Usability Evaluation of Domain-Specific Languages: A Systematic Literature Review

Ildevana Poltronieri Rodrigues(✉), Márcia de Borba Campos, and Avelino F. Zorzo

PUCRS - Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil
ildevana.rodrigues@acad.pucrs.br,
{marcia.campos,avelino.zorzo}@pucrs.br

Abstract. Software developers have always been concerned with the quality of the products they produce. Although software engineers use new methods to evaluate the quality of their software, there are still some concerns in several of the methods they use when developing software, for example, when using Domain-Specific Languages (DSLs). One of the main goals of DSLs is to ease the work of developers in different areas. However, to achieve this goal it is necessary to provide an evaluation of the usability of such languages. Although it is possible to find some experiments to evaluate such languages, usually this experiments are subjective and do not use techniques from the Human-Computer Interaction (HCI) area. Therefore, this paper presents a Systematic Literature Review (SLR) in which a discussion on the usability of DSLs is presented. This paper also presents a mapping to show how usability has been assessed by researchers in their work.

Keywords: Human-Computer Interaction · Domain-Specific Languages · Systematic Literature Review · Usability evaluation

1 Introduction

The use of Domain Specific Languages (DSLs) eases software development through the appropriate abstractions and notations [4, 12, 16]. Some studies [4, 5, 12, 15] show the importance of DSLs for increasing productivity when developing a system and also for information exchange among experts from a domain [20]. Furthermore, the use of DSLs may facilitate the understanding of programming code, write it faster, and make it less prone to include faults [13]. There are DSLs for several different domains, for example, robotics [9, 23], software architectures anomalies [1], games [11, 14, 29, 32] or performance testing [6–8].

Although their advantages, some studies [7, 18] show that DSLs usage is not widespread. Among the problems that may prevent their use is the lack of systematic approaches to validate DSLs, mainly regarding their quality of use, such as efficiency, efficacy and usage satisfaction [3]. Furthermore, it is difficult to identify or quantify usability problems in DSLs [13].

M. Kurosu (Ed.): HCI 2017, Part I, LNCS 10271, pp. 522–534, 2017.
DOI: 10.1007/978-3-319-58071-5_39

Therefore, this paper presents a Systematic Literature Review (SLR) to understand the main concerns when designing and using DSLs. This SLR allowed identifying primary studies in Human-Computer Interaction (HCI) and Software Engineering (SE) that performed some form of usability evaluation of DSLs. From those studies, several different used terms were mapped to a taxonomy in usability evaluation of DSLs.

This paper is organized as follows. Section 2 briefly describes the background on DSL and HCI. Section 3 presents the SLR protocol and discusses the main findings of this work. Section 4 describes, briefly, the main structure of a framework for usability evaluation of DSLs. Section 5 presents the final remarks of this work.

2 Background

This section presents the main related areas of this paper, *i.e.*, Domain-Specific Languages and Human-Computer Interaction.

2.1 Domain-Specific Languages

Domain Specific Languages (DSL), also called application-oriented, special purpose or specialized languages, are languages that provide concepts and notations tailored for a particular domain [7, 13]. They differ from a General Purpose Language (GPL), such as Java, C or Python, for example, since they are designed from the problem domain and not from the solution domain. When designing a DSL, it is important to analyze the domain, in order to identify and document features of that domain.

There are several tools that help to create and maintain DSLs. These tools are know as Language Workbench (LW), for example, Microsoft Visual Visualization Studio and Modeling SDK, Generic Modeling Environment (GME), Eclipse Modeling Framework (EMF) or MetaEdit+. Those tools are not restricted to analysis and code generation, but provide also a better experience to DSL developers, since they allow creating DSL editors that are very similar to modern Integrated Development Environments (IDEs) [13].

Similar to other programming languages, a DSL also has a well-defined syntax and semantic. Its syntax describes its structure, while its semantic defines the meaning of each construct. The DSL syntax usually contains: an abstract definition, which is normally specified in a meta model in which the language constructs, properties and relationships are defined; and a concrete definition, in which the elements of the DSL are represented, *e.g.* using tables, text, figures, or matrices [21].

The elements representation must meet the domain concepts that are being modeled. The goal is to meet the representation fidelity [17, 33], in which there is a clear mapping between each representation form to its domain concept. Furthermore, all concepts must be represented in the DSL. As an example, Fig. 1 shows a meta model scenario for a Performance Testing DSL [6–8]. In the figure, there are three types of system users: Browsing, Shopping and Ordering. Each type of user has a different probability of executing an action, *e.g.* the Browsing user has only 25% of chance of performing the Shop action and only 15% of chance of executing the Order action.

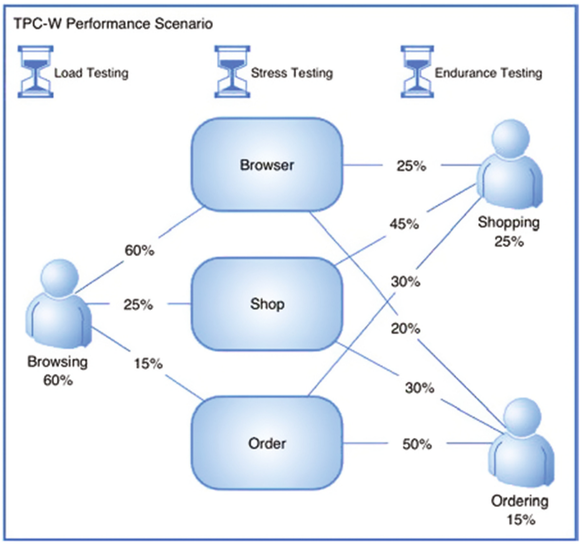

Fig. 1. Meta model scenario for stress testing

Figure 1 also shows that three different types of tests are possible: Load Test, Stress and Resistance [19].

A DSL should be easy to use and its use should be more, or at least as, efficient and efficacious as a GPL. However, experience and literature [10] show that developers consider that the learning curve may be too expensive when comparing the use of known GPLs. Hence more studies are necessary to evaluate the usability of DSLs.

2.2 Human-Computer Interaction

Human-Computer Interaction (HCI) is concerned with the quality of the use of interactive systems and with the impact of their daily use for users [26]. One of the goals of HCI is to obtain practical results related to user interfaces in systems projects [28]. To achieve this goal, researchers try to understand and to acquire methods and techniques that use different quality criteria for interactive systems. This work main focus is on the usability criteria. According to Rogers *et al.* [26], usability criteria includes to be efficient, efficacious, safe, useful, easy to learn and easy to remember how to use.

The design process of an interactive system contains the requirements, design alternatives, prototypes and evaluation. These are all activities that are complementary in an interactive cycle. The evaluation phase is responsible for guaranteeing that the system is adequate to its purpose to the final users [24]. Thus, the evaluation of the quality of use carried out during the process of developing an interactive system and not only in the final phase allows to identify (potential) problems that can affect the use of the system [30].

The evaluation process of an interactive system encompasses to know why, what, where, when and whom evaluates the system. There are different methods and techniques to evaluate system usability. Regarding DSLs, the evaluators could be domain users, analysts, developers, testers or HCI specialists, for example.

3 Systematic Literature Review

This section presents a systematic literature review (SLR) [18], in which the main focus was to identify and to analyze the evaluation process of DSLs. The period in which the SLR was executed was from March to June 2016. This study allowed us to identify primary studies in both Human-Computer Interaction (HCI) and Software Engineering (SE) areas.

3.1 SLR Planning

During this phase, the research goal, research questions, search strategy, and the inclusion and exclusion criteria are defined.

Research goals: based on preliminary studies on the subject, the following goals were established for this SLR: *(i)* whether HCI aspects were considered or not during the development of a DSL; *(ii)* to know the techniques and approaches used to evaluate DSLs; *(iii)* whether there were problems and limitations regarding DSL evaluation when HCI techniques were applied.

Research questions: based on the research goals, the following questions were asked: RQ1: Was the importance of usability considered during the DSL development? RQ2: What were the evaluation techniques that were applied in the context of DSLs? RQ3: What were the problems and limitations identified during the DSL usage?

Search strategy: *th*e following digital libraries were used: ACM (http://portal.acm.org/); IEEE (http://ieeexplore.ieee.org/); ScienceDirect (http://www.sciencedirect.com/); and, Scopus (https://www.scopus.com/).

Selection criteria: the following inclusion (IC) and exclusion (EC) criteria were used: IC1 - the study must contain at least one of the terms related to HCI evaluation in DSLs in the title or abstract; IC2 - the study must present some type of DSL evaluation; EC1 – the study is about evaluation but not DSLs; and, EC2 - the study is not written in English;

3.2 SLR Execution

During this phase, the search string construction, studies selection, quality evaluation, data extraction and synthesis were performed.

Search string construction: the search string was build based on terms from DSL and HCI, from usage evaluation and usability, and their synonyms (see Table 1). Table 2 presents the search string.

Table 1. Terms used to build the search string

Terms	Synonyms
Domain Specific Language	DSL, DSM, DSML, Domain Specific Modeling, Domain Specific Modeling Language
Human Computer-Interaction	HCI
Evaluation	Validation, Evaluating, Experiment

Table 2. Search string

(TITLE-ABS-KEY("Domain Specific Language", OR, dsl, OR, dsm, OR, "Domain Specific Modeling", OR, "Domain Specific Modeling Language"), AND, TITLE-ABS-KEY(evaluation, OR, evaluating, OR, experiment), AND, TITLE-ABS-KEY (usability, OR, "User Centered Design", OR, "User Experience", OR, hci, OR, "human computer interaction"))

Quality questions: Each quality question could have the following answers: yes, partially and no. Each answer would be graded as follows: 1 for yes, 0.5 for partially, and 0 for no. After answering the 5 quality questions, only studies that were marked with 2.5 to 5 were considered for further analysis. Table 3 shows only the papers that were considered to be read. The quality questions were: QQ1: Did the paper present any contribution to HCI?; QQ2: Did the paper present any usability evaluation technique?; QQ3: Did the paper present the results analysis?; QQ4: Did the paper describe the evaluated DSL?; and, QQ5: Did the paper describe the found usability problems?

Primary studies selection: the performed search, based on the search string (adapted for each database), returned the number of studies presented in Fig. 2.

Table 3. Quality assessment

Studies				QQ					Quality	
ID	Reference		Year	1	2	3	4	5	Sc	Desc
01	[29]	Sinhá	2006	Y	Y	Y	Y	Y	5	E
02	[4]	Barisic	2011	Y	Y	Y	Y	Y	5	E
03	[2]	Barisic	2012	Y	P	Y	Y	P	3,5	VG
04	[5]	Barisic	2012	Y	Y	Y	Y	P	4,5	E
05	[27]	Rouly	2014	Y	Y	Y	Y	P	4,5	E
06	[12]	Ewais	2014	Y	Y	Y	Y	Y	5	E
07	[3]	Barisic	2014	Y	Y	P	Y	Y	4,5	E
08	[15]	Gibbs	2014	Y	Y	Y	Y	P	4,5	E
09	[31]	Teruel	2015	Y	Y	Y	Y	P	4,5	E
10	[16]	Kabac	2015	Y	Y	Y	Y	P	4,5	E
11	[10]	Cueca	2015	Y	Y	Y	Y	Y	5	E
12	[1]	Albuquerque	2015	Y	Y	Y	Y	Y	5	E

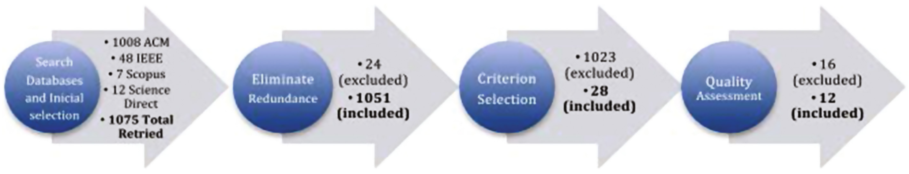

Fig. 2. Selection studies process

In the first phase of the SLR, 1008 papers returned from ACM, 48 from IEEE, 7 from Scopus and 12 from ScienceDirect, resulting in 1075 papers. After applying the inclusion, exclusion and quality criteria, 12 papers were thoroughly read. Figure 2 shows the papers that were selected after each phase.

3.3 Results Analysis

- RQ1 - Was the importance of usability considered during the DSL development?

Barisic *et al.* [2] introduce a methodology that considers usage quality criteria since the beginning of DSL language development to meet the user's expectations. Besides, their study considers usability criteria during the whole development of the DSL and also during the modeling of a system that was developed using the DSL. Their main conclusions were that it is necessary to have a clear definition on the quality criteria that will be used to evaluate the DSL; and, that it is important to integrate the IDE used for the development of the DSL.

Barisic *et al.* [3] propose systematic approaches based on experimental validation with real users. Their study main goal was to assess the DSL impact on domain specialists' productivity when using the DSL. One of the conclusions of their study is that there is a lack of systematic approaches to experimental evaluation of efficiency and efficacy of DSLs.

In order to reflect the user needs, Barisic *et al.* [5] included a usability engineer in the DSL development. The usability engineer participated in all phases of the DSL development, and also during the DSL assessment in an actual context. Their study presents four experiments whose main goals were to evaluate the DSL usability. The experiments were classified using the following attributes: type of evaluation (DSL x GPL, Visual DSL x Textual DSL) and quality (flexibility, productivity, usability, learnability, understanding, user satisfaction, language evolution, effectiveness, efficiency, perceived complexity).

Regarding usability concepts during the modeling process, Sinhá *et al.* [29] describe the use of a set of measures to evaluate DSL usability. This study presents the usability criteria during the functional testing of systems. The usability criteria are based on the ten Nielsen heuristics [22]. They applied the following four heuristics: learnability, effectiveness, efficiency and usage satisfaction. Those criteria were translated to a set of metrics to obtain quantitative data.

Rouly [27] performed a survey to evaluate 25 IDEs used for domains not focused on programming languages, *e.g.* diagrams modeling. The main goal of this work was to verify which functionalities were required in each different domain and how they were presented to the respective user's domain, *e.g.* 3D modeling, animation, music, prototyping, simulation, visual software development.

Similarly to Rouly [27], which was concerned with user-centered development, Gibbs *et al.* [15] propose a methodology and an architecture to separate front-end user interface and back-end coding. Their objective was to increase development environment flexibility, reduce development time and to improve development productivity. Basically the authors propose the use of DSL to describe the user interface, therefore separating the user interface from the rest of the system.

Albuquerque *et al.* [1] proposed the Cognitive Dimension Notation (CDN) to detect usability aspects for tasks maintaining in DSLs. Their study is based on four DSL aspects, *i.e.* expressiveness, conciseness, integration, and performance. For example, expressiveness is related to the fact that DSL artifacts reflects the domain they represent, while conciseness is used to verify whether fewer terms can be used to understand the meaning of the artifacts.

Teruel *et al.* [31] conducted an experiment to assess the usability of DSL for system requirements modeling. The experiment was executed by 38 students, which performed several different modeling tasks, to assess effectiveness, efficiency, and user's satisfaction. Effectiveness was assessed through task finishing rate and help required. Efficiency was measured by expended time and user's productivity when executing a task. Satisfaction was evaluated through questionnaires.

Some of the other selected papers [10, 12, 16], although discuss several aspects related to usability and DSL, do not focus directly on the usability criteria when developing the respective DSL.

Based on the analyzed papers, although few papers were found, all of them described some kind of concern regarding usability when developing DSLs. Some of the studies also were concerned with the easiness for developing DSLs, and, therefore, developed some mechanisms to allow that.

- RQ2 - What were the evaluation techniques that were applied in the context of DSLs?

Albuquerque *et al.* [1] suggested the use of quantitative and qualitative methods. In their study, they present an evaluation method called Cognitive Dimensions (CD) that contains 14 dimensions: viscosity, visibility, compromise, hidden dependencies, expressiveness role, error tendency, abstraction, secondary browsing, mapping proximity, consistency, diffusion, hard mental operations, provisional, and progressive evaluation. The authors presented their method and also applied Goals Question Metric (GQM) to corroborate the qualitative evaluation they performed. In their work, they present two metrics to evaluate DSLs: DSL expressiveness, which refers to in what extend the DSL represents the domain, and DSL conciseness, which refers to what terms can be deleted without compromising the domain artifact representativeness. These two metrics were also divided in other metrics, *i.e.*, expressiveness is composed of hidden dependencies, abstractions, mapping proximity; while conciseness is composed of viscosity, visibility, diffusion and hard mental operations.

Barisic *et al*. [3] suggested that for usability evaluation it is important to first define the usability requirements. Each requirement is assessed by a set of quantitative metrics using GQM. Regarding cognitive aspects, Barisic *et al.* [4] defined a cognitive model to languages based on user scenarios. The cognitive activities in the language are: syntax and semantic learning, syntax composition needed to fulfill a role, syntax understanding, syntax debugging, and changing a function that was written by any developer. In order to evaluate their method, they performed a controlled experiment with six participants. They used a DSL called PHysicist's EAsy Analysis Tool (Pheasant) and a baseline developed using a GPL, *i.e.* C++. At the end, the participants answered a questionnaire to verify whether the use of the DSL was intuitive, adequate and efficient.

Although Ewais *et al.* [12] did not explicitly describe an evaluation method, they used a strategy to evaluate the usability of a language before it would be implemented. To perform this evaluation, fourteen subjects (4 PhD students and 10 instructors) participated in the experiment. The evaluation was divided into 3 steps. In the first step the subjects were exposed to different notations, sample models created using languages, types of relationships, among others aspects. In the second step, the subjects had to construct and adapted 3D Virtual Learning Environment (VLE) using three languages. To realize that task, the subjects had to use paper and pen, since the goal was to assess the level of expressiveness of the language visual notations, and also the effort to create an adaptive 3D VLE using graphical language. In the third step, the subjects answered an online questionnaire.

As mentioned before, Sinha *et al.* [29] based their evaluation on four heuristics proposed by Nielsen, and for each heuristic there was a set of metrics. On one hand, learnability was measured through the number of errors a subject committed, divided by effort; while efficiency was measured by the size of the test set divided by effort. On the other hand, satisfaction was measured in four levels: frustrating, unpleasant, pleasant, and pleasurable. Therefore, it was possible to have a quantitative evaluation of a DSL when analyzing its usability.

Gibbs *et al.* [15] proposed an architecture that considers User Experience (UX) aspects with Model Driven Engineering (MDE). Three premises were used to illustrate the proposed architecture: role specialization to increase productivity and success; the communication gap that may cause confusion and inefficiency; and, communication gap between user interface architecture and code separation.

Cuenca *et al.* [10] described an empirical study to compare efficiency from a DSL to a GPL. Evaluation was performed through observations and interviews conducted with a pre-defined questionnaire. From these instruments conclusion rate, conclusion time, effort and difficulty were measured. These data were collected during programming tasks and usability evaluation during the programming tool usage.

Different from other studies, Barisic *et al.* [5] presented the analysis of four controlled experiments. The authors mentioned that the usability evaluation performed in each experiment was based on users interviews, open questionnaires, testing using tools support and multiple-choice questionnaires.

Barisic *et al.* [2] used a recommendation-based methodology that considers user-centered techniques. Therefore, some activities had to be applied to each phase of the DSL development. The main activities that their methodologies describe are:

domain analysis, language design, controlled experiment as testing, deployment and maintaining.

- RQ3 - What were the problems identified during the DSL usage?

Several studies performed some type of evaluation with users, and most of them did not find any kind of problem that would avoid the use of DSL. For example, Ewais *et al.* [12], Gibbs *et al.* [15] and Kabac *et al.* [16] performed experiments with end users, and analyzed their opinion. Basically, Ewais *et al.* [12] reported that whether the users were able to execute the tasks they were given; whether the users needs were achieved; or, whether the DSL was useful and easy to use.

Rouly *et al.* [27] mentioned that, although the interface design would have a good impact in efficiency, there were some problems regarding the interface of the modeling tool.

Albuquerque *et al.* [1] applied the same DSL to two different systems, *i.e.* HealthWatcher and MobileMedia, and did not present problems, but raised some limitation on their experiments. They applied their experiment considering a set of stable rules, which were not reusable, and they had a very small set of rules to build the systems.

Regarding the evaluation of modeling and developing tools, Barisic *et al.* [4] mentioned that there was a big error rate when inexperienced users used the proposed language. One of the conclusions, from the authors, was that this might have been caused by the lack of feedback that the tool provides for users. Therefore, users did not have the necessary feedback before the authors evaluated the experiment. If they had a feedback, they might have corrected the errors before submitting their results.

Teruel *et al.* [31], different from the other studies, report several problems regarding the representativeness of the used DSL. Based on their experiment, they proposed several modifications in the DSL, for example, change several elements that did not represent the domain.

As mentioned above, most of the studies did not report big problems when using a DSL. Actually, they even present several advantages of using DSL when compared to a GPL used as baseline. However, most of the authors that evaluate their own languages did not report problems in the use of the DSL, since they would use them in domains in which they had already found problems when using a GPL, hence they built a DSL to avoid those problems. Nonetheless, the advantages that were presented indicate that DSLs can provide great advantage in several aspects, for example, increase in productivity, representatives, less effort, and so on.

3.4 Taxonomy for DSL Evaluation

Based on the selected studies and the research questions in Sect. 3.3, this section presents a taxonomy of terms used during the evaluation of DSLs. This taxonomy was structured as a conceptual mapping and is shown in Fig. 3. The taxonomy is structured based on the terms that were mentioned in the studies selected in this SLR. Figure 3 shows the main groups of categories represented as the external rectangles: profile user, data type, usability evaluation methods, empirical methods, metrics, and collection

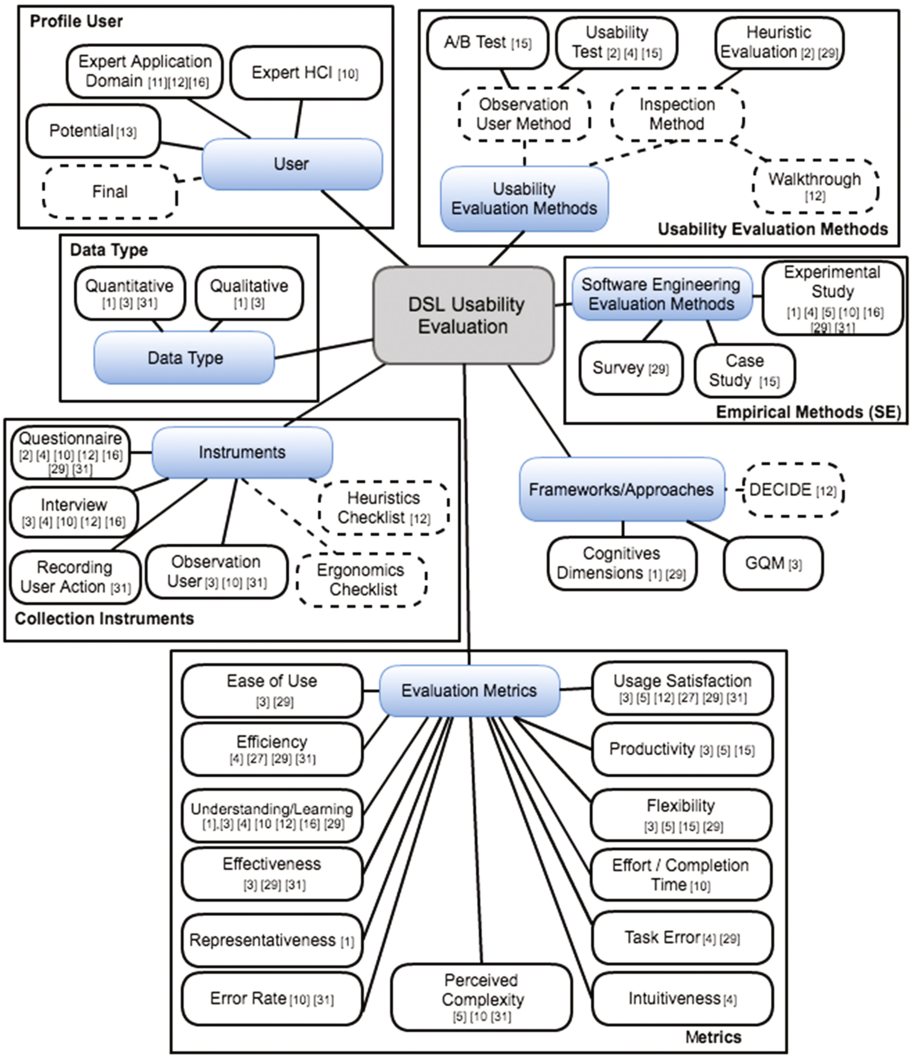
Profile User
Expert Application Domain [11][12][16]
Expert HCI [10]
Potential [13]
User
Final
A/B Test [15]
Usability Test [2] [4] [15]
Heuristic Evaluation [2] [29]
Observation User Method
Inspection Method
Usability Evaluation Methods
Walkthrough [12]
Usability Evaluation Methods
Data Type
Quantitative [1] [3] [31]
Qualitative [1] [3]
Data Type
DSL Usability Evaluation
Software Engineering Evaluation Methods
Experimental Study [1] [4] [5] [10] [16] [29] [31]
Survey [29]
Case Study [15]
Empirical Methods (SE)
Questionnaire [2] [4] [10] [12] [16] [29] [31]
Instruments
Interview [3] [4] [10] [12] [16]
Heuristics Checklist [12]
Recording User Action [31]
Observation User [3] [10] [31]
Ergonomics Checklist
Collection Instruments
Frameworks/Approaches
DECIDE [12]
Cognitives Dimensions [1] [29]
GQM [3]
Ease of Use [3] [29]
Evaluation Metrics
Usage Satisfaction [3] [5] [12] [27] [29] [31]
Efficiency [4] [27] [29] [31]
Productivity [3] [5] [15]
Understanding/Learning [1],[3] [4] [10 [12] [16] [29]
Flexibility [3] [5] [15] [29]
Effectiveness [3] [29] [31]
Effort / Completion Time [10]
Representativeness [1]
Task Error [4] [29]
Error Rate [10] [31]
Perceived Complexity [5] [10 [31]
Intuitiveness [4]
Metrics

Fig. 3. DSL usability evaluation taxonomy

instruments. Inside each of these rectangles, there is also a rounded rectangle that represents that group. In each of these groups, there is a set of categories, for example, user profile can have the following categories: HCI expert, DSL expert, potential user, final user. Notice that in the figure some of the categories are represented by dashed rounded rectangles. These categories are not directly mentioned in the studies presented in this SLR, but are important in the development of a framework to assess DSLs.

4 *Framework* Usa-DSL

Based on the SLR presented in this paper, a new framework to evaluate DSL usability has been proposed. This framework is called Usa-DSL and its structure is based on the project development life cycle [25]. The framework is organized in steps divided into phases. Each phase is composed of a set of activities. The steps are: Evaluator Profile, Ethical and Legal Responsibilities, Data Types, Empirical Experimental Methods, Evaluation Method, Metrics, Training, Evaluation, Data Packing, Evaluation Reporting. Phases are composed of: Definition, Execution, Analysis and Results. All steps from the framework were designed to meet the needs of each evaluation; therefore, the evaluator has the freedom to develop the evaluation according to the requirements of each language that is being assessed. Furthermore, it is possible to perform the evaluation in an interactive manner. The whole framework is not the subject of this paper and will be presented elsewhere.

5 Conclusion

Although most software is developed using general-purpose languages (GPLs), there are some software that may be developed using domain specific-languages (DSLs). These DSLs provide several advantages compared to GPLs, since they are designed for a specific application domain. Nonetheless, these languages have also to be tested in order to know whether they fulfill the requirements they try to meet. One of such test is regarding whether the languages meet the users needs and expectations. Hence it is important to evaluate the languages efficiency, efficacy, easiness of use, and user satisfaction. Some researchers also try to understand the cognitive effort or learning time users take to comprehend the language they are using.

Considering that, this paper presented a systematic literature review (SLR) to verify if the language designers take into consideration usability aspects when building a new DSL. The Kitchenhan protocol [18] was used to plan, execute and collect data for this SLR.

After applying the SLR protocol, only twelve papers were selected to be analysed. Those papers were used to answer three research questions, basically to understand how usability was considered in the design of DLS, to verify what techniques or approaches were used to evaluate their usability, and to identify what type of problems were raised when the DSL usability was assessed. These results helped to build a taxonomy that may aid researchers either to design new DSLs or, mainly, to evaluate DSLs usability. Furthermore, the results of this SLR helped to identify problems and resources to the proposal of a framework to evaluate DSL usability. This framework, called Usa-DSL, was briefly presented in Sect. 4 and will help new DSL designers to have a strategy to assess their final product in a systematic way. Currently, as this SLR showed, most researchers evaluate their DSL in an *ad hoc* way. This framework will use HCI and DSL knowledge; therefore it is expected to help in the design of better DSLs.

Acknowledgments. Study developed by the Research Group of the PDTI 001/2016, financed by Dell Computers with resources of Law 8.248/91.

References

1. Albuquerque, D., Cafeo, B., Garcia, A., Barbosa, S., Abrahão, S., Ribeiro, A.: Quantifying usability of domain-specific languages: An empirical study on software maintenance. J. Syst. Softw. **101**, 245–259 (2015)
2. Barisic, A., Amaral, V., Goulão, M.: Usability evaluation of domain-specific languages. In: Quality of Information and Communications Technology, pp. 342–347 (2012)
3. Barisic, A., Amaral, V., Goulão, M., Aguiar, A.: Introducing usability concerns early in the DSL development cycle: Flowsl experience report. In: MODELS, pp. 8–17 (2014)
4. Barisic, A., Amaral, V., Goulão, M., Barroca, B.: Quality in use of domain-specific languages: a case study. In: 3rd ACM SIGPLAN Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 65–72 (2011)
5. Barisic, A., Amaral, V., Goulão, M., Barroca, B.: Evaluating the usability of domain specific languages. In: Formal and Practical Aspects of Domain-Specific Languages: Recent Developments (2012)
6. Bernardino, M., Rodrigues, E., Zorzo, A.F.: Performance testing modeling: an empirical evaluation of DSL and UML-based approaches. In: 31st ACM Symposium on Applied Computing, pp. 1–6 (2016)
7. Bernardino, M., Zorzo, A.F., Rodrigues, E.: Canopus: a domain-specific language for modeling performance testing. In: 9th International Conference on Software Testing, Verification and Validation, pp. 1–8 (2016)
8. Bernardino, M., Zorzo, A.F., Rodrigues, E., de Oliveira, F.M., Saad, R.: A domain specific language for modeling performance testing: requirements analysis and design decisions. In: 9th International Conference on Software Engineering Advances, pp. 609–614 (2014)
9. Conrado, D.B.F.: Abordagem para criação de linguagem específica de domínio para robótica móvel. Dissertação de Mestrado – Universidade Federal de São Carlos (2012)
10. Cuenca, F., Bergh, J.V.D., Luyten, K., Coninx, K.: A user study for comparing the programming efficiency of modifying executable multimodal interaction descriptions: a domain-specific language versus equivalent event-callback code. In: 6th Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 31–38 (2015)
11. Dobbe, J.: A domain-specific language for computer games. Master thesis - Delft University of Technology, Netherlands (2007)
12. Ewais, A.B., De Troyer, O.: A usability evaluation of graphical modelling languages for authoring adaptive 3d virtual learning environments. In: 6th International Conference on Computer Supported Education (CSEDU), pp. 459–466 (2014)
13. Fowler, M.: Domain Specific Languages. Addison-Wesley Professional, Boston (2010)
14. Furtado, A.W.B., Santos, A.L.M.: Using domain-specific modeling towards computer games development industrialization. In: 6th OOPSLA Workshop on Domain-specific Modeling (DSM) (2006)
15. Gibbs, I., Dascalu, S., Harris, F.C.: A separation-based UI architecture with a DSL for role specialization. J. Syst. Softw. **101**, 69–85 (2015)
16. Kabac, M., Volanschi, N., Consel, C.: An evaluation of the DiaSuite toolset by professional developers: Learning cost and usability. In: 6th Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 9–16 (2015)
17. Kelly, S., Tolvanen, J.-P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley-Interscience: IEEE Computer Society, Hoboken (2008)
18. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering, ver. 2.3. Technical report, Evidence-Based Software Engineering (EBSE) (2007)

19. Meier, J.: Performance Testing Guidance for Web Applications: Patterns & Practices. Microsoft Press, Redmond (2007)
20. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37–4**, 316–344 (2005)
21. Mernik, M., Porubän, J., Kollár, J., Sabo, M.: Abstraction of computer language patterns: the inference of textual notation for a DSL. In: Formal and Practical Aspects of Domain-Specific Languages: Recent Developments. Information Science Reference (2012)
22. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: SIGCHI Conference on Human Factors in Computing Systems, pp. 249–256 (1990)
23. Nordmann, A., Hochgeschwender, N., Wrede, S.: A survey on domain-specific languages in robotics. In: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, pp. 195–206 (2014)
24. Prates, R.O., Barbosa, S.D.J.: Avaliação de interfaces de usuário–conceitos e métodos. Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo, vol. 6 (2003)
25. Pressman, R.: Software Engineering: A Practitioner's Approach, vol. 7. Palgrave Macmillan, New York (2005). Capítulo 1–2
26. Rogers, Y., Sharp, H., Preece, J.: Interaction Design. Wiley, New York (2002)
27. Rouly, J.M., Orbeck, J.D., Syriani, E.: Usability and suitability survey of features in visual ides for non-programmers. In: 5th Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 31–42 (2014)
28. Hewett, T.T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., Verplank, W.: ACM SIGCHI Curricula for Human-Computer Interaction. Technical report. ACM (1992)
29. Sinha, A.C., Smidts, C.: An experimental evaluation of a higher-ordered typed-functional specification-based test-generation technique. Empirical Softw. Eng. **11–2**, 173–202 (2006)
30. Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Longman Publishing Co., Boston (1997)
31. Teruel, M.A., Navarro, E., López-Jaquero, V., Montero, F., González, P.: A CSCW requirements engineering case tool: development and usability evaluation. Inf. Softw. Technol. **56–8**, 922–949 (2014)
32. Walter, R., Masuch, M.: How to integrate domain-specific languages into the game development process. In: 8th International Conference on Advances in Computer Entertainment Technology, pp. 1–8 (2011). Article No. 42
33. Weber, R., Zhang, Y.: An analytical evaluation of Niam's grammar for conceptual schema diagrams. Inf. Syst. J. **6–2**, 147–170 (1996)