



Systematic mapping study on domain-specific language development tools

Aníbal Iung¹  · João Carbonell¹ · Luciano Marchezan¹ · Elder Rodrigues¹ · Maicon Bernardino¹ · Fabio Paulo Basso¹ · Bruno Medeiros¹

Published online: 28 August 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Domain-specific languages (DSL) are programming or modeling languages devoted to a given application domain. There are many tools used to support the implementation of a DSL, making hard the decision-making process for one or another. In this sense, identifying and mapping their features is relevant for decision-making by academic and industrial initiative on DSL development. Objective: The goal of this work is to identify and map the tools, Language Workbenches (LW), or frameworks that were proposed to develop DSLs discussed and referenced in publications between 2012 and 2019. Method: A Systematic Mapping Study (SMS) of the literature scoping tools for DSL development. Results: We identified 59 tools, including 9 under a commercial license and 41 with non-commercial licenses, and analyzed their features from 230 papers. Conclusion: There is a substantial amount of tools that cover a large number of features. Furthermore, we observed that usually, the developer adopts one type of notation to implement the DSL: textual or graphical. We also discuss research gaps, such as a lack of tools that allow meta-meta model transformations and that support modeling tools interoperability.

Keywords DSL · Domain-specific language · Language workbench · Model driven engineering · DSL-supporting tools · Systematic mapping study · Systematic review

1 Introduction

The implementation of computer systems for a specific domain is becoming more complex, integrating many areas of knowledge. For example, the codification of modern Web systems demands several implementation concerns such as usability, security, persistence, and business rules. To assist the development of these different concerns independently of the coding technology, software engineers are adopting the development of Domain-Specific

Communicated by: Daniel Amyot

✉ Aníbal Iung
netoiung@gmail.com

Extended author information available on the last page of the article.

Languages (DSL) (Fowler 2010), which are one of the most widely used approaches for modeling and coding features from a given domain. DSLs have been developed for a large variety of domains including performance tests development (Bernardino et al. 2016), machine deep learning (Zhao and Huang 2018), object-oriented domain-driven design (Le et al. 2018), automatic creation of timetables for schools (Ribić et al. 2018), and database language for communication and query (Fowler 2010).

In this context, to design all the elements that capture software requirements, the development of graphical and/or textual DSLs is required to offer end-users the best front-end. A DSL allows a higher level of abstraction than that of programming/modeling languages commonly used in Software Engineering daily practice. For example, the current practice includes developers implementing code with General Purpose Languages (GPL) (Fowler 2010), such as C and Python, using code editors available in an Integrated Development Environment (IDE). Meanwhile, approaches built on platform independence concepts change this development practice, by introducing a higher-level language agnostic to C and Python implementations.

To reach this independence from platforms, *i.e.* through a software development project agnostic from implementation details, the creation of a new DSL may be needed to capture domain concepts. However, creating a new DSL is not a trivial task and it demands the use of many DSL development tools. These tools support the process of developing and maintaining DSLs, ensuring their consistency, evolution, and maintainability. Moreover, the tools may include code generators, validators, and model checkers, as well as lexical, semantic, and syntactic analyzers. Normally, these integrated tools are part of toolboxes known as Languages Workbenches (LWs) (Fowler 2010), and their usage processes are usually expressed as toolchains (Jakumeit et al. 2014).

To automate some software development process tasks (Whittle et al. 2015), each tool must be carefully selected before instantiating a toolchain (Basso et al. 2017). This is complex since several tools are helping in only one aspect of the DSL development (Mussbacher et al. 2014), whereas others help in more than one. This diversity is good for the practice, but it also increases the difficulty of understanding which features are necessary to start a process assisted by DSLs and to identify which complementary tools can support language construction in specific contexts (Liebel et al. 2014).

Although the literature of the area is full of references for DSL development tools, it misses a recent map of their quality attributes. In this sense, a mapping study surveying features and information useful for a technical decision about these tools could help software engineers to select the best option for a specific context on software development. Articles contributing to decision making in toolchains are welcome (Liebel et al. 2014), since the lack of this mapping hampers the selection of appropriate supporting meta-generators, which is a key activity in building domain-specific tools supporting some Software Engineering tasks (Jakumeit et al. 2014). Currently, this selection requires time to investigate many important elements affecting decision-making processes, including technicalities such as which tools meet the requirements and descriptive data for understanding their operation. For example, tools like MetaEdit+ (Kelly et al. 1996) have a high level of adoption and support the whole DSL development process in technical aspects. However, MetaEdit+ has a commercial license that may be incompatible with some business scenarios. Therefore, in some contexts, developers may need to look for free and/or non-commercial alternatives.

To characterize these tools for further decision-making, studies discussing the DSL development process and its application are essential. Current characterization studies (Pérez et al. 2013; Arkin and Tekinerdogan 2014), despite referencing which tools are

used in the DSL creation, do not provide enough details such as technical features. Other studies focus on systematic mapping to the DSL field (do Nascimento et al. 2012; Erdweg et al. 2013; Erdweg et al. 2015; Kosar et al. 2016; Méndez-Acuna et al. 2016), however, these reviews are not mapping tools that support some DSLs and DSMLs development process life-cycle stages and their features *e.g.*: highlighting and error marking. Due to these limitations, the technical decision for DSL building remains a hard task.

In this paper, we report a Systematic Mapping Study (SMS) (Kitchenham et al. 2011) presenting a panoramic view of supporting tools for DSL development. Also, we characterized for which domains DSLs are developed, providing a current overview of the research field of DSL development tools, and also identify research opportunities and gaps. This overview includes an analysis of cross-application domains where DSLs have been proposed and the adopted practices for building DSLs. To the best of our knowledge, our SMS characterizes technicalities used to build DSLs, including structural features and also business elements such as their license types and applicability. Thus, this work presents a significant contribution to this research field.

This article is organized as follows. Section 2 introduces the terms and concepts used through this paper. Section 3 depicts the SMS planning. Section 4 reports on how this SMS on DSL development tools was conducted. Section 5 presents the data collected to answer the research questions and Section 6 discusses research gaps. Section 7 presents the threats to validity and, finally, Section 8 concludes the work.

2 Background

An SMS is a secondary study aiming to map out a research area and to classify the most relevant contributions from the primary studies. When conducting an SMS, researchers may utilize search and data extraction methodologies, similar to Systematic Literature Review (SLR). However, different from SLRs, an SMS reviews a broader area and topics (Kitchenham et al. 2011). Also, the SMS results are more related to simpler categorizations and statistics, focusing on a more shallower assessment. In this context, we introduce the main technical terms founding this SMS in the following sections.

2.1 Domain-specific languages (DSL)

DSLs, also known as Little Languages, Small Languages, Special Purpose Languages, or Domain-Specific Modeling Languages (DSML), is defined as “computer programming languages with limited expressiveness that have focus on a specific domain” (Mernik et al. 2005).

To develop DSLs, tools called Language Workbenches (LWs) are used. According to Wachsmuth et al. (2014), LWs provide “high-level mechanisms for implementing programming languages and make affordable the development of new languages”. LWs facilitate not only the definition of semantic and syntactic analyzers, but also provide support to create a custom edition environment for the language, as well as other language-based tools such as model debuggers (Wu et al. 2008), model compilers (Henriques et al. 2002) and model test engines.

LWs may be used as standalone versions or in combination with frameworks. According to Johnson (1997), “a framework is a skeleton of an application that is able to be customized by a developer”. According to the Meta-Object Facility (MOF) specification (OMG 2019), “a metamodel is a model that consists of statements about models, a metamodel is also a

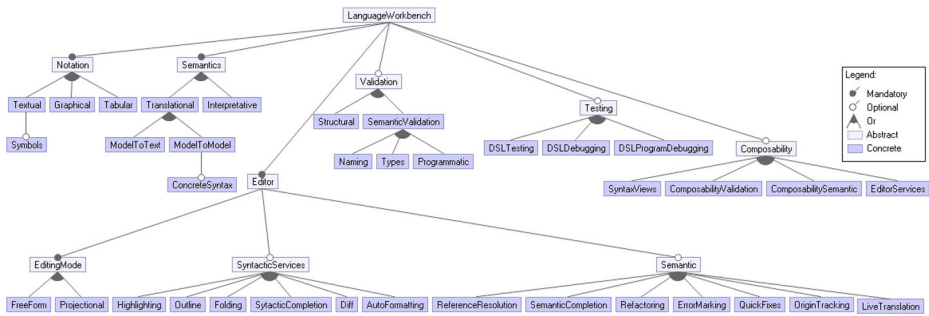


Fig. 1 Feature model for DSL language workbenches adapted from (Erdweg et al. 2015)

model but its universe of discourse is a set of models”. A metamodel contains statements about the constructs used in a system and is an abstraction of a model (Jeusfeld 2009). Furthermore, metamodels are used to define DSLs for graphical and textual languages (Schmidt 2006).

2.2 DSL development tool features

This section introduces features used as models for information gathering on the LWs in the selected primary studies of this SMS. We consider the term features as functionalities offered by the analyzed DSL development supporting tools. Thus, according to Erdweg et al. (2013) and Erdweg et al. (2015), the list of all tabulated features and sub-features are separated into six categories (Notation, Semantics, Editor, Validation, Testing, and Composability), as presented in Figure 1.¹ In the following, the six categories are presented in detail:

Notation Mandatory notation supported by LWs. It determines whether the models or programs will be presented in a textual, graphical, or tabular form. It is also possible to use them in a hybrid way, using one or more notations. Existing notation types are: Textual, Graphic, Tabular, Symbolic, which is a sub-feature of Tabular. According to Erdweg et al. (2013), “the notation can be a mix of textual, graphical, and tabular notations, where textual notation may optionally support symbols such as integrals or fraction bars embedded in regular text”.

Semantics Concerns the meaning of the models and can be categorized as translational or interpretative semantics. Translational semantics generates a program in some language from a model. Interpretative semantics, are directed at the execution of a model without a previous translation, *e.g.*, using Model@Runtime approaches (Mussbacher et al. 2014). According to Figure 1, both of them may be present in an LW. Also, translational semantics may be of different types: Model-to-Text, Model-to-Model, and Concrete Syntax.

Editor Support The editor support in LWs contains a sub-feature called editing mode, which is divided in two ways: free-form edition, as a common editor of the textual type, where the user is free to write and edit a template; and projectional edition, where the user

¹ An adapted feature model from (Erdweg et al. 2015) to remove duplicated features names.

works with predefined and fixed layouts. Also, the editor support has two types of services: Syntactic Service and Semantic Service. Syntactic Services consist of: Visual highlighting such as coloring text for textual language; Navigation support for outline view; Folding part of a model; Syntactic completion templates to suggest code; Comparison of programs, *i.e.* via diff tool; Auto formatting, restructuring, or aligning of a model's presentation.

On the other hand, semantic services consist of a reference resolution to link concepts such as declarations and variables; Code assists through semantic completion that blends semantic information such as reference resolution; Semantics-preserving refactoring of programs or models such as rename a class and move an attribute; Error markers for highlighting inconsistent structural features or violation of object constraints; Quick fixes to fix an error; Origin tracking and tracing models; and code derived from live translations.

Validation: LW's capability to identify structural, syntactic, and semantic disagreements and inconsistencies between models and metamodels. Types of validation: Structural validation and semantic validation, split into Naming, Types, and Programmatic.

Testing: Support provided by LWs for language testing, both for semantic (translation or interpretation) and syntactic testing (parser or projections). Testing aims language debugging and the development of language debuggers. Types of test support: DSL testing, debugging the DSL language definition (DSL debugging), and debuggers for the defined language (DSL Program Debugging).

Composability: A feature provided by LWs allowing the use of multiple languages to address different aspects of the system, such as support to incremental extension and linguistic alignment. For example, metamodels composed by other metamodels, or metamodels including structural features from others. Features related to composability: Syntax/Views, Composability Validation, Composability Semantics, and Editor Services.

2.3 Related work

Our work is inspired by the studies from (Erdweg et al. 2013; Erdweg et al. 2015), which are the basis for the data tabulation and extraction in our SMS. In other words, these are contributions highlighting features for a framework analysis applied to tools. However, they are not considered as systematic studies, since they are limited to tools presented at the Language Workbench Challenge (LWC'13).

As shown in Table 1, there are only two SMSs characterizing tool support for DSL development (do Nascimento et al. 2012; Kosar et al. 2016). In the following, we compare our contributions to these two works.

do Nascimento et al. (2012) conducted a systematic mapping of studies published until 2011, thus considering outdated technologies including: 1) tools to support the DSL development, and 2) application domains. Although presenting six tools and fifteen domains, their work failed to map tool features, so lacking elements for technical decision making. On the other hand, our SMS identified 59 tools, and also their features and application in more than 7 domains.

Kosar et al. (2016) performed an SMS considering works published in the interval from 2006 to 2012. Their focus is to understand the DSL research field, identifying research trends, and possible open issues in the area. Therefore, their study is lacking an analysis of tool support. Besides, the authors make a parallel with research questions inspired by another survey published in 2005, concluding that the number of publications remains stable

during the years from retrieved studies. Another important result is that the DSL community focuses more on the development of new techniques rather than investigating the integration of DSLs and software processes (toolchain). In this sense, our SMS provides a large number of DSL constructors whose diversity imposes difficulties for the configuration of toolchains, thus founding the area with compatibility features concerns.

Table 1 presents a comparison considering the aforementioned features. Our SMS analysis a wider range of studies, focusing on DSL, DSL development tools, and studies domain. Meanwhile, the studies (do Nascimento et al. 2012) (Kosar et al. 2016) focus only on the DSL domain. Although the state of the art presented by (Erdweg et al. 2013) has a focus similar to ours, they are limited to participants of the LWC'13, thus characterized as a survey. Differently, we considered all studies found in six scientific databases, published between 2012 and 2019, while the others only covered works until 2013.

3 SMS process

In this SMS, we followed a structured and established review process, presented by (Petersen et al. 2008). Figure 2 presents the process phases: Planning, Conduction, and Reporting. The first phase (Planning) and its activities are detailed in this section. The second phase (Conduction) is presented in Section 4. Finally, Section 5 presents and discusses the results obtained from the SMS data extraction activity.

3.1 Scope

Considering the creation of textual and graphical DSLs, our research goal is to map supporting tools, LWs, and frameworks, from works published between 2012 and 2019. Likewise, by providing an overview of tools features and licenses to provide guidance for DSL developers, we aim at helping MDE (Model-Driven Engineering) practitioners to choose the tools that best meet their needs for research and practice throughout four research questions and four quality assessment questions. In addition to these questions, which provide the classification of proposed DSLs for cross-application domains, we also included a broader analysis discussing research gaps for heterogeneity of the surveyed tools, asset evolution issues (model, transformation, metamodel, and toolchains), and usability issues.

3.2 Research questions

With this goal, we formulated the following Research Questions (RQ):

- RQ1.** What are the technologies used for DSL development? Our goal is to characterize these tools, thus qualifying their maturity in research and practice.
- RQ2.** What are the tools license types? Once tool acquisition and training are dependent on decisions such as business models built on non-commercial or commercial licenses, our goal is to map tools by licensing.
- RQ3.** For which application domains are the studies devoted to? Our goal is to identify the relevance of DSL roles in software development for each cross-domain such as Web applications, mobile applications, embedded systems, and others.
- RQ4.** What features of the DSL creation process do these tools support? Since DSL development tools are diverse and help in many phases of DSL development, we aim at characterizing these proposals by their technical features.

Table 1 Related Work Summary

Concept	Paper Type	Interval	Criteria Analysis	Domains	Tools
Our SMS	SMS	2012-2019	Supporting tools, tool features, domain, licenses	DSL, Supporting tool and paper domain	DSL-Supporting tools and features mapped from selected primary studies
(do Nascimento et al. 2012)	SMS	Not specified-2011	Supporting tools, techniques, methods, processes, DSL validation	DSL domains, approaches and paper contribution	DSL-Supporting tools mapped from selected primary studies
(Kosar et al. 2016)	SMS	2006-2012	Language product line, approaches Contribution type, research type, evaluation type, research fields, demographics	DSL domain, DSL research type, paper type	DSL-Supporting tools mapped from selected primary studies
(Erdweg et al. 2013)	The State of the Art	Not specified-2013	Empirical data, feature completeness	DSL-Supporting tools	DSL-Supporting tools and features mapped from Challenge participants

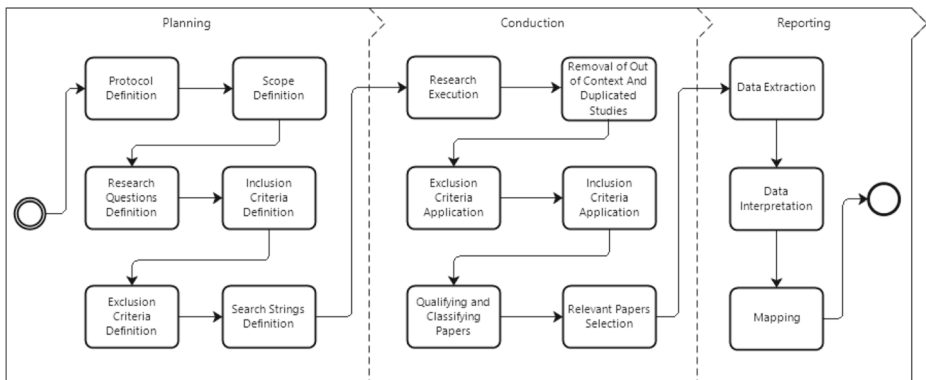


Fig. 2 Systematic Mapping Studies Process (Petersen et al. 2008) created with BPMN notation

3.3 Inclusion and exclusion criteria

The definition of the inclusion and exclusion criteria activity is based on the SMS scope, study objectives, and our research questions. The research is directed to most relevant studies within the context of this SMS, excluding articles with no relevance. This step is important because it determines which works may be classified and qualified in later stages of the SMS. The inclusion of a primary study must attend all Inclusion Criteria (IC) for later analysis and classification. Similarly, if the study meets at least one of the Exclusion Criteria (EC), it will be removed from the primary studies being analyzed.

Inclusion Criteria (IC)

IC1. The primary study must present an approach, technique, method, process, tool, framework, or LW to manipulate DSL or DSML.

Studies presenting a new technology that supports DSL or DSML development will be included, *e.g.* studies presenting the Xtext framework.

IC2. The primary study must mention a DSL-Supporting tool, framework, or LW.

Studies that mention a technology that supports DSL or DSML development will be included, *e.g.* studies that are focused on the geospatial domain but mention the use of a DSL and the technology that was used in its development.

Studies that focus on presenting a new DSL or DSML and also mention the technology that is used in its development will be included, *e.g.* studies focused on presenting a DSL and that also mention the technology that was used in its development.

Exclusion Criteria (EC)

EC1. Studies published before 2012.

EC2. Studies in any language other than English.

EC3. Duplicated and/or incomplete studies.

EC4. Studies only available in the form of abstract, slide presentation, poster or short paper.

Table 2 Definition of Search Strings

Terms	Synonyms
Domain-Specific Language	DSL, DSML, Domain Specific Language, Domain-Specific-Language, Domain Specific Modeling Language, Domain-Specific Modeling Language, Domain-Specific-Modeling-Language, Small Language, Small-Language, Little Language, Little-Language
Tool	Language Workbench

3.4 Search process

During the search process, we only considered the use of computer science databases providing web-based search engines through keywords. The following scientific databases/digital libraries in the computing area are used: Compendex (Engineering Village),² IEEE Xplore,³ ScienceDirect,⁴ Association for Computing Machinery (ACM) Digital Library,⁵ Scopus⁶ and SpringerLink.⁷

3.4.1 Search strings

To define the search strings,⁸ we used the terms and synonyms presented in Table 2. As Figure 3 shows, we used “OR” operators to include synonyms in the string. In addition, the “AND” operator is used to add more terms, narrowing the string. In some scientific databases, we had to adopt some strategies to create different versions of strings similar to the one presented in Figure 3. For evaluating the quality and comprehensiveness of the search string, exploratory research was carried out considering the definition of three control studies (Erdweg et al. 2013; Erdweg et al. 2015; Bernardino et al. 2017). Hence, the search string was executed in the digital libraries, thus certifying that the control studies are found among the retrieved publications.

3.5 Study quality assessment

To evaluate the quality of the primary studies, a set of Quality Assessment (QA) criteria was defined. The study’s target for QA is those remaining after application of the inclusion and exclusion criteria. These criteria aim to quantify the relevance of each primary study and to allow us to compare the selected primary studies. Primary studies with “zero (0) quality score” is removed from the relation of primary studies, even if they are in the domain of the research. Also, primary studies that did not score in QA1 consequently did not score on other quality assessment and were therefore removed from the list of primary studies. Each researcher applied the QA criteria in accordance with the following grade: Yes: 1.0; Partially: 0.5; No: 0.0. The QA criteria are defined as follows:

²Compendex: www.engineeringvillage.com
³IEEE: www.ieeexplore.ieee.org
⁴ScienceDirect: www.sciencedirect.com
⁵ACM: <https://dl.acm.org>
⁶Scopus: www.scopus.com
⁷SpringerLink: www.link.springer.com
⁸The search strings are available at our online repository (Iung et al. 2020)

(DSL OR DSML OR Domain Specific Language OR Domain-Specific Language OR Domain-Specific-Language OR Domain-Specific Modeling Language OR Domain Specific Modeling Language OR Domain-Specific-Modeling-Language OR Small Language OR Small-Language OR Little Language OR Little-Language) AND (Tool OR Language Workbench)

Fig. 3 Search String

QA1. Does the study present a tool that supports DSL development?

Evaluation: **Y:** The study presents a tool that supports DSL development; **P:** The study mentions a tool that supports DSL development but does not present details about the tool; **N:** The study does not present or mention a supporting tool to assist DSL development.

QA2. Does the tool support at least one of the notations (graphical or textual)?

Evaluation: **Y:** The study presents a tool supporting graphical or textual notations; **P:** The study does not present any supporting tools but the results indicate that graphical or textual notations were used; **N:** The study does not indicate evidence about a graphical or textual notation usage.

QA3. Does the study report how the tool was applied in the development of a DSL?

Evaluation: **Y:** The study presents the tool using coding examples, code snippets or images; **P:** The study argues that a tool was used, but does not present implementation details; **N:** The study does not present any implementation evidence.

QA4. Is the tool usage described in a clear/detailed manner?

Evaluation: **Y:** The study presents a tutorial, a process, steps to complete a task, or defines a flow to achieve a goal; **P:** The study only mentions activities that must be executed without further explanations; **N:** The study does not present any evidence to support the usage of the tool.

3.6 Data extraction strategy

We created a form to identify and extract relevant data from the selected studies. This information is used to answer the RQs. From each study, we extracted the following data:

- ★ Database: ACM, Compendex (Engineering Village), IEEE, SCOPUS, ScienceDirect and SpringerLink
- ★ Source: full reference conference, book, journal name
- ★ Title
- ★ Abstract
- ★ Authors
- ★ Year
- ★ Application Domain: domain to which the study is proposed
- ★ Tool, Framework, Language Workbench
 - Feature: notation, semantics, edit support, semantic and syntactic services, validation, testing and composability
 - License Type: commercial or non-commercial

Despite the year range (between 2012 and 2019) used during the studies search, our results present tools from years prior to 2012. Such tools were included because our

year range restriction was only applied to the study publications year, not to the tools release date.

3.7 Data analysis

We analyzed the collected data to: identify the DSL technologies (tools/ frameworks/ language workbenches) supporting the DSL development process (**RQ1**); identify DSL tools license types (commercial or non-commercial) and show the DSL technologies per notation type concerning their license types (**RQ2**); identify and map the application domains by the year of publication and study category (**RQ3**); map the DSL technologies from the primary studies supporting some DSL development process phase (**RQ4**).

4 Execution

The execution⁹ of this study was performed in July 2019 following the research protocol illustrated by the milestone “Conduction”, as shown in Figure 2. To support the planning and conduction phases of the SMS we used the Thoth tool (Marchezan et al. 2019). Two strings are used to search the IEEE Xplore digital library. We adopted this composite search strategy because the IEEE search engine limits the search to a maximum of 15 terms in each string. Therefore, keeping the other common terms in each compound search, the search string was split into two compound terms: DSL and DSML.

For all scientific databases, the search is limited to the fields “Abstract”, “Title” and “Keywords”. Besides, directly at the search engine, we limited the year range to include papers published from 2012 to 2019.

As Figure 4 illustrates, after duplicate removal, 1,862 studies remained. Then, studies considered out of context were removed, such as articles containing the word DSL or DSML, but that was not part of the research scope. In this step, a total of 1,780 remained for the application of exclusion and inclusion criteria.

As illustrated in the process shown in Figure 2, the activities Application of Exclusion Criteria and Application of Inclusion Criteria are executed. The result was 1,780 studies, read in the activity Qualifying and Classifying Papers. As a result, 1,430 studies remained after applying the EC. Next, studies were excluded because they did not relate to any IC. In total, 390 studies remained for qualification and classification.

4.1 Studies qualification

The results of the quality score are shown in Table 3, where each study can be identified through the column **ID**. The references and year of publication are presented in the column **References**. The scores based on the QA are showed in columns **1**, **2**, **3** and **4**. The final QA score is presented in column **QS**. Two researchers individually assessed each one of the 230 studies according to the four QA showed in Section 3.3.

This measure was applied to analyze nominal scales, and its results were interpreted by applying the table proposed by Landis and Koch (1977), available in the data-set package.

⁹Artifacts are available in our repository (Iung et al. 2020)

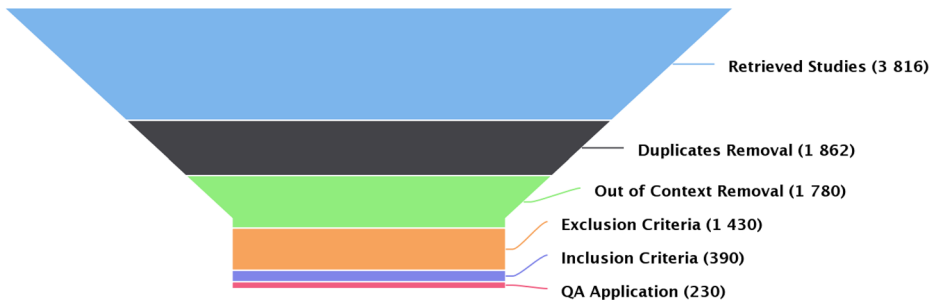


Fig. 4 Systematic Mapping Study Conduction Results

The Kappa's coefficient is used to analyze the inter-rater agreement between two observers. In this SMS, both researchers analyzed and applied the QA criteria for each study. The results are based on the number of agreement answers, considering a score between 0 and 1, where 1 represents a total agreement and 0 represents a total disagreement or a random agreement.

The Kappa analysis performed in the scores assigned to each judge of our SMS resulted in a coefficient of 0.64. Hence, this result means a substantial level of agreement between the researchers. To solve possible judgment discrepancies considering the studies classification, we consulted an expert in the DSL area.

4.2 Classification scheme

The classification scheme is generated based on the “Data Interpretation” activity. To acquire an abstract level of understanding from the selected studies, we merged their keywords. The results obtained during this step were used to give support when defining and categorizing the facets that represent the selected primary studies population. As shown in Table 4, the three main facets are: (i) DSL Development Mechanism Type: Tool, Language Workbench or Framework; (ii) Feature: Notation, Semantics, Edit Mode, Syntactic Service, Semantic Service, Validation, Testing, Composability; (iii) Publication Contribution: Approach, Research, Experience Report;

The classification scheme of “Feature” facet is based on studies by (Erdweg et al. 2013; Erdweg et al. 2015), exposing features of DSL-Supporting tools as shown in Table 5. These features range from the representation type used by the tool, such as Graphical and Textual, to tool assistance mechanisms, such as Quick Fixes and Refactoring.

5 Report

In this section, we present our outcome in answering the research questions, as follows.

RQ1. Which are the technologies used for DSL development?

After the analysis of the 230 selected papers, we identified 59 DSL development tools, as presented in Table 6. We observed that some technologies were cited by different authors.

It is important to highlight a large number of citations of the Xtext framework (102), as well as other tools derived from the Eclipse Modeling Framework (EMF) (114) and Graphical Modeling Framework (GMF) (30), such as the Papyrus tool (18) and the Sirius

Table 3 Studies Quality Scores

Studies		QA				Studies		QA					
ID	References	1	2	3	4	QS	ID	References	1	2	3	4	QS
S01	(Demirkol et al. 2013)	1.0	1.0	0.0	0.0	2.0	S116	(Mosteller et al. 2018)	1.0	1.0	0.5	0.5	3.0
S02	(Mavridou et al. 2018)	1.0	0.5	0.0	0.0	1.5	S117	(Jafer et al. 2017)	0.5	1.0	1.0	0.5	3.0
S03	(De Sousa and Da Silva 2016)	1.0	1.0	0.0	0.0	2.0	S118	(Ristić 2017)	0.5	0.0	0.0	0.0	0.5
S04	(Karol et al. 2018)	0.5	0.5	0.0	0.0	1.0	S119	(Rapos and Stephan 2019)	0.5	0.0	0.0	0.0	0.5
S05	(van den Berg et al. 2018)	1.0	1.0	0.0	0.0	2.0	S120	(Zikra 2012)	1.0	1.0	0.0	0.5	2.5
S06	(Hoyos et al. 2013)	1.0	1.0	0.0	0.0	2.0	S121	(Gamboa and Syriani 2019)	1.0	1.0	1.0	0.5	3.5
S07	(Mayr-Dorn and Laaber 2017)	1.0	1.0	1.0	0.0	3.0	S122	(Ratiu and Ulrich 2017)	1.0	1.0	0.0	0.0	2.0
S08	(Visic et al. 2015)	0.5	1.0	0.0	0.0	1.5	S123	(Pérez-Berenguer and García-Molina 2019)	0.5	0.0	0.0	0.0	0.5
S09	(Barbosa et al. 2019)	0.5	1.0	0.5	0.0	2.0	S124	(Schuts et al. 2018)	0.5	0.0	0.0	0.0	0.5
S10	(Heitkötter 2012)	1.0	1.0	0.0	0.0	2.0	S125	(Maschotta et al. 2019)	0.5	0.5	0.5	0.0	1.5
S11	(Cariou et al. 2018)	1.0	1.0	0.5	0.0	2.5	S126	(Semeráth and Varró 2018)	0.5	0.0	0.0	0.0	0.5
S12	(Nazir et al. 2017)	0.5	0.0	0.5	0.0	1.0	S127	(Åkesson and Hedin 2017)	1.0	0.5	1.0	1.0	3.5
S13	(Rensink and Aksit 2018)	0.5	0.0	0.0	0.0	0.5	S128	(Wachsmuth et al. 2014)	1.0	1.0	0.5	0.0	2.5
S14	(Heinrich et al. 2019)	0.5	0.5	0.0	0.0	1.0	S129	(Combemale et al. 2017)	1.0	0.5	0.0	0.0	1.5
S15	(Jr et al. 2019)	1.0	1.0	1.0	0.5	3.5	S130	(Voelter et al. 2017)	0.5	1.0	1.0	1.0	3.5
S16	(Koster et al. 2018)	0.5	0.5	0.5	0.5	2.0	S131	(Hojaji et al. 2019)	1.0	0.5	0.0	0.5	2.0
S17	(Pomante et al. 2015)	1.0	1.0	1.0	0.0	3.0	S132	(Visiers et al. 2017)	0.5	0.5	0.5	0.5	2.0
S18	(Salehi et al. 2018)	0.5	0.5	0.5	0.0	1.5	S133	(Jézéquel et al. 2015)	1.0	1.0	0.5	0.0	2.5
S19	(Antonelli et al. 2015)	1.0	1.0	0.0	0.0	2.0	S134	(Voelter et al. 2012)	1.0	1.0	0.0	0.0	2.0
S20	(HoseinDoost et al. 2017)	1.0	0.5	1.0	0.5	3.0	S135	(Burdusel et al. 2018)	0.5	0.5	0.5	0.0	1.5
S21	(Hasan et al. 2017)	0.5	1.0	1.0	0.5	3.0	S136	(Boubeta-Puig et al. 2017)	0.5	0.5	0.5	0.0	1.5
S22	(Kanav 2018)	0.5	0.0	0.0	0.0	0.5	S137	(Ries et al. 2018)	0.5	1.0	0.5	0.0	2.0
S23	(Arcaini et al. 2019)	0.5	0.5	0.0	0.0	1.0	S138	(Kövesdán and Lengyel 2019)	0.5	0.0	0.0	0.0	0.5

Table 3 (continued)

Studies		QA				Studies		QA					
ID	References	1	2	3	4	QS	ID	References	1	2	3	4	QS
S24	(Rieger and Kuchen 2018)	0.5	0.5	0.0	0.0	1.0	S139	(Kalnins and Barzdins 2019)	1.0	0.5	0.5	0.5	2.5
S25	(Dejanović et al. 2017)	1.0	1.0	1.0	0.5	3.5	S140	(Molina et al. 2013)	1.0	1.0	0.0	0.0	2.0
S26	(Barišić et al. 2017)	0.5	1.0	1.0	0.5	3.0	S141	(Pietrusiewicz 2019)	0.5	0.0	0.0	0.0	0.5
S27	(Gibbs et al. 2015)	1.0	1.0	0.5	0.0	2.5	S142	(Mendivelso et al. 2018)	0.5	0.5	0.0	0.0	1.0
S28	(Gómez-Abajo et al. 2018)	0.5	1.0	0.0	0.0	1.5	S143	(Gargantini and Radavelli 2018)	1.0	1.0	1.0	0.5	3.5
S29	(Jinzhí et al. 2017)	0.5	0.5	0.0	0.0	1.0	S144	(Denkers et al. 2018)	0.5	0.5	0.5	0.5	2.0
S30	(Bermúdez R. et al. 2017)	0.5	0.0	0.0	0.5	1.0	S145	(Ma and Sallai 2017)	1.0	1.0	0.5	0.5	3.0
S31	(Theobald et al. 2018)	1.0	1.0	1.0	0.5	3.5	S146	(Zhou et al. 2017)	0.5	0.5	0.5	0.0	1.5
S32	(Dwarakanath et al. 2017)	1.0	1.0	1.0	0.5	3.5	S147	(Yakymets et al. 2018)	0.5	0.0	0.0	0.0	0.5
S33	(Elaasar et al. 2018)	0.5	0.5	0.0	0.0	1.0	S148	(Wienke et al. 2018)	1.0	0.5	0.5	0.5	2.5
S34	(Bousse et al. 2019)	1.0	1.0	0.5	0.0	2.5	S149	(Vieira and Carvalho 2017)	1.0	1.0	1.0	0.5	3.5
S35	(Sorgalla et al. 2018)	0.5	1.0	0.5	0.5	2.5	S150	(Derakhshanmash et al. 2019)	0.5	0.0	0.0	0.0	0.5
S36	(García-Daz et al. 2018)	0.5	0.5	0.0	0.0	1.0	S151	(Nordmann et al. 2015)	1.0	1.0	0.0	0.0	2.0
S37	(López-Fernández et al. 2019)	0.5	0.5	0.0	0.0	1.0	S152	(Falkner et al. 2013)	1.0	1.0	0.0	0.0	2.0
S38	(Szabo et al. 2016)	1.0	1.0	0.5	0.0	2.5	S153	(Vinogradov et al. 2015)	1.0	1.0	0.0	0.0	2.0
S39	(Sandobalín et al. 2017)	0.5	0.5	0.5	0.0	1.5	S154	(Wigand et al. 2017)	0.5	0.5	0.5	0.5	2.0
S40	(Ratiu and Ulrich 2019)	1.0	0.5	0.0	0.5	2.0	S155	(Ratiu and Ulrich 2019)	0.5	0.5	0.0	0.0	1.0
S41	(Zarrin et al. 2018)	0.5	0.0	0.0	0.0	0.5	S156	(Alvarez and Casallas 2013)	1.0	1.0	0.0	0.0	2.0
S42	(Zarrin and Baumeister 2018)	0.5	0.0	0.0	0.0	0.5	S157	(Macías et al. 2019)	0.5	0.0	0.0	0.0	0.5
S43	(Häser et al. 2016)	1.0	1.0	0.0	0.0	2.0	S158	(Maróti et al. 2014)	1.0	1.0	0.0	0.0	2.0
S44	(Bergenti 2014)	1.0	1.0	0.0	0.0	2.0	S159	(Bonnet et al. 2016)	1.0	1.0	0.0	0.0	2.0
S45	(Lemazurier et al. 2017)	0.5	0.0	0.5	0.0	1.0	S160	(Bousse et al. 2018)	1.0	0.0	0.0	0.0	1.0
S46	(de Almeida Pereira et al. 2018)	0.5	0.0	0.0	0.0	0.5	S161	(Maro et al. 2015)	1.0	1.0	0.5	0.0	2.5

Table 3 (continued)

Studies ID	References	QA				Studies ID	References	QA			
		1	2	3	4			1	2	3	4
S47	(Brjancin et al. 2018)	0.5	0.5	0.5	0.0	S162	(Boßelmann et al. 2018)	0.5	0.0	0.0	0.0
S48	(Walter et al. 2014)	1.0	1.0	0.5	0.0	S163	(Bocciarelli et al. 2018)	0.5	0.5	0.0	0.0
S49	(Córdoba-Sánchez and de Lara 2016)	1.0	1.0	1.0	0.5	S164	(Oliveira and Belo 2017)	0.5	0.5	0.5	0.5
S50	(Li et al. 2018)	0.5	0.5	0.5	0.0	S165	(Challenger et al. 2014)	1.0	1.0	0.0	0.0
S51	(Pröll et al. 2018)	0.5	0.0	0.0	0.0	S166	(Ober et al. 2018)	0.5	0.5	0.0	0.0
S52	(Ruiz-Rube et al. 2019)	1.0	0.5	0.5	0.0	S167	(Tekinerdogan and Arkin 2019)	0.5	0.5	0.5	0.5
S53	(Mavropoulos et al. 2017)	0.5	0.0	0.0	0.0	S168	(Pescador et al. 2015)	1.0	1.0	0.0	0.0
S54	(Huang et al. 2015)	1.0	1.0	0.0	0.0	S169	(Korenkov et al. 2015)	1.0	1.0	0.0	0.0
S55	(Neubauer et al. 2017)	0.5	1.0	0.0	0.0	S170	(Gonnord and Mosser 2018)	0.5	0.0	0.0	0.0
S56	(Ángel et al. 2018)	0.5	0.0	0.0	0.0	S171	(Savić et al. 2014)	1.0	1.0	0.0	0.5
S57	(Ratiu and Voelter 2016)	0.5	1.0	1.0	1.0	S172	(Zweihoff et al. 2019)	0.5	0.5	0.5	0.5
S58	(Kahani 2018)	0.5	0.0	0.0	0.0	S173	(Nakamura et al. 2012)	0.5	1.0	0.0	0.0
S59	(Merino et al. 2018)	0.5	0.0	0.0	0.0	S174	(Ouared et al. 2018)	0.5	0.0	0.0	0.0
S60	(Yigitbas et al. 2018)	0.5	0.0	0.0	0.0	S175	(Arendt et al. 2013)	1.0	1.0	0.0	0.5
S61	(Dupont et al. 2018)	0.5	0.5	1.0	0.5	S176	(Molina 2019)	0.5	0.5	0.0	0.0
S62	(Koschke et al. 2018)	0.5	0.0	0.0	0.0	S177	(Zhu and Wang 2017)	0.5	0.5	0.5	0.5
S63	(Bernardino et al. 2016)	1.0	1.0	0.5	0.5	S178	(Nagele and Hooman 2017)	0.5	1.0	0.0	0.0
S64	(Zabawa and Hnatkowska 2017)	0.5	0.0	0.0	0.0	S179	(Trobo et al. 2019)	0.5	0.5	0.5	0.5
S65	(Naujokat et al. 2017)	0.5	1.0	1.0	1.0	S180	(Mohamad et al. 2015)	0.5	1.0	0.0	0.0
S66	(Gargantini and Vavassori 2012)	1.0	1.0	0.0	0.5	S181	(Hoisl et al. 2017)	0.5	0.5	0.0	0.0
S67	(Hiya et al. 2013)	1.0	1.0	0.0	0.0	S182	(Tikhonova 2017)	0.5	0.5	0.5	0.5
S68	(Stocker et al. 2017)	0.5	0.5	0.5	0.5	S183	(Monthe et al. 2016)	1.0	1.0	0.0	0.0
S69	(Zhu et al. 2017)	1.0	1.0	1.0	0.5	S184	(Caramujo et al. 2019)	0.5	0.5	0.5	0.0

Table 3 (continued)

Studies		QA				Studies		QA			
ID	References	1	2	3	4	ID	References	1	2	3	4
S70	(Bennani et al. 2018)	0.5	0.0	0.0	0.0	S185	(Ribeiro and da Silva 2017)	1.0	1.0	0.0	0.0
S71	(Ribeiro et al. 2016)	1.0	1.0	0.5	0.0	S186	(Iliasov and Romanovsky 2013)	0.5	1.0	0.0	0.0
S72	(Combemale et al. 2018)	0.5	0.5	0.0	0.5	S187	(Garmendia et al. 2019)	0.5	0.0	0.0	0.0
S73	(Butting et al. 2018)	0.5	0.5	0.5	0.0	S188	(Haitzer and Zdun 2014)	1.0	1.0	0.5	0.0
S74	(Tisi and Cheng 2015)	0.5	0.0	0.0	0.0	S189	(Coulon et al. 2018)	0.5	0.0	0.0	0.0
S75	(Tairas and Cabot 2015)	1.0	0.0	0.0	0.0	S190	(Pasternak et al. 2018)	0.5	0.5	0.0	0.0
S76	(Jager et al. 2016)	1.0	1.0	0.5	0.0	S191	(Vijović et al. 2014)	1.0	1.0	0.0	0.0
S77	(Krüger et al. 2018)	0.5	0.5	0.0	0.0	S192	(Corral Diaz 2018)	1.0	1.0	0.5	0.5
S78	(Rocha et al. 2017)	1.0	1.0	0.0	0.0	S193	(Bartman et al. 2017)	0.0	0.5	0.0	0.0
S79	(Bencharqui et al. 2019)	0.5	1.0	0.5	0.0	S194	(Strömbeck 2018)	0.5	0.0	0.0	0.0
S80	(Bünder 2019)	0.5	0.0	0.0	0.0	S195	(Jrad et al. 2019)	0.5	0.5	0.0	0.0
S81	(Butting et al. 2018)	0.5	0.5	0.5	0.0	S196	(Forbrig 2018)	0.5	0.5	0.0	0.0
S82	(Zhao and Huang 2018)	0.5	0.0	0.0	0.0	S197	(Sievaneic and Zdun 2018)	0.5	0.0	0.0	0.0
S83	(Zarrin and Baumeister 2014)	0.5	0.5	0.0	0.0	S198	(Tragatschnig et al. 2018)	0.5	0.5	0.0	0.0
S84	(Tariq et al. 2014)	1.0	1.0	0.5	0.0	S199	(Guelfi et al. 2017)	1.0	1.0	0.5	0.5
S85	(van den Berg et al. 2018)	0.5	0.0	0.0	0.0	S200	(Makedonski et al. 2019)	0.5	0.5	0.0	0.0
S86	(Montrieux et al. 2013)	1.0	0.5	0.0	0.0	S201	(Chlipala et al. 2017)	1.0	1.0	0.0	0.0
S87	(Rabiser et al. 2018)	0.5	0.0	0.0	0.0	S202	(Uhnák and Pergl 2016)	1.0	1.0	0.0	0.0
S88	(Betini 2014)	1.0	1.0	0.5	0.0	S203	(Dörndorfer et al. 2019)	0.5	0.5	0.0	0.0
S89	(Terzić et al. 2018)	0.5	0.5	0.5	0.5	S204	(Erdweg et al. 2013)	1.0	1.0	0.0	0.0
S90	(Schwaiger 2016)	0.5	0.5	0.5	0.0	S205	(Mettouris et al. 2018)	0.5	0.5	0.0	0.0
S91	(Arkin and Tekinerdogan 2014)	1.0	0.0	0.0	0.0	S206	(Kowalski and Magott 2012)	0.5	1.0	0.0	1.5
S92	(Krasts et al. 2012)	1.0	0.0	0.0	0.0	S207	(van Rozen and van der Storm 2019)	0.5	0.5	0.0	0.0

Table 3 (continued)

Studies		QA				Studies		QA			
ID	References	1	2	3	4	ID	References	1	2	3	4
S93	(Montenegro-Marin et al. 2012)	1.0	1.0	1.0	0.5	S208	(van Rozen and van der Storm 2017)	1.0	1.0	0.0	0.0
S94	(Tran et al. 2017)	0.5	0.0	0.0	0.0	S209	(Essadi and Anwar 2018)	0.5	0.5	0.0	0.0
S95	(Jacob et al. 2014)	1.0	1.0	1.0	0.5	S210	(Rodríguez-Echeverría et al. 2018)	0.5	0.0	0.0	0.0
S96	(Viana et al. 2013)	1.0	1.0	0.5	0.5	S211	(Idani et al. 2019)	0.5	0.5	0.5	0.0
S97	(Pescador and Lara 2016)	1.0	1.0	0.5	0.5	S212	(Visbakká et al. 2018)	0.5	0.5	0.0	0.0
S98	(Gossen et al. 2018)	0.5	0.0	0.0	0.0	S213	(Granchelli et al. 2017)	1.0	1.0	0.5	0.5
S99	(Tezel et al. 2018)	0.5	0.5	0.0	0.0	S214	(Hoffmann et al. 2018)	0.5	0.0	0.0	0.0
S100	(Azadi M. et al. 2015)	1.0	1.0	1.0	0.5	S215	(De F. et al. 2017)	0.5	0.0	0.5	0.0
S101	(Buisson and Rehab 2018)	0.5	0.0	0.0	0.0	S216	(Gavran et al. 2018)	0.5	0.0	0.0	0.0
S102	(Tolvanen and Kelly 2018)	1.0	1.0	0.5	0.0	S217	(Iglesias et al. 2019)	0.5	0.0	0.0	0.0
S103	(Herrera 2014)	1.0	1.0	0.5	0.0	S218	(Bettini 2019)	1.0	1.0	0.5	0.5
S104	(Rose et al. 2012)	1.0	1.0	0.5	1.0	S219	(Barišić et al. 2017)	1.0	1.0	0.0	0.5
S105	(Erdweg et al. 2015)	1.0	1.0	0.0	0.0	S220	(de Sousa and da Silva 2018)	0.5	0.5	0.0	0.0
S106	(Häser et al. 2018)	0.5	0.0	0.0	0.0	S221	(Hinkel et al. 2017)	1.0	1.0	0.0	0.0
S107	(Sutii et al. 2017)	1.0	1.0	0.0	0.0	S222	(Voelter et al. 2019)	0.5	0.5	0.5	0.5
S108	(Viana et al. 2013)	0.5	1.0	0.5	0.0	S223	(Crapo and Moltra 2019)	0.5	0.5	0.0	0.0
S109	(Mezhuyev et al. 2018)	0.5	0.0	0.0	0.0	S224	(Akhundov et al. 2016)	1.0	1.0	0.5	1.0
S110	(de la Vega et al. 2018)	0.5	0.0	0.0	0.0	S225	(Morgan et al. 2018)	0.5	0.5	0.0	0.0
S111	(Smits and Visser 2017)	1.0	1.0	0.5	0.0	S226	(Vögele et al. 2018)	0.5	1.0	0.5	0.0
S112	(Lelandais et al. 2018)	0.5	0.0	0.0	0.0	S227	(Efftinge et al. 2012)	1.0	1.0	0.5	0.0
S113	(Voelter 2018)	0.5	0.5	0.5	0.0	S228	(Santos and Gomes 2016)	1.0	1.0	0.0	0.0
S114	(Rein et al. 2016)	1.0	1.0	0.0	0.0	S229	(Ribeiro and Da S 2014)	1.0	0.0	0.0	0.0
S115	(Vaderna et al. 2018)	0.5	0.0	0.0	0.0	S230	(Le G. and Waltham 2013)	0.5	1.0	0.0	0.0

Table 4 Classification scheme

Dimensions	Categories	Description
DSL Development		
Mechanism Type	DSL Construction Tool	Generic classification for a tool used directly in DSL development, i.e., a tool that is essential to build a DSL.
	Language Workbench	Development technologies exclusively aimed to build DSL in an integrated environment (a package of tools, toolbox). The workbench supports the definition of languages and their integration with the environment of interest (Fowler 2010), thus characterizing supersets.
	Utility Tools	A unique component or a set of components which may be used to solve a specific problem in a specific domain or used in the DSL development, offering optional utility features for the other ones.
Features	Notation	Determine the presentation of models and programs. Such presentations may be in textual, graphical, or tabular form. Also, the models may be presented in a hybrid form.
	Semantics	The categorization of language semantics. This may be categorized as translational or interpretative semantics.
	Edit mode	The way how end-users interact with the Tool, LW, or Framework. It may be divided in two ways: free form edition and projectional edition.
	Syntactic service	Consists in support syntactic features such as navigation support for outline view, text coloring for textual language, folding a model, completion templates to suggest code and auto-formatting.
	Semantic service	Consists of semantic features such as code assist through semantic completion, semantics-preserving refactoring of programs or models, error markers to highlight the involved model and quick fixes to fix an error.
	Validation	Ability to identify structural, syntactic and semantic disagreements and inconsistencies between models and metamodels.
	Testing	Support for language testing, both for semantic and syntactic testing.
	Composability	Allows the use of multiple languages, aiming at different aspects of the system.
Publication		
Contribution	Approach	Procedure, technique, a method to deal with solving a problem. This work uses the definition provided by each paper author (<i>e.g.</i> (Barišić et al. 2017)).
	Research	Systematic investigation to reach conclusions. This work uses the definition provided by each paper author (<i>e.g.</i> (Ribeiro et al. 2016)).
	Experience Report	Report describing a previous experience from the authors in a Software Engineering activity. This work uses the definition provided by each paper author (<i>e.g.</i> (Zikra 2012)).

framework (42). Other LWs also received a fair number of citations, such as MetaEdit+ (27), belonging to MetaCase, MPS (Meta Programming System) (48) developed by JetBrains and Spoofox (21) by MetaBorg.

Table 5 List of Potential Features of DSL-Supporting Tools (Erdweg et al. 2015)

Features			
ID	Name	ID	Name
01	Textual	10	Naming
02	Graphical	11	Types
03	Tabular	12	Programmatic
04	Symbols	13	DSL Testing
05	Model2Text	14	DSL Debugging
06	Model2Model	15	DSL Progr. Debugging
07	Concrete Syntax	16	Syntax/Views
08	Interpretative	17	Validation
09	Structural	18	Semantics
19	Editor Services	20	Free-form
21	Projectional	22	Highlighting
23	Outline	24	Folding
25	Syntactic Completion	26	Diff
27	Auto Formatting	28	Reference Resolution
29	Semantic Completion	30	Refactoring
31	Error Marking	32	Quick Fixes
33	Origin Tracking	34	Live Translation

Figure 5 shows a Venn diagram of the notations supported by the tools: the A letter stands for Graphical notation; the B letter stands for Textual notation; the C letter stands for Graphical, and Textual notations; the D letter stands for Graphical, Tabular, and Symbolic notations; the E letter stands for Graphical, Textual, and Tabular notations; the F letter stands for Graphical, Textual, Tabular, and Symbolic notations.

There are 39 mapped tools that support graphical notations (the largest category), followed by 30 that support textual notations. Among the mapped tools, T38 and T55 provide support for all four types of notations. In terms of support for graphical, tabular and symbolic notations, T34 is the only one presenting these three features. Tool T32 supports a unique combination of graphical, textual and tabular notations. The tools T5, T16, T21, T28, T33, T43 and T48 support graphical and textual notations, but not symbolic or tabular ones.

It is important to mention that none of the tools implement all notations, because some tools only support one activity of the DSL development. Furthermore, T1, T27 and T58 are code generators, T57 is a Java dialect and T18 is a family of languages and tools for code generation and model validation, but they are here considered as part of textual notation.

RQ2. What are the tools license types?

In this SMS, we classified the studies into two categories: commercial or non-commercial. Normally, we extracted this information from the study, but in some cases, it was necessary to search on the official websites and manufacturers of the tools. Table 6 shows the tools license type where 41 tools are non-commercial, while only nine have commercial licenses. We were unable to identify the license type of nine tools, as these tools have not been officially released and are without online information¹⁰. As non-commercial software, it was also taken into account the tools under academic license and those available from online repositories open for evaluation, such as GitHub.

RQ3. For which application domains the study is devoted to?

In order to map the domains to which the construction tools for DSL are applied, as found in sections devoted to the evaluation/conceptual demonstration from the selected studies,

¹⁰We searched the tools on Google (www.google.com)

Table 6 List of DSL-Supporting Tools

ID	Name	Citations	Cited by	Licence
T01	Acceleo	84	S03 S04 S09 S11 S13 S14 S15 S18 S23 S24 S28 S34 S35 S37 S41 S42 S46 S47 S50 S51 S56 S58 S60 S62 S70 S72 S73 S74 S74 S89 S90 S96 S99 S101 S109 S112 S116 S118 S119 S123 S125 S126 S131 S135 S136 S138 S139 S142 S147 S150 S155 S156 S157 S160 S163 S166 S168 S172 S174 S176 S177 S179 S181 S184 S185 S187 S189 S191 S192 S196 S197 S198 S200 S203 S205 S207 S210 S211 S212 S214 S216 S217 S218 S220	Non-Commercial
T02	ADOxx	1	S08	Non-Commercial
T03	Argyle	1	S54	N/A
T04	AToMPM	2	S61 S121	Non-Commercial
T05	CINCO	4	S98 S138 S162 S172	Non-Commercial
T06	Clooca	2	S67 S73	Non-Commercial
T07	Meta3	1	S138	N/A
T08	DEViL	1	S65	Non-Commercial
T09	DoMe	1	S225	Non-Commercial
T10	DrRacket	2	S55 S201	Non-Commercial
T11	DSL-tao	2	S97 S168	Non-Commercial
T12	Eco	1	S55	N/A
T13	Edapt	1	S132	Non-Commercial

Table 6 (continued)

ID	Name	Citations	Cited by	Licence
T14	EMF	114	S03 S04 S06 S09 S10 S11 S13 S14 S15 S17 S18 S23 S24 S26 S28 S30 S34 S35 S37 S41 S42 S46 S47 S49 S50 S51 S53 S55 S56 S58 S60 S62 S63 S69 S70 S72 S73 S74 S84 S86 S88 S89 S90 S93 S99 S101 S107 S108 S109 S112 S114 S116 S117 S118 S119 S123 S125 S126 S127 S131 S135 S136 S138 S139 S142 S147 S150 S153 S155 S157 S160 S161 S163 S166 S172 S174 S175 S176 S177 S179 S180 S181 S182 S183 S184 S185 S186 S187 S189 S191 S192 S196 S197 S198 S199 S200 S203 S205 S207 S208 S210 S211 S212 S213 S214 S215 S216 S217 S218 S220 S226 S228 S229 S230	Non-Commercial
T15	EMFText	11	S4 S06 S13 S39 S48 S121 S171 S192 S206 S218 S228	Non-Commercial
T16	Enso	4	S105 S138 S204 S207	Non-Commercial
T17	Enterprise Architect	7	S51 S60 S71 S79 S141 S142 S202	Commercial
T18	Epsilon	8	S55 S58 S65 S110 S136 S149 S167 S200	Non-Commercial
T19	EuGENia Live	5	S39 S104 S156 S168 S219	Non-Commercial
T20	GEF	3	S191 S202 S139	Non-Commercial
T21	GEMOC	7	S11 S22 S34 S72 S129 S131 S160	Non-Commercial
T22	GME	6	S61 S63 S65 S146 S165 S228	Non-Commercial
T23	GMF	30	S19 S20 S33 S60 S61 S76 S93 S96 S108 S115 S117 S119 S127 S136 S139 S140 S152 S165 S167 S168 S179 S181 S191 S192 S200 S202 S205 S210 S211 S213	Non-Commercial
T24	GMP	2	S120 S202	Non-Commercial
T25	Gramada	1	S114	N/A
T26	Graphiti	8	S37 S60 S61 S168 S172 S191 S192 S228	Non-Commercial
T27	JET	5	S96 S108 S179 S181 S192	Non-Commercial

Table 6

(continued)

ID	Name	Citations	Cited by	Licence
T28	Kerneta	8	S11 S34 S46 S72 S131 S133 S160 S230	Non-Commercial
T29	Kitalpha	1	S159	Non-Commercial
T30	MagicDraw	3	S115 S137 S141	Commercial
T31	Marama	1	S65	Non-Commercial
T32	Más	2	S105 S204	N/A
T33	Melange	3	S11 S55 S72	Non-Commercial
T34	MetaEdit+	27	S29 S41 S42 S61 S63 S65 S69 S102 S105 S109 S120 S121 S123 S130 S133 S138 S139 S141 S165 S181 S192 S202 S204 S209 S219 S222 S228	Commercial
T35	Microsoft DSL tools	2	S42 S83	Commercial
T36	Microsoft Visual Studio	6	S41 S63 S92 S165 S176 S192	Commercial
T37	Monticore	7	S52 S72 S73 S80 S81 S133 S194	Non-Commercial
T38	MPS	48	S04 S10 S14 S16 S22 S38 S40 S41 S42 S43 S50 S52 S57 S60 S62 S68 S73 S80 S81 S101 S105 S106 S107 S111 S113 S121 S122 S123 S128 S130 S133 S134 S138 S148 S151 S153 S154 S162 S169 S170 S171 S204 S212 S216 S218 S222 S227 S228	Non-Commercial
T39	Nitra	1	S169	Commercial
T40	Obeco Designer	4	S61 S139 S171 S203	Commercial
T41	Onion	4	S41 S42 S105 S204	N/A
T42	Papyrus	18	S03 S31 S33 S58 S61 S71 S79 S86 S115 S119 S120 S137 S141 S147 S171 S200 S202 S220	Non-Commercial
T43	Rascal		S41 S42 S52 S59 S72 S105 S111 S114 S128 S130 S138 S189 S204 S207 S208	Non-Commercial
T44	RMT	1	S116	N/A

Table 6 (continued)

ID	Name	Citations	Cited by	Licence
T45	RSA	42	S71 S139	Non-Commercial
T46	Sirius		S15 S24 S26 S35 S37 S45 S60 S61 S65 S70 S71 S74 S76 S89 S90 S99 S112 S115 S123 S125 S126 S137 S139 S142 S155 S159 S160 S168 S174 S176 S179 S184 S187 S191 S192 S199 S200 S203 S205 S210 S211 S219	Non-Commercial
T47	Spoofax		S105 S114 S122 S128 S130 S133 S171 S204 S201 S4 S52 S72 S80 S81 S82 S138 S144 S189 S194 S218 S216	Non-Commercial
T48	Spray		S65 S168 S192	Non-Commercial
T49	SugarJ		S42 S41 S105 S194 S204	Non-Commercial
T50	Tau G2	1	S71	Commercial
T51	textX	1	S25	Non-Commercial
T52	Untitled tool	1	S169	N/A
T53	VL-Eli system	1	S168	N/A
T54	WebGME	6	S02 S21 S80 S145 S158 S209	Non-Commercial
T55	Whole	2	S105 S204	Non-Commercial
T56	Xbase	5	S08 S32 S44 S81 S227	Non-Commercial
T57	XMF-Mosaic	1	S71	Non-Commercial
T58	Xpand	7	S152 S153 S156 S166 S181 S192 S230	Non-Commercial
T59	Xtext	102	S01 S04 S05 S07 S08 S11 S12 S13 S17 S19 S23 S24 S26 S27 S28 S32 S36 S40 S41 S42 S44 S45 S47 S49 S50 S52 S56 S57 S58 S60 S62 S64 S66 S72 S73 S74 S75 S77 S78 S79 S80 S81 S82 S85 S87 S88 S89 S90 S91 S94 S95 S100 S101 S103 S105 S106 S110 S112 S114 S122 S123 S124 S126 S128 S130 S135 S137 S138 S143 S156 S161 S164 S167 S170 S173 S175 S176 S178 S181 S184 S185 S187 S188 S190 S192 S193 S194 S195 S199 S200 S204 S206 S207 S208 S211 S218 S221 S223 S224 S227 S228 S230	Non-Commercial

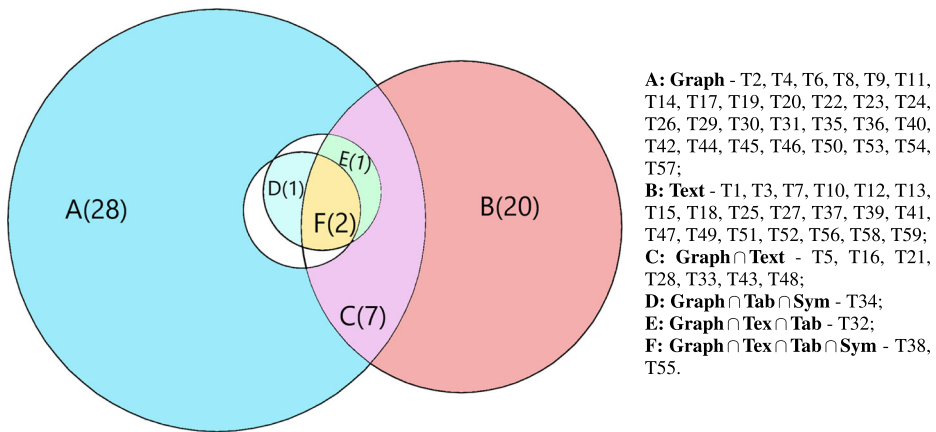


Fig. 5 Venn Diagram of Tools Notation Type

Figure 6 maps the application domain by publication year. Based on this data plot, it is possible to conclude that “DSL construction” is a mature research area, considering the many cross-application domains, thus characterizing an increasing interest by the research community.

It is noticed that 36% (83) of the considered studies are related to tools, approaches, or methods supporting some development process lifecycle stage of the DSL and DSML. Embedded systems correspond to 3% (seven studies). Web systems account for 1% (three studies), mobile applications appear in 3% (six studies) and multi-agent systems account for 2% (four studies). In addition, there are also four studies focusing on Cyber-Physical Systems, representing 2%. Domains mentioned by only one or two studies, such as aerospace systems, are classified as “Other”. This group accounts for 53%, (123 papers) of the studies, which shows the diversity of application domains.

RQ4. What features of the DSL creation process do these tools support?

Before discussing the results of this question, it is important to expose that when classifying the tools regarding their features, we grouped them into three different categories: LWs, DSL construction tools, and Utilities for toolchains. The main reason for doing this was due to some features not being relevant for some tools, *e.g.* editing mode features are not relevant for tools such as Acceleo and Xpand. It is also important to mention that the majority of studies do not provide complete information about the tools’ features, making it necessary to search for data on the gray literature to complement the information table. For example, several tools are proposed to assist only some aspects of the DSL development process (*e.g.* Acceleo, Xpand, Meta3, JET and XBase). As practitioners, it is acknowledged that these tools can be combined with several others, resulting in a set of tools complementing each other in a toolchain. However, their integration in possible toolchains is not discussed by the works.

As discussed on “DSL Development Mechanism Type” dimension in Table 4, the selection of studies presented tools that could be inserted in three main groups: 1) Features supporting the DSL construction through a common metamodel; 2) Features from toolboxes with integrated support for the context of DSLs, known as Language Workbenches; and 3) Tools providing utility features that could be adopted in more than one workbench, adopted as part of a toolchain for DSL construction.

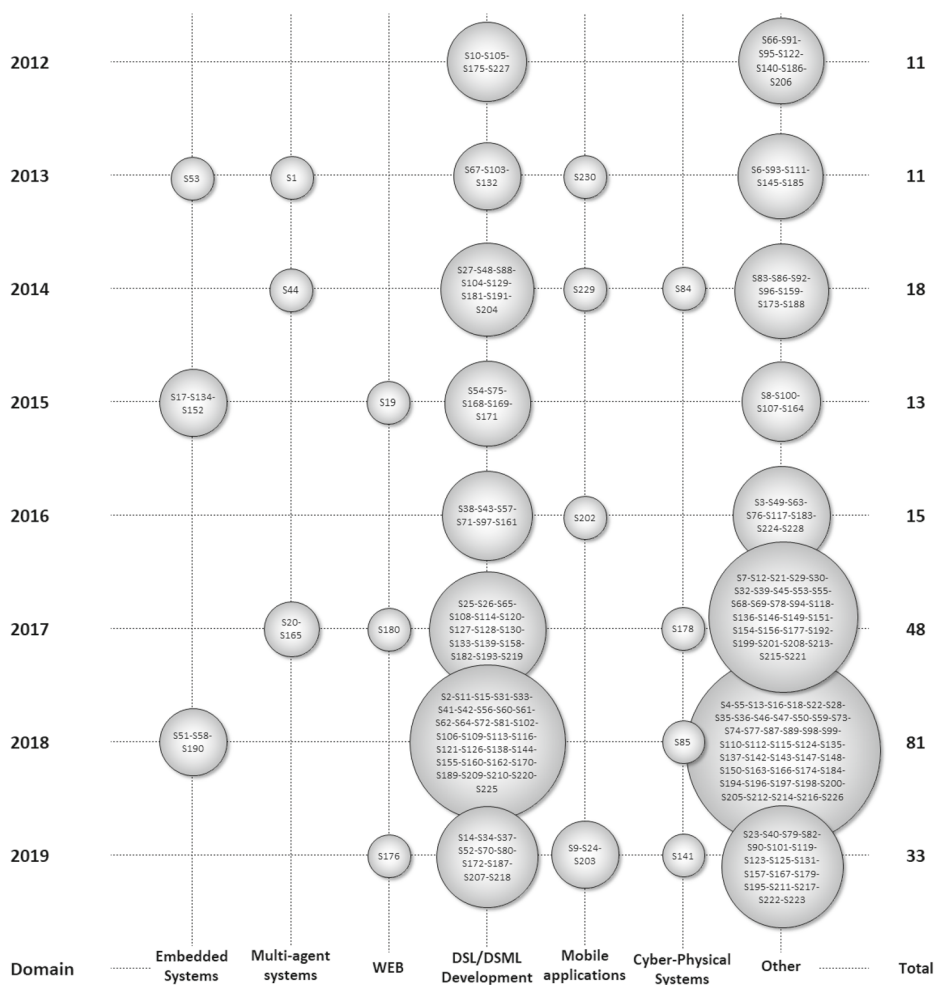


Fig. 6 Bubble Plot of the Studies Domain Distribution by Publication Year

Table 7 presents the tools classified as LWs. For this group, we used all the 34 features originally proposed for Language Workbenches (Erdweg et al. 2013), which are more complete toolboxes integrating homogeneous tools and delivering an environment that covers more aspects of the DSL development process. In this sense, it is worth highlighting the completeness of non-commercial LWs named Xtext, MPS and GEMOC Studio, as well as MetaEdit+. All of them cover several features on the DSL development process. MetaEdit+ (MetaCase 2017) covers 28 features; MPS (JetBrains 2017) covers 32 features, GEMOC Studio (GEMOC 2017) covers 31 features and Spoofox (Wachsmuth et al. 2014), Onion (Erdweg et al. 2013; Erdweg et al. 2015), and Whole (Erdweg et al. 2013; Erdweg et al. 2015) also cover many features of the DSL development process.

Six LWs cover both types of notation, graphical and textual. Among these LWs, we have GEMOC Studio and MPS, two non-commercial LWs that are still being maintained. GEMOC Studio is the only tool that contains a bi-directional representation of its models. That means that changes made in the textual model are also made in the graphical model in

Table 7 Language Workbenches Features

[illegible]

Legend: ¹Composability — ²Edit Mode — • Partial Graphical Support (Erdweg et al. 2015)

real-time and the opposite as well. GEMOC Studio is built on top of EMF, Xtext, and Sirius, being the reason why GEMOC Studio is one of the most complete open-source tools. GEMOC Studio also supports semantic execution and model simulation. Considering MPS, besides the graphical and textual notation, it also covers tabular specifications (feature 03). However, the graphical representation of the model (feature 02) was mentioned by (Erdweg et al. 2015) as being partially implemented in the LW. Argyle was also marked as covering graphical notation due to it supporting graphical representations of the specification of Software Product Lines as feature models.

The other two LWs that also cover both graphical and textual notations are not being maintained anymore: Enso, which is no longer being supported and Más, which was discontinued. Lastly, one of the most complete tools, MetaEdit+, despite its completeness, does not provide a specification for a textual notation, only a textual representation of graphic models.

Table 8 presents the features of the DSL construction tools. Among these, we emphasize the Eclipse DSL ecosystem including non-commercial frameworks such as EMF, covering 11 features, and Sirius (Eclipse F 2017a) covering 4.

Sirius, for example, is classified in terms of “translational or interpretative semantics” for the properties 5 “Model2Text”, 6 “Model2Model” and 8 “Interpretative”. In other words, Sirius may support DSL construction through the specifications of a Sirius model from an input EMF model, through code generation and interpretation for execution inside an Eclipse IDE.

It is also possible to notice that some DSL construction tools are covering many useful features. For example, Xtext (Eclipse F 2017b) covers 29 features. EMFText covers all features in the table, except the graphical notation due to the fact of EMFText allowing the definition of textual syntax from an Ecore model. Some tools are more focused and used just for modeling representations of a domain, such as MagicDraw used to modeling SysML, BPM, and UPDM languages. MagicDraw also provide transformation mechanisms, such as the transformation to XML and database models. Besides, Enterprise Architect provides support to data modeling and model simulation. Both tools, however, have commercial licenses. There are also non-commercial tools for domain modeling, such as Papyrus, a very complete solution that allows the use of UML profiles and SysML, being possible to perform real-time systems modeling.

Lastly, Table 9 presents Utilities Features for toolchains. For this classification, features such as validation and semantics were not considered as they are not relevant for these tools. From this group, it is important to mention Acceleo (Eclipse F 2020a) with 14 features, Xpand (Eclipse F 2020c) with 10 features, and XBase (Eclipse F 2020b) with nine. All tools in this table cover textual notation, this probably occurs because these tools have a textual aspect related to text and code generation. Also, they provide models transformations.

Meta3, Xpand, and Xbase covered both model-to-model and model-to-text transformation, while Acceleo and JET cover only model-to-text transformation. Lastly, the JET (Java Emitter Templates) tool, despite being cited by 5 studies, is not being maintained anymore.

6 Discussions

The previous section presented results strictly derived from the research questions. This section presents a broader discussion about how these contributions are supporting the automation of the whole DSL development process. Firstly, the results show that the number of studies related to DSL development technologies increased in the last years, more

Table 8 DSL Construction Tools Features

Feature																																				
Notation		Semantics					Validation					Testing					Comp ¹					EM ²					Syntactic services					Semantic Services				
ID	Tool	01	02	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34			
T2	ADOxx	✓																			✓															
T4	AToMPM	✓			✓																															
T5	CINCO	✓		✓	✓										✓	✓	✓	✓	✓																	
T6	Clooca	✓																																		
T8	DEViL	✓		✓	✓	✓	✓								✓		✓																✓			
T9	DoMe	✓			✓															✓																
T10	DrRacket	✓																																		
T11	DSL-Tao	✓			✓																		✓					✓	✓	✓	✓		✓			
T12	Eco	✓																																		
T13	Edapt	✓			✓																															
T14	EMF	✓			✓	✓											✓									✓		✓	✓	✓	✓					
T15	EMFText	✓		✓	✓	✓			✓	✓	✓		✓	✓	✓					✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓				
T17	Enterprise Architect	✓		✓	✓	✓						✓													✓											
T18	Epsilon	✓			✓							✓				✓					✓	✓						✓	✓	✓	✓					
T19	EuGENia Live	✓		✓	✓																															
T20	GEF	✓		✓	✓																															
T22	GME	✓		✓	✓	✓																						✓	✓	✓	✓					
T23	GMF	✓		✓	✓	✓																														
T24	GMP	✓																																		
T25	Gramada	✓					✓							✓	✓	✓																	✓			

Table 8 (continued)

Feature																																				
Notation		Semantics					Validation					Testing					Comp ¹					EM ²					Syntactic Services					Semantic Services				
ID	Tool	01	02	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34			
T26	Graphiti	✓		✓	✓														✓		✓		✓						✓	✓	✓					
T28	Kerneta	✓		✓	✓						✓										✓					✓				✓	✓					
T29	Kitalpha		✓																																	
T30	MagicDraw	✓		✓	✓					✓													✓							✓	✓	✓				
T31	Marama	✓		✓	✓																															
T33	Melange	✓		✓	✓																															
T36	MS Visual Studio	✓					✓																													
T39	Nitra	✓																																		
T42	Papyrus	✓		✓	✓																															
T44	RMT	✓		✓	✓																															
T45	RSA	✓		✓	✓												✓																			
T46	Sirius	✓		✓	✓																															
T48	Spray	✓		✓																																
T50	Tau G2	✓																																		
T51	textX	✓		✓	✓						✓	✓																								
T52	Tool with no name	✓																																		
T53	VL-Eli system		✓																																	
T54	WebGME		✓																																	
T56	XMF-Mosaic	✓		✓	✓					✓																										
T59	Xtext	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		

Legend: ¹Composability — ²Edit Mode

Table 9 Utility Features for Toolchains

ID		Feature																		Sem Serv ³			
		Not ¹			Semantics			Validation			Syntactic Serv ²												
		01	02		05	06	07	10	11	12	22	23	24	25	26	30	31	32					
T1	Acceleo	✓			✓		✓	✓	✓		✓		✓	✓	✓	✓	✓	✓					
T7	Meta3	✓			✓	✓		✓															
T27	JET	✓			✓																		
T57	XBase	✓			✓	✓	✓				✓		✓		✓	✓	✓	✓					
T58	Xpand	✓			✓	✓	✓		✓	✓	✓		✓		✓	✓	✓	✓					

Legend: ¹Notation — ²Semantic Services — ³Semantic Services

specifically since 2017. This result is more evident when comparing previous years, while a total of 162 papers were published between 2017 and 2019, only 68 were published from 2012 to 2016, increasing almost four times.

As we used previous works (Erdweg et al. 2013; Erdweg et al. 2015) as a basis for creating our protocol, it is important to mention how this SMS improved on the results from these works. Firstly, while their work only analyzed the features of LWs, ten in total, we considered also other types of technologies such as DSL construction tools, and utilities for toolchains. Another aspect that our work brings to improve in the last published related works is the number of technologies analyzed, 59 in total. In this sense, this SMS presents the feature classification of new LWs such as Argyle, GEMOC Studio, MS DSL Tools, Obeo Designer as presented in Table 7. Also, we contribute with 40 DSL construction tools presented in Table 8 and six utilities for toolchains presented in Table 9. Our work, however, does not extend the feature model proposed by (Erdweg et al. 2013) as our intention was not to update their model, only update the list of tools evaluated considering its features.

Considering the results for **RQ1** and **RQ4**, we have identified that the proposed tools support one activity of the DSL development, thus integration/toolchain mechanisms are essential. However, the surveyed studies do not provide details about these mechanisms, thus characterizing a limitation that hampers their selection for specific needs in DSL development processes. For this reason, regarding the contribution of the studies for specific domains, as illustrated in Figure 6, we also analyzed whether these studies are considering their DSLs not only as an isolated tool but also as an important part of the software development process. Through **RQ3**, we observed that DSLs have been constructed with several tools and applied to several cross-application domains. This characterizes a heterogeneous scenario where tool dependencies are very difficult to establish due to the lack of information provided by the studies. Thus, we present a list of research opportunities related to the limitations observed in the selected studies:

Bi-directional representation: Considering this topic, we only identified this feature to be present at GEMOC Studio, which handles real-time modifications for both the graphical and textual models. To the best of our knowledge, there is only one LW, Whole, allowing to choose among all four notations (Textual, Graphical, Tabular, Symbolic). This feature would make possible to visualize the different translations among these notations (*e.g.*, translation between graphical and textual notation). It is important to mention that MPS was considered to only implement the graphical notation partially.

Metamodel migration: Among the selected studies, it was not found any study presenting a tool supporting transformation of existing meta-metamodels. This feature can be found in two works (Kern 2016; Bruneliere et al. 2010), which allow the direct transformation among meta-metamodels by using bridges. Meta-model bridges must allow conversion among meta-metamodels such as Visio, GME, and ARIS. For example, (Kern 2016) allows transformations from the Ecore meta-metamodels to GOPRRR meta-metamodels, which are used by MetaEdit+. This kind of feature is related to some characteristics of MDE such as the use of models in all phases of the development process. These phases include the conception of the modeling tools, apply transformations to automatize software engineering related activities (Basso 2017), and solving interoperability problems (Bruneliere et al. 2010) among modeling tools.

Model animation and simulation : An emerging feature is the animation and simulation of DSL models. For instance, GEMOC Studio (GEMOC 2017) gives support to textual and

graphical DSL, besides offering models simulations and animations. This feature allows the definition of animators in graphical and textual DSL models. Then, these DSL models may be animated in run-time execution. The implementation of this feature could be a future trend for LWs.

7 Threats to validity

This section discusses the main threats to validity of our study and presents how we mitigated them based on (Cook and Campbell 1979; Wohlin et al. 2012).

Construct validity: A major threat to an SMS is the exclusion of relevant studies. To mitigate this threat, two researchers conducted an independent evaluation process, with well-defined exclusion and inclusion criteria, as well as quality assessment for qualification and classification of studies. Furthermore, during the quality assessment activity, Cohen's Kappa coefficient was applied to analyze the inter-rater agreement of the researcher's evaluations to mitigate the bias. Also, mapping studies might not guarantee the inclusion of all the relevant works in the field. For mitigating this threat, we performed a preliminary search in the scientific databases and analyzed whether the returned studies were relevant or not for answering our research questions.

However, although the number of selected studies is considered adequate for an SMS, we may not guarantee that all relevant studies are included. Another threat is the fact that only automated search was performed in this SMS, leaving aside other search approaches, such as snowballing and manual searches. Lastly, we understand that relevant studies may have been published before 2012, our start search year. However, as we included tools that were presented or only mentioned by the 230 studies analyzed, we believe that there is a very low chance of excluding relevant tools. Some evidence of this conclusion is presented in works already discussing tools prior to 2012 (Erdweg et al. 2013), as all tools covered by those works are also presented here.

Internal validity: This threat refers to the analysis of the selected studies.

In this sense, publication bias can be the inclusion of some studies which meet the study selection criterion but present low quality to answer the research questions.

We mitigated this threat by applying quality assessment questions, removing papers with scored zero (see Table 3). Other possible threat can be the low statistical power due to the selection of a small sample of studies. To mitigate this threat, we considered a large sample of studies when analyzing the DSL field to answer our research questions.

Another threat is the improper classification of the selected studies. In this sense, we mitigated this threat by adopting a classification proposed by other authors in the area (Erdweg et al. 2015), adding new features from tools to support technical decision making.

External validity: Issues with external factors, such as scientific digital libraries and paper unavailability, were solved by doing complementary searches on the Google Scholar database. In cases where studies did not mention the use of tools for developing DSL in the abstract or the conclusion, the study was entirely reviewed to find indications of the use of DSL development tools.

An additional threat is the exclusion of primary studies where authors did not use the specific DSL term when defining their language. This threat was mitigated by including more general terms, such as "Small Language", in the search string.

Other possible threat was the insufficient data about the tool features in the selected studies. We mitigated this threat by searching this information on websites and repositories of the tools. We are aware that some evidence of the existence of particular features for some of the tools might have been missed by the authors, for instance, the possibility of generating web-based and collaborative editors for the DSL usage.

Conclusion validity: In order to mitigate the conclusion bias, we followed the research protocol presented by (Petersen et al. 2008), a well-accepted research methodology for planning and conducting SMS. The low statistical power derived from the low quantity of the selected studies can be another threat. This is mitigated through the inclusion of six scientific databases, adopting an expressive search string adapted for each base, a search for tools mentioned in the papers and a rigorous analysis of tool features, which resulted in an adequate set of selected studies. Finally, we draw the conclusions only after collecting the results, thus avoiding the fishing and the error rate problem (Wohlin et al. 2012). Also, researchers, being unaware of their own bias during the analysis and classification of the studies, could have impacted the conclusion of the study. Beyond this, another threat to this SMS is a possible inaccuracy in data extraction. Hence, to mitigate both threats, two researchers independently performed the quality assessment and data extraction of each retrieved study. Moreover, in case of any divergences about a quality assessment or data extraction activities, two other specialists of the DSL area were assigned for discussing doubts. Then, they decided whether the study should or should not be included as primary study or whether a data should or should not be classified in such a way.

8 Conclusion

DSL development tools are reported as essential to construct and execute processes built on DSL tools. Therefore, it is relevant to map out these tools and their features. According to Kitchenham et al. (2010), “mapping studies can be of significant benefit to researchers in establishing baselines for further research activities”. Thus, this paper contributes with a Systematic Mapping Study (SMS), reporting DSL development tools and their features.

Based on the SMS results, it is possible to conclude that the DSL development area is an active research field. Several studies show DSL solutions that could be used to solve problems of specific domains (Selgrad et al. 2016; Hoyos et al. 2013; Córdoba-Sánchez and de Lara 2016; Zarrin and Baumeister 2014) or present some supporting tool to assist on DSL development (Rose et al. 2012; Wachsmuth et al. 2014; Viyović et al. 2014; Efftinge et al. 2012). To the best of our knowledge, this SMS is the first study focused on mapping DSL development tools and their features. In this SMS, the mapping focused on gathering information regarding the stages of the DSL development process, as well as tool features. In addition, we also sought to identify whether the tools are non-commercial or commercial and their respective application domains.

Our results indicate that there are only a few tools supporting a bidirectional DSL transformation between different workbenches. Besides, we also concluded that few DSLs tools support bi-directional/multiple notations. Bi-directional DSLs tools support more than one notation in the same tool, including graphical, textual, symbolic, and/or tabular notations. Likewise, another less explored tool feature is the support for customized graphical elements to represent concepts of the language application domain (e.g. .SVG, .EPS, .JPG), which help improving language expressiveness in relation to the domain.

Finally, this mapping study is relevant for research and practice in building DSLs. Considering the practice, the reported results may be used by DSL developers to identify which tools better fit their project's needs, the tool license type, and the domains benefiting from the proposed DSLs. The analysis of the results helps DSL researchers and practitioners to choose the tools for the development of a new DSL or to choose a set of tools to work together during the DSL development process. For researchers, this SMS may help to align new researches with a discussion of new trends, such as model animation and simulation.

References

- Åkesson A, Hedin G (2017) Jatte: A tunable tree editor for integrated DSLs, vol 2
- Akhundov J, Werner M, Schaus V, Gerndt A (2016) Using timed automata to check space mission feasibility in the early design phases. In: 2016 IEEE Aerospace Conference, pp 1–9
- Alvarez C, Casallas R (2013) MTC Flow: A Tool to Design, Develop and Deploy Model Transformation Chains. In: Proceedings of the Workshop on ACadeMics Tooling with Eclipse, (ACME'13). ACM, New York, NY, USA, pp 7:1–7:9
- Ángel MS, de Lara J, Neubauer P, Wimmer M (2018) Automated modelling assistance by integrating heterogeneous information sources. *Computer Languages, Systems & Structures* 53:90–120
- Antonelli HL, da Silva EAN, Fortes RPM (2015) A Model-driven Development for Creating Accessible Web Menus. *Procedia Computer Science* 67:95–104
- Arcaini P, Mirandola R, Riccobene E, Scandurra P (2019) A Pattern-Oriented Design Framework for Self-Adaptive Software Systems. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), IEEE, pp 166–169
- Arendt T, Taentzer G, Weber A (2013) Quality assurance of textual models within eclipse using OCL and model transformations. In: CEUR Workshop Proceedings, vol 1092, Miami, FL, United states, pp 1–10
- Arkin E, Tekinerdogan B (2014) Domain specific language for deployment of parallel applications on parallel computing platforms. In: ACM International Conference Proceeding Series (ICPS'14), Vienna, Austria, p University of Vienna
- Azadi M. E, Azadi M. E, Challenger M (2015) DSML4CP: A Domain-specific Modeling Language for Concurrent Programming. *Computer Languages, Systems and Structures* 44:319–341
- Barbosa A, Silva F, Coutinho L, Santos D, Teles A (2019) A Domain-Specific Modeling Language for Specification of Clinical Scores in Mobile Health. In: Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, SCITEPRESS-Science and Technology Publications, Lda, pp 104–113
- Barišić A, Amaral V, Goulão M. (2017) Usability driven DSL development with USE-ME. *Computer Languages, Systems and Structures* 51:1339–1351
- Barišić A, Blouin D, Amaral V, Goulão M (2017) A Requirements Engineering Approach for Usability-driven DSL Development. In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, (SLE'17). ACM, New York, NY, USA, pp 115–128
- Bartman B, Newman CD, Collard ML, Maletic JI (2017) SrcQL: A syntax-aware query language for source code. In: 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER'17), pp 467–471
- Basso FP (2017) RAS++: representing hybrid reuse assets for MDE as a service, Ph.D. Thesis, Universidade Federal do Rio de Janeiro
- Basso FP, Werner CML, de Oliveira TC (2017) Revisiting Criteria for Description of MDE Artifacts. In: 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, JSOS@ICSE, Buenos Aires, Argentina, May 23, 2017, pp 27–33
- Bencharqui H, Haidrar S, Anwar A (2019) Dealing with Requirement Inconsistencies Based on ReqDL Language. In: 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), IEEE, pp 1–6
- Bennani S, El Hamlaoui M, Nassar M, Ebersold S, Coulette B (2018) Collaborative model-based matching of heterogeneous models. In: 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), IEEE, pp 443–448
- Bergenti F (2014) An Introduction to the JADEL Programming Language. In: 2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI'14), pp 974–978

- Bermúdez R. FJ, Sánchez Ramón Ó, García M. J (2017) A tool to support the definition and enactment of model-driven migration processes. *Journal of Systems and Software* 128:106–129
- Bernardino M, Zorzo AF, Rodrigues EM (2016) Canopus: A Domain-Specific Language for Modeling Performance Testing. In: 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST'16), pp 157–167
- Bernardino M, Rodrigues E, Zorzo A, Marchezan L (2017) A Systematic Mapping Study on Model-Based Testing: Tools and Models. *IET Software* (2017) 11:141–155
- Bettini L (2014) Developing user interfaces with EMF parsley. In: Proceedings of the 9th International Conference on Software Paradigm Trends (ICSOFT-PT'14), Vienna, Austria, pp 58–66
- Bettini L (2019) Type errors for the IDE with Xtext and Xsemantics. *Open Computer Science* 9(1):52–79
- Bocciarelli P, D'Ambrogio A, Paglia E, Giglio A (2018) On the Performance Prediction Capabilities of the eBPMN-based Model-driven Method for Business Process Simulation. In: CIISE, pp 71–78
- Bonnet S, Voirin JL, Exertier D, Normand V (2016) Not (strictly) relying on SysML for MBSE: Language, tooling and development perspectives: The Arcadia/Capella rationale, pp 1–6
- Boßelmann S, Naujokat S, Steffen B (2018) On the difficulty of drawing the line. In: International Symposium on Leveraging Applications of Formal Methods, Springer, pp 340–356
- Boubeta-Puig J, Díaz G, Maciá H, Valero V, Ortiz G (2017) MEdit4CEP-CPN: an approach for complex event processing modeling by prioritized Colored Petri Nets. *Information Systems* 81:267–289
- Bousse E, Leroy D, Combemale B, Wimmer M, Baudry B (2018) Omniscient debugging for executable DSLs. *Journal of Systems and Software* 137:261–288
- Bousse E, Mayerhofer T, Combemale B, Baudry B (2019) Advanced and efficient execution trace management for executable domain-specific modeling languages. *Software & Systems Modeling* 18(1): 385–421
- Brdjanin D, Banjac D, Banjac G, Maric S (2018) An Online Business Process Model-driven Generator of the Conceptual Database Model. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, ACM, p 16
- Bruneliere H, Cabot J, Clasen C, Jouault F, Bézivin J (2010) Towards model driven tool interoperability: bridging Eclipse and Microsoft modeling tools. In: European Conference on Modelling Foundations and Applications, (ECMFA'2010), Springer, pp 32–47
- Buisson J, Rehab S (2018) Effective Bridging Between Ecore and Coq: Case of a Type-Checker with Proof-Carrying Code. In: International Symposium on Modelling and Implementation of Complex Systems, Springer, pp 259–273
- Bünder H (2019) Decoupling Language and Editor-The Impact of the Language Server Protocol on Textual Domain-Specific Languages. In: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, SCITEPRESS-Science and Technology Publications, Lda, pp 129–140
- Burdusel A, Zschaler S, Strüber D (2018) MDEoptimiser: a search based model engineering tool. In: Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, ACM, pp 12–16
- Butting A, Dalibor M, Leonhardt G, Rumpe B, Wortmann A (2018) Deriving Fluent Internal Domain-specific Languages from Grammars. In: Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2018, 2018, pp 187–199
- Butting A, Eikermann R, Kautz O, Rumpe B, Wortmann A (2018) Controlled and Extensible Variability of Concrete and Abstract Syntax with Independent Language Features. In: Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems, ACM, pp 75–82
- Caramujo J, da Silva AR, Monfared S, Ribeiro A, Calado P, Breaux T (2019) RSL-IL4Privacy: a domain-specific language for the rigorous specification of privacy policies. *Requirements Engineering* 24(1):1–26
- Cariou E, Le Goar O, Brunschwig Léa, Barbier F (2018) A generic solution for weaving business code into executable models. In: MODELS Workshops, pp 251–256
- Challenger M, Demirkol S, Getir S, Mernik M, Kardas G, Kosar T (2014) On the use of a domain-specific modeling language in the development of multiagent systems. *Engineering Applications of Artificial Intelligence* 28:111–141
- Chlipala A, Delaware B, Duchovni S, Gross J, Pit-Claudel C, Suriyakarn S, Wang P, Ye K (2017) The end of history? Using a proof assistant to replace language design with library design. In: Leibniz International Proceedings in Informatics, (LIPIcs'17), vol 71
- Combemale B, Barais O, Wortmann A (2017) Language engineering with the GEMOC studio. In: Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, (ICSAW'17): Side Track Proceedings, vol 54, pp 189–191
- Combemale B, Kienzle J, Mussbacher G, Barais O, Bousse E, Cazzola W, Collet P, Degueule T, Heinrich R, Jézéquel J-M, et al. (2018) Concern-oriented language development (COLD): Fostering reuse in language engineering. *Computer Languages, Systems & Structures* 54:139–155

- Cook TD, Campbell DT (1979) Quasi-Experimentation: Design and Analysis Issues for Field Settings, Houghton Mifflin
- Córdoba-Sánchez I, de Lara J (2016) Ann: A domain-specific language for the effective design and validation of Java annotations. *Computer Languages, Systems and Structures* 45:164–190
- Corral Diaz MA (2018) Software Development Tools in Model-Driven Engineering. In: *Proceedings-2017 5th International Conference in Software Engineering Research and Innovation, CONISOFT 2017, 2018-January*, volumen 2018, *Proceedings-2017 5th International Conference in Software Engineering*
- Coulon F, Degueule T, Van Der Storm T, Combemale B (2018) Shape-diverse DSLs: languages without borders (vision paper). In: *Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, ACM*, pp 215–219
- Crapo AW, Moitra A (2019) Using OWL ontologies as a domain-specific language for capturing requirements for formal analysis and test case generation. In: *2019 IEEE 13th International Conference on Semantic Computing (ICSC), IEEE*, pp 361–366
- de Almeida Pereira DI, Malki O, Bon P, Perin M, Collart-Dutilleul S (2018) An MDA approach for the specification of relay-based diagrams. In: *International Conference on Model and Data Engineering, Springer*, pp 17–29
- De F. CD, Moreira A, Araújo J, Amaral V (2017) Towards security modeling of E-voting systems
- de la Vega A, García-Saiz D, Zorrilla M, Sánchez P (2018) Flandm: a development framework of domain-specific languages for data mining democratisation. *Computer Languages, Systems & Structures* 54:316–336
- De Sousa LM, Da Silva AR (2016) A domain specific language for spatial simulation scenarios. *GeoInformatica* 20(1):117–149
- de Sousa LM, da Silva AR (2018) Usability evaluation of the domain specific language for spatial simulation scenarios. *Cogent Engineering* 5(1):1436889
- Dejanović I, Vadera N, Milosavljević G, Vuković (2017) TextX: A Python tool for Domain-Specific Languages implementation. *Knowledge-Based Systems* 115:1–4
- Demirkol S, Challenger M, Getir S, Kosar T, Kardas G, Memik M (2013) A DSL for the development of software agents working within a semantic web environment. *Computer Science and Information Systems* 10(4 SPEC.ISSUE):1525–1556
- Denkers J, van Gool L, Visser E (2018) Migrating custom DSL implementations to a language workbench (tool demo). In: *Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, ACM*, pp 205–209
- Derakhshanmanesh M, Ebert Jürgen, Grieger M, Engels G (2019) Model-integrating development of software systems: a flexible component-based approach. *Software & Systems Modeling* 18(4):2557–2586
- do Nascimento LM, Viana DL, Neto PAS, Martins DA, Garcia VC, Meira SR (2012) A systematic mapping study on domain-specific languages. In: *Proceedings of the 7th International Conference on Software Engineering Advances (ICSEA'12)*, pp 179–187
- Dörndorfer J, Hopfensperger F, Seel C (2019) The SenSoMod-Modeler—A Model-Driven Architecture Approach for Mobile Context-Aware Business Applications. In: *International Conference on Advanced Information Systems Engineering, Springer*, pp 75–86
- Dupont G, Mustafiz S, Khendek F, Toeroe M (2018) Building Domain-specific Modelling Environments with Papyrus: An Experience Report. In: *Proceedings of the 10th International Workshop on Modelling in Software Engineering. MiSE '18, 2018*, pp 49–56
- Dwarakanath A, Era D, Priyadarshi A, Dubash N, Podder S (2017) Accelerating Test Automation through a Domain Specific Language
- Eclipse F (2017) Sirius. Available in: <https://www.eclipse.org/sirius/>
- Eclipse F (2017) Sirius. Available in: <https://www.eclipse.org/Xtext/>
- Eclipse F (2020) Aceleo. Available in: <https://www.eclipse.org/aceleo/>
- Eclipse F (2020) Xbase. Available in: <https://wiki.eclipse.org/Xbase>
- Eclipse F (2020) Xpand. Available in: <https://www.eclipse.org/modeling/m2t/?project=xpand>
- Efftinge S, Eysholdt M, Köhnlein J., Zarnekow S, von Massow R, Hasselbring W, Hanus M (2012) Xbase: Implementing Domain-specific Languages for Java. *SIGPLAN Not.* 48(3):112–121
- Elaasar M, Noyrit F, Badreddin O, Gérard Sébastien (2018) Adaptation and Implementation of the ISO42010 Standard to Software Design and Modeling Tools. In: *International Conference on Model-Driven Engineering and Software Development, Springer*, pp 236–258
- Erdweg S, Van Der Storm T, Volter M, Boersma M, Bosman R, Cook WR, Gerritsen A, Hulshout A (2013) The State of the Art in Language Workbenches. In: *Software Language Engineering, (SLE'13). Springer International Publishing, Cham*, pp 197–217
- Erdweg S, Van Der Storm T, Völter M, Tratt L, Bosman R, Cook WR, Gerritsen A, Hulshout A, Kelly S, Loh A, et al. (2015) Evaluating and comparing language workbenches: Existing results and benchmarks for the future. *Computer Languages, Systems & Structures* 44:24–47

- Essadi N, Anwar A (2018) Towards A Language Interface Design to Coordinate Between Heterogeneous DSLs. In: 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), IEEE, pp 12–17
- Falkner K, Chiprianov V, Falkner N, Szabo C, Puddy G (2013) Modeling scenarios for the performance prediction of distributed real-time embedded systems. In: Military Communications and Information Systems Conference (MilCIS'13) 2013, pp 1–6
- Forbrig P (2018) Supporting Collaborative Decision Making in Software Engineering. In: In Proceedings of The 2018 Workshop on PhD Software Engineering Education: Challenges, Trends and Programs (SWEPHD2018)
- Fowler M (2010) Domain-specific languages, Pearson Education
- Gambo A, Syriani E (2019) Improving user productivity in modeling tools by explicitly modeling workflows. *Software & Systems Modeling* 18(4):2441–2463
- García-Daz V, Espada JP, Crespo RG, Pelayo G, Bustelo BC, Cueva Lovelle JM (2018) An approach to improve the accuracy of probabilistic classifiers for decision support systems in sentiment analysis. *Applied Soft Computing* 67(C):822–833
- Gargantini A, Radavelli M (2018) Migrating combinatorial interaction test modeling and generation to the web. In: 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, pp 308–317
- Gargantini A, Vavassori P (2012) CITLAB: A Laboratory for Combinatorial Interaction Testing. In: Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, (ICST'12). IEEE Computer Society, Washington, DC, USA, pp 559–568
- Garmendia A, Guerra E, De Lara J, García-Domínguez A, Kolovos D (2019) Scaling-up domain-specific modelling languages through modularity services. *Information and Software Technology* 115:97–118
- Gavran I, Mailahn O, Muller R, Peifer R, Zufferey D (2018) Tool: accessible automated reasoning for human robot collaboration. In: Proceedings of the 2018 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, ACM, pp 44–56
- GEMOC, I (2017) GEMOC Studio. Available in: <http://gemoc.org/studio.html>
- Gibbs I, Dascalu S, Harris Jr. FC (2015) A Separation-based UI Architecture with a DSL for Role Specialization. *Journal of Systems and Software*. 101(C):69–85
- Gómez-Abajo P, Guerra E, de Lara J, Merayo MG (2018) A tool for domain-independent model mutation. *Science of Computer Programming* 163:85–92
- Gonnord L, Mosser S (2018) Practicing domain-specific languages: from code to models. In: Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, ACM, pp 106–113
- Gossen F, Margaria T, Murtovi A, Naujokat S, Steffen B (2018) DSLs for decision services: a tutorial introduction to language-driven engineering. In: International Symposium on Leveraging Applications of Formal Methods, Springer, pp 546–564
- Granchelli G, Cardarelli M, Francesco PD, Malavolta I, Iovino L, Salle AD (2017) Towards recovering the software architecture of microservice-based systems. In: Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, (ICSAW'17): Side Track Proceedings, pp 46–53
- Guelfi N, Jahić B, Ries B (2017) TESMA: Requirements and design of a tool for educational programs. *Information (Switzerland)* 8(1):37
- Haitzer T, Zdun U (2014) Semi-automated architectural abstraction specifications for supporting software evolution. *Science of Computer Programming* 90(PART B):135–160
- Hasan S, Dubey A, Chhokra A, Mahadevan N, Karsai G, Koutsoukos X (2017) A modeling framework to integrate exogenous tools for identifying critical components in power systems, 2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES'17), pp 1–6
- Häser F, Felderer M, Breu R (2016) An integrated tool environment for experimentation in domain specific language engineering. In: ACM International Conference Proceeding Series, (ICPS'16), vol 01-03-June
- Häser F, Felderer M, Breu R (2018) Evaluation of an Integrated Tool Environment for Experimentation in DSL Engineering. In: International Conference on Software Quality, Springer, pp 147–168
- Heinrich R, Strittmatter M, Reussner RH (2019) A Layered Reference Architecture for Metamodels to Tailor Quality Modeling and Analysis. *IEEE Transactions on Software Engineering*
- Heitkötter H (2012) A Framework for Creating Domain-specific Process Modeling Languages. *Icsoft*, pp 127–136
- Henriques PR, Pereira MJV, Mernik M, Lenič M., Avdičaušević E, Žumer V (2002) Automatic generation of language-based tools. *Electronic notes in theoretical computer science* 65(3):77–96
- Herrera AS-B (2014) Enhancing ttext for general purpose languages. In: CEUR Workshop Proceedings, vol 1321, Valencia, Spain
- Hinkel G, Goldschmidt T, Burger E, Reussner R (2017) Using internal domain-specific languages to inherit tool support and modularity for model transformations. *Software and Systems Modeling*, pp 1–27

- Hiya S, Hisazumi K, Fukuda A, Nakanishi T (2013) clooca: Web based tool for Domain Specific Modeling. In: Demos/Posters/StudentResearch@ ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS'13), pp 31–35
- Hoffmann B, Chalmers K, Urquhart N, Farrenkopf T, Guckert M (2018) Towards Reducing Complexity of Multi-agent Simulations by Applying Model-Driven Techniques. In: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, pp 187–199
- Hoisl B, Sobernig S, Strembeck M (2017) Reusable and generic design decisions for developing UML-based domain-specific languages. *Information and Software Technology* 92:49–74
- Hojaji F, Zamani B, Hamou-Lhadj A, Mayerhofer T, Bousse E (2019) Lossless compaction of model execution traces. *Software & Systems Modeling*, pp 1–32
- HoseinDoost S, Adamzadeh T, Zamani B, Fatemi A (2017) A model-driven framework for developing multi-agent systems in emergency response environments. *Software and Systems Modeling*, pp 1–28
- Hoyos JR, Garcia-Molina J, Botia JA (2013) A Domain-specific Language for Context Modeling in Context-aware Systems. *Journal of Systems and Software* 86(11):2890–2905
- Huang C, Osaka A, Kamei Y, Ubayashi N (2015) Automated DSL Construction Based on Software Product Lines. In: Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development, (MODELSWARD'15). SCITEPRESS - Science and Technology Publications, Lda, Portugal, pp 247–254
- Idani A, Ledru Y, Wakrime AA, Ayed RB, Bon P (2019) Towards a tool-based domain specific approach for railway systems modeling and validation. In: International Conference on Reliability, Safety, and Security of Railway Systems, Springer, pp 23–40
- Iglesias A, Iglesias-Urkia M, López-Davalillo B, Charramendieta S, Urbietta A (2019) Trilateral: Software product line based multidomain iot artifact generation for industrial cps. In: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, Modelsward
- Iliasov A, Romanovsky A (2013) SafeCap domain language for reasoning about safety and capacity. Proceedings - 2012 Workshop on Dependable Transportation Systems/Recent Advances in Software Dependability, (WDTS-RASD'12), pp 1–10
- Jung A?B, Carbonell J, Marchezan L, Rodrigues E, Bernardino M, Basso F, Medeiros B (2020) Systematic Mapping Study on Domain-Specific Language Development Tools - Data Repository, Zenodo. <https://doi.org/10.5281/zenodo.3963379>
- Jacob F, Wynne A, Liu Y, Gray J (2014) Domain-specific languages for developing and deploying signature discovery workflows. *Computing in Science and Engineering* 16(1):52–64
- Jafer S, Chhaya B, Durak U (2017) Graphical specification of flight scenarios with aviation scenario definition language (ASDL). In: AIAA Modeling and Simulation Technologies Conference, 2017 (AIAA SciTech'17)
- Jager S, Maschotta R, Jungebloud T, Wichmann A, Zimmermann A (2016) Creation of domain-specific languages for executable system models with the Eclipse Modeling Project. In: 10th Annual International Systems Conference, (SysCon'16) - Proceedings
- Jakumeit E, Buchwald S, Wagelaar D, Dan L, Hegedus Á, Herrmannsdorfer M, Horn T, Kalnina E, Krause C, Lano K, Lepper M, Rensink A, Rose L, Watzoldt S, 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. <http://dx.doi.org/10.1016/j.scico.2013.10.009>
- JetBrains (2017) MPS. Available in: <https://www.jetbrains.com/mps/>
- Jeusfeld MA (2009) Metamodel. In: Encyclopedia of Database Systems. Springer US, Boston, MA, pp 1727–1730
- Jézéquel J-M, Combemale B, Barais O, Monperrus M, Fouquet F (2015) Mashup of Metalanguages and Its Implementation in the Kermeta Language Workbench. *Software and Systems Modeling* 14(2):905–920
- Jinzh L, Törngren M, Chen De-Jiu, Wang J (2017) A tool integration language to formalize co-simulation tool-chains for Cyber-Physical System (CPS). In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems A satellite event of SEFM2017-15th International conference on Software Engineering and Formal Methods, Springer
- Johnson RE (1997) frameworks = (Components + Patterns). *Communications of ACM* 40(10):39–42
- Jr A, Benedito F, Coutinho L, Silva F, Roriz M, Endler M (2019) A mobility restriction authoring tool approach based on a domain specific modeling language and model transformation. In: International Conference on Enterprise Information Systems (ICEIS), pp 525–534
- Jrad AB, Bhiri S, Tata S (2019) STRATFram: A framework for describing and evaluating elasticity strategies for service-based business processes in the cloud. *Future Generation Computer Systems* 97:69–89
- Kahani N (2018) AutoModel: A Domain-Specific Language for Automatic Modeling of Real-Time Embedded Systems. In: 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), pp 515–517

- Kalnins A, Barzdins J (2019) Metamodel specialization for graphical language support. *Software & Systems Modeling* 18(3):1699–1735
- Kanav S (2018) A modular approach to integrate verification tools in model based development. In: *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ACM, pp 150–155
- Karol S, Nett T, Castrillon J, Sbalzarini IF (2018) A Domain-Specific Language and Editor for Parallel Particle Methods. *ACM Trans. Math. Softw.* 44:34:1–34:32. <https://doi.org/10.1145/3175659>
- Kelly S, Lyytinen K, Rossi M (1996) MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: *8th International Conference on Advances Information System Engineering (CAISE'96)*. Springer-Verlag, London, UK, pp 1–21
- Kern H (2016) Model Interoperability between Meta-Modeling Environments by using M3-Level-Based Bridges. Ph.D. Thesis, Universität Leipzig
- Kitchenham BA, Budgen D, Brereton OP (2010) The value of mapping studies-A participant-observer case study. In: (EASE'10), vol 10, pp 25–33
- Kitchenham BA, Budgen D, Brereton OP (2011) Using mapping studies as the basis for further research—a participant-observer case study. *Information and Software Technology* 53(6):638–651
- Korenkov Y, Loginov I, Lazdin A (2015) PEG-based language workbench. In: *Conference of Open Innovation Association, (FRUCT'15)*, vol 2015-June, Yaroslavl, Russia, pp 75–81
- Kosar T, Bohra S, Mernik M (2016) Domain-specific languages: A systematic mapping study. *Information and Software Technology* 71:77–91
- Koschke R, Schmidt U, Berger B (2018) [engineering paper] built-in clone detection in meta languages. In: *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp 165–170
- Koster N, Wrede S, Cimiano P (2018) A Model Driven Approach for Eased Knowledge Storage and Retrieval in Interactive HRI Systems. In: *2018 Second IEEE International Conference on Robotic Computing (IRC)*, 2018, pp 113–120
- Kövesdán G, Lengyel L (2019) Meta3: a code generator framework for domain-specific languages. *Software & Systems Modeling* 18(4):2421–2439
- Kowalski M, Magott J (2012) Time coordination of heterogeneous distance protections using a domain specific language. *E-Informatica Software Engineering Journal* 6(1):7–26
- Krasts O, Kleins A, Teilans A (2012) Domain specific language for securities settlement systems. In: *Digital Information Processing and Communications (ICDIPC'12)*, Second International Conference on, pp 80–83
- Krüger S, Späth J, Ali K, Bodden E, Mezini M (2018) CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. In: *32nd European Conference on Object-Oriented Programming (ECOOP 2018)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
- Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. *biometrics*, pp 159–174
- Le DM, Dang D-H, Nguyen V-H (2018) On Domain Driven Design Using Annotation-Based Domain Specific Language. *Computer Languages, Systems & Structures*
- Le G. O, Waltham S (2013) Yet Another DSL for Cross-platforms Mobile Development. In: *Proceedings of the First Workshop on the Globalization of Domain Specific Languages, (GlobalDSL'13)*. ACM, New York, NY, USA, pp 28–33
- Lelandais B, Oudot M-P, Combemale B (2018) Fostering Metamodels and Grammars Within a Dedicated Environment for HPC: The NabLab Environment (Tool Demo). In: *Proceedings of the 11th ACM SIG-PLAN International Conference on Software Language Engineering, SLE 2018*. ACM, New York, NY, USA, pp 200–204
- Lemazurier L, Chapurlat V, Grossetête A (2017) An MBSE Approach to Pass from Requirements to Functional Architecture. *IFAC-PapersOnLine* 50(1):7260–7265
- Li X-S, Tao X-P, Song W, Dong K (2018) AocML: A Domain-Specific Language for Model-Driven Development of Activity-Oriented Context-Aware Applications. *Journal of Computer Science and Technology* 33(5):900–917
- Liebel G, Marko N, Tichy M, Leitner A, Hansson Jörgen (2014) Assessing the state-of-practice of model-based engineering in the embedded systems domain. In: *Model-Driven Engineering Languages and Systems, MODELS'14*, pp 166–182
- López-Fernández JJ, Garmendia A, Guerra E, de Lara J (2019) An example is worth a thousand words: Creating graphical modelling environments by example. *Software & Systems Modeling* 18(2):961–993
- Ma T, Sallai J (2017) MiW: A domain specific modeling environment for complex molecular systems. *Procedia Computer Science* 108:1232–1241
- Macías F, Wolter U, Rutle A, Durán F, Rodríguez-Echeverría R (2019) Multilevel coupled model transformations for precise and reusable definition of model behaviour. *Journal of Logical and Algebraic Methods in Programming* 106:167–195

- Makedonski P, Adamis G, Käärik M, Kristoffersen F, Carignani M, Ulrich A, Grabowski J (2019) Test descriptions with ETSI TDL. *Software Quality Journal*, pp 1–33
- Marchezan L, Bolfé G, Rodrigues E, Bernardino M, Basso FP (2019) Thoth: A Web-based Tool to Support Systematic Reviews. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp 1–6
- Maro S, Steghöfer J-P, Anjorin A, Tichy M, Gelin L (2015) On Integrating Graphical and Textual Editors for a UML Profile Based Domain Specific Language: An Industrial Experience. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering, (SLE'15). ACM, New York, NY, USA, pp 1–12
- Maróti M, Kecskés T, Kereskényi R, Broll B, Völgyesi P, Jurác L, Levendoszky T, Lédeczi A. (2014) Next generation (Meta)modeling: Web- and cloud-based collaborative tool infrastructure. *CEUR Workshop Proceedings* 1237:41–60
- Maschotta R, Wichmann A, Zimmermann A, Gruber K (2019) Integrated Automotive Requirements Engineering with a SysML-Based Domain-Specific Language. In: 2019 IEEE International Conference on Mechatronics (ICM), vol 1, IEEE, pp 402–409
- Mavridou A, Kecskés T, Zhang Q, Sztipanovits J (2018) A common integrated framework for heterogeneous modeling services. *CEUR Workshop Proceedings* 2245:416–422
- Mavropoulos O, Mouratidis H, Fish A, Panaousis E (2017) ASTo: A tool for security analysis of IoT systems. In: Proceedings - 2017 15th IEEE/ACIS International Conference on Software Engineering Research, Management and Applications, (SERA'17), pp 395–400
- Mayr-Dorn C, Laaber C (2017) A Domain-Specific Language for Coordinating Collaboration. 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA'17), pp 57–60
- Méndez-Acuna D, Galindo J, Degueule T, Combemale B, Baudry B (2016) Leveraging software product lines engineering in the development of external DSLs: a systematic literature review. *Computer Languages, Systems & Structures* 46:206–235
- Mendivelso LF, Garcés K, Casallas R (2018) Metric-centered and technology-independent architectural views for software comprehension. *Journal of Software Engineering Research and Development* 6(1):16
- Merino MV, Vinju J, van der Storm T (2018) Bacatá: a language parametric notebook generator (tool demo). In: Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, ACM, pp 210–214
- Mernik M, Heering J, Sloane AM (2005) When and how to develop domain-specific languages. *ACM computing surveys (CSUR)* 37(4):316–344
- MetaCase (2017) MetaEdit+. Available in: <http://www.metacase.com/mep/>
- Mettouris C, Achilleos A, Kapitsaki G, Papadopoulos GA (2018) The UbiCARS Model-Driven Framework: Automating Development of Recommender Systems for Commerce. In: European Conference on Ambient Intelligence, Springer, pp 37–53
- Mezhuyev V, Al-Emran M, Fatehah M, Hong NgChin (2018) Factors affecting the Metamodelling Acceptance: A Case Study from Software Development Companies in Malaysia. *IEEE Access* 6:49476–49485
- Mohamad RP, Kolovos DS, Paige RF (2015) Resource requirement analysis for web applications running in a virtualised environment. *Proceedings of the International Conference on Cloud Computing Technology and Science, (CLOUDCOM'15)* 2015-Febru(February):632–637
- Molina AI, Gallardo J, Redondo MA, Ortega M, Giraldo WJ (2013) Metamodel-driven Definition of a Visual Modeling Language for Specifying Interactive Groupware Applications: An Empirical Study. *Journal of Systems and Software* 86(7):1772–1789
- Molina PJ (2019) Quid: prototyping web components on the web. In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM, p 3
- Montenegro-Marin CE, Cueva-Lovelle JM, Sanjuán-Martínez O, García-Díaz V (2012) Domain specific language for the generation of learning management systems modules. *Journal of Web Engineering* 11(1):23
- Monthe VM, Nana L, Kouamou GE, Tangha C (2016) RsaML: A domain specific modeling language for describing robotic software architectures with integration of real-time properties. In: *CEUR Workshop Proceedings*, vol 1697
- Montrieux L, Yu Y, Wermelinger M (2013) Developing a domain-specific plug-in for a modelling platform: The good, the bad, the ugly. In: 2013 3rd International Workshop on developing Tools as Plug-ins (TOPI'13), pp 1–6
- Morgan R, Grossmann G, Schrefl M, Stumptner M, Payne T (2018) VizDSL: a visual DSL for interactive information visualization. In: *International Conference on Advanced Information Systems Engineering*, Springer, pp 440–455

- Mosteller D, Haustermann M, Moldt D, Schmitz D (2018) Graphical Simulation Feedback in Petri Net-based Domain-Specific Languages within a Meta-Modeling Environment. In: PNSE@ Petri Nets/ACSD, pp 57–76
- Mussbacher G, Amyot D, Breu R, Bruel J-M, Cheng BHC, Collet P, Combemale B, France RB, Hel-dal R, Hill J, Kienzle J, Schöttle M, Steimann F, Stikkolorum D, Whittle J (2014) The relevance of model-driven engineering thirty years from now. In: Model-Driven Engineering Languages and Systems, pp 183–200
- Nagele T, Hooman J (2017) Rapid Construction of Co-Simulations of Cyber-Physical Systems in HLA Using a DSL. 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA'17), pp 247–251
- Nakamura H, Nagano R, Hisazumi K, Kamei Y, Ubayashi N, Fukuda A (2012) QORAL: An External Domain-Specific Language for Mining Software Repositories. In: Proceedings of the 2012 Fourth International Workshop on Empirical Software Engineering in Practice, (IWESEP'12). IEEE Computer Society, Washington, DC, USA, pp 23–29
- Naujokat S, Lybecait M, Kopetzki D, Steffen B (2017) CINCO: A simplicity-driven approach to full generation of domain-specific graphical modeling tools. International Journal on Software Tools for Technology Transfer, pp 1–28
- Nazir A, Alam M, Malik SUR, Akhunzada A, Cheema MN, Khan MK, Ziang Y, Khan T, Khan A (2017) A high-level domain-specific language for SIEM (design, development and formal verification). Cluster Computing 20(3):2423–2437
- Neubauer P, Bill R, Mayerhofer T, Wimmer M (2017) Automated generation of consistency-achieving model editors. In: 24th International Conference on Software Analysis, Evolution and Reengineering (SANER'17), 2017 IEEE, IEEE, pp 127–137
- Nordmann A, Wrede S, Steil J (2015) Modeling of movement control architectures based on motion primitives using domain-specific languages. In: Proceedings - IEEE International Conference on Robotics and Automation (ICRA'15), vol 2015-June, Seattle, WA, United states, pp 5032–5039
- Ober I, Palyart M, Bruel J-M, Lugato D (2018) On the use of models for high-performance scientific computing applications: an experience report. Software & Systems Modeling 17(1):319–342
- Oliveira B, Belo O (2017) On the specification of extract, transform, and load patterns behavior: A domain-specific language approach. Expert Systems, 34(1)
- OMG (2019) META-OBJECT FACILITY. Available in: <https://www.omg.org/spec/MOF/2.4.1/PDF>
- Ouared A, Ouhammou Y, Bellatreche L (2018) QoS MOS: QoS metrics management tool suite. Computer Languages, Systems & Structures 54:236–251
- Pasternak M, Kahani N, Bagherzadeh M, Dingel J, Cordy JR (2018) Simgen: a tool for generating simulations and visualizations of embedded systems on the unity game engine. In: Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, ACM, pp 42–46
- Pescador A, Garmendia A, Guerra E, Sanchez Cuadrado JS, De Lara J (2015) Pattern-based development of Domain-Specific Modelling Languages. In: 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems, (MODELS'15) - Proceedings, Ottawa, ON, Canada, pp 166–175
- Pescador A, Lara JD (2016) DSL-Maps: From Requirements to Design of Domain-Specific Languages. 31st IEEE/ACM International Conference on Automated Software Engineering (ASE'16), pp 438–443
- Petersen K, Feldt R, Muftaba S, Mattsson M (2008) Systematic Mapping Studies in Software Engineering. 12th Int. Conf. on Evaluation and Assessment in Software Engineering 17(1):1–10
- Pietrusiewicz K (2019) Metamodelling for Design of Mechatronic and Cyber-Physical Systems. Applied Sciences 9(3):376
- Pomante L, Candia S, Incerto E (2015) A Model-Driven approach for the development of an IDE for Spacecraft on-board software. In: 2015 IEEE Aerospace Conference, pp 1–17
- Pérez F, Valderas P, Fons J (2013) A domain-specific language for enabling doctors to specify biomechanical protocols. In: 2013 IEEE Symposium on Visual Languages and Human Centric Computing, (IEEE-VL/HCC'13), pp 99–102
- Pérez-Berenguer D, García-Molina J (2019) INDIEAuthor: A Metamodel-Based Textual Language for Authoring Educational Courses. IEEE Access 7:51396–51416. <https://doi.org/10.1109/ACCESS.2019.2911884>
- Pröll R., Rumpold A, Bauer B (2018) Applying integrated domain-specific modeling for multi-concerns development of complex systems. In: Pires LF, Hammoudi S, Selic B (eds) Model-Driven Engineering and Software Development. Springer International Publishing, Cham, pp 247–271
- Rabiser R, Thanhofner-Pilisch J, Vierhauser M, Grünbacher P, Egged A (2018) Developing and evolving a DSL-based approach for runtime monitoring of systems of systems. Automated Software Engineering 25(4):875–915

- Rapos EJ, Stephan M (2019) IML: Towards an Instructional Modeling Language. In: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, SCITEPRESS-Science and Technology Publications, Lda, pp 417–425
- Ratiu D, Ulrich A (2017) Increasing usability of spin-based C code verification using a harness definition language: Leveraging model-driven code checking to practitioners. In: SPIN 2017 - Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software (SPIN'17), pp 60–69
- Ratiu D, Voelter M (2016) Automated Testing of DSL Implementations - Experiences from Building mbeddr. In: 2016 IEEE/ACM 11th International Workshop in Automation of Software Test (AST'16), pp 15–21
- Ratiu D, Ulrich A (2019) An integrated environment for Spin-based C code checking. *International Journal on Software Tools for Technology Transfer* 21(3):267–286
- Rein P, Hirschfeld R, Taeumel M (2016) Gramada: Immediacy in programming language development. In: Onward! 2016 - Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (SPLASH'16), pp 165–179
- Rensink A, Aksit M (2018) A Java Bytecode Metamodel for Composable Program Analyses. In: Software Technologies: Applications and Foundations: STAF 2017 Collocated Workshops, Marburg, Germany, July 17–21, 2017, Revised Selected Papers, vol 10748, Springer, p 30
- Ribeiro A, Da S AR (2014) XIS-Mobile: A DSL for mobile applications. In: Proceedings of the ACM Symposium on Applied Computing, (SIGAPP'14), pp 1316–1323
- Ribeiro A, De S. L, Da S. AR (2016) Comparative analysis of workbenches to support DSMLs: Discussion with non-trivial model-driven development needs. In: Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD'16), pp 323–330
- Ribeiro A, da Silva AR (2017) RSLingo4Privacy Studio - A Tool to Improve the Specification and Analysis of Privacy Policies. Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS'17), pp 52–63
- Ribić S, Turčinhodžić R, Muratović-Ribić A, Kosar T (2018) REDOSPLAT: A readable domain-specific language for timetabling requirements definition. *Computer Languages, Systems & Structures* 54:252–272
- Rieger C, Kuchen H (2018) A process-oriented modeling approach for graphical development of mobile business apps. *Computer Languages, Systems & Structures* 53:43–58. <https://doi.org/10.1016/j.cl.2018.01.001>
- Ries B, Capozucca A, Guelfi N (2018) Messir: a text-first DSL-based approach for UML requirements engineering (tool demo). In: Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, ACM, pp 103–107
- Ristić S (2017) How to apply model-driven paradigm in information system (Re) engineering. In: 2017 IEEE 14th International Scientific Conference on Informatics, IEEE, pp 6–11
- Rocha H, Durelli RS, Terra R, Bessa S, Valente MT (2017) DCL 2.0: modular and reusable specification of architectural constraints. *Journal of the Brazilian Computer Society*, 23(1)
- Rodriguez-Echeverria R, Izquierdo JLC, Wimmer M, Cabot J (2018) Towards a Language Server Protocol Infrastructure for Graphical Modeling. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, ACM, pp 370–380
- Rose LM, Kolovos DS, Paige RF (2012) EuGENia Live: A Flexible Graphical Modelling Tool. In: Proceedings of the 2012 Extreme Modeling Workshop (XM'2012), XM'12. ACM, New York, NY, USA, pp 15–20
- Ruiz-Rube I, Person T, Doderio JM, Mota JM, Sánchez-Jara JM (2019) Applying static code analysis for domain-specific languages. *Software & Systems Modeling*, pp 1–16
- Salehi P, Hamou-Lhadj A, Toeroe M, Khendek F (2018) A model-driven approach for the generation of configurations for highly available software systems. *Innovations in Systems and Software Engineering* 14(4):273–307
- Sandobalin J, Insfran E, Abrahao S (2017) An Infrastructure Modelling Tool for Cloud Provisioning. Proceedings - 2017 IEEE 14th International Conference on Services Computing (SCC'17), pp 354–361
- Santos AL, Gomes E (2016) Xdiagram: A declarative textual DSL for describing diagram editors (Tool Demo -). In: Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering, co-located with SPLASH 2016 (SLE'16), (SLE'16), pp 253–257
- Savić D, Da S AR, Vlajić S, Lazarević S, Antović I, Stanojević V, Milić M (2014) Preliminary experience using JetBrains MPS to implement a requirements specification language. Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, (QUATIC'14) 1:134–137
- Schmidt DC (2006) Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-* 39(2):25
- Schuts M, Hooman J, Tielemans P (2018) Industrial Experience with the Migration of Legacy Models Using a DSL. In: Proceedings of the Real World Domain Specific Languages Workshop 2018. RWDSL2018, 2018, pp 1:1–1:10


- Schwaiger WSA (2016) REA Business Management Ontology: Conceptual Modeling of Accounting, Finance and Management Control. In: CAiSE Forum
- Selgrad K, Lier A, Dörntlein J., Reiche O, Marc Stamminger M (2016) A High-Performance Image Processing DSL for Heterogeneous Architectures. In: Proceedings of the 9th European Lisp Symposium on European Lisp Symposium, (ELS'16). European Lisp Scientific Activities Association, pp 5:38–5:37
- Semeráth Oszkár, Varró Dániel (2018) Iterative Generation of Diverse Models for Testing Specifications of DSL Tools. In: FASE, pp 227–245
- Smits J, Visser E (2017) FlowSpec: Declarative Dataflow Analysis Specification. In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, (SLE'17). ACM, New York, NY, USA, pp 221–231
- Sorgalla J, Wizenty P, Rademacher F, Sachweh S, Zündorf A. (2018) Ajil: enabling model-driven microservice development. In: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ACM, p 1
- Stevanetic S, Zdun U (2018) Supporting the analyzability of architectural component models-empirical findings and tool support. Empirical Software Engineering 23(6):3578–3625
- Stocker K. AB, Washizaki HB, Fukazawa YB (2017) Closing the Gap between Unit Test Code and Documentation. Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW'17), pp 304–308
- Strömbäck F. (2018) Storm: A Language Platform for Interacting and Extensible Languages (Tool Demo). In: Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering. SLE 2018, 2018, pp 60–64
- Sutii AM, van den Brand M, Verhoeff T (2017) Exploration of modularity and reusability of domain-specific languages: An expression DSL in MetaMod. Computer Languages, Systems and Structures
- Szabo T, Alperovich S, Voelter M, Erdweg S (2016) An extensible framework for variable-precision data-flow analyses in MPS. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, (ASE'16), Singapore, Singapore, pp 870–875
- Tairas R, Cabot J (2015) Corpus-based Analysis of Domain-specific Languages. Software and Systems Modeling 14(2):889–904
- Tariq MU, Florence J, Wolf M (2014) Design specification of cyber-physical systems: Towards a domain-specific modeling language based on simulink, eclipse modeling framework, and giotto. In: CEUR Workshop Proceedings, vol 1250, pp 6–15
- Tekinerdogan B, Arkin E (2019) ParDSL: a domain-specific language framework for supporting deployment of parallel algorithms. Software & Systems Modeling 18(5):2907–2935
- Terzić B., Dimitrieski V, Kordić S., Milosavljević G., Luković I. (2018) Development and evaluation of MicroBuilder: a Model-Driven tool for the specification of REST Microservice Software Architectures. Enterprise Information Systems 12(8-9):1034–1057
- Tezel B, Challenger M, Kardas G (2018) Dsm14bdi: A modeling tool for bdi agent development. In: 12th turkish national software engineering symposium (uysm 2018)
- Theobald M, Palladino L, Virelizier P (2018) About DSML design based on standard and open-source - REX from SAFRAN tech work using Papyrus-SysML. International Journal of Electrical and Electronic Engineering and Telecommunications 7:70–75. <https://doi.org/10.18178/ijeetc.7.2.70-75>
- Tikhonova U (2017) Reusable specification templates for defining dynamic semantics of DSLs. Software and Systems Modeling, pp 1–30
- Tisi M, Cheng Z (2018) CoqTL: an Internal DSL for Model Transformation in Coq. In: International Conference on Theory and Practice of Model Transformations, Springer, pp 142–156
- Tolvanen J-P, Kelly S (2018) Effort Used to Create Domain-Specific Modeling Languages. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, ACM, pp 235–244
- Tragatschnig S, Stevanetic S, Zdun U (2018) Supporting the evolution of event-driven service-oriented architectures using change patterns. Information and Software Technology 100:133–146
- Tran N-H, Chiba Y, Aoki T (2017) Domain-specific language facilitates scheduling in model checking. In: 2017 24th Asia-Pacific Software Engineering Conference (APSEC), IEEE, pp 417–426
- Trobo IP, Díaz VG, Espada JG, Crespo RG, Moreno-Ger P (2019) Rapid modeling of human-defined AI behavior patterns in games. Journal of Ambient Intelligence and Humanized Computing 10(7):2683–2692
- Uhnák P, Pergl R (2016) The OpenPonk Modeling Platform. In: Proceedings of the 11th Edition of the International Workshop on Smalltalk Technologies, (IWST'16). ACM, New York, NY, USA, pp 14:1–14:11
- Vaderna R, Vuković Ž, Dejanović I, Milosavljević G (2018) Graph Drawing and Analysis Library and Its Domain-Specific Language for Graphs' Layout Specifications. Scientific Programming, pp 2018

- van den Berg F, Hooman J, Haverkort BR (2018) A Domain-Specific Language and Toolchain for Performance Evaluation Based on Measurements. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10740 LNCS:295–301. https://doi.org/10.1007/978-3-319-74947-1_21
- van den Berg F, Garousi V, Tekinerdogan B, Haverkort BR (2018) Designing cyber-physical systems with aDSL: A domain-specific language and tool support. In: 2018 13th Annual Conference on System of Systems Engineering (SoSE), IEEE, pp 225–232
- van Rozen R, van der Storm T (2017) Toward live domain-specific languages: From text differencing to adapting models at run time. *Software & Systems Modeling* 18:1–17
- van Rozen R, van der Storm T (2019) Toward live domain-specific languages. *Software & Systems Modeling* 18(1):195–212
- Viana M, Penteado R, Do P. A, Durelli R (2013) F3T: From features to frameworks tool. In: *Proceedings - 2013 27th Brazilian Symposium on Software Engineering, (SBES'13)*, Brasilia, DF, Brazil, pp 89–98
- Viana MC, Penteado RAD, do Prado AF (2013) Domain-Specific Modeling Languages to improve framework instantiation. *Journal of Systems and Software* 86(12):3123–3139
- Vieira MA, Carvalho ST (2017) Model-driven Engineering in the Development of Ubiquitous Applications: Technologies, Tools and Languages. In: *Proceedings of the 23rd Brazilian Symposium on Multimedia and the Web, (WebMedia'17)*. ACM, New York, NY, USA, pp 29–32
- Vinogradov S, Ozhigin A, Ratiu D (2015) Modern model-based development approach for embedded systems practical experience. In: *1st IEEE International Symposium on Systems Engineering, (ISSE'15)* - *Proceedings*, Rome, Italy, pp 56–59
- Visic N, Fill H-G, Buchmann RA, Karagiannis D (2015) A domain-specific language for modeling method definition: From requirements to grammar. In: *Proceedings - International Conference on Research Challenges in Information Science, (RCIS'15)*, vol 2015-June, Athens, Greece, pp 286–297
- Vissers Y, Mengerink JGM, Schiffelers RRH, Serebrenik A, Reniers MA (2017) Maintenance of specification models in industry using Edapt. In: *Specification and Design Languages (FDL)*, 2017 Forum on, IEEE, pp 1–6
- Vistbakka I, Barash M, Troubitsyna E (2018) Towards creating a DSL facilitating modelling of dynamic access control in Event-B. In: *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Springer, pp 386–391
- Viyović V, Maksimović M, Perišić B (2014) Sirius: A rapid development of DSM graphical editor. *IEEE 18th International Conference on Intelligent Engineering Systems, Proceedings (INES'14)*, pp 233–238
- Voelter M, Kolb B, Szabó T, Ratiu D, van Deursen A (2017) Lessons learned from developing mbeddr: a case study in language engineering with MPS. *Software and Systems Modeling*, pp 1–46
- Voelter M (2018) Fusing modeling and programming into language-oriented programming. In: *International Symposium on Leveraging Applications of Formal Methods*, Springer, pp 309–339
- Voelter M, Kolb B, Birken K, Tomassetti F, Alff P, Wiart L, Wortmann A, Nordmann A (2019) Using language workbenches and domain-specific languages for safety-critical software development. *Software & Systems Modeling* 18(4):2507–2530
- Voelter M, Ratiu D, Schaez B, Kolb B (2012) Mbeddr: An extensible c-based programming language and IDE for embedded systems. In: *Proceedings of the 2012 ACM Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH'12)*, Tucson, AZ, United states, pp 121–140
- Vögele C, van Hoorn A, Schulz E, Hasselbring W, Krcmar H (2018) WESSBAS: extraction of probabilistic workload specifications for load testing and performance predictions a model-driven approach for session-based application systems. *Software & Systems Modeling* 17(2):443–477
- Wachsmuth GH, Konat GDP, Visser E (2014) Language Design with the Spoofox Language Workbench. *IEEE Software* 31(5):35–43
- Walter T, Parreiras FS, Staab S (2014) An ontology-based framework for domain-specific modeling. *Software & Systems Modeling* 13(1):83–108
- Whittle J, Hutchinson J, Rouncefield M, Håkan B, Rogardt H (2015) A taxonomy of tool-related issues affecting the adoption of model-driven engineering. *Software & Systems Modeling*, pp 1–19
- Wienke J, Wigand D, Koster N, Wrede S (2018) Model-based performance testing for robotics software components. In: *2018 Second IEEE International Conference on Robotic Computing (IRC)*, IEEE, pp 25–32
- Wigand DL, Nordmann A, Goerlich M, Wrede S (2017) Modularization of domain-specific languages for extensible component-based robotic systems. *Proceedings - 2017 1st IEEE International Conference on Robotic Computing, (IRC'17)*, pp 164–171
- Wohlin C, Runeson P, Höst M, Ohlsson M, Regnell B (2012) *Experimentation in Software Engineering*, Springer
- Wu H, Gray J, Mernik M (2008) Grammar-driven generation of domain-specific language debuggers. *Software: Practice and Experience* 38(10):1073–1103

- Yakymets N, Sango M, Dhoubi S, Gelin R (2018) Model-Based Engineering, Safety Analysis and Risk Assessment for Personal Care Robots. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 6136–6141
- Yigitbas E, Anjorin A, Leblebici E, Grieger M (2018) Bidirectional Method Patterns for Language Editor Migration. In: European Conference on Modelling Foundations and Applications, Springer, pp 97–114
- Zabawa P, Hnatkowska B (2017) CDMM-F—domain languages framework
- Zarrin B, Baumeister H (2014) Design of a Domain-Specific Language for Material Flow Analysis Using Microsoft DSL Tools: An Experience Paper. In: Proceedings of the 14th Workshop on Domain-Specific Modeling, (DSM'14). ACM, New York, NY, USA, pp 23–28
- Zarrin B, Baumeister H (2018) An integrated framework to specify domain-specific modeling languages. In: 6th International Conference on Model-Driven Engineering and Software Development, pp 83–94
- Zarrin B, Baumeister H, Sarjoughian H (2018) An integrated framework to develop domain-specific languages: Extended case study. In: International Conference on Model-Driven Engineering and Software Development, Springer, pp 159–184
- Zhao T, Huang X (2018) Design and implementation of DeepDSL: A DSL for deep learning. *Computer Languages, Systems & Structures* 54:39–70
- Zhou N, Li D, Li S, Wang S, Liu C (2017) Model-Based Development of Knowledge-Driven Self-Reconfigurable Machine Control Systems. *IEEE Access* 5:19909–19919
- Zhu M, Wang AI (2017) RAIL: A Domain-Specific Language for Generating NPC Behaviors in Action/Adventure Game. In: International Conference on Advances in Computer Entertainment, Springer, pp 868–881
- Zhu Z, Lei Y, Zhu Y, Sarjoughian H (2017) Cognitive Behaviors Modeling Using UML Profile: Design and Experience. *IEEE Access* 5:21694–21708
- Zikra I (2012) Implementing the unifying meta-model for enterprise modeling and model-driven development: An experience report. In: Sandkuhl K, Seigerroth U, Stirna J (eds) *The Practice of Enterprise Modeling*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 172–187
- Zweihoff P, Naujokat S, Steffen B (2019) Pyro: Generating Domain-Specific Collaborative Online Modeling Environments. In: International Conference on Fundamental Approaches to Software Engineering, Springer, pp 101–115

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Aníbal lung¹  · João Carbonell¹ · Luciano Marchezan¹ · Elder Rodrigues¹ · Maicon Bernardino¹ · Fabio Paulo Basso¹ · Bruno Medeiros¹

João Carbonell
joaocarbonellpc@gmail.com

Luciano Marchezan
lucianomarchp@gmail.com

Elder Rodrigues
eldermr@gmail.com

Maicon Bernardino
bernardino@acm.org

Fabio Paulo Basso
fabiopbasso@gmail.com

Bruno Medeiros
brunobragamedeiros@gmail.com

¹ Federal University of Pampa (Unipampa), Av. Tiarajú, 97546-550, Alegrete, RS, Brazil