering.

Better technology transfer implies economical benefits for our industrial partners and ultimately more business for fortiss itself.

4 Context

The history of software engineering is one of inventing increasing levels of abstraction as a means to cope with the increasing complexity of software systems. High level languages such as C, Pascal or FORTRAN introduced

programmers to structured code, C++, Java or C# added the notion of *object* as a means to encapsulate code. UML, together with design patterns, gave software developers principles of architecture and design.

Software modelling is a broad term that proposes many different approaches to the different stages of software development (requirements, design, coding and verification/validation), making use of formal, semi-formal and informal languages. Based on the principles of software modelling, several tools such as Enterprise Architect or MATLAB Simulink have had significant impact in the industry.

One of the currently most promising approaches to again raise the level of abstraction at which software is built are Domain-Specific Languages (DSLs). DSLs propose encapsulating the entities of the domain on which computations are to be performed through languages that encapsulate and manipulate the concepts of that same domain. Here, the Turing machine stops being the absolute reference – rather, it becomes important to adjust the computational power and the expressiveness of the language to the task at hand. DSLs thus concentrate on providing adequate abstractions to describe computations for specific domains. They abstract from the machine- architecture- and mathematical-oriented properties of Turing-machine inspired languages and favor abstractions closer to the human understanding of the domain of interest.

Industry is constantly on the lookout for automation that can provide an edge regarding their competitors. This is particularly true in Bavaria, where large automotive, aerospace and software companies exist and compete in a globalised market.

While it is expected and understood that new techniques and tools will provide such competitive edge, even large companies with dedicated research departments cannot perform their own research in-house due to time or expertise constraints.

The IDS2 group will work hand-in-hand with industry to provide solutions to software development problems that may be solved using domain-specific languages. This is a proven business model: for example itemis AG or PROTOS GmbH are currently using it with success. Both companies are expert in their own modelling frameworks (MPS and eTrice respectively) and provide domain-specific solutions based on those tools. Their areas of expertise of their clients range from automotive, to aerospace, health, heavy machinery or manufacturers of power tools.

Given fortiss' technology transfer mission, IDS2 does not intend to specialize in one specific technological platform but rather to build expertise around the the construction and evaluation of Domain-Specific Languages. In particular, we intend on leveraging and experimenting with machine learning as well as with practical formal methods in order to consistently provide innovative and cutting-edge solutions.

When implementing projects with the industry we will follow a formula that has proven successful in the past: for each project we intend on building a prototype focused on solving one specific problem, delivering the solution in a fashion such that the advantages of the research idea are clearly transmitted to the industry. In practice, this means that our focus will not be on building an encompassing framework where all solutions may be plugged in, but rather to impress our clients with clean and

clear-cut solutions to well-specified problems. Technology will be used opportunistically and towards specific goals.

We strongly believe that such an approach will be impactful, as it is well understood that the industry does not use prototypes built by fortiss directly – they rather re-implement the research idea for their proprietary toolchains when the research is deemed to be useful. Equally important, such an approach will allow us to build a range of impressive prototypes that will not only serve as a presentation portfolio for the group, but will also allow us to attract more business for fortiss.

5 Motivation

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 [28] 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?".

IDS2 will be in a privileged position to find answers to the questions above. The proposed main members of the group have considerable experience with DSL workbenches, modelling and verification, both through their studies and their professional accomplishments. Additionally, we rely on a national and international academic and industrial network that is stable and currently growing.

By developing DSLs for the industry as the main business model, we have and will be exposed to the difficulties in the conception of DSLs, in their adoption and evaluation in practice.

Our main goal is to build a conceptual and systematic framework to build DSLs and the right level of abstraction for the task at hand, as well as evaluating their impact in the productivity of the industry and discovering best practices.

RQ1: What is the quality of an abstraction in a DSL?
RQ2: How do formal methods and machine learning help out?

Our mid-to-longterm goal is to inseminate the community with the practical experiences and struggles of applying DSLs in practice, leading to the theoretical establishment of the domain. fortiss is a very privileged context to do this, as we have the kind of access to projects and partners that occurs very seldom in academia.

6 State of the Art

DSLs have a rich history of contributions in many areas. Notable examples of DSLs that left permanent marks in their domains are *lex* and *yacc* [13] for compiler construction, HTML [5] for the world-wide web or VHDL [3] for hardware specification.

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 [21]. 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 [26, 30], much in the light of what had been very successfully done by the "gang of four" for design patterns [9]. Such research culminated in the very well-cited work of Mernik, Heering and Sloane in 2005 [20] 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 [31] 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 [6] or Microsoft's DSL tools [22], GME [2] or AToM3 [1]. In particular, EMF was heavily adopted by academia for pursuing research on DSLs and together with GMF [7] 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 [29], other authors began to question the state of practice in general. In particular Kelly from MetaCase raises in [15] 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.* [8] 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 [17] 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 [17] have concentrated on proposing means to quantitatively assess such and other characteristics [25, 10, 4, 14]. A notable study dating from this period from Kosar *et al.* [18] 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 [12], Sirius used by Obeo as a core technology [24], Xtext from itemis AG [11], among others [16]. 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 [23]. MPS [12] has been used to develop commercial tools for branches such as software development, health, financial companies or automotive, among others. Xtext [11] 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

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

actual state of the art. Mernik [19] 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 [**?**]).

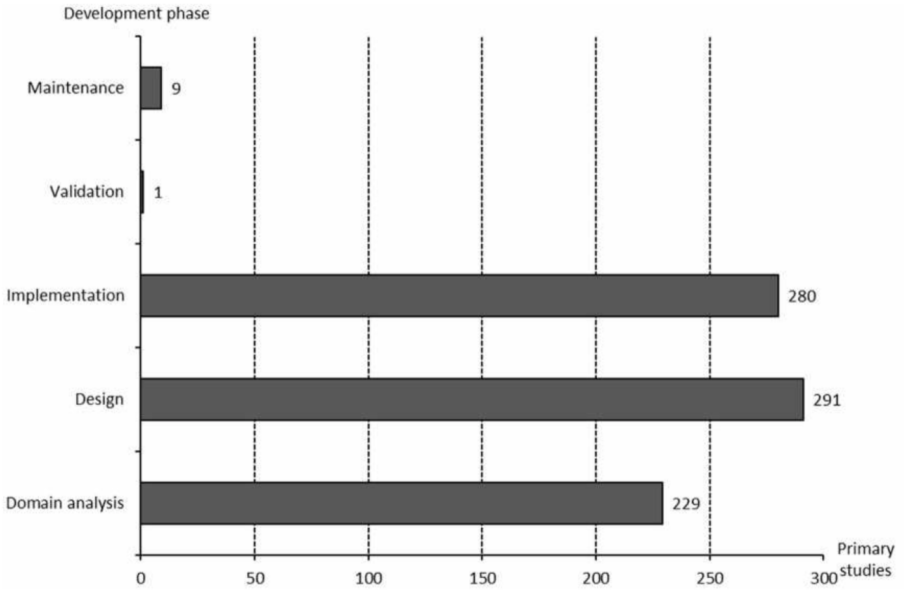

Figure 1: Distribution of 810 scientific contributions by phases of DSL construction

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+ [27]. 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 even if such productivity increases do exist, corporate secrecy would avoid losing competitive advantages. Tolvanen and Kelly do mention in [27] 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 used in academic research. This claimed is shared by us.

7 Proposed Group Members

- Dr. Levi Lúcio (group lead)
- Dr. Saad Bin Abid (topic leader for process modelling and machine

learning)

- Sudeep Kanav (topic leader for formal methods in practice)
- Ananya Misra (HiWi)
- Vishal Mahajan (HiWi)

Dr. Saad Bin Abid and Sudeep Kanav have expressed keen interest on becoming part of the IDS2 group. They are nonetheless members of MbSE and are expected to work part-time for IDS2 where synergies exist. Depending on acquired funds in 2019 and personal and group interests the percentage of time they work for each group might be subjected to adjustments.

8 Synergies with other 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 IDS2 is with the 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. ID2S 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 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, IDS2 may 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 i4 group has been so far very successful, having lead to the implementation (together with Vincent Aravantinos) 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 A Context

The proposal for the formation of this group stems from three years of work at fortiss developing DSLs in the context of several projects. The three main members of the foreseen group (Dr. Levi Lúcio, Dr. Saad Bin abid and Sudeep Kanaav) all have extensive experience working together both for the IETS3 the CBMD and the MAGNET projects.

A.1 Completed Projects

- IETS3
  - Consortium: fortiss, itemis, ZF, Diehl aerospace
  - Running time, funding and personnel: 2 years / 300K Euro / 3 people

A.2 Running Projects

- 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)

A.3 Projects in the Acquisition Pipeline

- FaktorBUILD
  - Consortium: fortiss, University of Lübeck, Siemens, LMU, ?
  - Running time, funding and personnel: 3 years / 400K Euro / 3 people (Levi + Dhiraj + HiWi)
- Follow-up for CBMD
  - Consortium: fortiss, PROTOS, ?
- Follow-up for ARTEMIS

  - Consortium: fortiss, Airbus
- H2020, project on "agility in model-based software engineering"
  - Consortium: fortiss, University of Antwerp, University Nova de Lisboa, TU Wien, Hasso-Platner Institut, Telecom-Paristech, Unit Bilisim Technologies

Appendix B 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 FactorBUILD)

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 FaktorBUILD 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 C Plan for the first year of activity

In the first year of activity we intend on solidifying the group by building on existing results and enlarging the scope of our national and international network, both at the academic and industrial levels. In particular, we aim at achieving the following goals:

C.1 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.

C.2 Project acquisition:

- Build on our existing collaboration with Airbus defense by delivering excellent results for the ARTEMIS project and acquiring a second project in the context of model checking C code based on EARS requirements.
- Acquire one or more national and/or international projects. This work will be based on current acquisition efforts for a BMBF and an H2020 projects (as described in section A.3).
- Acquire a project on the continuation fn CBMD on the compositional verification of models of embedded software. This is well underway and desired by PROTOS GmbH (as described in sections A.2 and A.3).
- Acquire a project with an industrial partner (PROTOS is a promising option) on applying the results of MAGNET Eigenforshungs project in practice.
- Work in the direction of establishing a funded consortium with Professors Eugene Syriani and Michalis Famelis from the University of Montréal. A collaboration with those researchers is already in place in the context of the MAGIC project (as described in A.2) with the goal of acquiring a large Canadian-Germany partnership project[2].

C.3 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.

C.4 Organisation:

- Contribute to and aid in the delivery of tutorials for AF3.
- Contribute to the organization of MODELS 2019 in Munich.
- Conduct interviews leading to the the hiring of post-docs, PhD students, Msc students or HiWis.

[2] http://www.nserc-crsng.gc.ca/International-Internationale/CanadaGermany_call-Appel_CanadaAllemagne_fra.asp

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 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.

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

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

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

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

[9] 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.

[10] 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.

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

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

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

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

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

[16] 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.

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

[18] 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.

[19] 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.

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

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

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

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

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

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

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

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.