DSML Success Factors and Their Assessment Criteria

Abdelilah KAHLAOUI, Alain ABRAN, Éric LEFEBVRE

Abstract

Over the past few years, a number of Domain Specific Modeling Languages (DSMLs) have been developed, and their use has increased in approaches such as Model Driven Engineering (MDE), software factories and even MDA (Model Driven Architecture). However, developing a DSML is still a challenging and time-consuming task. Issues to tackle include the DSML development process, DSML quality and DSML model verification and validation (V&V). Therefore, techniques and solutions are needed to make DSML development easier and more accessible to software developers and domain experts. This paper recommends a list of success factors to consider when developing or choosing a DSML for those developing it, and for software developers and domain experts interested in using it. The paper then maps these success factors to a set of assessment criteria that can be used to assess DSML quality.

1. Introduction

Models play a central role in the Model Driven Engineering (MDE) approach, and they constitute the main artifacts to develop in the software development life cycle. While they have traditionally been used mainly for documentation purposes, models are considered in MDE as first-class entities that can (and should) be used for code generation.

This use of models as inputs to code generation increasingly demands high-quality domain-specific modeling languages capable of producing formal models that can be processed by tools (i.e. generators, interpreters, compilers, etc.) [1]. Examples of the quality characteristics required include formality, domain specificity and expressiveness, among others.

Most of the existing modeling languages lack such characteristics. To help developers have a clear idea of what makes a good DSML, and to help deciders choose the right DSML to meet their needs, a set of assessment criteria for both functional and quality attributes, as well as their related measures, should be set up.

In this paper, we provide a list of the success factors we consider important for domain-specific modeling languages and propose a technique for converting them into assessment criteria. The technique was designed to be generic, so that it can be used for domains other than DSMLs.

This paper is organized as follows. Section 2 summarizes related work on the quality of models and modeling languages. Section 3 identifies a set of success factors that should be considered when building a DSML. Section 4 describes a technique for converting success factors into assessment criteria. Finally, section 5 concludes the paper with a discussion.

2. Related work

The subject of domain-specific modeling languages has been studied from a variety of perspectives, among them DSML design, the DSML definition process, DSML building tools and DSML quality. Three of these topics constitute a good starting point for the study of DSML success factors. They are:

a. **Quality of models:** The effort in this domain has been focused on finding solutions to improve the quality of models by proposing methods and techniques which help build better-quality models. Here, a distinction is made between studies which have focused on models built using the Unified Modeling Language (UML) [2;3] and those which have extended their scope to cover conceptual models in general, regardless of the modeling language used to build them.

   Quality characteristics that have been found to be essential in the case of UML can be directly applied to DSML. However, it is to be noted that these are not enough, and do not take into account some of the specific aspects of DSMLs, namely those characteristics related to domain specificity, models transformation and code generation.

   Similarly, it has been noted that the studies that have examined conceptual models in general usually focus on specific categories of models, such as process models [4;5], requirements models, data models [6;7], etc., and also that there is a need for research to investigate the quality of models from a domain-specific perspective.

b. **Quality of modeling languages:** Authors in this field have looked at the issue of modeling language quality and assessment from a variety of perspectives. These studies cover modeling language evaluation [8-10], the development of evaluation methodologies [11] [12;13] and design principles for modeling languages [14].

   Similarly, it has been noted that aspects related to the nature of domain-specific modeling languages are missing.

c. **DSML design experiences:** In the last fifty years or so, hundreds of DMSLs have been built. The experience accumulated in developing these languages can serve as a good resource for identifying success factors. For example, lessons reported by Wile [15] can be very easily transformed into success factors.

3. DSML Success Factors

3.1. Identification of success factors

The following success factors have been identified by combining the results of work carried out in the three dimensions described in the previous section:

- ***Domain expertise***: DSML development requires an in-depth knowledge of the domain of interest, and domain knowledge facilitates the identification of domain concepts, terminology, rules and constraints. This can be achieved using domain analysis methods.
- ***Domain scoping***: Defining the appropriate scope for the domain is a critical task, as it determines the utility and usefulness of the DSML. If the scope is too broad, DSMLs will be less specific and less expressive; if it is too narrow, the return on investment might be low.
- ***Effective support tools***: DSML development is difficult. It needs to be supported by a set of tools that can automate some of the more tedious tasks in the DSML development process (i.e. analysis, verification, validation, code generation, etc.).
- ***Effective meta-model***: Developers should choose the meta-models used to define their DSML carefully. An effective meta-model will make it easier to define formal, unambiguous and expressive DSMLs. By contrast, an inappropriate meta-model may have a negative impact on a DMSL's quality.
- ***Effective underlying generator***: Since the aim of domain-specific modeling is to increase productivity by eliminating, or at least reducing, manual coding, generators capable of transforming DSML models into code are required. Without these transformation tools, DSML models will only be used for documentation purposes.
- ***High level of abstraction***: For a DSML to be effective and useful, it should define abstractions that use domain experts' vocabulary; in other words, it should raise the level of abstraction to bring the implementation world closer to the specification world. This can be done by defining languages based on domain concepts rather than on code concepts.
- ***Domain engineering environment (DEE)***: Ideally, DSML development should occur within a DEE. This is where all the core assets (i.e. reusable components, architectures, patterns, design, etc.) should be developed. DEE include, among others, domain engineers, domain experts, domain developers and DSML designers. Their primary goal is to collect, organize and model domain knowledge. The availability of rich domain knowledge is critical in identifying concepts and defining DSML elements.
- ***Language development expertise*:** Defining a domain-specific language is not an easy task. Skilled specialists in language development are required to define convenient DSMLs, and a lack of expertise may lead to some awkward and unfitted DSMLs. Any organization that decides in favor of in-house DSML development should consider assigning (or possibly hiring) the appropriate staff to accomplish the job;

- ***Viewpoint orientation*:** Viewpoints are a great way to separate and organize stakeholder concerns, and a viewpoint-oriented DSML is most likely to fit the needs of its users. Focusing on one perspective of the system at a time makes DSML models more specialized and useful;
- ***Purpose-orientation*:** A DSML is a specialized language designed to deal with a particular problem within a single domain;
- ***Domain expert support***: Domain experts are the primary users of DSMLs. Their praise for the DSML and their approval of it are critical to its adoption. Developers of these languages should make sure that they provide a DSML that fills domain experts' needs.
- ***Effective DSML definition process*:** As with any engineering activity, DSML development should be based on a set of well-defined processes, practices and tools. The process describes the activities to perform and the artifact to deliver when developing a DSML.

3.2. Categorization of success factors

In this section, a categorization scheme similar to that proposed by Wile [15] is given to help differentiate among the success factors listed above (see Table 1).

- **Organizational**: related to the organizational culture (i.e. the organization's mission, values, beliefs, norms, etc.);
- **Personal**: related to human resources' capabilities (i.e. competencies, experiences, expertise, etc.);
- **Social**: related to social behaviors and relationships;
- **Technical**: related to technical issues, such as tools and technologies.

Table 1 Success Factor Classification

Success Factor	Technical	Organizational	Social	Personal
Domain expertise				X
Domain scoping		X		
Effective supporting tools	X			
Effective meta-model	X			
Effective underlying generator	X			
Domain engineering environment		X		
High level of abstraction	X			
Language expertise				X
Viewpoint orientation	X			
Purpose-orientation		X		
Domain expert support			X	
Effective DSML definition process		X		

4. Transformation Technique

To transform DSML success factors into criteria for DSML assessment, a four-step technique has been designed (see

Figure **1**). The technique is presented below, and the way in which it can be used to extract DSML assessment criteria from the above DSML success factors is explained:

1. **Identifying success factor impact**: the existence (or non-existence) of a success factor has a direct (or indirect) impact on the quality of the DSML. The existence/absence of a success factor has a positive/negative impact and, usually, affects one or more elements of the subject (here, the subject is a DMSL).
2. **Identification of the EAI (*Elements Affected by the Impact*):** identification of the elements (i.e. some aspects related to the subject) affected by the success factor impacts.
3. **EAI attribute identification**: For each EAI, we determine the features and properties that characterize it and assist in its assessment. These features and properties are then organized into categories to facilitate the selection of those that affect the quality of the subject;
4. **Selecting attributes that have a direct effect on quality**: selection of the attributes that affect the quality of the subject (i.e. internal quality, external quality and quality in use as defined in the ISO/IEC 9126 models of the quality of software products [16]).

Table 2 illustrates an application of this technique to derive assessment criteria for DMSLs:

- From the list of DSML success factors identified in section 2, we identified a set of positive and negative impacts related to the existence or the absence of these factors;
- Then, we identified the elements affected by this impact for each success factor. For a DMSL, we have identified four elements:

    a. ***Abstract Syntax:*** defines the essential concepts and structures to be modeled in a DSML;
    b. ***Concrete Syntax:*** defines a notation (i.e. visual appearance) for the concepts defined by the abstract syntax and how these abstract concepts are realized in a concrete notation such as text or graphics;
    c. ***Semantics:*** gives a meaning to the abstract concepts and their relationships;
    d. ***Views:*** defines a perspective from which a given aspect of a software product can be described [1].

- Finally, we extracted a list of assessment criteria for each element, each time taking into account the impact at hand.

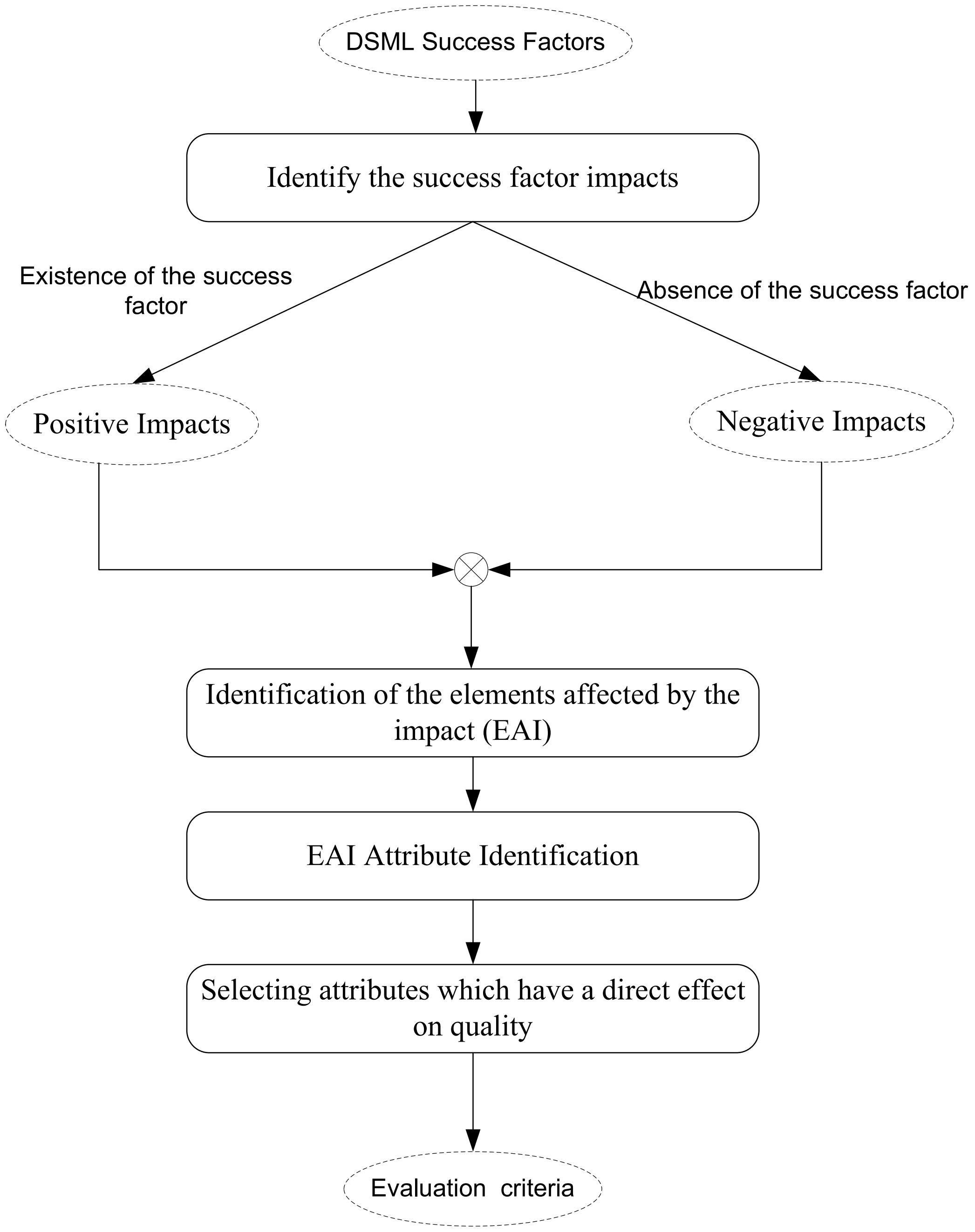
DSML Success Factors
Identify the success factor impacts
Existence of the success factor
Absence of the success factor
Positive Impacts
Negative Impacts
Identification of the elements affected by the impact (EAI)
EAI Attribute Identification
Selecting attributes which have a direct effect on quality
Evaluation criteria

Figure 1 Process of transformation of success factors into assessment criteria

Table 2 Success Factors and Assessment Criteria Mapping

Success Factors	**Positive Impact**	**Negative Impact**	**EAI**	**Assessment Criteria**
Domain expertise	**Good knowledge of domain concepts, vocabulary and terminology => Expressive DSML**	**Incomplete domain knowledge => Inexpressive DSML, lacking functionalities**	***Abstract syntax***	***Expressiveness Completeness***
Domain scoping	**Fitting business needs**	**Domain too broad or too narrow**	***Abstract syntax Concrete syntax***	***Simplicity Suitability Completeness***
Effective supporting tools	**Faster, better and cheaper DSML development**	**Costly DSML**	***All elements***	***Cost-effectiveness Productivity***
Effective meta-model	**Easy definition and upgrade of expressive, high-level abstract DSML.**	**Ambiguity Poor semantics**	***Abstract syntax Semantic***	***Formality Expressiveness Comprehensibility Scalability***
Effective underlying generator	**Better exploitation of DSML models**	**Models exclusively intended for documentation purposes**	***Abstract syntax Semantic***	***Utility Transformability Interpretability***
Domain engineering environment	**Specialized and dedicated teams for DSML definition**	**Weak domain expertise**	***All elements***	***Supportability Suitability***
High level of abstraction	**Close to the real-world separation of concerns**	**Platform-dependent DSML**	***Abstract syntax***	***Simplicity Expressiveness***
Language expertise	**Coherent models, useful and non-redundant functionality.**	**Incoherent models' useless, redundant functionalities**	***All elements***	***Uniqueness Coherence***
View point orientation	**Specialized DSMLs**	**Some stakeholders may be neglected**	***Views***	***Utility Suitability***
Purpose-orientation	**Focused DSMLs**	**Incoherent models**	***Views***	***Consistency Utility***
Domain expert support	**More support and fast adoption**	**Resistance and sabotage**	***All elements***	***Usability Utility***
Effective DSML definition process	**Well-established practices for DSML definition**	**Individual approaches' unintended results**	***All elements***	***Scalability Maintainability***

5. Discussion

While in the past few years, a number of DSMLs have been developed and their use has increased in approaches, such as Model Driven Engineering (MDE) and MDA (Model Driven Architecture), there has been little work done on the quality of such languages. The motivation for the work reported here was to identify DSML assessment criteria that should be built in by those developing these languages, and that should be looked for by those software developers and domain experts interested in using them.

On the basis of success factors documented in the literature for DSMLs, we have proposed a technique to convert them into assessment criteria.

The list of DSML success factors and their corresponding assessment criteria is aimed at helping evaluators and decision makers assess these languages. Evaluators in other areas of knowledge may also use the success factor assessment criteria conversion technique to identify criteria that help them assess their products.

We do not claim that the list of success factors and assessment criteria presented in this paper is exhaustive. More effort is needed to investigate its completeness. Case studies are also required to verify and validate the relevance of these factors and assessment criteria in industrial contexts.

Bibliography

[1] Greenfield, J., & Short, K. (2004). *Software factories: assembling applications with patterns, models, frameworks, and tools*: Wiley Pub.

[2] Lange, C. F. J. (2006). Improving the quality of UML models in practice http://doi.acm.org/10.1145/1134285.1134472 in *Proceeding of the 28th International Conference on Software Engineering* (pp. 993-996). Shanghai, China ACM Press.

[3] Unhelkar, B. (2005). *Verification and validation for quality of UML 2.0 models*. Hoboken, N.J.: John Wiley.

[4] List, B., & Korherr, B. (2006). An evaluation of conceptual business process modelling languages. http://doi.acm.org/10.1145/1141277.1141633 in *Proceedings of the 2006 ACM Symposium on Applied Computing* (pp. 1532-1539). Dijon, France ACM Press.

[5] Krogstie, J., Sindre, G., & Jørgensen, H. (2006). Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems, 15*, 91-102.

[6] Moody, D. L. (2005). Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. http://dx.doi.org/10.1016/j.datak.2004.12.005 *Data Knowl. Eng. , 55* (3), 243-276

[7] Moody, D. L., Sindre, G., Brasethvik, T., & S\&\#248;lvberg, A. (2003). Evaluating the quality of information models: empirical testing of a conceptual model quality framework, in *Proceedings of the 25th International Conference on Software Engineering* (pp. 295-305). Portland, Oregon IEEE Computer Society.

[8] Nysetvold, A., & Krogstie, J. Assessing Business Processing Modeling Languages Using a Generic Quality Framework. In *Proceedings of the CAiSE '05 Workshops, Vol. 1, Tenth International Workshop on Exploring Modeling Methods in Systems Analysis and Design, J. Castro and E. Teniente (eds.), FEUP Edições, Porto, 2005, pp. 545-556.*

[9] Krogstie, J. (2003). Evaluating UML using a generic quality framework In *UML and the unified process* (pp. 1-22): Idea Group Publishing.

[10] Krogstie, J., & Arnesen, S. Assessing Enterprise Modeling Languages using a Generic Quality Framework. *Information Modeling Methods and Methodologies*, 63-79.

[11] Morris, S., & Spanoudakis, G. (2001). *UML: an evaluation of the visual syntax of the language.* Paper presented at the System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34), ed. R.H. Sprague, Jr. (IEEE Computer Society, 2001)..

[12] Bobkowska, A. (2005). Modeling pragmatics for visual modeling language evaluation. http://doi.acm.org/10.1145/1122935.1122950 in *Proceedings of the 4th International Workshop on Task Models and Diagrams* (pp. 75-78). Gdansk, Poland, ACM Press.

[13] Bobkowska, A. (2005). A Methodology of Visual Modeling Language Evaluation. In *SOFSEM 2005: Theory and Practice of Computer Systems, LNCS 3381*: Springer.

[14] Paige, R. F., Ostroff, J. S., & Brooke, P. J. (2000). Principles for modeling language design. *Information and Software Technology, 42*(10), 665-675.

[15] Wile, D. (2004). Lessons learned from real DSL experiments. *Science of Computer Programming, 51*(3), 265-290.

[16] ISO/IEC 9126:1999. Software Engineering – Product quality. Int. Org. for Standardization, ISO 9126, 1999.