The Journal of Systems and Software 101 (2015) 245–259

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Quantifying usability of domain-specific languages: An empirical study on software maintenance

Diego Albuquerque[a,b,*], Bruno Cafeo[b,2], Alessandro Garcia[b,2], Simone Barbosa[b,2], Silvia Abrahão[c,3], António Ribeiro[a,1]

[a] University of Minho, Campus de Gualtar, 4710-057, Braga, Portugal
[b] Pontifical Catholic University of Rio de Janeiro – PUC-Rio, Rua Marquês de São Vicente, 225, 22453-900, Rio de Janeiro, Brazil
[c] Valencia University of Technology, Camino de Vera, 46022, Valencia, Spain

ARTICLE INFO

Article history:
Received 2 April 2014
Revised 20 October 2014
Accepted 27 November 2014
Available online 22 December 2014

Keywords:
DSL
Usability
Metrics

ABSTRACT

A domain-specific language (DSL) aims to support software development by offering abstractions to a particular domain. It is expected that DSLs improve the maintainability of artifacts otherwise produced with general-purpose languages. However, the maintainability of the DSL artifacts and, hence, their adoption in mainstream development, is largely dependent on the usability of the language itself. Unfortunately, it is often hard to identify their usability strengths and weaknesses early, as there is no guidance on how to objectively reveal them. Usability is a multi-faceted quality characteristic, which is challenging to quantify beforehand by DSL stakeholders. There is even less support on how to quantitatively evaluate the usability of DSLs used in maintenance tasks. In this context, this paper reports a study to compare the usability of textual DSLs under the perspective of software maintenance. A usability measurement framework was developed based on the cognitive dimensions of notations. The framework was evaluated both qualitatively and quantitatively using two DSLs in the context of two evolving object-oriented systems. The results suggested that the proposed metrics were useful: (1) to early identify DSL usability limitations, (2) to reveal specific DSL features favoring maintenance tasks, and (3) to successfully analyze eight critical DSL usability dimensions.

1. Introduction

A domain-specific language (DSL) aims to facilitate construction of software artifacts through specialized abstractions and notations (Langlois et al., 2007). DSLs are increasingly being used in many software engineering activities, including designing and checking architectural rules (e.g. Gurgel, 2012; Moha et al., 2010). Nevertheless, the difficulties of using DSLs have become more apparent when exposed to software maintenance circumstances (Van Deursen et al., 2000; Van Deursen and Klint, 1998). Several studies (Van Deursen et al., 2000; Langlois et al., 2007; Nishino, 2012; Van Deursen and Klint, 1998; Mernik et al., 2005) concluded that these difficulties might adversely lead to higher maintenance effort. An important factor that contributes to increased maintenance effort is the low usability of such DSLs (Barišic et al., 2011). The usability of a DSL artifact (e.g., a specification built using the DSL) is the quality that makes it easy for users to understand, learn, and interact with it (Langlois et al., 2007; Barišic et al., 2011).

Recently, we observed some studies concerned with analyzing the usability of DSLs from several point of views (Van Deursen et al., 2000; Humm and Engelschall, 2010; Nishino, 2012). There is, however, a lack of studies which rely on quantitative analysis to complement the qualitative analysis of the DSLs usability. The creation of a metric suite to support the quantitative analysis of DSLs would allow an objective comparison between DSLs (Barišic et al., 2011; Sobernig et al., 2011; Gabriel et al., 2011), therefore complementing the qualitative analysis approaches found in the literature (Rainer and Hall, 2003; Mernik et al., 2005; Prieto-Díaz, 1990; Hudak, 1998). The results would be more reliable and provide extra information at early design stages of a DSL than approaches without any quantitative analysis. Moreover, such a metric suite would support the early evaluation of DSL usability in order to help choose the most appropriate DSL given the nature of the software maintenance tasks.

* Corresponding author. Tel.: +351 253604430; fax: +351 253604430.
E-mail addresses: diego.l.albuquerque@gmail.com, pg19789@alunos.uminho.pt (D. Albuquerque), bcafeo@inf.puc-rio.br (B. Cafeo), afgarcia@inf.puc-rio.br (A. Garcia), simone@inf.puc-rio.br (S. Barbosa), sabrahao@dsic.upv.es (S. Abrahão), anr@di.uminho.pt (A. Ribeiro).
[1] Tel.: +351 253604430; fax: +351 253604430.
[2] Tel.: +55 21 3527 1500; fax: +55 21 3527 1500.
[3] Tel.: +34 96 3877000; fax: +34 96 3877000.

http://dx.doi.org/10.1016/j.jss.2014.11.051
0164-1212/

Concerned with the aforementioned issues, we report a study conducted to compare the usability of textual DSLs[4] for detecting architectural problems (Gurgel, 2012; Moha et al., 2010; Moha and Guéhéneuc, 2007; Gurgel et al., 2014). In particular, we defined a usability metrics suite that was developed based on the cognitive dimensions of notations (CDN) framework (Blackwell and Green, 2003). We instantiated these cognitive dimensions for evaluating DSLs and assessed them by a qualitative process. These instantiations of the CDN capture usability aspects of DSL artifacts relevant to software maintenance tasks. Data were collected from two DSLs (Gurgel, 2012; Moha et al., 2010) for detecting architectural problems. The two chosen DSLs explicitly embed constructs to define architectural design rules so that they can be checked in the source code. In addition, both DSLs were designed for different categories of stakeholders, including software architects, programmers and code reviewers.

The remainder of this paper is organized as follows: Section 2 gives some background information needed to better understand the scope of this paper. Section 3 describes the steps required to create the metrics. Section 4 describes the design of a qualitative study aimed at assessing the proposed instantiation of the cognitive dimensions. Section 5 describes the metrics suite developed to analyze the usability of DSLs. Section 6 describes the design of an exploratory study aimed at comparing the two textual DSLs and assessing the usefulness of the proposed metrics. The results of the study are analyzed and discussed in Section 7. Section 8 describes the threats to the validity of our study. Section 9 discusses related work. Finally, Section 10 concludes the work and suggests future developments.

2. Background

A DSL is a type of programming language or specification language in software development dedicated to a particular problem or solution domain (Van Deursen et al., 2000; Humm and Engelschall, 2010; Consel and Marlet, 1998). A DSL facilitates software development through appropriate abstractions and notations. Several studies (Van Deursen et al., 2000; Visser, 2008; Humm and Engelschall, 2010; Gray and Karsai, 2003) identify various benefits of using DSLs in the area of software engineering, including the provision of an idiom at the level of abstraction of the problem domain. These studies also show how the expressive power of DSLs is significant when they are properly designed for one specific domain.

In our study, we selected the domain of architectural rules. In this domain, DSLs are used by software architects, programmers and code reviewers to specify and check the adherence of the source code with respect to architectural rules. It is particularly challenging to design a usable DSL in this domain for several reasons (Garcia et al., 2009; Macia et al., 2012; Mitschke et al., 2013), including: (1) it needs to offer a concise set of abstractions in order to enable architects to express the high-level design rules, (2) it needs to be concise and expressive enough in order to support programmers and code reviewers in understanding which program elements are affected by the architectural rules, and (3) it needs to be expressive enough to allow users to tailor the architecture rules as they implement, maintain and evolve modules of a program.

Thus, the next subsection briefly describes the framework used for developing the usability metrics suite. This framework characterizes important usability properties to be assessed in the design of languages, such as DSLs (Section 2.1).

2.1. CDN framework

The CDN framework is "a set of discussion tools for use by designers and people evaluating designs" (Blackwell and Green, 2003). We chose this framework because we found that it is a widely used technique to support usability evaluation in the literature (Maia et al., 2012; Neto et al., 2010; Green and Petre, 1996). This framework provides cognitive dimensions[5] of general use in different domains, as shown in Table 1. These CDs are conceptual tools defined to help the designer or evaluator to reason about the system or language being assessed (Maia et al., 2012; Blackwell and Green, 2003). In addition, these CDs allow "to improve the exchange of experience, opinions, criticism and suggestions" (Maia et al., 2012). This framework was originally proposed to evaluate notational systems for designing artifacts, aiming "to improve the quality of discussion" (Blackwell and Green, 2003, p. 107). These CDs cover a wide range of issues and, consequently, their definitions may lead to different interpretations. Previous work has employed this framework to qualitatively evaluate the design of DSLs in different contexts (Maia et al., 2012; Neto et al., 2010).

Table 1
Cognitive dimensions originally defined by CDN (Blackwell and Green, 2003, pp. 116–118).

Cognitive dimension	Description
Viscosity	Resistance to change
Visibility	Ability to view entities easily
Premature Commitment	Constraints on the order of doing things
Hidden Dependencies	Relevant relations between entities are not visible
Role-Expressiveness	The purpose of an entity is readily inferred
Error-Proneness	The notation invites mistakes and the system gives little protection
Abstraction	Types and availability of abstraction mechanisms
Secondary Notation	Extra information in means other than formal syntax
Closeness of Mapping	Closeness of representation to domain
Consistency	Similar semantics are expressed in similar syntactic forms
Diffuseness	Verbosity of language
Hard Mental Operations	High demand on cognitive resources
Provisionality	Degree of commitment to actions or marks
Progressive Evaluation	Work-to-date can be checked at any time

However, to the best of our knowledge, no previous study has defined a CDN-based metrics suite to support a quantitative evaluation of DSLs. We selected a subset of the CDs to support the evaluation of DSLs in evolving systems (Section 3). According to the literature, DSLs comprise four important aspects: expressiveness, conciseness, integration, and performance (Humm and Engelschall, 2010). However, only the first two characteristics are considered in this paper, since they are important in terms of the language itself. In other words, we aim to evaluate the specifications that the user–developer needs to understand and/or produce and not the interaction of the language with some tool. These two characteristics are defined as: (1) ***DSL expressiveness***, *which refers to the extent a domain-specific language allows to directly represent the elements of a domain*, and (2) ***DSL conciseness***, *which refers to the economy of terms without harming the artifact comprehension*.

2.2. DSLs for detecting architectural problems

Nowadays there are currently hundreds of DSLs, in a wide range of domains in the context of software systems, engineering, and telecommunications, among others (Van Deursen et al., 2000). In particular, there are several DSLs in software engineering particularly intended to support developers in specifying design rules at different levels of abstraction (e.g. Gurgel, 2012; Silva Filho et al., 2011; Moha et al., 2010; Terra and Valente, 2009; Ubayashi et al., 2010; Morgan et al., 2007). For instance, some DSLs are intended to support programmers in defining low-level design rules that are relevant at the implementation level (e.g. Silva Filho et al., 2011; Morgan et al., 2007). As mentioned in Section 2, we chose to apply

[4] From hereafter, we use the term "DSLs" to refer only to textual DSLs.

[5] From hereafter, we use the term "CDs" to refer cognitive dimensions.