Designing a Framework for the Development of Domain-Specific Process Modelling Languages

Sven Jannaber[1(✉)], Dennis M. Riehle[2], Patrick Delfmann[3], Oliver Thomas[1], and Jörg Becker[2]

[1] Institute for Information Management and Information Systems, University of Osnabrück, Osnabrück, Germany
{sven.jannaber, oliver.thomas}@uni-osnabrueck.de
[2] European Research Center for Information Systems, University of Münster, Münster, Germany
{dennis.riehle, joerg.becker}@ercis.uni-muenster.de
[3] Institute for Information Systems Research, University of Koblenz-Landau, Koblenz, Germany
delfmann@uni-koblenz.de

Abstract. Domain-specific process modelling has gained increased attention, since traditional modelling languages struggle to meet the demands of highly specialized businesses. However, methodological support on the development of such domain-specific languages is still scarce, which hampers the specification of adequate modelling support. To this end, the paper applies a design-oriented research approach to create an integrated framework that facilitates the development of domain-specific process modeling languages. The framework is a result of 23 consolidated requirements from relevant literature and contains essential building blocks that need to be considered during the development process. It is demonstrated that the framework satisfies the identified requirements by structuring and systematizing the development of domain-specific languages, which increases language adequacy and quality.

Keywords: Business process management · Domain-specific process modelling · Framework · Modelling language development

1 Introduction

Business Process Management (BPM) has become increasingly important in today's enterprise due to the high complexity of organizational operations. It is widely acknowledged in research and practice that sophisticated BPM effort in organizations is closely tied to increased operational performance [1]. Crucial prerequisite for any successful BPM endeavor is the identification and documentation of business processes, which represent one of the main success factors for companies [2]. For this purpose, business process modelling languages (BPMLs) have emerged that provide a variety of concepts and constructs to represent these organizational processes as semi-formal process models.

A. Maedche et al. (Eds.): DESRIST 2017, LNCS 10243, pp. 39–54, 2017.
DOI: 10.1007/978-3-319-59144-5_3

However, traditional BPMLs suffer from severe shortcomings: Nowadays, common process modelling standards struggle to meet the requirements of highly diversified and specialized businesses. Studies indicate that especially niche domains grow more and more unsatisfied with generic languages, since they do not integrate domain knowledge [3, 4], Naturally, this issue translates to the end-users, who continue to rely on common visualization software (e.g. Microsoft PowerPoint) rather than on sophisticated modelling suites and traditional languages for process modelling [5]. Recently, domain-specific process modelling Languages (DSPMLs) have gained increased attention, which are designed to address the outlined shortcoming by capturing the needs of specific domains. By adapting, for instance, the level of abstraction, the terminology or the available subset of modelling elements to the needs of a specific application domain, DSPMLs significantly contribute a successful application of BPM. However, methodological support and guidance regarding the development of DSPMLs is still scarce. Although the creation domain-specific languages (DSL) are an already matured topic in computer science (e.g. [6]), insights have not yet been fully transferred to the BPM domain, which is a crucial gap considering the growing importance of DSPMLs with respect to the increasing specialization of business models and new emerging technology [7]. In BPM literature, DSPMLs have primarily been addressed from an application point of view, while neglecting conceptual design and development [8]. Hence, a structural view on DSPML building blocks and development can be seen as a first step to systematize modelling language development towards current and future demands.

To this end, the paper at hand aims at structuring and ultimately systematizing the development process of future-proof DSPMLs. For this purpose, a design-centered research methodology is applied in order to develop a DSPML framework as main IT artifact of this contribution. The framework consists of major building blocks deducted from insights of a literature review that need to be considered when designing or modifying process modelling languages towards specific business demands. Additionally, the building blocks are aligned in a way that highlights the interdependencies of the language constructs and concepts in order to shed light on the inner core of modelling languages. The DSPML framework supports language designers in research and practice by providing a comprehensive overview over a DSPMLs inner structure as a starting point for language development.

This paper is structured as follows: Succeeding the introduction, a brief outline on topic fundamentals is given in Sect. 2, followed by details on the applied research methodology in Sect. 3. The IS artifact design of this paper is presented in Sect. 4. First, the artifact's design requirements are deducted from the results of a structured literature review. Second, the DSPML framework is introduced as main outcome of this paper. Third, the artifact is being evaluated against the identified requirements. The paper concludes with a discussion and a summary of the findings in Sect. 5.

2 BPM and Domain-Specific Process Modelling

BPM is about "concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes" [9] (p. 5). One core concept of BPM is the representation of business processes in form of semi-formal

process models. While traditionally process models have primarily been used for mere process documentation and knowledge sharing, models nowadays are being processed by sophisticated algorithms for process intelligence or process mining. Due to their semantic formalization, process models can also include several technical details and thus be directly translated into executable workflows. The way a process model is structurally created and visually notated is called business BPML. Due to the different objectives and purposes that process models can be created for, there exist a large variety of BPMLs. An overview can be found in [10] or [11]. In literature, BPMLs have been heavily featured, resulting in numerous application scenarios across different domains and multiple extensions to enrich their expressiveness towards new demands and domains [12, 13]. Some of the most popular BPMLs are the Business Process Model and Notation (BPMN), for which a full specification is available by [14], and the Event-driven Process Chains (EPC). Typically, BPMLs come alongside with modeling instructions, e.g. [15], modeling frameworks or reference architectures, such as SOM, MEMO or icebricks, for which a comparison can be found in [16].

While traditional standardized languages such as BPMN are widely-used, they provide rather generic constructs for process modelling, since those languages are intended to be generally applicable, i.e., they can be used to create process models for all kind of organizations independent of their domain. This purpose differs from DSPMLs, which focus on particular application domains and, by introducing domain-specific concepts, specifically integrate domain knowledge required to capture the highly specialized business processes of that domain. Therefore, DSPMLs directly contribute to model quality, model integrity and efficiency of process modelling [6, 17]. An example for a DSPML is PICTURE, which is used in BPM projects in the public administration domain and provides 24 pre-defined process bricks. Each brick describes an standard activity in public administration [18]. Hence, typical business processes of that domain can be captured more closely than with traditional languages. According to [18], PICTURE, on the one hand, is designed to represent complex administration processes that involve several different departments, while on the other hand, through its pre-defined process bricks, it is simple enough to be used by non-experts. As the level of abstraction and the used terminology are already defined with the available process bricks, models created with PICTURE are likely to be more standardized and comparable to each other, even if many different users were involved in creating the models. In contrast, modelling languages like BPMN or EPC are highly flexible and, therefore, put higher demands towards the process modeler.

DSPMLs are subject of ongoing discussion in BPM literature. Exemplarily, [8] provides guidelines for the conception of domain-specific modeling languages, while [17] specifies generic and meta model-based requirements. In addition, a macro process for designing a domain-specific language is provided. In [19], different guidelines are proposed within the categories language purpose, language realization, language content, concrete syntax and abstract syntax. A framework for deriving DSPML from a generic BPML is provided by [6], however a software development point of view is taken by providing model-to-model transformation rules. To the best of our knowledge, a framework that integrates existing knowledge in terms of DSPML development by highlighting required building blocks has not yet been proposed.

3 Research Design

For the conceptualization and design of a DSPML framework as primary outcome of this contribution, the paper at hand adheres closely to the design science (DS) research paradigm that has been predominantly introduced in the IS domain as a research framework by [20] and which is nowadays popularized and applied in the field of IS [21, 22]. Originated from the lack of legitimate Information Systems (IS) research methodologies that serve the need of IS as being an "'applied' research discipline" [22], the design science research approach addresses this issue by adapting design-oriented approaches taken from related disciplines such as natural sciences or engineering to the field of IS and thus closing the gap between IS research and practice [22, 23]. In his three cycle view on DS, [24] characterizes DS research as an interplay of relevance, design and rigor. The design cycle is the main cycle of DS research and iterates between the building artifacts and evaluating them against certain requirements [24]. These requirements are considered in the relevance cycle, which connects the artifact to the desired environment and thus determines the organizational problem to be addressed, the intended application domain and also defines the criteria for evaluation of the research result [24]. The rigor cycle relates the research activities with the knowledge base. This ensures that the artifact design is grounded established foundations and previous work, while also add new insights to the knowledge base [24]. Main outcome of design-centered research is an IS artifact, which may be a prototype, but also models, methods or instantiations [20].

In literature, multiple DS research conceptualizations have been proposed, for example the Design Science Research Cycle [25], which differentiates six phases of DS research in an iterative DS procedure model, or Wieringa's [26] DS framework, which is based on work by [20] and further specifies the relationship between environment, knowledge base and actual IS design. However, common DS approaches ultimately incorporate a standard build-evaluate pattern, which prescribes an ex post artifact evaluation, leading to delayed insight about the artifact's truth resp. validity. In their work, [27] propose a design science research approach that is applied as the research design of this paper. In particular, the proposed approach addresses the stated issues of the traditional build-evaluate pattern an instead introduces a more agile way of DS evaluation. Figure 1 visualizes the design science research cycle according to [27] and highlights the phases covered in this contribution. Essentially, the DSR cycle is divided into two main phases, namely ex ante evaluation and ex post evaluation. Besides the baseline DS research activities *problem identification*, *design*, *construct* and *use*, the novelty of this approach is a separate evaluation step after each activity. Subsequently, this allows for a meaningful evaluation even in early design stages of the artifact [27]. In addition, the proposed DS research procedure model comes with specific guidance on methods and criteria for each evaluation step, which we will adhere to in the following.

Adapting the framework in Fig. 1, we focus on the ex-ante phase in this paper. In doing so, we provide a problem statement, which is evaluated by conducting an extensive literature review, thus fulfilling evaluation step 1. As a result of both problem statement and design objectives deducted from the review, the DSPML framework is developed. Concluding the design step, the artifact is evaluated against the proposed

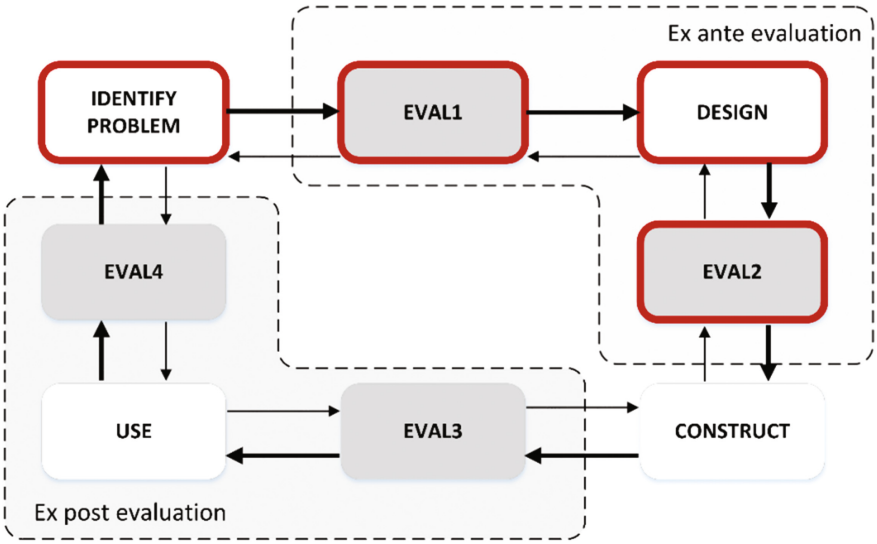

Fig. 1. Design science research cycle according to Sonnenberg and vom Brocke [27]

design objectives and requirements obtained through the literature review. Hereby, we focus on the criteria internal consistency, completeness and clarity, as mentioned by [27]. In accordance with [24], the contribution at hand makes heavy use of existing work included the IS knowledge base, since a consolidation of previous research effort is extracted via a systematic literature review. The review results do not only form the basis of the artifact's requirements, but are also integrated as building blocks into the final result. The presented work contributes a novel IS artifact to the knowledge base by providing a structured framework of essential DSPML building blocks, which may serve as a blueprint for future language design.

4 A Framework for Domain-Specific Modelling Languages

4.1 Literature Review and Design Requirements

To gather design requirements for the DSPML framework and thus query the IS knowledge base, a systematic literature review according to [28] has been conducted using the databases SpringerLink, Google Scholar, ScienceDirect and EbscoHost. The search terms "*model* *develop*", "*process* *model* *develop*", "*process* *language*" and "*model* *language*" have been used, as well as their German equivalents and brief abbreviations fit each search engine's modus operandi.

The choice of search terms has been generic on purpose, since literature specifically addressing crucial building blocks of process modelling language development is wide-spread across the BPM and conceptual modelling domain. Furthermore, the contribution at hand aims providing an overview over previous work existing in the IS knowledge base with the ultimate objective to obtain a holistic integration of the

widely-spread literature on modelling language development. For this reason, the literature review has not been restricted to a specific time interval or IS journals and conferences.

After scanning titles and abstracts of the initial review results (564), an amount of 253 publications remained, of which 97 publications were considered to be relevant. Each publication has been assessed regarding insights on (domain specific) modelling language requirements and integral core components of conceptual modelling languages. After consolidating and clustering the findings, 23 meta requirements that address structure and development of DSPMLs have been identified, which serve as design requirements for the development of the framework in Sect. 4.2. The requirements are depicted in Table 1. Requirements 1–14 represent requirements that directly

Table 1. Meta requirements for DSPML framework design

	No.	Framework design requirement	Reference (ex.)
Requirements analysis	1	The DSPML has a defined scope and purpose	[8, 17, 29]
	2	The language is based on requirement analysis	[8, 17, 29]
	3	Stakeholder groups are considered during development	[17, 29, 30]
	4	Language building blocks integrate domain relevance	[8, 17, 19, 31]
Language specification	5	The modelling language is specified by a language meta model	[32, 33]
	6	The DSPML adheres to concrete syntax	[19, 32, 33]
	7	The DSPML adheres to abstract syntax	[19, 32–35]
	8	The language provides (formal) language semantics	[32, 33]
	9	The language considers modelling pragmatics	[17, 36]
Development process	10	The DSPML is a result of a systematic development approach	[19, 29, 37]
Concepts, constructs and elements	11	The language development is based on existing concepts	[8, 17, 19, 29]
	12	The DSPML language contains (domain-specific) modelling constructs to represent business processes:	[8, 17, 19, 31, 35, 38]
		Process elements	[34, 35, 39–41]
		Control flow pattern	[42–46]
		Resources	[34, 35, 40]
		Modularization	[35]
	13	The DSPML provides a graphical notation	[17, 47]
Evaluation	14	The language is assessable regarding its quality and correctness	[17, 48]
Language quality	15	*Uniqueness*	[37]
	16	*Consistency*	[37]
	17	*Scalability*	[37]
	18	*Supportability*	[17, 37]
	19	*Simplicity*	[37, 49–51]
	20	*Space economy*	[37]
	21	*Reversibility*	[37]
	22	*Reliability*	[37]
	23	*Seamlessness*	[37]

refer to modelling language core components that have been stated in the relevant literature and thus need to be considered for DSPML development. Requirements 15–23 concern the resulting language itself. However, since the framework aims at supporting the development of such languages, we argue that these result-centric requirements need to be reflected in the DSPML framework as well.

4.2 DSPML Framework Design

On the basis of the identified requirements, a framework for the development of domain specific process modelling languages is featured as the main IT artifact of this paper. Figure 2 shows the fully developed DSPML framework, which sheds light on crucial building blocks to a (domain-specific) process modelling language. Each of the building blocks is deducted from the framework design requirements presented in Table 1 and substantiated by corresponding literature. The DSPML framework consists of three succeeding main phases, which are aligned as iterating layers. The *requirements layer* lays ground for the subsequent language specification and evaluation. Initially, the development of domain specific languages requires the definition of scope and purpose of the language to be designed [17]. Essentially, this building block represents an initial planning phase, which on the one side details the indented value and long-term usage of the language, since any language component needs to be tailored towards a determined purpose [19]. On the other side, this building block also provides for first analyses regarding the feasibility and applicability. Succeeding scope and purpose, the identification of requirements is the core task in the requirements layer. The framework differentiates between two types of requirements: Generic requirements of any domain specific modelling language are closely tied to language pragmatics and encompass for instance abstraction level [17]. In the context of process modelling, such generic requirements also reflect necessity of a well-defined language specification. Hence, the specification of language syntax and semantics are treated as generic requirements that transits from the requirements layer into the language specification layer. Specific requirements however are set to shape the language towards the intended application domain. Exemplarily, studies demonstrate that the financial industry [4] has fundamental different demands regarding language concept and constructs to reflect their business domain than businesses in the chemical domain [3]. Accordingly, any modelling language needs to reflect the concepts and constructs relevant to their particular domain [8, 17, 19, 31]. In this paper, emphasis is also put on the technology involved in the intended modelling effort. Primarily, this building block refers to technical devices on which the language is applied. While literature on this matter is still scarce, it is undisputed that both mobile [52] and wearable [53] devices impose different requirements (e.g. limited screen size and interaction space) on process modelling languages than traditional modelling suites running on desktop computers. Each process domain is connected to a certain set of stakeholders involved. Gaining insight into stakeholder groups is considered a crucial task for language development [17, 29]. Potential stakeholders encompass persons involved in the language development process, hence domain experts or language designer [17, 29, 30]. Additionally, the target audience has to be kept in mind, since in practice process

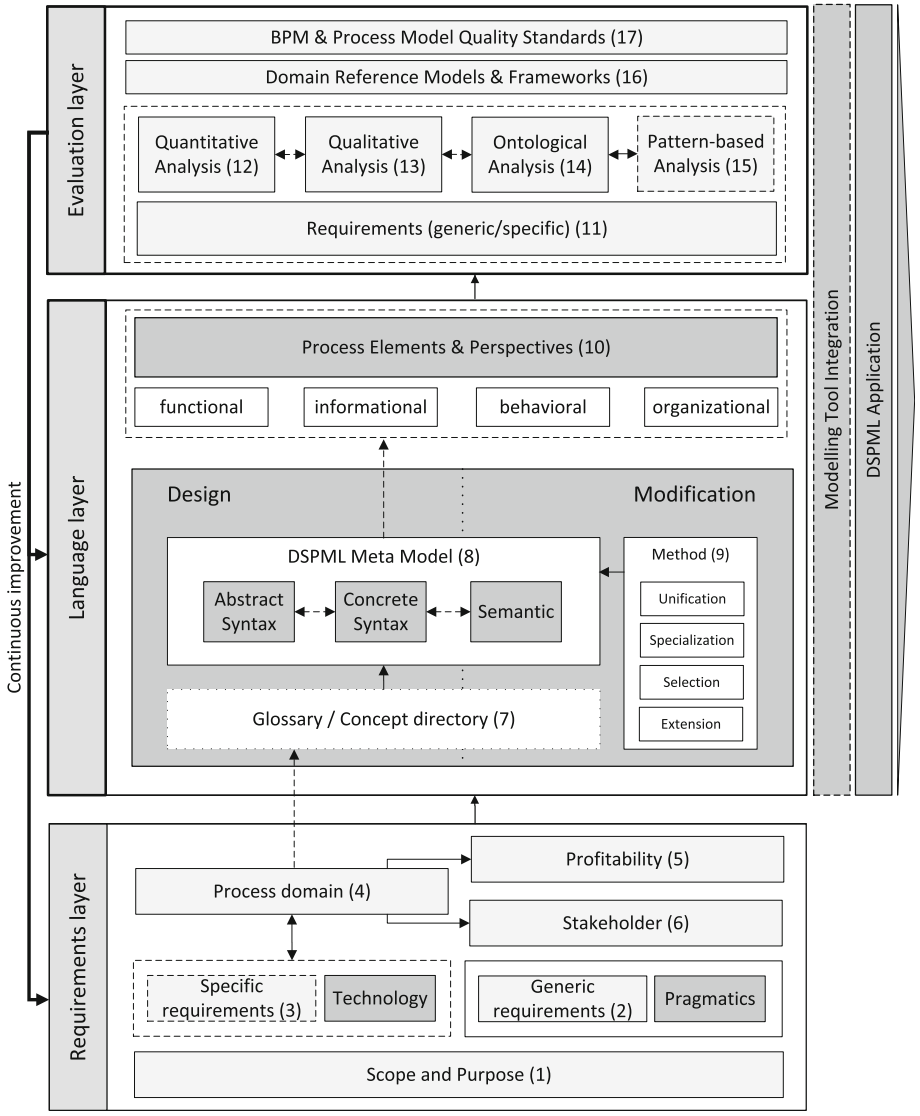

Fig. 2. The DSPML development framework

modelling is often conducted by employees who only possess limited modelling knowledge [54]. Influenced by both the intended application purpose and process domain, the profitability of the development process is embedded on the requirements layer. [29] states that every language development is associated with monetary investment, which is also reflected in the DSML development process of [17]. [8] also attached a profitability criterion at the evaluation layer of a modeling language.

Succeeding the requirements layer, the DSPML has to be designed and specified according to the elaborated requirements. The *language layer* consists of the two primary components language specification and design of process elements and perspectives. Core of the language specification is the meta-model of DSPML to be developed. In this case, the meta-model based specification of a modelling language subsumes the determination of the language's abstract and concrete syntax as well as its semantics [32]. The abstract syntax determines and details concepts, constructs and elements that are being used within the language [19]. The concepts, constructs and elements defined within the language specification are detailed using a glossary and concept directory, which is directly influenced by the process domain. Primary purpose of the directory is to ensure that the intended domain is correctly captured by applied terms and constructs [17].

For abstract syntax specification, language designers can draw from insight gained in [8, 17, 19, 31, 35, 38] to determine constructs essential to a modelling language, while maintaining a domain-driven point of view. Ultimately, decisions regarding potential language modularization, e.g. via sub processes, have to be made at this point in the development process [19], as well as a determination of relevant control structures to be provided by the language [43]. The concrete syntax specifies the grammar of the determined constructs, hence their interconnection and relationships. The concrete syntax can be represented by both textual syntax rules and the language meta-model. Lastly, (formal) semantics are essential to modelling languages in order to provide additional meaning to the language's elements and to prevent ambiguity in their definition. Most importantly, formal semantics are required for model automatization and execution, thus preventing deadlocks or livelocks in the resulting process model [55]. In accordance to [37], the language specification of this framework is divided into two development approaches: The first approach, Design, refers to a design from scratch approach in which a new DSPML is created bottom-up. As a second approach, Modification subsumes applicable methods to tackle language development on the basis of already existing languages. Unification refers to the integration of languages in order to benefit from their combined advantages. Specialization is achieved by restricting certain aspects of the language in order to specialize a given language towards a given purpose or domain [29]. Extension reflects the enhancement of a modelling language in order to obtain larger expressional capabilities (e.g. BPMN extensions provided in [13]). Lastly, Selection requires a partial usage of only a small subset of language constructs [29]. All aforementioned methods represent different design entry points which may also skip the requirements layer. However, all methods provide modifications on the language meta-model, hence the framework underlines their interconnection. Subsequent to the language specification, the definition of process elements and their graphical notation represents the last building block of the language layer. According to [38], relevant elements are associated with the four different perspectives functional, behavioral, informational and organizational when considering process modelling. Within these perspectives, concrete modelling elements need to be deducted from the abstract syntax [31]. Finally, each identified element needs to be assigned to a distinct graphical notation in order to enable the creation of semi-formal process models [17].

Following the language and element specification, the *Evaluation layer* aims at assessing the language against predefined criteria with the overall purpose to identify potential need for refinement and optimization. The initial building block of the evaluation layer refers to the requirements criterion. Here, the specified language needs to be checked against the generic and specific requirements that have been elaborated in the requirements layer. For DSPML analysis, we adhere to approaches proposed in [48]: On the one hand, qualitative and quantitative analyses are applied in order to gather intelligence about the language's applicability and feasibility as well as its profitability, for example via semi-structured interview or surveys [48]. For ontological analysis, the Bunge-Wand-Weber ontology, e.g. [56], can be used to investigate ontological deficits within a specified language grammar [48]. In [57], the capability of process pattern for language evaluation purposes is discussed. Henceforth, a pattern-based analysis, for example using the workflow patterns proposed in [58], is included. Reference and domain models are integrated into a separate building block in order to ensure adherence to domain-specific terms and constructs as well as best practices [59]. Lastly, although not particularly designed towards evaluating process modelling languages, common BPM standards, such as process model quality frameworks and modelling guidelines, can be applied. Here, the core task is to determine whether a process model resulting from the developed DSPML framework is able to sufficiently fulfill these quality standards. Potential quality deficit in resulting models can thus be treated as hint for wrongly specified language building blocks. Exemplarily frameworks are, for example, the SEQUAL framework [60]. Modelling guidelines to take into consideration encompass for example the7PMG [61]. The results of the evaluation layer pass into the continuous improvement cycle, since all layers are interconnected iteratively. Depending on the detected faults and gaps, improvement can on the requirement as well as language layer.

As carried out in Sect. 2, the proposed DSPML framework is solely limited to the modelling language, hence it does not provide for implementation, application or algorithms. However, modelling tool integration and application are included as potential interfaces for further framework enhancement, directly attached to the deliverables of the language and evaluation layer, as their outputs are likely to be applied, whether the requirements only concern the language specification.

4.3 Evaluation and Discussion

For framework evaluation, we specifically address the criteria internal consistency, clarity and completeness as proposed in [27]. We argue that internal consistency is provided through the DSPML framework being deeply anchored in literature, as a structured literature review served as the foundation for the framework building blocks. Furthermore, additional consistency is given by the framework's alignment to related work in the field of language development, especially to the work of [17], whose development workflow is being closely reflected within the framework. In terms of clarity, we argue that the clear structure and visualization of the framework, thus the ordering of language building blocks and their interconnection, facilitates comprehensibility and understandability of the language development process. Therefore,

clarity and transparency is provided. For completeness, we again refer to the extensive literature review that has been conducted, which, to the best of our knowledge, provides an overview over the main components that need to be considered when developing process modelling languages. To further strengthen the argument for completeness, the alignment of the identified requirements from literature with the buildings blocks of our DSPML framework follows.

Meta requirement 1 (Req. 1) is fulfilled by integrating building block Scope and Purpose (1) into the framework as preliminary planning phase. REquation 2 has been divided into multiple building blocks. Whereas building blocks (2) and (3) cover core requirements engineering, (4), (5) and (6) are tailored towards particular, domain-driven requirements. In terms of stakeholder (Req. 3), we decided to integrate a designated building block (6) to embrace the importance of stakeholders to the language development side as well as on the application side. Req. 4 is incorporated in multiple building blocks. First, domain specific relevance is ensured by conducting specific requirements (3) and taking characteristics of the process domain (4) into consideration. These characteristics are directly translated into a corresponding concept directory (7), which is the basis for meta-model development (Req. 5) and process element specification. Req. 6–8 are covered by the inner core of the DSPML meta-model, abstract syntax, concrete syntax and semantics. Req. 9 is considered in building block (2) in an early stage of language development, since pragmatics is closely tied to generic and stakeholder-related requirements. Req. 10 is primarily covered by building block (9), since the language designer can choose between different development approaches. However, the differentiation between Design and Modification also corresponds to this requirement. Regarding Req. 11, specifically building blocks (7) and (9) ensure that each developed languages is to a certain degree built on existing concepts. For once, these concepts can be referred to when determining the concept directory of the intended domain. Additionally, each method in (9) provides that at least one existing language is used as a basis for development. Req. 12 with its corresponding sub-requirements is particularly addressed within the concept directory (7), the meta-model specification (8) and determination of process elements (10). Designing the framework, we refrained from integrating element-specific building blocks (*"Process elements"*, *"Control flow Pattern"*) into the framework to maintain a coherent abstraction level. We argue that these requirements are covered nevertheless, because the abstract syntax specifically address constructs, concepts and elements relevant to the intended application purpose, which is domain-specific process modelling. For Req. 13, building block (10) defines both determinations of process elements as well as their visualization. Regarding Req. 14, the evaluation layer of the framework enables various forms of language assessment, for example via ontological analyses or alignment with domain reference models and process quality standards. While Req. 15–20, refer to actual modelling languages, we argue that the DSPML framework lays the foundation to develop languages that adhere to these requirements: Regarding Req. 15, domain-specific languages are unique per nature, since they are limited to concepts relevant to their particular domain. Additionally, sophisticated requirements analysis, existing work in the concept directory and syntactical and semantical formalizations ensure that there is no language overload and ambiguity. Consistency refers to "a purpose of the design of the language" in a way that this

purpose translates through the whole development process [37]. The framework provides a consistent blueprint to language development, since its bottom-up approach ensures the permeation of scope and purpose throughout the development process. Req. 17 is covered by the frameworks abstraction level and domain focus. In terms of Req. 18, requirement analyses with a specific focus on pragmatics as well as the formalization of the language when specifying syntax and semantic ensures that the resulting language is both usable for humans and tools. Similar to Req. 15, Req. 19 is covered by strict adherence to requirements, pragmatics and existing constructs to be used. In addition, the domain-orientation ensures the usage constructs limited to the particular domain. While space economy (Req. 20) is hard to assess, we argue that the consideration of general requirements and pragmatics at an early development stage steer language designers to consider this requirement during their process. The framework is conceptualized iteratively, so that evaluation results and new insights can be integrated into language refinement, thus covering Req. 21. Req. 22 is fulfilled in two ways: First, the modular structure of the framework facilitates its implementation, for instance in a language meta-modelling tool. Second, the modeling language as a result of the framework can be implemented into common process modelling tools, since its meta-model is formalized and processable. For Req. 23, the argumentation of Req. 16 holds. Essentially, the strict usage of concept included in the concept directory as well as the suggested usage of already existing constructs for abstract syntax and element specification ensures a congruent usage of abstractions throughout the development process.

5 Discussion and Conclusion

The DSPML framework presented in this paper consolidates and integrates existing work in the field, and can be applied to systematize and structuring the development of modelling languages tailored towards specific domains or technology, which is an emerging issue in BPM. Hereby, the framework supports language designers in both research and practice, who can draw upon the identified building blocks when developing novel, domain and technology-specific modelling languages. However, limitations have to be considered: First, the framework represents a rather high-level overview over crucial components required for language development. For actual framework application, each building block needs to be detailed with respect to methods or tools (e.g. meta-modelling platform) and substantiated regarding its content. Second, at this point the framework is limited to the mere modelling language without addressing application, modelling software integration or algorithms. However, the stated areas pose additional challenges and requirements to the development of modelling languages that are not yet reflected in the model. Furthermore, only a descriptive evaluation is proposed. Applying the framework in practice may reveal different requirements that have not yet been considered. Subsequently, the artifact and its evaluation as well as the outlined limitations open up new possibilities for further research need: On the one hand, further work need to specify and substantiate each building block in detail regarding processed input and output as well as applied methodologies or tool support. On the other hand, the application of the framework in

research and practice to develop domain-specific languages will reveal valuable insights to be incorporated into a subsequent DS iteration. An elaborate ex-post evaluation against DSPML quality or usability criteria as well as the degree of domain coverage may reveal the framework's applicability from both the language designer and resulting modelling language perspective. Furthermore, future work may enhance the framework by taking the structure and specific characteristics of existing modeling languages into account. Lastly, tool support that implements all layers of the framework with appropriate features and software components proves to be a fruitful expansion of the research presented.

Adhering to the applied DS methodology, this paper motivates the topic and provides a problem statement. Following a brief introduction of fundamentals, the results of an extensive literature review are condensed into 23 design requirements that lay ground for DSPML framework design. The evaluation demonstrates that the framework sufficiently addresses the needs expressed in relevant literature. Using the proposed artifact, the engineering of domain-specific process modelling languages can be methodologically grounded, which structures and systematizes the development process. Ultimately, this leads to an increased adequacy and quality of resulting languages, which need to be designed towards increasingly complex requirements driven by domain, technology and end-user.

References

1. Melenovsky, M.J.: Business process management's success hinges on business-led initiatives. Gart. Res. 1–6 (2005). https://www.gartner.com/doc/483847/business-process-managements-success-hinges
2. Becker, J., Mathas, C., Winkelmann, A.: Geschäftsprozessmanagement. Springer, Heidelberg (2009)
3. Eggersmann, M., Krobb, C., Marquardt, W.: A modeling language for design processes in chemical engineering. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) ER 2000. LNCS, vol. 1920, pp. 369–382. Springer, Heidelberg (2000). doi:10.1007/3-540-45393-8_27
4. Becker, J., Breuker, D., Weiß, B., Winkelmann, A.: Exploring the status quo of business process modelling languages in the banking sector – an empirical insight into the usage of methods in banks. In: ACIS 2010 Proceedings, Paper 8 (2010)
5. Harmon, P., Wolf, C.: The State of Business Process Management (2016)
6. Heitkötter, H.: A framework for creating domain-specific process modeling languages. In: 7th International Conference on Software Paradigm Trends (ICSOFT), Rome, Italy, pp. 127–136 (2012)
7. Houy, C., Fettke, P., Loos, P., Aalst, W.M.P., Krogstie, J.: Business process management in the large. Bus. Inf. Syst. Eng. **3**, 385–388 (2011)
8. Frank, U.: Some guidelines for the conception of domain-specific modelling languages. In: Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011, Hamburg, Germany, 22–23 September 2011, pp. 93–106 (2011)
9. Weske, M.: Business Process Management. Springer, Heidelberg (2012)
10. List, B., Korherr, B.: An evaluation of conceptual business process modelling languages. In: 2006 ACM Symposium on Applied Computing, pp. 1532–1539 (2006)

11. Lu, R., Sadiq, S.: A survey of comparative business process modeling approaches. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 82–94. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72035-5_7
12. Riehle, D.M., Jannaber, S., Karhof, A., Thomas, O., Delfmann, P., Becker, J.: On the de-facto standard of event-driven process chains: how EPC is defined in literature. In: Modellierung 2016, Karlsruhe, 2–4 März 2016, pp. 61–76. Köllen Druck+Verlag, Bonn (2016)
13. Braun, R., Esswein, W.: Classification of domain-specific BPMN extensions. In: Frank, U., Loucopoulos, P., Pastor, Ó., Petrounias, I. (eds.) PoEM 2014. LNBIP, vol. 197, pp. 42–57. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45501-2_4
14. Object Management Group: Business Process Model and Notation (BPMN) Version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0
15. Thomas, O.: Fuzzy Process Engineering. Gabler Verlag | GWV Fachverlage GmbH, Wiesbaden (2009)
16. Becker, J., Riehle, D.M., Clever, N.: Ansätze zur Unternehmensmodellierung – Eine Einordnung. In: Benker, T., Jürck, C., Wolf, M. (eds.) Geschäftsprozessorientierte Systementwicklung — Von der Unternehmensarchitektur zum IT-System, pp. 415–425. Springer, Wiesbaden (2016). doi:10.1007/978-3-658-14826-3_25
17. Frank, U.: Domain-specific modeling languages: requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering: Product Lines, Languages, and Conceptual Models, pp. 133–157. Springer, Heidelberg (2013)
18. Becker, J., Algermissen, L., Falk, T.: Prozessorientierte Verwaltungsmodernisierung: Prozessmanagement im Zeitalter von E-Government und New Public Management. Springer, Dordrecht (2009)
19. Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., Völkel, S.: Design guidelines for domain specific languages. In: Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modelling (2009)
20. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information research. MIS Q. **28**, 75–105 (2004)
21. March, S.T., Storey, V.C.: Design science in the information systems discipline: an introduction to the special issue on design science research. MIS Q. **32**, 725–730 (2008)
22. Peffers, K., Tuunanen, T., Gengler, C.E., Rossi, M., Hui, W., Virtanen, V., Bragge, J.: The design science research process: a model for producing and presenting information systems research. In: Proceedings of the First International Conference on Design Science Research in Information Systems and Technology, DESRIST 2006, vol. 24, pp. 83–106 (2006)
23. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decis. Support Syst. **15**, 251–266 (1995)
24. Hevner, A.R.: A three cycle view of design science research. Scand. J. Inf. Syst. **19**, 87–92 (2007)
25. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**, 45–77 (2008)
26. Wieringa, R.: DS as nested problem solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST 2009, Philadelphia, Pennsylvania (2009)
27. Sonnenberg, C., vom Brocke, J.: Reconsidering the Build-Evaluate Pattern in Design Science Research. In: Proceedings of 7th Design Science Research in Information Systems and Technology, pp. 381–397 (2012)

28. vom Brocke, J.M., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., Cleven, A.: Reconstructing the giant: on the importance of rigour in documenting the literature search process. In: 17th European Conference on Information Systems, Verona, Italy, pp. 1–13 (2013)
29. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37**, 316–344 (2005)
30. Cho, H., Gray, J., Sun, Y., White, J.: Key challenges for modeling language creation by demonstration. In: ICSE 2011 Workshop on Flexible Modeling Tools, pp. 1–4 (2011)
31. Lin, F.-R., Yang, M.-C., Yu-Hua, P.: A generic structure for business process modeling. Bus. Process Manag. J. **8**, 19–41 (2002)
32. Clark, T., Sammut, P., Willans, J.: Applied Metamodelling. A Foundaton for Language Driven Development (2008)
33. Klör, B., Bräuer, S., Beverungen, D., Monhof, M.: A domain-specific modeling language for electric vehicle batteries. In: Wirtschaftsinformatik Proceedings 2015 (2015)
34. Casanova-Brito, V., Patig, S.: Requirements of process modeling languages – results from an empirical investigation. In: Wirtschaftsinformatik Proceedings 2011, pp. 756–765 (2011)
35. Zamli, K.Z., Ashidi, N., Isa, M.: A survey and analysis of process modeling languages. Malays. J. Comput. Sci. **17**, 68–89 (2004)
36. Seel, C.: Reverse Method Engineering: Methode und Softwareunterstützung zur Konstruktion und Adaption semiformaler Informationsmodellierungstechniken. Logos Verlag, Berlin (2010)
37. Paige, R.F., Ostroff, J.S., Brooke, P.J.: Principles for modeling language design. Inf. Softw. Technol. **42**, 665–675 (2000)
38. Curtis, B., Kellner, M.I., Over, J.: Process modeling. Commun. ACM **35**, 75–90 (1992)
39. de Cesare, S., Serrano, A.: Collaborative modeling using UML and business process simulation. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS 2006), pp. 1–10 (2006)
40. Derniame, J.-C., Kaba, B.A., Wastell, D.: The software process: modelling and technology. In: Derniame, J.-C., Kaba, B.A., Wastell, D. (eds.) Software Process: Principles, Methodology, and Technology. LNCS, vol. 1500, pp. 1–13. Springer, Heidelberg (1999). doi:10.1007/3-540-49205-4_1
41. Chou, S.-C.: A process modeling language consisting of high level UML diagrams and low level process language. J. Object Technol. **1**, 137–163 (2002)
42. Figl, K., Mendling, J., Strembeck, M., Recker, J.: On the cognitive effectiveness of routing symbols in process modeling languages. In: Abramowicz, W., Tolksdorf, R. (eds.) BIS 2010. LNBIP, vol. 47, pp. 230–241. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12814-1_20
43. Pichler, H., Eder, J.: Business process modeling and workflow design. In: Embley, D.W., Thalheim, B. (eds.) Handbook of Conceptual Modeling, pp. 259–286. Springer, Heidelberg (2011)
44. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: On the suitability of BPMN for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A. P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006). doi:10.1007/11841760_12
45. Schmidt, G., Braun, O.: Process language GPN. In: Bernus, P., Mertins, K., Schmidt, G. (eds.) Handbook on Architectures of Information Systems, pp. 197–214. Springer, Heidelberg (2006)

46. van Hee, K.M., Sidorova, N., van der Werf, J.M.: Business process modeling using petri nets. In: Jensen, K., Aalst, W.M.P., Balbo, G., Koutny, M., Wolf, K. (eds.) Transactions on Petri Nets and Other Models of Concurrency VII. LNCS, vol. 7480, pp. 116–161. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38143-0_4
47. Schalles, C., Creagh, J., Rebstock, M.: A causal model for analyzing the impact of graphical modeling languages on usability. Int. J. Softw. Eng. Knowl. Eng. **24**, 1337–1355 (2014)
48. Recker, J.: Evaluations of Process Modeling Grammars: Ontological, Qualitative and Quantitative Analyses Using the Example of BPMN. Springer, Heidelberg (2011)
49. Conradi, R., Liu, C.: Process modelling languages: one or many? In: Schäfer, W. (ed.) EWSPT 1995. LNCS, vol. 913, pp. 98–118. Springer, Heidelberg (1995). doi:10.1007/3-540-59205-9_47
50. Atkinson, D.C., Weeks, D.C., Noll, J.: The design of evolutionary process modeling languages. In: 11th Asia-Pacific Software Engineering Conference, pp. 73–82 (2004)
51. Luo, W., Tung, Y.A.: A framework for selecting business process modeling methods. Ind. Manag. Data Syst. **99**, 312–319 (1999)
52. Kolb, J., Rudner, B., Reichert, M.: Towards gesture-based process modeling on multi-touch devices. In: Bajec, M., Eder, J. (eds.) CAiSE 2012. LNBIP, vol. 112, pp. 280–293. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31069-0_24
53. Metzger, D., Niemöller, C., Berkemeier, L., Brenning, L., Thomas, O.: Vom Techniker zum Modellierer - Konzeption und Entwicklung eines Smart Glasses Systems zur Laufzeitmodellierung von Dienstleistungsprozessen. In: Thomas, O., Nüttgens, M., Fellmann, M. (eds.) Smart Service Engineering, pp. 193–213. Springer, Heidelberg (2017)
54. Recker, J.: Opportunities and constraints: the current struggle with BPMN. Bus. Process Manag. J. **16**, 181–201 (2010)
55. Fellmann, M., Bittmann, S., Karhof, A., Stolze, C., Thomas, O.: Do we need a standard for EPC modelling? The state of syntactic, semantic and pragmatic quality. Lecture Notes Informatics (LNI), vol. P-222, pp. 103–117. Gesellschaft fur Inform (2013)
56. Wand, Y., Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. Inf. Syst. J. **3**, 217–237 (1993)
57. Recker, J., Rosemann, M., Krogstie, J.: Ontology- versus pattern-based evaluation of process modeling languages: a comparison. Commun. AIS. **20**, 774–799 (2007)
58. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**, 5–51 (2003)
59. La Rosa, M., Gottschalk, F., Dumas, M., Van Der Aalst, W.M.P.: Linking domain models and process models for reference model configuration. In: Hofstede, A., Benatallah, B., Paik, H.-Y. (eds.) BPM 2007. LNCS, vol. 4928, pp. 417–430. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78238-4_43
60. Krogstie, J., Sindre, G., Jørgensen, H.: Process models representing knowledge for action: a revised quality framework. Eur. J. Inf. Syst. **15**, 91–102 (2006)
61. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Inf. Softw. Technol. **52**(2), 127–136 (2010)