Evaluating the Usability in Domain-Specific Languages

Christian Moreira
Universidad de Cuenca
Cuenca, Ecuador *alexander.moreira@ucuenca.edu.ec*

Juan Cobos-Q
Universidad de Cuenca
Cuenca, Ecuador *juan.cobosq94@ucuenca.edu.ec*

Wilson Valdez Solís
Universidad de Cuenca
Cuenca, Ecuador *wilson.valdezs@ucuenca.edu.ec*

Cristina Sánchez-Zhunio
Universidad de Cuenca
Cuenca, Ecuador *cristina.sanchezz@ucuenca.edu.ec*

Priscila Cedillo
Universidad de Cuenca
Cuenca, Ecuador *priscila.cedillo@ucuenca.edu.ec*

Abstract

Domain-Specific Languages (DSL) represent software abstractions that provide semantic to a specific class or domain. DSLs use the concepts and rules from the field or domain to strengthen ties between the domain expert and the technology. Although several DSLs are modeled by considering quality features, most do not have high-quality characteristics related to their usability or do not address them. However, this characteristic constitutes a key aspect for software engineers and domain experts while designing their solutions. Therefore, this paper presents a usability quality model for DSLs aligned to the ISO / IEC 25010 standard and an evaluation model based on the ISO / IEC 25040 standard. Finally, the evaluation method was applied in order to show its feasibility by a case study that addresses the use of two DSLs (related to Ambient Assisted Living and Cloud Computing domain) and assesses their usability.

Keywords: Domain-Specific Languages, Usability, Quality Model, Evaluation Method.

1. Introduction

The software has become a valuable tool to optimize efficiency and satisfy organizations' needs [26]. Hereof, software development has gained the interest of the research and industrial community. In this context, software engineering provides several methodologies to reduce the time spent developing an application and considering criteria to guarantee quality [2,3]. In this sense, to ensure quality, various criteria are defined according to each researcher or entity, and also quality standards must be implemented in each software development stage [21].

Nowadays, several types of software can be identified depending on their approach and generally can be separated into two types: General Purpose Languages (GPL) and Domain-Specific Languages (DSL) [5,6,7]. Regarding DSL, it is a formal limited expressiveness software language that processes algorithms of a specific domain [12]. DSLs could be divided into three categories (i.e., internal, external, language workbench). However, there are highlighted two of them. On the one hand, an internal DSL is the one that is developed in a general-purpose language but only uses a subset of the language's features in a particular style to handle one small aspect of the overall system. As an example of internal DSL, K. Hawick proposes an internal DSL software for lattice-based simulations [11]. Here, the DSL implements lattice-based operations for the entire family of simulation models on different lattices and with different neighborhood locations. On the other hand, an external DSL is built from scratch, implementing each module with custom syntax. Usually, an external DSL has a custom syntax, but it using another language's syntax is also common such as XML [5],[10].

Undoubtedly, a DSL has to consider the qualitative software characteristic called usability to enhance user adherence and experience. Here a starting point could be the techniques used to evaluate the standard user interfaces (UI) or user experience (UX) during the use of DSLs [3]. However, the main problem related to a DSL quality evaluation is the lack of systematic evaluation approaches, guidelines, and a complete set of tools that do not let DSL developers neglect the quality evaluation step [3]. Increased usability is seen as one of the key benefits of DSL over GPL. This has a major impact on the achieved productivity of DSL users. Therefore, it is essential to build good usability while developing DSL. According to A. Barišić. et al., usability assessment is often relaxed or omitted in articles that report on DSL development [2].

However, another point to consider in the problem is that it is difficult to identify its usability strengths and weaknesses at an early stage. This happens because there is no guide on how to reveal these strengths and weaknesses objectively. Usability is a multifaceted quality feature, which is challenging to quantify in advance by DSL stakeholders. There is even less support on how to quantitatively assess DSL usability [1].

Therefore, this paper presents a Method to Evaluate the Usability of DSLs (MEUD). The method is based on the standard ISO/IEC 25040 [15]; MEUD contains activities, artifacts, guides, and roles applied to the DSL usability evaluation domain. Also, a quality model is presented. This model is based on the ISO/IEC 25010 standard [13], which defines eight essential characteristics for product quality, being one of them the usability. From this, the proposed quality model is an essential guide when evaluating the usability of DSLs[13].

This document is structured as follows: Section 2 presents the related work, Section 3 describes each task of method MEUD, Section 4 shows the proposed quality model with its sub-characteristics and attributes, Section 5 presents the evaluation of two DSL by using the MEUD method and the proposed model; finally, Section 6 presents the conclusions and future work.

2. Related Work

In recent years, it is an evident increase in research on the evaluation of usability in DSL. Therefore, this section presents a set of related work to contextualize and deepen this research topic. Some studies explain quality models to evaluate DSL, and in others, the use of a specific technique to evaluate a DSL.

Firstly, Poltronieri et al. [25] present a DSL usability evaluation framework called Usa-DSL. It considers usability concepts and aspects of Human-Computer Interaction (HCI) to evaluate a DSL; also, the authors used focus groups composed of seven subjects during their research in order to validate the framework in the first instance. Moreover, the study presents a case study to evaluate the DSL, conducted by experts in three areas: HCI, software engineering, and performance testing. However, although the study has considered relevant HCI aspects, it has no considered either a quality product evaluation or standards for product quality such as ISO-25010 [13] or ISO-9126 [31].

Besides, A. Barišić. et al. [3], emphasizes that the measure of success of a DSL should be determined by evaluating the impact of its use in practice by the users of the specific context of the DSL. The study uses an evaluation method is known as Domain Specific Inspection (DSI) and is developed using traditional evaluations such as Heuristic Evaluation (HE) and Usability Testing (UT); the evaluations are based on experimentation with the users who interact and the DSL, analyzing the results obtained among the users [3]. This method provides optimal results regarding identifying broad usability problem areas and the usability evaluation method's relevant metrics. The authors concluded that evaluating a DSL has several similarities with evaluating User Interfaces (UI). However, the evaluation presented in [3] does not consider the use of standards and focuses only on the language's engineering and not its usability, which shows a gap to cover in the DSL evaluation research.

Unlike the presented studies, Lopez et al. [20] address the functional suitability (functional correctness, functional completeness, and functional appropriateness) of DSLs derived from different metamodels. The study gives a different approach to DSL quality evaluation since it is based on the ISO-25010 standard [13]. That study also proposes evaluating two metrics to verify the functional suitability of the DSL: assessing the programming complexity implied by a metamodel and the model's domain-specificness. Despite being based on the ISO-25010 standard and the aforementioned metrics, this research only motivates its use.

Moreover, Montenegro et al. [23] present a DSL called KiwiDSM, a tool that allows modeling modules that make up a learning management system (LMS) in the communications area. For the DSL validation, they proposed a model made with KiwiDSM on a platform LMS called ATutor, and a hypothesis is raised "When working with MDA (Model Driven Architecture), the time and effort in the generation of solutions are reduced". The tests consisted of measuring the time and effort that 16 users spend to create modules in five topics in the following order: 1 Chat, 1 Forum, 1 Wiki, 1 Announcement, 1 News, and 1 Note, both in ATutor and KiwiDSM; then, the effort is measured according to the number of occasions in which the user selects or enters some type of information to the system, this according to Yamada. [14]. From the results, it can be highlighted that 58.87% more efficiency is obtained when performing a DSL task than when using the ATutor tool. However, although the study presents a different perspective to address the DSL quality, it is needed to evaluate this DSL's usability is required through the use of a quality model structured from a standard.

To sum up, once analyzed several studies already carried out by various researchers, it was found that existing quality models do not address the evaluation of DSL usability using a quality standard. In this sense, the research conducted in this paper seeks to assess all types of DSL more objectively and systematically through DSL measurement quality solutions. This contribution breaks down the usability sub-characteristic from the ISO/IEC 25010 standard into other sub-characteristics and attributes.

Table 1 is presented a comparison between existing methods for evaluating DSLs with the method proposed in this paper.

Table 1. Main characteristics of the related works and of the proposed method.

Reference	Evaluate	Method used	Standards
[25]	• Usability aspects. • Aspects of Human-Computer Interaction (HCI)	• Focus group	None
[3]	• The impact of your use of DSLs in practice.	• Domain Specific Inspection (DSI). • Heuristic Evaluation (HE). • Usability testing (UT).	None
[20]	• The functional suitability composed of functional correction, functional integrity and functional adequacy. • The complexity of programming. • The specificity of the domain.	Does not explain	• ISO/IEC 25010
[23]	• Weather • Effort to generate solutions.	• Model made with Kiwi DSM on an LMS platform called ATutor	None
Proposed method	Usability through the following characteristics: • Intelligibility • Learning • Operability • Protection against user errors. • Esthetic	• Method to evaluate the usability of DSL (MEUD) based on ISO / IEC 25040	• ISO/IEC 25010 • ISO/IEC 25040

3. A Usability Model for DSL

This section presents a usability model for evaluating DSL tools. The attributes and quality metrics were obtained using sub-characteristics that correspond to the usability characteristic defined by the ISO/IEC 25010 [13]. Five of the six characteristics of the referred standard have been considered: intelligibility, learning, operability, protection against errors, and aesthetics. The remaining feature is accessibility, and this feature is not considered because it focuses on users' ability with limitations to using DSL. Besides, it is relevant to mention that some of these characteristics are only applicable to Graphical User Interface (GUI) – based DSL software tools. The method was designed based on a general user without either motor or cognitive limitations, and who knows the DSL domain, and has basic knowledge for technology use [13].

3.1. Intelligibility

Intelligibility is the product's ability that allows the user to understand whether the software is adequate or not and allows to identify how it can be used to perform different tasks [15]. The considered sub-characteristics and attributes for this quality model are shown in Table 2.

The *symbolism* within the DSL is evaluated through the level of perception, that the user has of the functionalities of the icons and the functionality they provide [14].

The *visual readability* is evaluated by means of four attributes, the first is the arrangement of the components and is measured through the relationship between the number of visible components and the total number of components of the DSL [14], the next attribute is the size of the components and is measured through the relationship between the number of components of adequate size and the total number of components [14], on the other hand, we have the complete screen of components, and it is measured by the level of user satisfaction when viewing the DSL components [14]; finally, the adequacy of the component screen is evaluated, and it is measured through the visualization that the user has of the DSL components, and for this, the background and the color of the components are considered[14].

The *familiarity* is evaluated through two attributes, the first is the popularity of the component and is measured by the relationship between the total number of components known to the user and the total number of components present in the DSL [14], the second is the adequacy of the graphical interface and is measured by the level of adequacy [14].

The *textual semantics* is evaluated by means of the attribute and is the understanding of textual information. It is measured by the user's understanding when reading text in the DSL [14].

Table 2. Sub characteristics and attributes of intelligibility.

Sub characteristics	Attribute	Meaning
1.1 Symbolisms	1.1.1 Significance	Level of perception of the functionalities of the icons and the functionality they provide
1.2.- Visual readability	1.2.1 Component arrangement	The ratio of the number of visible components to the total number of DSL components
	1.2.2 Component size	The ratio of the number of appropriately sized components to the total number of components
	1.2.3 Complete component display	Level of satisfaction when viewing the components of the DSL
	1.2.4 Adequacy of component display	The DSL allows to correctly view the components considering the DSL background color and the color of the components.
1.3 Familiarity	1.3.1 Component Popularity	The ratio between the total number of components known to the user and the total number of components present in the DSL
	1.3.2 Adequacy of the graphical interface	Level of the adequacy of the graphical interface.
1.4 Textual semantics	1.4.1 Understanding of textual information.	Level of understanding of textual information.

3.2. Learning

Learning feature is associated with the ease with which target users of the system (domain experts) can learn to use the system [13]. Table 3 describes the decomposition of this characteristic into understanding, help, and predictability.

The *understanding* is evaluated in two points. The first one represents the average time that the user needs to understand the DSL operation. The second refers to the level of understanding that the user has when observing the DSL interface [13].

The *help* is measured through two attributes; the first one is the effectiveness of the documentation, which seeks to calculate the need that the user has to use the DSL documentation. The second attribute is the activity guide that seeks to calculate the level of feedback that the DSL offers to the user in the different activities that can be carried out [13].

The *Predictability* allows evaluating the user's ability to foresee the operation of the different components of the DSL [13].

Table 3. Sub characteristics and attributes of learning.

Sub characteristics	Attribute	Meaning
2.1 Understanding	2.1.1 Training time	Average time required for the user to understand DSL operation
	2.1.2 Easy to understand interface	Level of understanding of the DSL interface without using support tools
2.2 Help	2.2.1 Documentation effectiveness	Level of need to use the documentation for the use of DSL
	2.2.2 Activities guide	Level of feedback towards the user of the actions that can or are being carried out
2.3 Predictability	2.3.1 Predictability of component actions	Relationship between the number of components with foreseeable actions and the total number of components of the DSL
	2.3.2 Determination of possible permitted actions	The ratio of the total number of shares allowed in a DSL section to the total number of shares available

3.3. Operability

In [13], operability is defined as the product's ability that allows users to operate and control a software product easily. The associated sub-characteristics are adaptability, manageability, efficiency, reliability, and graphical interface adjustment. Table 4 details this characteristic.

Here, i) the *adaptability* seeks to answer the following question: How often does the user use the DSL?, ii) the *manageability* allows the evaluation of the level of complexity of the DSL perceived by the user, iii) the *efficiency* is one of the most important factors within the quality model, with which it is possible to evaluate the time it takes the user to perform specific tasks; also, productivity can be evaluated when using the DSL, iv) the *reliability* evaluates the relationship between the number of actions performed with errors and the number of actions performed in total and v) the *graphical interface adjustment* evaluates if the DSL maintains an adequate size of the components concerning the screen that is being projected.

Table 4. Sub characteristics and attributes of operability.

Sub characteristics	Attribute	Meaning
3.1 Adaptability	3.3.1 Action Time	Time obtained when performing a certain action
3.2 Manageability	3.2.1 Complexity	Level of complexity of the DLS perceived by the user
3.3 Efficiency	3.3.1 Action Time	Time obtained when performing a certain action
	3.3.2 Productivity	Number of actions carried out in a given time
3.4 Reliability	3.4.1 Reliability	List of the number of actions carried out with errors and the number of actions carried out in total
3.5 Graphical interface adjustment	3.5.1 Window size and DSL components	DSL maintains proper component size by adjusting the viewing area

3.4. Protection against user errors

User error protection is a feature that helps protect users from making mistakes when handling DSL. This characteristic consists of two sub-characteristics, error prevention and reversibility. Table 5 shows each of these sub-characteristics in detail.

The error prevention evaluates if the data entered by the user in the DSL are validated; for this, a relationship is made between the number of data entered with errors and the total number of data entered. Reversibility makes it possible to assess whether the user can retrace his steps, that return to a screen or action before the current one.

Table 5. Sub characteristics and attributes of protection against user errors.

Sub characteristics	Attribute	Meaning
4.1 Error prevention	4.1.1 Data entry validation	Relationship between the amount of data with errors entered and the total number of data entered
4.2 Reversibility	4.2.1 Previous state	The ratio of the total number of actions that allow backtracking and the total number of actions that the DSL allows

3.5. Aesthetics

Aesthetics is the user interface’s ability to satisfy the interaction with the user. The sub-characteristics that compose aesthetics are proportionality and visual consistency. Hence, aesthetics is only appliable to GUI-based software tools. This characteristic’s composition is shown in Table 6.

The proportionality evaluates the relationship between the components and the screen; it is observed if the relationship between the area occupied by the DSL screen and the total area of the DSL screen is proportional. Visual consistency allows evaluating the DSL interface's different characteristics, such as the uniformity of colors on the screens, the coherence in the grouping of the components, the organization of the components, etc. [21[19].

Table 6. Sub characteristics and attributes of aesthetics.

Sub characteristics	Attribute	Meaning
5.1 Proportionality	5.1.1 Size ratio between components and screen	The proportional relationship between an element occupies the DSL screen and the DSL screen's total area.
5.2 Visual consistency	5.2.1 Color uniformity on screens	The ratio of the number of displays with similar colors to the total number of DSL displays
	5.2.2 Consistency in the grouping of components	The ratio of the number of grouped components to the total number of DSL components
	5.2.3 Component organization	Level of identifiability and accessibility to the different elements of the DSL in an organized way
	5.2.4 Component layout in the graphical interface	Relationship of the arrangement of the components on the screen to the total number of components of the DSL

4. Method to Evaluate the Usability of DSLs (MEUD)

This section presents a Method to Evaluate the Usability of DSLs (MEUD) based on ISO/IEC 25040 [29]. The evaluation process consists of five steps: establish the evaluation requirements, specify the evaluation, design the evaluation, execute the evaluation, and conclude the evaluation [15]. Fig. 1. represents this process by using the Software Process Engineering Metamodel (SPEM) specification [24]. The figure shows each stage and role of the evaluation method, joint with the most relevant documents obtained through the process.

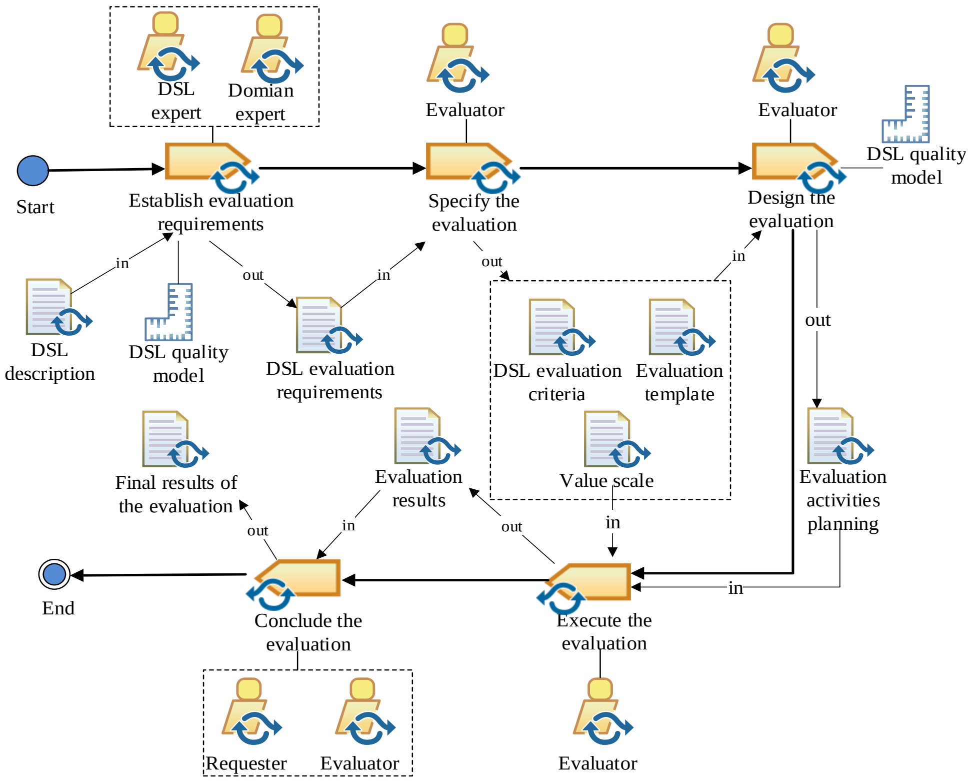

Fig. 1. Method to Evaluate the Usability of DSLs (MEUD) process.

The process starts with *establishing the evaluation requirements* that consist of defining the objective of the evaluation, the product quality requirements by selecting the characteristics and sub-characteristics to be evaluated, and the parts of the product to be included in the evaluation. The second stage is the *specification of the evaluation*, which involves three activities such are selecting the metrics, define metrics threshold or values scales, and obtaining the evaluation template are obtained. The third stage is the *evaluation design* by using the evaluation template and the quality model presented in Section 3. The fourth stage, *execution of the evaluation*, is carried out by the evaluator. The evaluation consists of four activities starting from the measurement of variables defined in the planning, then applying the thresholds for each metric, afterward that the criteria established for evaluation are applied using the evaluation template, and finally, the evaluation is made to the DSL, and the document with the evaluation results is generated. The last stage is the *conclusion of the evaluation*; here, the evaluator and the applicant interact to review the evaluation results and the processing of the evaluation data to create the final evaluation results report finally.

5. Case Study Applying the MEUD Process

MEUD process is appliable to any DSL usability evaluation by following the proposed steps below enhanced with an application example. The selected scenarios to evaluate the quality model and the evaluation method are two GUI-based DSL software tools for modeling Ambient Assisted Living (AAL) solutions. These DSLs let model both Fog Computing networking and Microservices for devices in AAL.

On the one hand, FOGAAL DSL [7]. This DSL tool lets modeling Fog Computing Architectures for AAL. This DSL is composed of *environments, devices, fog nodes, users, network parameters*, *communication protocols*, and *data format* for the devices within AAL scenarios, all these elements were chosen by following the SOPRANO ontology [27]. The modeling interface lets the user creating his/her Fog Computing architectures to address specific intelligent environments oriented to people with degenerative diseases or the elderly.

On the other hand, MICAAL DSL [8]. This DSL tool complements FOGAAL DSL; it lets model microservice architectures for IoT devices within an AAL environment. This graphical DSL is based on the Spring Cloud Netflix microservices architecture, and it allows modeling microservices capabilities to the created devices. The modeling interface is composed of the core element of the Spring Cloud Netflix microservices architecture, which is the *Container* and thus *Load Balancer*, *Circuit breaker*, and *Log Management*. Besides, the *edge server* for the devices and *Configuration, Authorization, Register,* and *Domain services* [28].

The evaluation process has been applied to evaluate the quality of the GUI-based DSL previously described. Then, in the lines below is described the step-by-step journey to assess the quality of these DSLs by using MEUD.

5.1. Establish evaluation requirements

Establish the objective of the evaluation

To define the objective of the evaluation, the Goal-Question Metric (GQM) proposed by Basili et al. [4] was applied (See Table 7).

Table 7. Objective definition with GQM

GQM variable	*Response*
Evaluate	Domain Specific Languages (DSLs)
With the purpose of	Evaluate the usability of DSLs
From the viewpoint of	Quality Engineers
In the context of	A group of undergraduate students of Computer Science

The undergraduate students are the future professional generation, and they have close technical capability to the interested population. Hence, to consider students in the evaluation process is not a problem as it is essential to evaluate the method with expert or nonexpert professionals [17]. The students that carried out the assessment had a preparation stage where both the DSL tools and MEUD were presented in detail to deeply immerse them into the domain. Then, the students had one hour to know and use the DSLs and then solve doubts. After that, they applied MEUD to evaluate the DSLs usability.

Establish quality requirements

The evaluation is applied in FOGAAL and MICAAL DSL tools to provide feedback during their development and use [9]. The development process's objective is to predict usability and correct potential errors, and in the use process by end-users, to evaluate usability.

The evaluation is conditioned to i) the process of using the graphic tool since it has an impact on the proper functioning of the DSLs, ii) the method of design and creation of DSL used by the author because it must be flexible to integrate changes and iii) for the context of use of the DSLs (age, language, operating system and device).

Identify parts to evaluate

In this activity, the usability model proposed in Section 3 is applied. The attributes to be evaluated in FOGAAL and MICAAL DSL tools are selected.

5.2. Specify the evaluation

In this phase, you have to establish the evaluation criteria and define criteria for the metrics. The specification of this evaluation is described below:

Establish evaluation criteria

For this evaluation, since the evaluation was carried out on DSLs already developed, the selection of characteristics to evaluate and the evaluation design was based on the quality model focused on usability. Therefore, from this model, it was possible to obtain the sub-characteristics to be evaluated in both cases. In this sense, the values established for the limits representing the degree of usability of the DSLs are presented in Table 8.

Table 8. Usability levels.

Usability level	Threshold limits
High	0.80 < Usability level < 1
Medium	0.50 < Usability level < 0.79
Low	0 < Usability level < 0.49
Non-existent	Usability level = 0

Define criteria for metrics

Hereof, to calculate the usability that each DSL sub-characteristic has, it is made a first average of the attributes evaluated concerning the sub-characteristic by using the following equation:

$$X = \frac{\sum i(attribute)}{Num\ attribute} \tag{1}$$

Subsequently, a second average is performed within all the sub-characteristics to obtain the DSL usability level. Hereof, equation one is used as in the first average, and then, a final average is generated from the obtained percentages to determine the DSLs usability level by using equation 2:

$$Usability_{\ level} = \frac{\sum i(sub\text{−}characteristics)}{Num\ characteristics} * 100\% \tag{2}$$

However, beyond following the same procedure for both cases, different results were obtained for each DSL.

5.3. Design the evaluation

In this phase, usability problems are reported by using the template proposed by Fernández [9] (See Table 9), and the evaluation plan is also elaborated here. The plan of this evaluation is to execute all the artifacts necessary to execute the DSLs and thus obtain the metrics.

Table 9. Template to report usability problems.

ID	Code to identify the usability problem detected
Description	Description of the problem identified
Affected attribute	ID sub-characteristic/ID Attribute (Use the Usability Model for DSL of Section 3)
Severity level	Criticality level of the intervals defined above for the measure
Occurrences	Number of appearances of the same usability problem detected
Recommendations	Recommendations to correct the usability problem detected
Priority	Priority Importance of the usability problem (High, Medium, Low)
Resources	Resources needed to correct the proposed changes

5.4. Execute the evaluation

In this phase, the measures are executed by applying the decision criteria for metrics to the DSL evaluation; hence, the evaluation of the DSLs are detailed below:

Here, the metrics are applied for each selected attribute. To evaluate all the attributes of each sub-characteristic, the metric and its form of evaluation are considered as shown in detail in the quality model. Moreover, for the attributes whose metrics are evaluated using a Likert scale, a usability percentage was established according to the option to obtain the sub-characteristics averages. Then, the results of this process are presented in detail in Table 10.

Table 10. DSLs sub-characteristics detailed attributes results.

Sub-Characteristic	Attribute	Metric value FOGAAL DSL	Metric value MICAAL DSL
Intelligibility Results	Symbolisms	1	0.50
	Visual readability	0.75	0.66
	Familiarity	0.14	0.12
	Textual semantics	0.10	0.10
Learning Results	Understanding	0.80	0.80
	Help	0.20	0.10
	Predictability	0.59	0.50
Operability Results	Adaptability	1	1
	Manageability	0.80	0.50
	Reliability	1	1
	GUI tuning	1	1
Results of protection against user errors	Error prevention	1	1
	Reversibility	0.80	1
Esthetic Results	Proportionality	0.1	1
	Visual consistency	0.95	0.95

From this, are determined the usability levels for each sub-characteristic as shown in Table 11.

Table 11. DSLs sub-characteristics results.

Sub-Characteristic	FOGAAL DSL		MICAAL DSL	
	Metric value	Usability level	Metric value	Usability level
Intelligibility	0.49	Low	0.35	Low
Learning	0.53	Medium	0.46	Low
Operability	0.95	High	0.88	High
Protection against user errors	0.90	High	1	High
Esthetic	0.98	High	0.98	High

5.5. Conclude the evaluation

Review evaluation results

As a result, FOGAAL DSL has an average usability level of 77%, calculated as presented in equation 3.

$$FOGAAL\ Usability_{level} = \frac{(0.49+0.53+0.95+90+0.98)}{5} * 100\% = 77\% \quad (3)$$

As presented before, the type of metric and its form of evaluation must be considered for MICAAL DSL, the average of usability is calculated as presented in equation 4. Then, the level of usability is 73.4%.

$$MICAAL\ Usability_{level} = \frac{(0.35+0.46+0.88+1+0.98)}{5} * 100\% = 73.4\% \quad (4)$$

Evaluation data processing

As a conclusion of the evaluation method, the results indicate that both FOGAAL and MICAAL have a medium usability level. Hence, the proposed evaluation method meets expectations and generates acceptable results when evaluating DSLs.

The evaluators gave their feedback about the DSLs assessment through MEUD and the quality model. Overall, they were able to apply the provided tools and highlighted the ability to evaluate DSLs in that way easily. Besides, they mentioned the need to assess another feature beyond usability, such as the case of functional suitability.

6. Conclusions and Future Work

The Quality of Software is an essential part of Software Engineering considered when developing any software. In this sense, regarding Domain-Specific Languages (DSL), on the one hand, knowing that these are software tools; and on the other hand, understanding the need of having models which let measuring the quality of these tools. This paper has presented a quality model aligned to ISO 25010 standard to evaluate Usability characteristics for any DSL and an evaluation method aligned to ISO 25040 standard to assess the quality model.

The quality model considers twenty-four attributes divided into five sub-characteristics, and each attribute is evaluated with a specific metric. Thus, the model lets the evaluation of DSL tools' usability by measuring the most important features. Moreover, to verify the effectiveness of the evaluation method and thus the quality model, this paper presents an evaluation of two GUI-based DSL tools. As a result, was obtained 73.4% usability level for the MICAAL DSL and a 77% usability level for the FOGAAL DSL. In this sense, it can be concluded that the level of usability presented by the DSL tools evaluated is medium.

Hence, the presented model's benefit is faced with the need for DSL tools to evaluate their quality as described in the related work Section. Consequently, this is the first step towards a robust quality model that considers more DSLs characteristics beyond usability.

However, there are necessary future evaluation stages, such as controlled experiments with domain expert groups. In that sense, it could be used methodologies for evaluating user perceptions about the use of MEUD and the quality model.

Acknowledgement

This work is part of the following research projects: “Fog Computing applied to monitor devices used in assisted living environments; case study: a platform for the elderly people” and “Design of architectures and interaction models for assisted living environments aimed at older adults. Case study: playful and social environments”. Therefore, we thank the Research Department of Universidad de Cuenca (DIUC) for its support.

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. Barišić, A., Amaral, V., Goulão, M.: Usability evaluation of domain-specific languages. In: Proceedings - 2012 8th International Conference on the Quality of Information and Communications Technology, QUATIC 2012. pp. 342–347. (2012)
3. Barišić, A., Goulão, M., Amaral, V., Barroca, B.: Evaluating the usability of domain-specific languages. Softw. Des. Dev. Concepts, Methodol. Tools, Appl. 4–4 2120–2141 (2013)
4. Basili, V.R., Agresti, W.W.: The experimental paradigm in software engineering. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 706 LNCS 33–37 (1993)
5. Brady, M.H., Cioffi, J.M.: The Worst-Case Interference in DSL Systems Employing Dynamic Spectrum Management. EURASIP J. Adv. Signal Process. 2006 (1), 078524 (2006)
6. Callejas-Cuervo, M., Alarcón-Aldana, A.C., Álvarez-Carreño, A.M.: Modelos de calidad del software, un estado del arte. Entramado. 13 (1), 236–250 (2017)
7. Cedillo, P., Valdez, W., Lara, F.: FOGAAL: A Domain Specific Language for Fog Computing Architectures Design for Ambient Assisted Living Environments (Working Paper). (2021)
8. Cedillo, P., Valdez, W., Moyano, J.: MICAAL: A Domain Specific Languaje for Fog

Computing Microservices Based Architectures Design for Ambient Assisted Living Environments (Working Paper). (2021)

9. Fernández Martínez, A.: A Usability Inspection Method for Model-driven Web Development Processes. Universitat Politècnica de València (2012)
10. Fowler, M.: Domain-specific languages. Pearson Education (2010)
11. Hawick, K.A.: Engineering internal domain-specific language software for lattice-based simulations. Proc. IASTED Int. Conf. Softw. Eng. Appl. SEA 2012. 314–321 (2012)
12. Hinsen, K.: Domain-Specific Languages in Scientific Computing. Comput. Sci. Eng. 20 (1), 88–92 (2018)
13. ISO: ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, https://www.iso.org/standard/35733.html, Accessed: December 09, 2020, (2011)
14. ISO: ISO - ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, https://www.iso.org/standard/35733.html, Accessed: January 13, 2021, (2011)
15. ISO: ISO - ISO/IEC 25040:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process, https://www.iso.org/standard/35765.html, Accessed: February 03, 2021, (2011)
16. Kihlman, L.: Producing domain-specific languages from strategy patterns. In: 2015 7th Computer Science and Electronic Engineering Conference (CEEC). pp. 9–12. IEEE (2015)
17. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. 28 (8), 721–734 (2002)
18. Kowalski, M., Wilkosz, K.: A Domain Specific Language in Dependability Analysis. In: 2009 Fourth International Conference on Dependability of Computer Systems. pp. 324–331. IEEE (2009)
19. Liu, Y., Zhang, Q.: Interface Design Aesthetics of Interaction Design. (2019)
20. López, J.: Aseguramiento de la calidad en el diseño del software. 170 (2014)
21. Marcela, L., Constanzo, A.: Comparación de modelos de calidad, factores y métricas en el ambito de la Ingeniería de Software. Inst. Tecnol. Apl. 78 1–36 (2014)
22. MK Bajuri: Análisis, diseño e implementación de un prototipo de un Lenguaje de Dominio Específico (DSL) interno, orientado al modelado de problemas de programación lineal. Phys. Rev. E. (1993), 24 (2015)
23. Montenegro, C., Cueva, J., Martinez, Ó., Geona, P.: Desarrollo de un lenguaje de dominio específico para sistemas de gestión de aprendizaje y su herramienta de implementación "KiwiDSM" mediante ingeniería dirigida por modelos. Ing. 15 (2), 67–81 (2010)
24. Object Management Group: Software & Systems Process Engineering Meta-Model Specification. (2008)
25. Poltronieri, I., Zorzo, A.F., Bernardino, M., De Borba Campos, M.: USA-DSL: Usability evaluation framework for domain-specific languages. Proc. ACM Symp. Appl. Comput. (April), 2013–2021 (2018)
26. Singh, B., Kannojia, S.P.: A review on software quality models. Proc. - 2013 Int. Conf. Commun. Syst. Netw. Technol. CSNT 2013. 801–806 (2013)
27. Sixsmith, A., Meuller, S., Lull, F., Klein, M., Bierhoff, I., Delaney, S., Savage, R.: SOPRANO - An ambient assisted living system for supporting older people at home. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 233–236. Springer Verlag (2009)
28. Spring: Spring Cloud Netflix, *Spring Cloud Netflix*, (2020)
29. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process: ISO/IEC 25040:2011, (2011)
30. Vijayasarathy, L.R., Butler, C.W.: Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? IEEE Softw. 33 (5), 86–94 (2016)
31. ISO - ISO/IEC 9126-1:2001 - Software engineering — Product quality — Part 1: Quality model