A DSL Framework for requirements engineering.

Tea Tavanxhiu
University of Tirana, tea.tavanxhiu@unitir.edu.al

Kozeta Sevrani
University of Tirana - Faculty of Economy, kozeta.sevrani@unitir.edu.al

A DSL Framework for requirements engineering.

Tea Tavanxhiu[1], Kozeta Sevani[1]

[1]University of Tirana
Tirana, Albania
tea.tavanxhiu@unitir.edu.al
kozeta.sevrani@unitir.edu.al

Abstract. This research paper explores the integration of Domain-Specific Languages (DSLs) as a modeling framework for requirement engineering in software development lifecycle. The instantiation of the DSL is enabled form a proposed architecture of the Framework. The study investigates the benefits and challenges of using DSLs, emphasizing increased involvement of domain experts, reduced delivery time gaps, wider visibility, and reduced technology dependency. Through an Active Design Research (ADR) methodology, the paper consists in the execution of the first ADR cycle, proposing design principles for a DSL Framework. The findings highlight the importance of integrating domain knowledge, conceptual modeling, and semantic enrichment in requirement engineering. Further refinement of the empirical studies and feedback gathering from users on the proposed DSL framework will be part of the research project of the authors starting with this.

Keywords: requirement engineering, information system design theory, action design research, domain specific language.

1 Introduction

The software development life cycle is complex and, in most cases, seen as an interdisciplinary project when correlated to the context of the usage of the application itself as well as the human interaction with the software. Requirement engineering is one of the initial phases of the software development lifecycle and is ofen referred to process of understanding of the stakeholder's needs and documenting such needs once analysed. [1] It is as well considered as one of the most relevant topics of evaluation of the software quality and this calls for the need of continuos improvement of the undersatndng, approaches and tools used to conduct such phase during the development process of the software.

There are multiple suggestions in the literature of what can be used for conducting the requirement engineering phase, and the scope of this research paper is to focus on a framework that considers the specificities of the domain and the problem/need at hand. We strongly believe that the interdisciplinarity of the information systems derives in an important rate from the requirement engineering process, and in this contex, we present two main arguments related to requirement engineering frameworks and domain-specific language (DSL) as a modeling framework for requirement engineering. The first argument discusses various requirement engineering frameworks and theories, highlighting the need for a comprehensive approach that integrates domain knowledge, conceptual modeling, and semantic enrichment. The second argument focuses on the potential benefits and challenges of using DSLs as a modeling framework for requirement engineering, emphasizing the advantages of increased involvement of domain experts, gap reduction in delivery time, wider visibility, and reduced dependency on technology.

There are two research questions being addressed in this paper.

Research Question 1: How can the integration with DSL enhance the requirement engineering aprocesses?

Research Question 2: What are the benefits and challenges of using Domain-Specific Languages (DSLs) as a modeling framework for requirement engineering?

The chosen methodology to conduct the research is through Active Design Research due to its ability to intercorrelate the domain expertise of practitioners with the literature and theory research during all the phases of an ADR cycle. This research conducts one ADR cycle while proposing the outputs of the evaluation, reflection, and learning phase as an input to the second ADR cycle and future work.

1.1 Requirement engineering frameworks and relevant theories

Requirement engineering is a discipline that is primarly related to software engineering, although there is trace in the literature that there are correlations between requirement engineering and knowledge problem in the theory of a problem domain [33]. When engineering the requirements an iterative process starting with the problem and context identification, moving toward domain analysis and theory adeguation, and concluding with implementation and evaluation. Given the nature of the requirement engineering discipline mostly focused on action design problems, we have chosen for this research work the Action Design Research methodology.

Consulting the body of literature [2][3], there is trace of proposed ideas for a framework to depict requirements using a specific DSL, the Unified Modeling Language (UML). The main phases identified in a framework of interest for our research [2] include the feasibility study, the requirement collection and specification, the analysis of business requirements, the system requirement modeling, and the system design, which executed in a cycle might propose some main building blocks for such a framework.

The NATURE framework [4] is a more mature framework and provides a rigorous foundation for requirements engineering and it integrates various disciplines by suggesting that the major divisions in the domain theory are [4] the conceptual model construction and the enrichment of the modeling languages and it's semantics. During our research on the NATURE framework, we came across some really useful publications [5],[6],[7],[8],[9],[10] that actually have conducted research work based on the Nature Framework. These publications have progressed and applied the framework to the domain of action research and this plays in favor of the maturity of the franmewor. Publications such as [11], [12], [13] are of important interest ot our work given the very close nature of research to ths paper.We will refer to the NATURE framework as an important example of an early work that identified and highlighted the need of a comprehensive approach to requirements engineering.

1.2 Domain specific language as a modeling framework for requirement engineering

Requirement engineering is a knowledge problem and as such the solving of the problem requires action design research and a design problem analysis [14]. Domain knowledge is a critical component in the analysis and assessment of requirement engineering. The framework proposed in this paper attemots to integrate the convergence of influences coming from domain knowledge and domain-specific language, thereby proposing an operative approach to the requirement engineering and modeling procedures.

DSL enable domain experts to directly contribute to the development effort by autonomously specifying parts of the solution.[15] The advantages of using a DSL based framework for requirement gathering analysis, modeling and engineering would be as following:

- Increased involvement of the business domain experts in the requirement analysis and modeling processes.
- Gap reduction in terms of time to delivery of the operational and functional requirements.
- Greater insight from industry professionals and experts into the comprehensive scope of designed requirements for the software application.
- Higher understanding of the intercorrelated engineered or to be engineered functionalities due to the multidisciplinary background of the business domain experts.
- Reduced dependency from the technology stack and provide an abstract meta-model of design that can be applied in different applications.

Even though DSL can be a good solution to building abstract modeling languages that do not need to be instantiated but might as well be re-used in different software

environments, it is important to acknowledge the challenges and limits it poses. As discussed in [16] these challenges and problems lie within the following categories of model abstraction, model decomposition, model translation and a main noted problem is the model adaption in other applications.

To bypass these problems this paper proposes an instantiable framework that lies upon an application and the DSL is implemented within the framework, leaving every possible application untouched with the DSL logic, syntax, and complexity. This would enable the maintenance of the framework and the DSL, the communication with multiple applications regardless from their architecture or technology stack and the enrichment in a single DSL meta-framework of all identified and mapped operational, functional, transactional, or non-functional requirements.

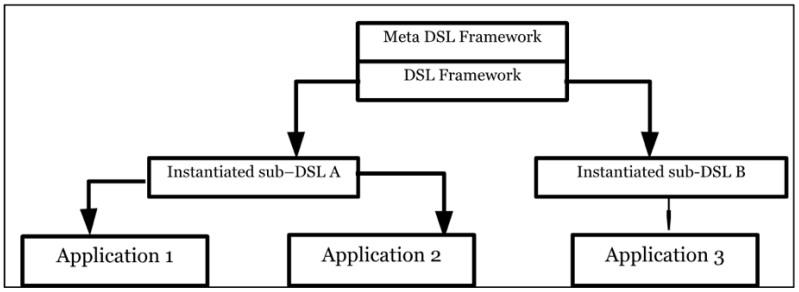

Table 1. Proposed DSL Framework

2 Information Systems Design Theory

The present paper focuses on the Theory for Design and Action as the primary theory type of interest, which is action-oriented and primarily concerned with practical aspects of designing, building, and implementing information systems. As per the categorization of theories, there exist five macro types [17], and Information Systems Design Theory (ISDT) falls under this category. The ISDT provides design principles that facilitate the accomplishment of specific objectives in real-world settings, without restricting hypothetical evaluations to controlled experimental environments. Also, it equips theoretical instructions for developing and enabling a specific type of information system, while also contributing to the expansion of knowledge in the field [18], [19].

Design Product		
1.	Meta-requirements	Describes the class of goals to which the theory applies
2.	Meta-design	Describes a class of artifacts hypothesized to meet the meta-requirements
3.	Kernel theories	Theories from natural or social sciences governing design requirements
4.	Testable design product hypotheses	Used to test whether the meta-design hypotheses satisfies the meta-requirements
Design Process		
1.	Design method	A description of procedure(s) for artifact construction
2.	Kernel theories	Theories from natural or social sciences governing design process itself
3.	Testable design process hypotheses	Used to verify whether the design hypotheses method results in an artifact which is consistent with the meta-design

Table 1. Components of an Information System Design Theory [18]

3 Research approach

The research approach of this paper to formulate an Information System Design Theory follows the framework proposed by Walles et al [18] and the design principles of the framework are based on the reusability principles of Iivari et al [20] and the approach of Sein et al [21] to construct, intervene and evaluate IT-artifact within an organization which will lead to the contribution into design principles and theories.

The sensitivity toward the usage of theory while conducting the research is inspired by Iivari [22] and as such it is important for this research to incorporate theoretical foundations. ADR as a method that combines action research and design science research, aims to develop innovative IT solutions while simultaneously addressing organizational or societal problems. To be aware of the challenges that ADR poses as a chosen methodology and to be able to correctly address the nascent problems from using it, we will base the usability of this methodology to the finding of Haj-Bolouri et al [23].

Empirical findings and kernel theories will contribute to the creation of the Information System Design Theory of this paper.

4 Action Design Research cycle

The ISTD of this paper has been generated by running one cycle of the ADR - method. While identifying the practice inspired research principle which derives from multiple workshops with supply chain management experts and practitioners, we have intentionally gone through a thorough literature research to gather a full understanding and to base the paper on a theory-integrated base.

These two principles from the problem formulation phase have been used to generate a class of features for the proposed framework which after implementation are evaluated.

The first ADR cycle will be followed by to other ADR cycles to complete the whole research project and the outputs, evaluations, and reflections from the first ADR cycle will serve as an initial analysis in the problem formulation phase of the second ADR cycle.

The whole three predicted cycles will be complemented with the reflection and learning principle to finalize the research project with generalized outcomes and a contribution to the exiting knowledge on the usage of DSL for requirement engineering.

The first activities toward the identification and formulation of the problem have been 2 workshops with practitioners from the supply chain management industry and the abstract outcomes are as following:

- DSL shall empower the practitioners and consultants of the organization to model and engineer new requirements (functional, transactional, operational) form the frequently changing dynamics of the organization due to inner and outer factors.
- A DSL Framework laying outside of the applications that provides requirement engineering for the applications, would ease the process of technology change, reengineering of applications and the procurement process of applications.
- The DSL syntax is a layer of complexity that need to be overcome with the proper interactive training and documentation for the practitioners.
- The technology stack of the Framework should be able to offer microservices and operate in different communication methods. Very new but relevant communication method should be implemented within the Framework.
- The implementation process should ensure the completeness of all requirement engineering categories possible in the domain of interest. All novel requirement engineering cases should fall within the identified, analyzed and implemented categories of requirement engineering.

The scope of these two workshops was to address questions, issues and problems

deriving from the industry in understanding the system requirements and needs in terms of modeling and engineering day-to-day requirements through a DSL. The workshop topics of discussion were combined with the theoretical background from the literature review process. An alpha prototype was generated which had implemented 5 macro categories of requirement engineering areas and 497 types of specific requirement engineering cases in total for all 5 categories.
The identified categories of interest for the alpha prototype are displayed in Table 2.

Macro category of requirement engineering area in the alpha prototype	**Total number of requirement engineering cases implemented through a DSL in the framework**
Accounting	103
Business	100
Finance	100
Marketing	99
Statistics	95

Table 2. Identified macro categories of requirement engineering in the alpha prototype and the respective number of requirement engineering cases analyzed and coded in the Framework.

The alpha prototype of the framework was developed with the primary objective of providing an evaluation basis for both researchers or practitioners who are part of this research project. Furthermore, it aimed to offer insightful input for the subsequent ADR cycle that would be subject to future design decisions.
The reflection and learning stage provided important feedback toward the challenges that such Framework would pose to a possible implementation in an organization such as the ability from a practitioner perspective to abstract upon the granularity of the specific case of requirement engineering and the capability to categorize such specific case. The maintenance of the framework also requires further investigation, analysis and poses a governance topic for discussion in future ADR cycles.

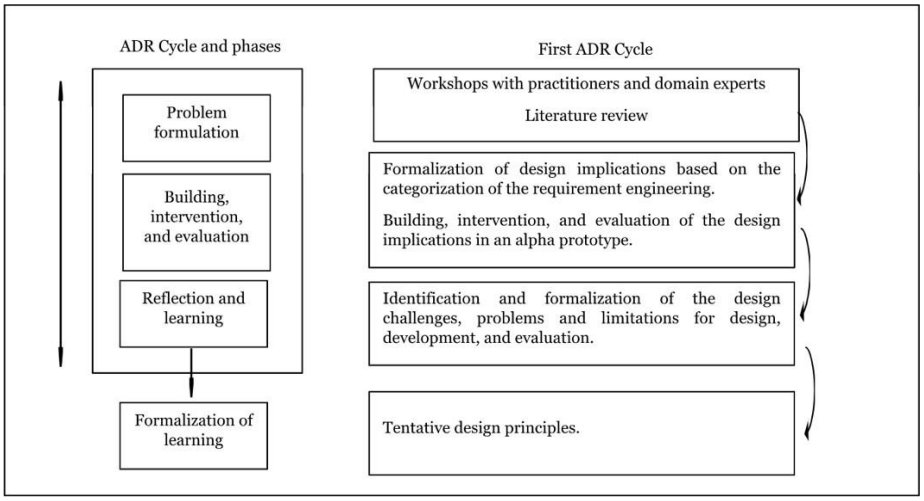

Table 3. ADR executed cycle.

In the first cycle of ADR, the evaluation phase plays a very important role in providing crucial feedback regarding the feasibility of the project and its potential to contribute to the design principles of ISDT. The design principles themselves have undergone evaluation through the interaction with practitioners. In the following paragraph, we shall delve further into the findings and contributions of said evaluation.

5 Information Systems Design Theory for the DSL Framework

Relying on the framework for Information System Design Theory proposed by Walls et al [18], we have executed all the necessary phases prescribed by the DSR methodology. This involved identification of the core theories regading DSL and requirement engineering, determination of the requirements for the proposed artifact, establishment of features that would direct the design process of the artifact, development of the artifact, implementation, and eventual evaluation.

We used Action Design Research in this research project to undertake an analysis of the relationship between requirement engineering and information system design (ISD).

We are aware that requirement engineering is more frequently linked to software engineering than to Information System Design. This study focuses on understanding and proposing a multidisciplinary approach in the requirement identification, modeling, and engineering processes, particularly in the context of ISD's design process and theory, by examining the existing literature. The studies cited as [24], [25], and [26] offer insightful information and support our initial presumtion.

Based on the input from the workshop, and a consultation to existing literature on the classification, categorization, and modelling of DSL [27], [28], [29], we have modeled 5 main categories which result with a common usage in different industries and can be a good ground for the evaluation of the alpha prototype.

In this paper, we have used the FEDS framework proposed by Venable et al [30] as guiding principle to ensure that the contribution constituent of the structure focuses on the impact of the investigation on the subject region during the assessment stage.

The evaluands in this case are the requirement engineering cases that were categorized. The artefacts/evaluands are evaluated altogether divided by category, after the design artefacts is developed. A naturalistic evaluation assessed the impact made by the framework by using the categories of implemented DSL in the framework above two applications in the Supply Chain Management digital environment of a retail company. The evaluation is performed using two specific information systems in an organization with real and impacting problems deriving from the requirement engineering processes. The evaluation phase was successfully completed with a good level of user acceptance of the DSL Framework and a set of challenges, comments, and ideas that will be used as input during the problem formulation stage of the upcoming cycle of the ADR.

The kernel theories for the Information System Design Theory used for this research are categorized as following:

- Requirement engineering
- Literature on DSL and DSL Frameworks
- DSL Modeling and Semantics
- Domain knowledge on business processes.

The design principles produced by the design theory consultation and the design process of the artifact are explicites in the next paragraphs. .

The first design principle identifies the multidisciplinary character of requirement engineering and calls for a significant contribution for a significant contribution from domain knowledge and cross-correlation with practitioners and experts in the field. [1], [4],[6].

From the literature research we have identified that an underlying set of operational, transactional, functional, and non-functional requirements deriving from the organizations will conduct a reciprocal interaction process between the researcher, domain experts, practitioners, and end users of the proposed framework.

The second design concept focuses on the definition of a model and the semantics of the innovative DSL that is used in the proposed framework based on the rules already in place and a review of other relevant used DSLs. [15], [28], [31], [32]. To properly conduct the construction of the DSL we have conducted an iterative process of

domain definition and scope of the DSL, semantics and the communication microservices toward the information systems, evaluation, testing and continuous refinement and enrichment of the cases and categories of DSLs.

6 Conclusion and future work

Following the recommendations from Walls et al. [18] and Gregor et al. [19] for creating, formulating, and formalizing the design principles of Information System Design theory, this paper describes the execution of the first ADR cycle for the development of an implemented artifact.

Such principles can help practitioners and research in providing a theory-based guidelines for the construction of Framework that uses DSL in requirement engineering, as well providing theory-based principles that can be subject to further empirical evaluation.

These design principles are formulated and sustained both by empirical activities and from kernel theories and as such can be validated and considered a valuable contribution to the ISDT for a DSL Framework.

We acknowledge that the generalization of the rules and knowledge to the Design Theory is limited by the reflection and learning stages of the ADR cycle described in this paper. More additional ADR cycles will be required to validate, improve, and generalize the suggested design principles before they can be considered as novelty and an ISDT contribution to a DSL Framework for Requirement Engineering.

Drawing upon the investigations conducted in our paper, it has been determined that the assimilation of domain expertise, conceptual modeling, and semantic augmentation into requirement engineering frameworks can yield noteworthy enhancements in the quest for a comprehensive approach to information system modeling. This is particularly significant when contemplating the proficiency of end users. Theoretically, our research refers to a framework that comprises domain knowledge and provides a Domain-Specific Language (DSL) to expert users of the organizations for modeling information systems based on business requirements. This is mostly beneficial in terms of defining data flows and engineering functional requirements.

By utilizing domain-specific concepts, terminology, and syntax through the DSL, expert users can effectively capture and represent the complex details of the organization's requirements that might change over time. This facilitates better communication between domain experts and system designers, leading to a more exact and comprehensive representation of the desired information system.

Additionally, the framework can now capture the semantic meaning underlying the requirements thanks to the introduction of conceptual modeling approaches and semantic enrichment, ensuring that the system's architecture closely matches the intended business objectives. This not only enhances the clarity and understanding of the information system but also improves its overall effectiveness and efficiency.

Nevertheless, it is important to acknowledge that our conclusions are derived from a theoretical examination. Further empirical studies and real-world implementations are necessary to authenticate the practical benefits and potential challenges associated with integrating domain knowledge, conceptual modeling, and semantic enrichment into requirement engineering frameworks.

Using Domain-Specific Languages (DSLs) as a modeling framework for requirement engineering offers several benefits, as already mentioned in the research work, but it also presents challenges across different domains. These challenges should be taken in consideration while addressing the second research question and they are divided into three main areas. The first area consist in the technological gap from a system perspective in accepting modeling requests from the DSL Framework. Second area consists in the exper - user perspective and their ability to bypass the important learning curbe for the usage of the specific implemented DSL. Last, there are challenges to be addressed from a system architectural perspective, as this approach

would require a convergence in the Framework of all the existing architectural infrastructures (or at least the selected ones) that will exchange requirement engineering and modeling with the DSL framework.
This paper indicates that there will be future ADR cycle and that the problem formulation phase will start with the challenges and issues regarding the proposed artifact identified during the Reflection and Learning phase of the first ADR cycle. The problem formulation phase of the future ADR cycle will be subject to empirical studies, workshops and response gathering from end-users that experienced the requirement engineering through the alpha prototype of proposed DSL Framework.

References

[1] Batool, A., Hafeez Motla, Y., Hamid, B., Asghar, S., Riaz, M., Mukhtar, M., Ahmed, M., (2013), "Comparative Study of Traditional Requirement Engineering and Agile Requirement Engineering", Advanced Communication Technology (ICACT), 2013 15th International Conference.
[2] Pandey. Dh., Suman. U., Ramani, A.K. (2011). "A Framework for Modelling Software Requirements", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011
[3] S. Konrad, H. Goldsby, K. Lopez, (2006) "Visualizing Requirements in UML Models." International Workshop REV'06: Requirements Engineering Visualization,
[4] Jarke. M., Bubenko. J., Rolland. C., Sutcliffe. A., Vassiliou. J., (1993) "Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis"
[5] Pohl, K. (1994). "The three dimensions of requirements engineering: a framework and its applications", Proceedings of the Second International Conference on Requirements Engineering (ICRE'94). IEEE.
[6] Pohl, K., Haumer, P., Jarke, M. (1996). "Process-centered Requirements Engineering", Proceedings of the Third International Conference on Requirements Engineering (ICRE'96). IEEE.
[7] Pohl, K., Sikora, E. (1995). "Structured Natural Language: A Framework for the Integration of Formal and Informal Specification Techniques", Proceedings of the 17th International Conference on Software Engineering (ICSE'95). ACM.
[8] Broy, M., Krüger, I. H., Pretschner, A. (2001). "Engineering Automotive Software", Proceedings of the IEEE, 89(2), 183-197.
[9] Zowghi, D., Offen, R. (2002). "A Logical Framework for Modeling and Reasoning about the Evolution of Requirements." Requirements Engineering, 7(1), 1-30.
[10] Pohl, K., Böckle, G., van der Linden, F. J. (2005). "Software Product Line Engineering: Foundations, Principles and Techniques." Springer Science & Business Media.
[11] Dalpiaz, F., Giorgini, P., Mylopoulos, J. (2013). "An Architecture-Centric Approach for Producing Quality Software Requirements", IEEE Transactions on Software Engineering, 39(6), 789-806.
[12] Fernández, D. M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T. (2017). "Naming the Pain in Requirements Engineering: Contemporary Problems, Causes, and Effects in Practice", Empirical Software Engineering, 22(5), 2298-2338.
[13] Uchitel, S., Chechik, M. (2016). "Merging State Machines in Requirements Engineering", Proceedings of the 38th International Conference on Software Engineering Companion (ICSE'16), 579-581.
[14] Wieringa, R. J. (1996), "Methodologies of Requirements Engineering Research and Practice: Position Statement."
[15] Nussbaumer, M., Freudenstein, P., Gaedke, M., (2006), "Towards DSL-based Web Engineering".
[16] France, R., Rumpe, B., (2007), "Model-driven Development of Complex Software: A Research Roadmap", Proceedings - International Conference on Software Engineering.

[17] Gregor, S., (2006). “The Nature of Theory in Information Systems”, MIS Quarterly, 30(3).
[18] Walls, J., Widmeyer, G. R., El Sawy, O. A., (1992), “Building an Information System Design Theory for
Vigilant EIS”, Information Systems Research, 3(1), pp. 36-59.
[19] Gregor, S., Jones, D., (2007), “The Anatomy of a Design Theory”, Journal of the Association for Information Systems 8, no. 5, pp. 312-335.
[20] Iivari, J., Perlt Hansen, M.R., Haj-Bolouri, A., (2021), “A Proposal for Minimum Reusability Evaluation of Design Principles” European Journal of Information Systems
[21] Sein, K.M., Henfridsson, O., Purao, S., Rossi, M., Lindgren, R., (2011), “Action Design Research”, MIS Quarterly, 35(1), pp. 37-56.
[22] Iivari, J., (2020), “A Critical Look at Theories in Design Science Research”, Journal of the Association for Information Systems 21(3), 502-519
[23] Haj-Bolouri, A., Purao, S., Rossi, M., Bernhardsson, L., (2018), “Action Design Research in Practice: Lessons and Concerns”.
[24] Nuseibeh, B., Easterbrook, S., (2000), “Requirements Engineering: A Roadmap.”, Proceedings of the Conference on The Future of Software Engineering (pp. 35-46). ACM.
[25] Horkoff, J., Maiden, N., (2018), “Creativity and Goal Modeling for Software Requirements Engineering.” ACM Transactions on Software Engineering and Methodology (TOSEM), 27(3), 1-44
[26] Pacheco, C., García, A., (2012). “A Systematic Literature Review of Stakeholder Identification Methods in Requirements Elicitation.” Journal of Systems and Software, 85(7), 1481-1504.
[27] Mernik, M., Heering, J., Sloane, A. M., (2005), “When and How to Develop Domain-Specific Languages”, ACM Computing Surveys (CSUR), 37(4), 316-344.
[28] van Deursen, A., Klint, P., Visser, J., (2000), “Domain-specific languages: An annotated bibliography.”, ACM Sigplan Notices, 35(6), 26-36.
[29] Kelly, S., Tolvanen, J. P., (2008), “Domain-Specific Modeling: Enabling Full Code Generation.” Wiley-IEEE Computer Society Press.
[30] Venable, J., Pries-Heje, J., Baskerville, R., (2016) “FEDS: a Framework for Evaluation in Design Science Research”, European Journal of Information Systems (2016) 25, 77–89.
[31] Poltronieri, I., Bernardino, M., Zorzo, A. F., de Borba Campos, M., (2018), “Usability evaluation framework for domain-specific language: a focus group study”, ACM SIGAPP Applied Computing Review.
[32] Kosar, T., Bohra, S., Mernik, M., (2015), “Domain-Specific Languages: A Systematic Mapping Study”, Elsevier.
[33] Wieringa, R. J. 2023. “Methodologies of Requirements Engineering Research and Practice: Position Statement”