Quality in Use of DSLs: Current Evaluation Methods

Ankica Barišić, Vasco Amaral, Miguel Goulão, and Bruno Barroca

CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Campus de Caparica, 2829-516 Caparica, Portugal
barisic.ankica@gmail.com, vasco.amaral@di.fct.unl.pt,
miguel.goulao@di.fct.unl.pt, bruno.barroca@di.fct.unl.pt

Abstract. Domain Specific Languages (DSLs) are claimed to contribute to increment productivity, while reducing the required maintenance and programming expertise. In this context, the usability of these languages becomes a major issue: if the language is not easy to learn and use, it is unlikely to be successfully adopted.
It is important to foster high quality DSLs during its engineering process. We argue that a systematic approach based on User Interface Experimental validation techniques should be used to assess the impact of the introduction of DSLs in the productivity of DSL user. Productivity can be fostered by assessing important usability attributes early in the language construction . This work's contribution, besides highlighting the problem of the absence of systematic approaches for experimental validation of DSLs in general, is to identify existing evaluation approaches that can be adapted from the field of User Interfaces.

Keywords: Domain Specific Software Languages Evaluation, Quality in Use

1 Introduction

Domain Specific Languages (DSLs) are meant to close the gap between the Domain Experts and the Solution domain. The claim is that the closer we get to fill this gap, the closer we are to increase the user's productivity[9].

Software Languages Engineering (SLE) as a systematic approach to DSL construction is becoming a mature activity, building upon the collective experience of a growing community, and the increasing availability of supporting tools [9]. A typical SLE process starts with the Domain Engineering phase, in order to elicit the domain concepts. Language design follows, capturing the referred concepts and their relationships. Then, the language is implemented, documented,and goes on to testing, deployment, evolution, recovery, and retirement phases. A good DSL is hard to build because it requires both domain knowledge and language development expertise, and few people have both [10]. DSL evaluation is not a trivial task, and it can be both expensive and time consuming.

As DSLs are meant to close the gap between domain experts and the solution domain, their main purpose is similar to **Human-Computer**(H/C)**Interaction**. In what matters to DSLs' usability evaluation, we can reuse existing techniques for evaluating general User Interface (UI).

In this paper we identify candidate techniques for evaluating DSL's Quality in Use. We present a definition of DSLs that relates them with UI and different interpretations of Usability (section 2). Then, in section 3, we outline the results of a systematic review on DSLs' usability evaluation approaches, and contrast them with the common practices in General Purpose Language's (GPLs) evaluation. Section 4 discusses existing evaluation methodologies and which quality attributes are relevant for evaluation of DSLs quality in use. Conclusions and further work are presented in section 5.

2 Domain Specific Languages and Usability

Intuitively, a language is a means of communication among peers who exchange sentences that are composed by signs in a particular order. According to the context of a conversation, these sentences can have different interpretations.

In this paper, we consider languages as communication interfaces between humans and computers i.e. UI. We argue that any UI is actually a realization of a language, where in this context a language is considered a theoretical object (a.k.a. model) that describes the allowed terms and how to compose these terms into the sentences involved in a particular H/C interaction. Fig. 1 puts quality in use in perspective with respect to other quality attributes.

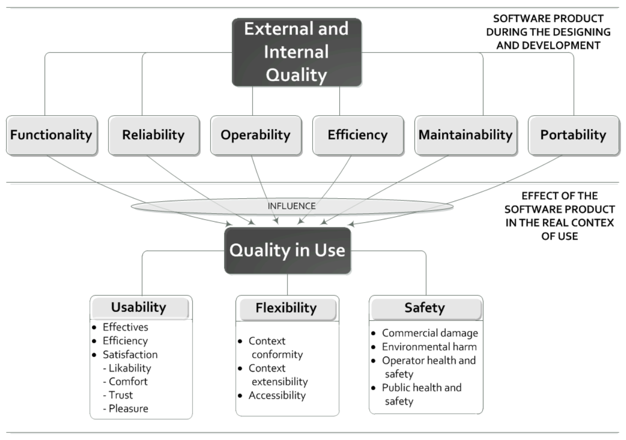

Fig. 1. Quality model for achieving Quality in Use adapted from [13]

The **Context of Use** defines i.e. *'the users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a* *product is used'* [7]. The same problem can be solved in different ways for different contexts of use in language user's mind. Languages that reduce the use of *computation domain concepts* and focus on the *domain concepts* of the *contexts of use's* problem, are called **Domain Specific Languages**. Each DSL is defined for a specific context of use and that is why context of use should be also considered as important factor in DSL's usability evaluation.

Usability is defined as: '*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.*' [4]. Moreover ISO 9126 estimated this definition with the notion of **Quality in Use** i.e the quality as perceived by the user during actual utilization of a product in real **Context of Use**.

Quality in Use is measured in terms of the result of using the software, rather than properties of the software itself. The standard states that achievement of *Quality in Use* can be assured by achieving internal and external Quality that provides us with metrics that can be used early in software development process. A quality model for achieving *Quality in Use* can be found in [13].

3 Language evaluation

GPLs usability can be indirectly assessed by the size of the community that uses the GPL. Other, more direct, evaluations are presented as language benchmarks with automatic quantitative data collection, covering memory resources, runtime memory consumption, source text length, comment density, program structure, reliability, and the amount of effort required to perform a given task [14]. Language comparison can also focus on some characteristics that indicate the language suitability to its intended Context of Use.

There is an increasing awareness to the usability of languages, fostered by the competition of language providers. Better usability is a competitive advantage, although evaluating it remains challenging, because it is hard to interpret existing metrics in a fair, unbiased way, which resists to external validity threats concerning the broad user groups, or internal ones - it is very easy to end up comparing apples with oranges, when evaluating competing languages.

The methods used to evaluate usability of GPLs are not always adequate for DSLs because they are not systematic and are centred only on *computation domain concepts*. The GPLs intended users are expected to have high knowledge of technical and computational concepts, while the DSLs intended user group are domain experts that are more familiar with the logical domain concepts. We need a different approach to perform evaluation of DSLs.

In general, the software industry does not invest much on the evaluation of the usability of DSLs [6]. It is unclear whether this results from an insufficient understanding of the SLE process which should include the evaluation of the produced DSLs. Language engineers may perceive the investment in evaluation as an unnecessary cost and prefer to risk providing a solution which has not been validated, w.r.t. its usability, by end users. With anecdotal reports of 3-10 times productivity improvements [12, 8, 15], or "clearly boosted development speeds"

[11] in industrial settings, why bother with validation? The problem is that those reports lack external validity e.g. they cannot be safely extrapolated in general.

We conducted a systematic literature review to assess the extent to which DSLs are evaluated and how they are evaluated [6]. The level of DSL evaluation found in our survey can be considered low, and the details on the few performed evaluations are clearly insufficient. There was a predominance of toy DSLs with unsubstantiated claims to the merits of DSLs. Most authors present reports of usability evaluations that are impossible to replicate and to extract a precise rationale from (e.g it is hard to reason about the representativeness of their DSL's users due to the poor characterization of subjects involved in the evaluation).

4 Usability Evaluation

We will now discuss some common techniques for usability evaluation of UIs that are candidates for adaptation to DSL evaluation:

UCA (Usability Context Analysis) is a methodology for eliciting detailed information about a product and how it will be used [3]. This method ensures that user-based evaluation produces valid results, by specifying how important factors are to be handled in an evaluation, and by defining how well the evaluation reflects real world use.

MUSiC (Metrics for Usability Standards in Computing) is a methodology for measuring Usability. It can be applied at different stages of the software lifecycle and is composed by four modules which can be used independently or in an integrated way concerning *user satisfaction*, *user performance measurement*, *cognitive workload* and *analytic measurement* (of a dynamic model of the UI and user tasks) [5].

MAGICA is a methodology that involves the measurement of [1] *user satisfaction* assessed with a questionnaire, *task completion time*, the *cognitive effort* (i.e. the stress level the user suffers when completing a task) and the adherence to *heuristic principles* for the product. As mentioned before, the finished DSL product can be seen as a UI and its evaluation should essentially consider H/C interaction. Standard methodologies can be adapted to DSLs. To capture Context of Use we can reuse UCA methodology and at earliest phase as possible of development cycle we should perform User analysis, to identify characteristics of the target user population, and Task analysis, to capture goals and preconditions. To evaluate the achieved Quality in Use of the DSL we can adapt MUSiC and MAGICA methods and tools.

DSLs are built for a more confined context of use, capturing one particular set of domain concepts. While evaluating these languages, the universe of users is smaller, and they have less diversity of skills, so the validity of the results to their target population is much higher.

Although DSLs' development process is not the same as the one of UIs — typically DSLs have an explicit underlying model (by means of metamodels or grammars), while UIs models are usually implicit in their implementation — in general there is no distinction between finished DSLs' and general software products' evaluation. The influencing characteristics to achieve Quality in Use can be very diverse, namely if we compare two languages in different domains such as the ones presented in Fig. 2 ((H)ALL [2] and Lego [1]). Here we present an assessment of language requirements for two languages with different contexts of use. These requirements are presented in terms of internal and external quality attributes that will contribute to achieving Quality in Use of language. Different contexts of use lead to different priorities, with respect to quality attributes requirements. For example, we consider operability to be much more important for LEGO than for (H)ALL.

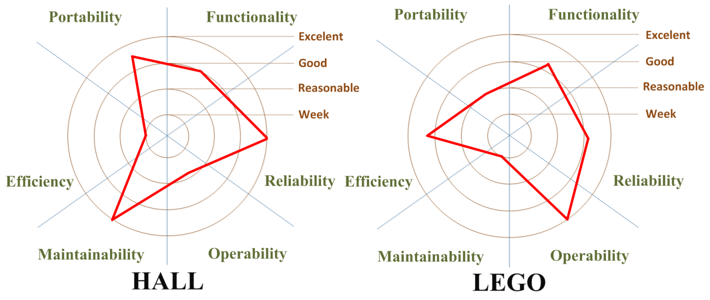

Fig. 2. Kiviat diagrams for (H)ALL and Lego DSLs

To capture achieved Quality in Use of DSLs we should evaluate the following characteristics: *Effectiveness* that should determine the accuracy and completion of the implementation of the sentences; *Efficiency* which tell us what level of effectiveness is achieved at the expense of various resources, such as mental and physical effort, time or financial cost, commonly measured in the sense of time spent to complete a sentence; *Satisfaction* that captures freedom from inconveniences and positive attitude towards the use of the language; and *Accessibility* with accent on learnability and memorability of the language terms.

Different quality attributes will bring success in achieved Quality of Use — it depends on its context of use and the target user population, so they can be identified just after performing the Context of Use analysis. Also, it is not always possible to achieve optimal scores for all usability attributes simultaneously, so when usability trade-offs seem inevitable, it is necessary to find a win-win solution that can satisfy both requirements.

5 Conclusions and Future Work

In this paper, we showed that the software industry does not seem to invest much on the evaluation of DSLs. We find that it is essential to evaluate its usability. We conclude that current methods are adequate to evaluate DSLs'

[1] http://mindstorms.lego.com/en-us/Software/Default.aspx

usability, however since they essentially focus on general UIs, they mostly lack both precision and concrete metrics that we could have if we take advantage of DSL's unique characteristic of having a smaller context of use. As future work, we will seek for user-centered models and languages to help us devise a framework for DSL's usability evaluation in order to be able to effectively indicate the Usability problems during DSLs' development, and measure its potential Quality in Use.

Acknowledgments

This work was partially funded by the grant PEst-OE/EEI/UI0527/2011 Centro de Informática e Tecnologias da Informação (CITI/FCT/UNL) - 2011-2012.

References

1. M. Alva, A. Martínez P, J. Cueva L, T. Sagástegui Ch, and B. López P. Comparison of Methods and Existing Tools for the Measurement of Usability in the Web. *Web Engineering*, pages 386–389, 2003.
2. Bruno Barroca and Vasco Amaral. (h)all: a dsvl for designing user interfaces for control systems. In *Proceedings of the 5th Nordic Workshop on Model Driven Engineering NW-MoDE 2007 27-29 August 2007*. Blekinge Institute of Technology, 2007. URL=http://www.ituniv.se/ miroslaw/node.htm.
3. N. Bevan. *Usability Context Analysis: a Practical Guide*. NPL Usability Services, Teddington, UK, 1997.
4. N. Bevan. Extending quality in use to provide a framework for usability measurement. *Human Centered Design*, pages 13–22, 2009.
5. L. Binucci. Software quality of use: Evaluation by MUSiC. *Objective Software Quality*, pages 165–178, 1995.
6. P. Gabriel, M. Goulão, and V. Amaral. Do Software Languages Engineers Evaluate their Languages? In *Proceedings of the XIII Congreso Iberoamericano en" Software Engineering"(CIbSE'2010)*, pages 149–162, 2010.
7. International Standard Organization. Iso/iec 9126 quality standards.
8. Steven Kelly and Juha-Pekka Tolvanen. Visual domain-specific modelling: benefits and experiences of using metacase tools. In Jean Bzivin and J. Ernst, editors, *International Workshop on Model Engineering, at ECOOP'2000*, 2000.
9. A.G. Kleppe. *Software language engineering: creating domain-specific languages using metamodels*. Addison-Wesley, 2009.
10. Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, 2005.
11. MetaCase. Eads case study, http://www.metacase.com/papers/metaedit_in_eads.pdf. Technical report, MetaCase, 2007.
12. MetaCase. Nokia case study, http://www.metacase.com/papers/metaedit_in_nokia.pdf. Technical report, MetaCase, 2007.
13. H. Petrie and N. Bevan. The evaluation of accessibility, usability and user experience. *The Universal Access Handbook*, 2009.
14. Lutz Prechelt. An empirical comparison of seven programming languages. *IEEE Computer*, 33(10):23–29, 2000.
15. David M. Weiss and Chi Tau Robert Lai. *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison Wesley Longman, Inc., 1999.