Characterizing and evaluating the quality of software process modeling language: *Comparison of ten representative model-based languages*

J.A. García-García[*], J.G. Enríquez, F.J. Domínguez-Mayo

Computer Languages and Systems Department, University of Seville, Escuela Técnica Superior de Ingeniería Informática, Web Engineering and Early Testing (IWT2) Group, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Keywords:

Quality Evaluation Framework (QuEF)

Quality Model

Software Process Modeling Language comparative study

A B S T R A C T

Software organizations are very conscious that deployments of well-defined software processes improve software product development and its quality. Over last decade, many Software Process Modeling Languages (SPMLs) have been proposed to describe and manage software processes. However, each one presents advantages and disadvantages. The main challenge for an organization is to choose the best and most suitable SPML to meet its requirements. This paper proposes a Quality Model (QM) which has been defined conforms to QuEF (Quality Evaluation Framework). This QM allows to compare model-based SPMLs and it could be used by organizations to choose the most useful model-based SPML for their particular requirements. This paper also instances our QM to evaluate and compare 10 representative SPMLs of the various alternative approaches (metamodel-level approaches; SPML based on UML and approaches based on standards). Finally, this paper concludes there are many model-based proposals for SPM, but it is very difficult to establish with could be the commitment to follow. Some non-considered aspects until now have been identified (e.g., validation within enterprise environments, friendly support tools, mechanisms to carry out continuous improvement, mechanisms to establish business rules and elements for software process orchestrating).

1. Introduction

Today, software applications affect indirectly or directly to any kind of company or organization because computer systems support day to day business activities in diverse scenarios including air traffic control, automotive mechanical, healthcare processes, industrial or manufacturing processes, processes associated with nuclear reactors, and service processes, among others. The process management of these scenarios is crucial because it affects many millions of people and many financial resources. Also, competitiveness and productivity are key mechanisms to improve the management of any organization. For this purpose, there are mature management models by which organizations could implement and maintain their processes achieving an edge over their competitors using different technologies such as cloud solutions [1], among others.

In this context, BPM (Business Processes Management) is the term most used and known around the world to identify management strategies on business processes. BPM defines methods, techniques and tools to support the process lifecycle and improve the business management [2]. Reducing costs and improving the management are goals of BPM [3,4]. For this purpose, BPM proposes to perform a continuous improvement lifecycle what may increase and improve return on investments through reducing production costs [5]. In fact, many institutions that promote, by means of their standards and guidelines, the application of BPM as a process-oriented mechanism to improve productivity, competitiveness, quality and efficiency at organizations [6]. Many companies follow these advices in all areas of business.

This situation is not different in software organizations. This kind of organizations knows the importance of deploying well-defined processes to improve the software development process throughout its lifecycle [7]. In this context, one of the most important challenge is to decide how software processes should be modeled and managed. Software process could be defined as an ordered sequence of steps, objects, human resources, automatic resources (i.e., computer systems) and constraints, as well as information structures. Its goal is to obtain and manage software products [8]. This meaning shows the large number of factors involved in processes.

Over last decades, researchers throughout the world have proposed different methods and techniques to define software processes what shows the great interest existing as scientific subject in software engineering. This affirmation is the conclusion of different works such as the Zamli‘s et al. papers [9,10] and the García-Borgoñón's et al. paper

[*] Corresponding author.
E-mail address: juliangg@us.es (J.A. García-García).

[11]. Both papers establish their own taxonomy to classify SPML (Software Process Modeling Languages). Historically, the first generation relates to those SPML which are Petri nets-based, rule-based or programming language-based. These focus on process execution and formality, which make them become complex, inflexible and difficult to understand. The second generation coincides with the moment when UML became mature as a standard language in the software industry. However, authors as García-Borgoñón et al. [11] conclude that to talk about generations of languages is not particularly appropriate, since although more second-generation proposals have been launched from some years ago up to date, there are recent proposals which can be classified a s fi rst-generation la nguages. Th us, Ga rcía-Borgoñón et al. propose a taxonomy for SPMLs in three groups [11]. The first group is named grammar-based languages and includes all SPMLs that focus on formal languages, mathematical and programming, by means of rules or restrictions. A second group contains several versions of UML-based SPMLs, and finally, the last group includes metamodel-based SPMLs or DSLs. García-Borgoñón et al. also categorize 41 SPMLs within the groups of their taxonomy [11].

Anyway, each SPML has advantages and disadvantages according to requirements of each organization at a specific moment. In this context, the decision of using the most suitable SPML is the main challenge for any software organization because the dependency created is very strong. If another modeling language is necessary to use by the company, this one has to translate their process models[1] to the new language.

This paper aims to explore in more detail SPML in order to establish their scope, domain, features or benefits, among other aspects. Besides, it suggests a characterisation model to compare and evaluate software SPML (those are based on models) taking into account the defined scope. For this purpose, this paper uses the methodology defined by the Quality Evaluation Framework (QuEF) [12], and we propose a quality model to evaluate and compare model-based SPMLs. This model allows organization can choose the most suitable SPML for their purposes and requirements. QuEF also provides mechanisms for improving the quality model itself. The objective of this paper is twofold; to: (*i*) identify measured features for software SPML by means of a quality model conform to QuEF; and (*ii*) demonstrate the value QuEF offers to support the process modeling.

Finally, after this introduction, Section 2 and Section 3 describe, respectively, some related works and our research questions, which is followed by the research method that tries to approach it. Section 4 describes briefly the theoretical foundations of QuEF. Then, Section 5 describes each feature of our quality model. Sections 6 and 7 present the evaluation of our quality model to evaluate model-based SPMLs and our analysis and discussion, respectively. Finally, Section 8 states learned lessons as well as ongoing work.

2. Related works

Over last years, some research papers have emerged as literature reviews and surveys in SPMLs [9–11], but we have found too few on quality models or comparative studies in this area. Below, the most relevant works are briefly described.

Sutton et al. [13] proposes a simple classification scheme which is based on process enactment support. This classification only allows comparing SPML in terms of: non-enactable (i.e., SPMLs only support process modeling, but not process enactment); simulated (i.e., SPMLs enable a high-level simulation, but do not provide finegrained control of the software process); or enactable (i.e., SPMLs permit the process model to be enacted to control a software process). Authors also compare each proposal in terms of semantic richness, ease of use, mechanism of abstraction and composition and graphic representation.

Some years later, in [14], authors compare and evaluate six UML-based SPMLs taking into account few requirements previously identified in the literature related to SPML. Authors compare each proposal in terms of semantic richness, formality and conformity to UML, mechanism of composition, executability and tooling support.

Pichler et al. [16] perform an empirical investigation to compare process modeling languages. In this case, the scope of this publication is different to the one presented in this work, presenting a comparison between just two kind of languages: imperative and declarative. Authors have carried out this comparative study in terms of semantic richness and understanding of process models.

Muehlen et al. [18] presents a representational analysis of four rule modeling specifications: The Simple Rule Markup Language (SRML), the Semantic Web Rules Language (SWRL), the Production Rule Representation (PRR), and the Semantics of Business Vocabulary and Business Rules (SBVR) specification comparing their Bunge-Wand-Weber (BWW) representation capabilities. Authors compare each proposal taking into account representation capabilities and conclude that, no single language is internally complete with respect to the BWW representation model being better to use a combination of two of them (SRML and BPMN) for this purpose.

Kopp et al. [19] discuss the core characteristics of graph-based and block-structured business process modeling languages and compare them with respect to their join and loop semantics. Authors compare each proposal in terms of semantic richness. Similarly, Lu et al. [17] present a comparison of graphical process models modeling and rule-based SPMLs. Lu et al. also compare each proposal in terms of semantic richness in the control flow definition of process models.

3. Research question and methods

After presenting the context of this paper and our motivating scenario, we establish our hypothesis and research question in order to define our proposal rigorously: *«Is it possible to useful define a quality model that allows objectivity evaluating and comparing model-based SPMLs?»*.

To answer this question, this paper proposes a solution following methodological principles, as suggested in [20]. Specifically, this paper follows the Design Science Research Methodology (DSRM) [21] that establishes five phases:

Phase 1. Problem identification and motivation. Section 1 describes the context of this paper and introduces the problem of software companies to choose the more appropriate SPML to model their processes. In [11], Systematic Literature Review (SLR) is carried out by us to identify SPML proposals that have been suggested in last decades. the conclusion we reach is a great amount of SPMLs have been proposed (specifically, authors analyse and categorize 41 SPMLs in [11]) This makes us reflect on the idea that there is no perfect SPML. Each language has strong and weak points depending on the requirements of each organization at a specific time. In fact, the main challenge for an organization is to choose the best and most suitable SPML to meet its requirements. This is an important decision and it should be critical because once a specific SPML is chosen, a very close dependency is created. Problems related to needs of migration between process model notations could append what constitutes a complex and costly task in itself [11].

Phase 2. Objective of the solution. As mentioned above, the García-Borgoñón's et al. paper [11] identifies large number of SPMLs. Specifically, these authors study and analyse 41 SPMLs of which, 25 SPMLs are based on model. In this context, it could be interesting to establish a characterization mechanism that allows us to evaluate and compare

[1] Process model is a formal representation that provides a unified environment integrates the production and management activities. This integration controls and improves the information flow by which the management activities control the production activities [15]. When the management context is focused on software environment, the concept of *"process model"* is known as *"software process model"*.

model-based SPMLs objectively. This mechanism is important because preliminary conclusions from [11] confirm t hat m any model-based SPMLs do not provide mechanisms to support all phases of a typical process lifecycles which usually comprise, at least, four phases [22,23]: (*i*) modeling; (*ii*) execution and orchestration; (*iii*) monitoring; and (*iv*) continuous improvement.

A s mentioned above, our goal is to define a nd p ropose a quality model by which, software organizations can be able to objectivity compare and evaluate model-based SPMLs to choose the more appropriate SPML to their requirements. We propose a flexible, e asily extensible, scalable, and practical quality model which is going to allow objectively measuring how each model-based SPML supports each feature included in our quality model. For this purpose, we use and follow QuEF [12] after defining a specific quality model to compare and evaluate model-based SPMLs. QuEF is a framework to analyze and evaluate the quality of any kind of entity (e.g., model-based SPMLs approaches in this paper). In this framework, the evaluation of an entity is calculated in terms of a set of information needs and quality features. The first one (information needs) are requirements demanded by final users and the second one (i.e., quality features) are specific aspects that the entities provide to their users. In addition, it could be interesting to indicate that QuEF has been successfully applied in different contexts and business areas. For example, in Web Engineering [24], Product Lifecycle Management [25], Enterprise Document Management [26], among others.

Phase 3. Design. This phase aims to design a novel theoretical quality model and its associated features to evaluate SPMLs. These features are proposed and improved taking into account requirements for model-based SPMLs identified b y s everal r esearch studies [9,27–29]. These requirements are described in Section 5 in detail, but, in summary, these ones are nine features: (*i*) semantic richness and expressiveness; (*ii*) understandability; (*iii*) conformity to standards such as UML (Unified Modeling Language); (*iv*) granularity; (*v*) executability and orchestrability; (*vi*) measurability; (*vii*) business rules[2] support; (*viii*) support tools; and (*ix*) validation in real environments.

Phase 4. Demonstration. Once defined our quality model in the previous model, it is necessary and important to develop a supporting tool to automatically compare and evaluate each model-based SPMLs. In this sense, we present a supporting Web tool by which, software organizations can be able to execute our theoretical quality model.

Phase 5. Evaluation. This phase implies the use of our quality model defined previously on a set of representative model-based SPMLs (based on metamodels, UML or based on standards). Sections 6 and 7 evaluate these approaches in detail and present an analysis, respectively. This evaluation and analysis process has been methodologically carried out by three senior researchers (authors of this paper). Each researcher analyzes individually each model-based SPML and evaluates each feature of our quality model. However, some doubts may appear during this evaluation process. For this reason, *face-to-face* meetings between researchers have been performed to jointly discuss and agree on the evaluation final. These meetings also minimize the bias of each researcher.

4. QuEF: Quality evaluation framework

QuEF is a model-based framework that it allows managing quality requirements of any kind of entity (products, processes or services, among others) within any kind of business or theoretical environment (we manage quality of model-based SPMLs in the context of our paper). For this purpose, QuEF is aligned with several quality standards. These are some of them: (*i*) ISO 9000:2000 [30], ISO 9001:2008 [31], ISO 9004:2000 [32], which lay the foundations to carry out improvements and organization excellence; (*ii*) ISO/IEC 9126 [33] and ISO/IEC 25000:2005 (SQuaRE) [34], which provide quality characteristics to evaluate any kind of entity what lays the foundation for the definition of the quality metamodel defined by QuEF; (*iii*) ISO/IEC 20000 [35–39] and ITIL v3 (Information Technology Infrastructure Library) [40] that define best practices to improve service quality based on continuous improvement lifecycles.

Regarding theoretical foundations, QuEF provides model-based mechanisms to define your own quality model for the domain under study as well as methods to instantiate and evaluate this quality model. It is important to mention that these mechanisms are framed into a continuous improvement lifecycle what allows managing improvements in your quality model. Fig. 1 shows how QuEF is used in practice. On the one hand, *providers* are who have expertise and knowledge on the specific domain. They also define needs to analyze, control, evaluate and improve the entities under study. On the other hand, QuEF considers *consumer* roles, who instance the quality model and decide what is the most suitable entity for them according to their needs.

As mentioned above, QuEF provides model-based mechanisms to define your own quality model and evaluate any kind of entity. These mechanisms include a metamodel (which is considered the core of the framework) and a continuous improvement lifecycle (which allows to apply QuEF in order to achieve quality management becomes strategically active). Both mechanisms are described in detail in [12], but we are going to briefly present them.

On the one hand, the QuEF metamodel allows defining our quality model and laying the foundations to specify quality requirements and evaluate them. This quality model metamodel contains a set of characteristics and its relationships, and represents the core and quality management revolves around it what allows the model instantiation can be more flexible and practical.

On the other hand, QuEF defines a continuous improvement lifecycle based on several stages which allow to methodologically improve our quality model. Firstly, *providers* have to determine what is the strategy to manage the quality on their domain entities. This stage is known as *Quality Model Strategy phase*. The second stage is named *Quality Model Design phase* and its goal is the definition of the quality model in terms of all strategic entities under study. This stage is also performed by *provider* roles. Once quality model is defined, it is time to analyse and evaluate each entity according to the quality model and its features (*Quality Model Operation phase*). Finally, QuEF also includes and defines methods to execute changes on the quality model as well as improve its own lifecycle in a controlled manner. These aspects are performed in *Quality Model Transition phase* and *Quality Continuous Improvement phase*, respectively.

After briefly introducing QuEF, it might be interesting to indicate that, a defining comparison criteria is just part of QuEF. This framework offers techniques, methods and tools to support a quality continuous improvement of a quality model (in this case, a quality model for SPMLs). So, QuEF is not just applied to analize and evaluate SPMLs (*Quality Model Operation phase* in QuEF) but QuEF is also used to apply a set of phases (strategy, design, operation, etc.) that assure a quality continuous improvement of a quality model for SPMLs. It offers a set of benefits, for instance, try to agree a quality model for SPMLs in order to be standarized by the community. In addition, the criteria applied to the quality model might be adapted according the context. So, QuEF offers you the possibility to modify and weigh these criteria as well as adjusting them automatically to the context, without needing to redo in each case. It also allows to play with both quantitative and qualitative aspects.

5. An approach to characterise the quality of model-based SPML

At present, there are a lot model-based SPML and process notations

[2] A business rule is a rule that defines or constrains some aspect of business and always resolves to either true or false. Also, business rules are intended to assert business structure or to control or influence the corporate behavior of an organization (people, processes, software systems, etc.) [41].

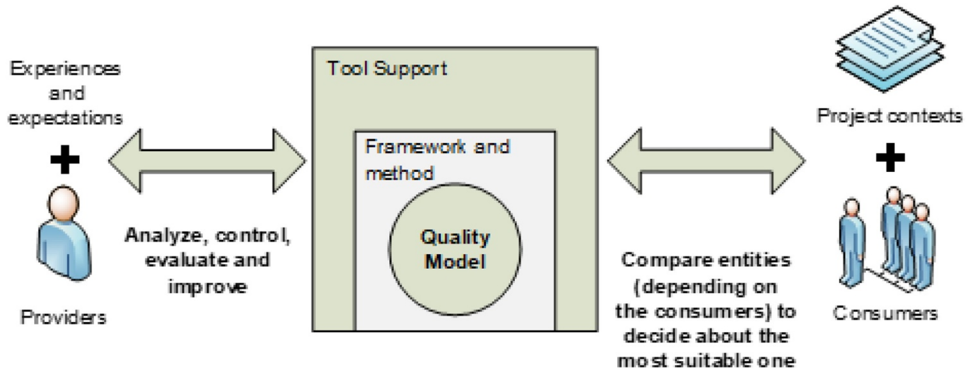
Experiences and expectations
Providers
Analyze, control, evaluate and improve
Tool Support
Framework and method
Quality Model
Compare entities (depending on the consumers) to decide about the most suitable one
Project contexts
Consumers

Fig. 1. Conceptual scheme of QuEF [13].

what reflect that there is not perfect SPML. This statement is a conclusion obtained from [11] where authors analyse and categorize 41 SPMLs; each one has strong and weak points because each one was proposed to support specific requirements of an organization. In any software organization, the election of the best and most suitable SPML to meet its requirements is a challenge itself.

This paper tries to improve the decision-making when a software organization needs to choose the most suitable SPML. For this purpose, we propose a quality model which focuses on nine specific quality features (e.g., human understanding, automated execution support as well as knowledge of real successful applications of each evaluated proposal, among others).

Before proposing our dimensions, other comparative studies have considered to establish appropriate criteria. In [9,27–29,42], many authors have identified requirements related to SPMLs. After taking into account these references, we have selected most predominant and common criteria (that is, semantic richness & expressiveness, understandability, conformity to standards, granularity, and executability) and we have included other ones as features to be evaluated (that is, orchestrability, measurability, business rules support, support tools, and validation in real environments). Following subsections, we introduce each feature in detail.

5.1. *Expressiveness*

It is possible to consider different perspectives in order to define process models [43]: control-flow, data-flow, resource and operation perspectives. On the one hand, the first one (control-flow perspective) describes how activities are ordered during the process execution flow. This ordering is usually built using control elements such as sequence, choice, parallelism, and synchronization elements, among others. The second perspective is data flow which represents how each information object (business documents, workflow variables, etc.) is processed during the execution of the control flow. On the other hand, the *resource perspective* is the third perspective. It allows defining roles and devices machine who/which perform each activity of the process. Finally, the atomic actions executed by activities are represented with the *operation perspective*.

Workflow language's effectiveness and expressiveness is mainly provided by the control flow perspective. The data perspective rests on it, while the organizational and operational perspective are ancillary. Expressiveness is an objective criterion which establishes how much modeling constructs are offered by a specific SPML. Given a certain set of these constructs, it is possible to show that a specific SPML can, or cannot, be modeled.

In this context, there are many issues related to the expressive power of process model languages, but our quality model is going to take into account constructs to model: human and automatic activities, conditional branches, parallel branches, exception handling, products and roles.

5.2. *Understandability*

The understandability of process models is, at present, one challenge in BPM [44]. This paper considers understandability as the ability to easily manage, read process flow without additional explanation. It is possible to establish numerous and intricate dependencies among many elements within a process model. In this context, it is sometimes complex and hard to explain process models to stakeholders, and to maintenance these models over time (e.g., due to unforeseen circumstances or changing business requirements). In literature, some studies have been carried out on process model understandability. For instance, metrics for measuring the understandability process models [45], features that favor process models understandable [46] or influence factors of understanding process models [47], among others.

In this context, our quality model includes this feature to evaluate how each evaluated SPML allows defining understandable and readable process models. This evaluation is based on the use of familiar metaphors, number of symbols, structures used and unstructured statements.

5.3. *Conformity to standards*

The standardization of software process descriptions is an important need because of the proliferation of model-based SPMLs over last decades. In addition, most of these SPML present its own format, content and level of prescription because most of them has been proposed for specific purposes.

For this reason, ISO/IEC TR 24774:2010 [48] standard is defined by ISO (International Organization for Standardization) to present guidelines about the most frequently elements used when any software process is modeled (e.g., the title, purpose, outcomes, activities, task and information item, among other aspects) in order to standardize the definition of process models. Whilst the primary purpose of ISO/IEC TR 24774:2010 is to encourage consistency in standard process reference models, the guidelines it provides can be applied to any process model developed for any purpose.

Moreover, over last decade, UML also have emerged as alternative to define SPML because provides a rich set of notations, diagrams, and extension mechanisms. In addition, thanks the extension mechanisms of UML (in form of an MOF [49] metamodel, of a UML-Profile, or simply reuse UML diagrams for their language), this alternative allows reducing costs of developing a supporting tool and efforts of users when they use the process language into their favourite UML tool. In this context, many SPML based on UML have been proposed since several years.

Finally, and as conclusion, our quality model also includes these features in order to measure if each SPML is defined conforms to these standards.

5.4. *Granularity*

Reusing concepts is a psychological mechanism own of humans [50] that provokes reaction in the brain to specific input patterns, scripts and other visual models. Many application areas as learning or knowledge management, among other areas have successfully used these mechanisms as method for solving problems.

These psychological mechanisms of humans are also present within software and technological organizations [51] where the application of reuse mechanisms are very attractive in design and management tasks (e.g., code and component reuse, reuse of reference models, design patterns, reuse of conceptual models, etc.). In fact, it is possible to establish qualitative improvements in daily work [52]. In this context, reusing methodological patterns or business process models gain importance because of their higher abstraction. In addition, it allows improving the decision making and reducing cost in an enterprise [53].

Although there are research studies that evaluate impacts of granularity on business process understandability and detect limited degree of granularity [54], we consider very important to include this feature in our quality model to evaluate how each SPML supports the granularity as understandability mechanism.

5.5. *Executability and orchestability*

Many organizations have successfully applied BPM to improve their internal management. However, there are still limitations and difficulties during the execution of processes [15]. For example, this situation is usual in software organizations because of characteristics of the software process itself. Some authors have identified the main differences between software process and other processes such as industrial processes [15]. These differences are: (*i*) they are constantly evolving, as they usually incorporate new lifecycles and technologies and they frequently comprise several iterations that produce different software products versions; (*ii*) they are complex because they are strongly influenced by many unpredictable circumstances and many work teams; and (*iii*) they often rely on communication, coordination and cooperation of different frameworks and development technologies as well as on the different roles they play.

In this context, many software organizations usually focus their efforts only on modeling their processes, but the execution is forgotten because many activities cannot be easily and effectively automated [15]. This situation causes each role execute its tasks manually, which complicates the execution [55] and orchestration [56] of the process. We have contrasted this statement with many international and Spanish software companies with which we work on R&D projects.

Therefore, executability and orchestability seem to be critical and essential features in any proposal to manage processes because companies are being driven by the need to extensively automate their processes with enterprise management systems and process engines (known as BPMS, Business Process Management Suite [57]). Coordinating between process' participants or enforcing artifacts routing, etc., could be improved if this goal is achieving. For this reason, executability and orchestability are two features included in our quality model to know how each SPML support them.

5.6. *Measurability*

Once the process is deployed and is being executed, effectiveness of the software process can be evaluated in order to know its productivity. For this purpose, it is necessary to define key performance indicators (KPIs) as mechanism for performing a continuous improvement lifecycle. Quality, efficiency, effectiveness and performance levels are important aspects to achieve in the process execution [58]. In this sense, therefore, the model-based SPML should support the definition of indicators or metrics to measure process elements (the process itself, its activities, products, etc.). However, it is not only important defining metrics or indicators, but also the calculation associated with these metrics or indicators at runtime of the process.

Therefore, we have considered to include this feature in our quality model. After studying this feature per each SPML, it will be possible to know the support degree of each evaluated SPML.

5.7. *Business rules*

As mentioned previously in Sections 5.1 and 5.2, a good understanding of a domain (e.g., process models) is a prerequisite to effective communication and design of models which improve human cognition of the domain represented (i.e., process models defined). However, process models mainly focus on the modeling of activities within taking into account aspects very important to support for the remaining stages of the BPM lifecycle. One of this forgotten aspects is the definition of business rules when process engineer models his/her processes [59].

In practice, business rules are indispensable in the design and implementation of process models because they could be extracted from laws, policies, procedures, etc. Business rules could be represented in different ways: (*i*) an integrated manner into the model (i.e., using graphical mechanisms, such as graphical links or text annotations) or in a separated manner (i.e., using separate documents or rule engines, but the relations and connections of process models and the rules are not explicitly represented in the process models [59]).

Therefore, we have considered to include this feature in our quality model taking into account the statements previously argued. After studying this feature per each SPML, it will be possible to know the support degree of the definition of business rules within process models in an integrated manner.

5.8. *Supporting tools*

Using theoretical proposals (based on models and metamodels) for process modeling guarantees uniformity, formalized terminology and correct definition. Designing and developing friendly software tools allows ensuring the applicability of this kind of proposals in real contexts. In fact, maintenance could become too complex and expensive in enterprise environments when it is not possible to have a suitable supporting tool [60]. This statement is especially important when the theoretical proposal is based on models and paradigms such as MDE (Model-Driven Engineering) [61]. The application of MDE may become monotonous and very expensive if there are no software tools that automate the process.

In this sense, therefore, we have also considered to include this criterion in our quality model because it could be very important when any organization has to choose a SPML among different alternatives.

5.9. *Validation in real environments*

Although it is not frequently a characteristic supported in studies, there are different studies in the literature [62–64] that demonstrate that the research literature is full of examples that were never applied in practise. Even when they tried to be applied to full and complex example they were not successful being only useful for academia. Therefore, we consider to include this criterion in our quality model because it could be important to show the technical viability of a SPML when this one is successfully applied in practice. This feature is evaluated taking into account public accessible references and papers where these applications are described.

6. Evaluation of model-based process modeling languages

As justified previously in Section 3, this paper is going to compare a concrete set of model-based SPMLs (PLM$_4$BS, UML$_4$SPM, UML-EWM, UPME, Ferreira's approach, SPEM 2.0, Combemale's approach, xSPEM, eSPEM and MODAL). This set regroups representative initiatives from industry and academic research groups. In addition, these SPMLs that have been evaluated in this paper have been selected taking into account their transcendence and use in the community. These aspects are

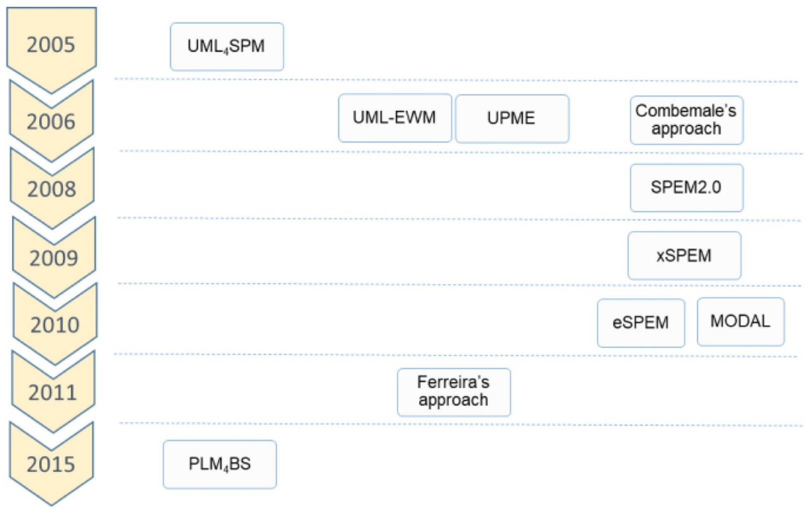

Fig. 2. Summary of SPML analysed in this paper.

conclusions of the systematic literature review presented in [11]. These proposals could be considered representative of different kinds of research lines and initiatives [11]: metamodel-level approaches with full executability support; high-level approaches based on UML diagrams and an ad-hoc low-level approaches; standard SPMLs and other approaches based on standards.

Next Fig. 2 summarizes the SPMLs analysed in this paper and grouped by year. Below, this section is organized according to the different kinds of initiatives identified previously and next subsections evaluate each SPML in detail taking account our quality model defined in Section 5.

6.1. PLM4BS evaluation

PLM$_4$BS (Process Lifecycle Management for Business-Software) [65,66] defines a theoretical MDE-based framework to apply BPM in real business environments. Below, PLM$_4$BS is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. PLM$_4$BS defines a Process Definition MetaModel (PDMM) that supports three kinds of ***activities or actions*** to develop the process: (*i*) «OrchestrationActivity», which means a task carried out by an automatic device; (*ii*) «ComplexActivity», which allows including a process within another process; and (*iii*) «HumanActivity», which is performed by someone.

Defining the process ***control flows*** are supported by PLM$_4$BS with two mechanisms. Firstly, PLM$_4$BS supports different types of «ControlElement» to distinguish among diverse kinds of control flow: (*i*) «InitialElement» and «FinalElement», which mean entry and exit point of the process, respectively; (*ii*) «Conditional», which establishes exclusive branches into process workflow; and (*iii*) «Fork» or «Join», which allows starting and ending parallel branches, respectively. Secondly, PLM$_4$BS defines the «Link» element to build the process workflow between any other elements.

Moreover, the ***exception flow*** is not supported by PLM$_4$BS yet. However, the ***data flow*** is considered by PLM$_4$BS using the «Product» element which can be typed as an input, an output or an input/output product. PLM$_4$BS also allows defining data flows between activities using «BusinessVar» elements which provide to share data between activities.

Finally, the PDMM of PLM$_4$BS includes the «Stakeholder» element to represent human ***resources*** that can be either participant or responsible of a «HumanActivity». These ones differ in that first one could complete some aspects of the product (which are produced by the activity), but they cannot complete the activity without being authorized by the responsible member, who can fulfill the activity in order to continue the process actions sequence.

Understandability. Authors have designed PLM$_4$BS metamodels using UML-Profiles what has made it possible to take advantage of all the semantic and visual richness of artefacts and basis models of UML. PLM$_4$BS also provides familiar metaphors and 8 symbols.

Conformity to standards. PLM$_4$BS has been defined conforms to MOF, UML and ISO/IEC 24774:2010. PLM$_4$BS also use OCL (Object Constraint Language) [67] to include formal semantic constraints (which guarantee the building of well-defined models) as well as QVT (Query/View/Transformation) [68] and MOFM2T [69] to define transformation rules (model-to-model and model-to-text rules, respectively).

Granularity. This feature is supported by PLM$_4$BS with its «ComplexActivity» element, which is used when it is necessary to include a process within another one.

Executability & Orchestability. PLM$_4$BS theoretically defines an execution metamodel (named Process Execution & Orchestration MetaModel; PEOMM) [66] which is generated from PDMM using QVT rules. PEOMM describes static information of each concept or attributes and OCL semantic constraints to ensure that process models are built well. In addition, PEOMM also describes information at runtime to define how and when each element of PEOMM can be instantiated. In this sense, PLM$_4$BS associates a state machine each element to react and evolve its status at runtime along its own lifecycle. In addition, transformation rules are defined using MOFM2T to generate executable code (WS-BPEL [70] and BPEL4People [71]) from PEOMM.

Measurability. This feature is partially supported. Users can use the PDMM of PLM$_4$BS in order define metrics and indicators associated with their processes [65]. However, PLM$_4$BS does not support the automatic calculation in runtime of these metrics/indicator yet.

Business rules. PLM$_4$BS also supports the definition of business rules using special and specific associations between «BusinessVar» elements (mentioned above). This mechanism also provides indirect data flow paths when a «ControlElement» (type «Conditional») node is reached and read. In consequence, the «Conditional» node is the node where variables are checked to select the next action to be performed. The checking action is represented by «checkBusinessVar» relationship between «ControlElement» metaclass and «BusinessVar» metaclass. This relationship also has «comparison_value» and «logic_operator» properties to config the business rule. The result is calculated using the equation below, when the process model can be executable and a «Conditional» node checks a collection (C) of n variables:

$$\begin{aligned} &Result\ (C) \\ &\quad = \wedge_{i=1}^{n} \{C_i(initValueVar)\ C_i(logicOperator)\ C_i(comparisonValue)\} \end{aligned}$$

where:

- C_i is the *i*th checked variable.
- C_i*(initValueVar)* is the initial value of C_i variable.
- C_i *(logicOperator)* is the selected logic operator to evaluate C_i variable. It can be *equal* or *non-equal*.
- C_i *(comparisonValue)* is the value used to compare the initial value of C_i variable.

Support tools. PLM$_4$BS is supported by CASE (Computer Aided Software Engineering) tools named PLM$_4$BS Suite [65]. This suite is integrated into Enterprise Architect (EA) [72] and it provides a professional working space to use UML-Profiles of each metamodel of PLM$_4$BS, apply transformation rules between models and automatically verify each OCL constraints in runtime.

Validation in real environments. After finding some references, some real projects are identified where PLM$_4$BS has been successfully applied and validated (e.g., healthcare environments [73] and business consulting environments [74]).

6.2. UML4SPM evaluation

UML$_4$SPM [75,76] is acronym of UML for Software Process Modelling and has been defined as MOF-compliant metamodel extending UML2.0. Below, UML$_4$SPM is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, we evaluate aspects like Activities or tasks, control flow, exception flow, data flow and resources. Firstly, UML$_4$SPM metamodel includes «SoftwareActivity» concept, which represents an individual and general task (machine or human). Secondly, UML$_4$SPM also supports the definition of ***control flow*** in process models. As mentioned above, UML$_4$SPM has been defined by extension of the UML «Activity» metaclass, which is related to itself to build the control flow of the process. Therefore, UML$_4$SPM also verifies this feature by extension.

Moreover, the ***exception flow*** is supported by UML$_4$SPM with the «RaiseExceptionAction» and «ExceptionHandler» concepts, which are provided by UML2.0. UML$_4$SPM uses these concepts with «SoftwareActivity» in order to redirect the control flow to a predefined behaviour when an unexpected situation appends during the «SoftwareActivity» execution. In this situation, the «SoftwareActivity» is stopped and UML$_4$SPM assigns the appropriate handler in the flow.

Regarding ***data flow***, UML$_4$SPM defines «WorkProduct» concept, which means data consumed, produced or modified during the software process. It is also related to ***resources*** of the process. In this case, UML$_4$SPM defines the resources as activity performers and it establishes two categories: «Tool», which identifies an automatic resource (machine, code batch, etc.); and «Performer Role», which may be an agent with specific skills or a team.

Understandability. As mentioned above, UML$_4$SPM reuses some UML2.0 diagrams and notations. Therefore, UML$_4$SPM takes advantages of UML regarding the understandability because UML2.0 is standard, graphical, intuitive, and easy to understand.

Conformity to standards. UML$_4$SPM has been defined as MOF2.0 metamodel extending UML. The UML$_4$SPM metamodel organizes the elements in the UML$_4$SPM Process Structure UML$_4$SPM Foundation packages. Primary process elements of UML$_4$SPM and the subset of UML2.0 concepts extended these process elements are stored in the first and second package, respectively.

Granularity. Although this aspect is not described in detail in the reference identified, UML$_4$SPM relates its «Process» concept with itself what could allow defining any process within another one at several levels [76].

Executability & Orchestability. This aspect is supported by UML$_4$SPM with a set of mapping rules to generate WS-BPEL from UML$_4$SPM models [77].

Measurability. This feature is not supported by UML$_4$SPM yet.

Business rules. This feature is not supported by UML$_4$SPM yet.

Support tools. UML$_4$SPM provides an editor integrated into the Eclipse environment.[3] This prototype also implements each mapping rule between UML$_4$SPM and WS-BPEL.

Validation in real environments. In [32], authors mention the elaboration of two case studies to evaluate UML$_4$SPM in the context of two European research projects (ModelWare and ModelPlex). However, we do not have been able to find references about these projects in order to know how UML$_4$SPM was applied. Anyway, it has not been possible to locate publications on the validation of the proposal in industrial and real environments.

6.3. UML-EWM evaluation

UML-EWM [78,79] proposes an UML Extended Workflow Metamodel. Especially, it extends UML Activity Diagram (UML-AD) metamodel to incorporate the capacity to modelling processes workflows. However, UML-ADs does not provide elements enough for modeling in detail software process models. Then, authors extend UML-AD to include new concepts such as of organizational elements in the process models. Below, UML-EWM is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. After studying the UML-EWM metamodel, this one just includes the «WorkflowActivity» which represents an elementary activity or task of the process. In addition, it is possible to find some concepts or metaclasses with which define the ***control flow*** of the process. For example, this metamodel defines conditional branches, but not parallel branches in the control flow. The ***exception flow*** is not supported neither by UML-EWM.

Moreover, the ***data flow*** and ***resources*** are considered with the «WorkflowParticipant» and «WorflowRelevantData» metaclass, which represents the participants and, data that are produced during of the software process, respectively.

Understandability. UML-EWM presents an extension of UML what guarantee the understandability of UML-EWM because UML is standard, graphical, intuitive, and easy to understand. However, UML-EWM does not include familiar metaphors for non-technical users.

Conformity to standards. As mentioned above, UML-EWM is based on UML and it extends UM UML-AD metamodel. Therefore, this feature is supported by UML-EWM.

Granularity. Although this aspect is not described in the cited reference identified, we could consider that this feature is also supported by UML-EWM because extends UML-AD which allows the composition of activities by actions. This composition of activities and events could allow to define a workflow in a granular way.

Executability & Orchestability. In [78], authors mention that UML-EWM is also based on the Activity Graphs metamodel of WfMC what allows the automation of process models using UML-EWM. However, authors do not indicate references and it has not been possible to find research results about the execution and automation of processes modelled with UML-EWM.

Measurability. This feature is not supported by UML-EWM yet.

Business rules. This feature is not supported by UML-EWM yet.

Support tools. Authors describes UML-EWM in a theoretical way and it has not been possible to find references on modeling tools for UML-EWM.

Validation in real environments. Authors briefly mention in [78] an academic case study to model and map between SPEM and UML-EWM. However, we have not found research results that evidence the application of UML-EWM within real projects.

[3] Manual Editor tool of UML$_4$SPM. Official website. Retrieved January 2018 from, http://pagesperso.lip6.fr/Reda.Bendraou/IMG/pdf/prover_-user_guide.pdf

6.4. UPME evaluation

UPME [80] is a UML-based proposal that extends UML Class Diagrams (UML-CD), UML-AD, and UML State Diagrams (UML-SD), among others. UPME establishes a framework comprising three phases for SPM: (*i*) metamodels for building the software process domain; (*ii*) model instantiation for building the model using IDL (Instantiation Description Language scripts); and (*iii*) model compilation for translating the model into object-oriented code skeletons which allows the process execution. Below, UPME is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, we evaluate aspects like Activities or tasks, control flow, exception flow, data flow and resources. In this context, the definition of **work and control flows** are partially supported by UPME because authors delegate this feature on the UML-AD metamodel. After studying cited-above references, defining the ***exception flow*** is also not supported by UPME.

Regarding the ***data flow*** and ***resources***, UPME is partially supported this feature is delegated on UML-AD metamodel and its «Artifact» and «Actor» elements. The first one represents data that are either used or produced by a process or by a system operation and, the second one specifies a role (user or other system) that interacts with an activity.

Understandability. Taking into account the definition of this feature, we can conclude that it is not supported by UPME. Process models defined with the UPME metamodel have a graphic representation similar to UML-CD which has a technical notation and does not include familiar metaphors for non-technical users.

Conformity to standards. As introduced above, UPME extends UML to model some common basic elements of software process. Especially, UPME extends UML-CD and UML-SD metamodels. Therefore, this feature is supported by UML-EWM.

Granularity. It is possible to consider that UPME supports this characteristic because it is based on UML-AD, which allow granularity in their models. In addition, UPME metamodel includes the «SEL_SUB_CLS_EX» metaclass which allows the creation of subactivities into process models.

Executability & Orchestability. In this sense, authors have developed IDL which is a script language to describe the instantiation tasks explicitly [80]. For this purpose, UPME extends UML-CD metamodel to include special behaviors (related to execution) in elements of UML-AD. For instance, UPME extends the meaning of the activity concept (human activity, machine activity, software activity, etc.) with the «SUB_CLS_EX» metaclass in the UPME metamodel. Once modeled the process with «SUB_CLS_EX» metaclasses, UPME allows transforming this one into object-oriented code skeletons.

Measurability. This feature is not supported by UPME yet.

Business rules. This feature is not supported by UPME yet.

Support tools. Authors mention that there are CASE tools (for instance, Rational Rose and Visual Paradigm) which support the modeling and the kinds of tranformation rules defined by UPME. However, authors mention that these tools do not support full the UPME's framework.

Validation in real environments. Authors explain in [80] how several academic examples are modelled with UPME what demonstrates that UPME makes process modeling more reusable and easier. However, it has not been possible to locate publications on the validation of the proposal in industrial and real environments.

6.5. The Ferreira's approach evaluation

Ferreira et al. proposes an approach for software process modelling [81] extending UML Component Diagrams (UML-CPD). Authors consider that developing a process model should entail developing phases of process design and process implementation. Authors use UML-CPD models to support the design phase and define model-to-text transformation rules for obtaining executable code which could be deployed and executed in authors' engine. Below, the Ferreira's proposal is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, we evaluate aspects like workflow, control flow, exception flow, data flow and resources.

On the one hand, regarding the **workflow**, the proposal defines «ModuleLibrary» metaclass that provides a repository of the process model. This concept aims to support the management of text descriptions of applied methods in software development process. Within the Ferreira's metamodel, software process concept is represented by the «ArchitectureProcess» metaclass which is composed of UML Component. In addition, the UML Component metaclass is extended by authors to define: (*i*) «SoftComponent» metaclass, which is used to declare architecture components that do not have a semantic defined (for example, criteria of quality or satisfaction); and (*ii*) «HardComponent», which has its semantics defined by means of process modules objects. These process modules represent specific tasks within the workflow. Moreover, the Ferreira's metamodel described in [81] does not include supporting for ***exception flow, data flow*** or ***resources*** modelling.

Understandability. The Ferreira's proposal presents an extension of UML-CPD what guarantee the understandability of his proposal because UML is standard, graphical, intuitive, and easy to understand. However, we can conclude that it is not supported by it, because UML has a technical notation and the Ferreira's proposal does not include familiar metaphors for non-technical users.

Conformity to standards. As mentioned above, this SPML is based on UML and it extends UML-CPD metamodel. Therefore, this feature is supported by the Ferreira's proposal.

Granularity. Although this aspect is not explicitly described in [81], we could consider that this feature is also supported by this proposal because it extends UML-CPD which allows the composition of component models. This composition of elements could allow to define a workflow in a granular way.

Executability & Orchestability. Transformations rules are defined by authors to generate executable code (conform to Little-JIL [82]) from their process models. Little-JIL is a programming language that allows coordinating and running tasks among autonomous systems.

Measurability. This feature is not supported by the Ferreira's proposal.

Business rules. This feature is not supported by the Ferreira's proposal.

Support tools. It has not been possible to find a modeling tool to use the authors' proposal.

Validation in real environments. Authors describe in [81] how a small software development process scenario is modelled with their proposal, but it has not been possible to locate publications on the validation of the proposal in industrial and real environments.

6.6. SPEM2.0 evaluation

SPEM2.0 [83] is known by proposing a split between usage and content in software development processes. It also introduces extension mechanisms, compliance points and concepts to distinguish method contents from processes. Below, SPEM2.0 is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, SPEM2.0 includes into its metamodel aspects like activities or tasks, control flow, exception flow, data flow and resources. Firstly, SPEM2.0 supports «Activity» elements, which mean general work units assignable to performers. These performers are represented by role user. SPEM2.0 also defines other structural elements, such as «Guidance» and «Process». The first one associates elements of SPEM, to provide more detailed information to the executors about the associated elements; for example: Guides, Techniques, Metrics, Checklists, etc.; and the second one contains a set of activities to achieve a specific goal); among others.

Secondly, defining the ***control flow*** is mainly carried out by SPEM2.0 using proactive controls with the «WorkSequence» element. The control flow is also built using precedence between activities (e.g., *start-start, start-finish, finish-start* and *finish-finish*).

Moreover, ***exception flow*** is not supported by SPEM2.0 yet. However, the ***data flow*** or processing data is considered using «WorkProduct» which are in most cases tangible work products consumed, produced, or modified by Tasks. Also it is possible to define relationship among work products.

Finally, SPEM2.0 includes «RoleDefinition» and «ToolDefinition» elements to represent human and machine ***resources***, respectively. The first one defines a set of related skills of individuals; and the second one describes the capabilities of automation units (e.g., CASE tools, systems, etc.).

Understandability. SPEM2.0 is designed using UML-Profiles what has made it possible to take advantage of all the semantic and visual richness of artefacts and basis models of UML. SPEM2.0 also provides familiar metaphors to represent each its element (total 26 symbols) [83].

Conformity to standards. SPEM2.0 comes in the form of MOF2.0-compliant metamodel that reuses UML-Infrastructure and UML-Diagram Interchange specifications. The standard defines also a UML-Profile which contains a UML stereotype per each element of the SPEM2.0 metamodel.

Granularity. SPEM2.0 also allows reusing and extending process models using its «ActivityUseKind» property (enumeration) of the «Activity» metaclass. Depending on the value of this property, a SPEM2.0 activity can: (*i*) extend an activity from another process; (*ii*) be extended by another activity; or (*iii*) completely replace an activity in another existing process. This would allow to redefine, reuse, or replace another process models [56].

Executability & Orchestability. At present, SPEM2.0 does not support execution of models. Anyway, SPEM2.0 suggests two common mechanisms for that [83]: (*i*) mapping the processes into Project Plans and enacting these with project planning and enactment systems such as IBM Rational Portfolio Manager[4]; (*ii*) mapping the process to a business flow or execution language and, then executing this representation of the processes using a workflow flow engine such as a BPEL-based workflow engine. Nevertheless, SPEM2.0 supposes that this task is the tool implementer's responsibility.

Measurability. This feature is supported by SPEM2.0 with the «Metric» which allows measuring instances of SPEM2.0 elements.

Business rules. This feature is not supported by SPEM2.0 yet.

Support tools. SPEM2.0 is supported by EA, among others.

Validation in real environments. After finding some references, it is possible to identify some real projects of various areas where SPEM2.0 has been used, especially in Multi-Agent Systems [84] and Agile Software Development [85], among others.

6.7. *The Combemale's et al. approach evaluation*

Combemale et al. [86] suggest a SPEM1.1-based [87] proposal for SPM. This proposal arises due to the difficulty of using SPEM1.1, as it is very general and provides no guidelines on its use. SPEM1.1 semantics is essentially expressed in a natural language that leads to the construction of inconsistent process models, as it lacks a formal definition of concepts. Authors define a specialization of the SPEM metamodel whose purpose is to clearly define concepts and formally express their semantics with OCL. Below, the Combemale's proposal is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, we evaluate aspects like activities or tasks, control flow, exception flow, data flow and resources. The authors' approach aims to clearly and formally defines SPEM1.1 concepts and their semantic, but authors do not add new concepts in their metamodel respect to SPEM1.1.

However, it could be interesting to evaluate semantic richness and expressiveness on SPEM1.1. Firstly, SPEM1.1 supports «WorkDefinition» and «Activity» elements, which represent general units of work assignable to specific performers represented by role use. An activity also can be composed of «Steps», which describes an assignable unit of work and it usually affects one or only a small number of work products. Secondly, defining the ***control flow*** is mainly carried out by SPEM1.1 using control and object flows. SPEM1.1 also supports proactive controls with the *start-start, finish-start* and *finish-finish* precedence. Moreover, ***exception flows*** are not supported by SPEM1.1. However, the ***data flow*** and ***resources*** are considered using «WorkProduct» and «ProcessRole», respectively.

Finally, we are going to consider that the authors' proposal partially supports the semantic richness and expressiveness because this proposal itself does not provide new concepts respect to SPEM1.1 for SPM.

Understandability. This approach does not support this feature because this proposal itself does not provide familiar metaphors when its own concepts are instanced, neither any advantage on understandability respect to SPEM1.1, which uses UML1.4 diagrams and includes customized icons.

Conformity to standards. This approach is based on SPEM1.1 which is a standard itself. SPEM1.1 is defined as MOF1.3-compliant metamodel and an UML-Profile.

Granularity. It is important to evaluate this feature on SPEM1.1 since this approach is based on this standard. SPEM1.1 provides mechanisms for composing and reusing process models thanks to «ProcessComponent» compositions. However, it is necessary to manually rename process elements when «ProcessComponents» are combined in order to get one coherent process. This issue is a well-known issue of SPEM1.1 that the Combemale's approach does not solve.

Executability & Orchestability. This feature is not supported by this approach.

Measurability. This feature is not supported by this approach.

Business rules. This feature is not supported by this approach.

Support tools. Regarding the tooling support, authors do not provide any supporting tool to use their proposal in practice.

Validation in real environments. Regarding this feature, we have not found research results that evidence the application of this approach within real projects.

6.8. *xSPEM evaluation*

The xSPEM [88] (eXecutable SPEM) proposal is proposed as extension of SPEM2.0 to provide executable mechanisms in SPEM2.0. It validates process model execution using Petri-net. Below, xSPEM is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, xSPEM does not introduce new improvements in the semantic richness and expressiveness of the SPEM2.0 notation because xSPEM just focuses on defining executable mechanisms in SPEM2.0. In this context, we are going to consider that xSPEM partially supports this feature because this proposal itself does not provide new concepts respect to SPEM2.0 for SPM. This one is the similar criteria used to previously evaluate the Combemale's approach (Section 6.7).

Understandability. Although xSPEM is based on SPEM2.0, authors have proposed xSPEM in a theoretical manner and it has not been possible to find cites where authors explain mechanisms for friendly defining xSPEM models.

Conformity to standards. The author's proposal is based on SPEM2.0 (which is a standard itself) and also use OCL to add formal semantic constraints. Authors also uses ATL,[5] Kermeta [89] and xOCL[6]

[4] Rational Portofolio Manager (RPM). Website: https://www.ibm.com/developerworks/ssa/library/IBM_Rational_Portfolio_Manager.html

[5] ATL (ATL Transformation Language) is a transformation language of model using Eclipse technologies. Website: http://www.eclipse.org/atl/

[6] XOCL is a combination of Eclipse Modeling Framework (EMF) [91] and OCL to define domain specific languages. Website: http://www.montages.com/xocl.html

to define the behavior of xSPEM models in an imperative manner.

Granularity. xSPEM does not define any additional improvement with respect to SPEM2.0 in terms of modularity or granularity. In this context, we consider it appropriate to evaluate this characteristic partially for xSPEM since xSPEM is focused on the executability of SPEM2.0 (following the same criterion previously used).

Executability & Orchestability. As mentioned above, xSPEM extends SPEM2.0 and it provides concepts related to execution of process models. For instance, xSPEM stores status of each process element during runtime. In addition, xSPEM defines two complementary mechanisms to execute process model. The first one allows validating process models with model-checkers based on Petri-nets; and the second mechanism is the establishment of a mapping into WS-BPEL from SPEM2.0 models.

Measurability. This feature is not directly supported by xSPEM because it is out of scope.

Business rules. This feature is not supported by xSPEM neither SPEM2.0 yet.

Support tools. As mentioned before, authors perform an equivalence of their xSPEM models in Petri-net models to evaluate and validate advantages or disadvantages of xSPEM through experiments. These Petri-nets are executed and simulated within TINA[7]. Once xSPEM process models are validated, this tool allows generating WS-BPEL code to orchestrate process activities.

Validation in real environments. In [88], authors mention that xSPEM has been evaluated in several real R&D projects (IST MODELPLEX and TOPCASED projects). However, it has not been possible to find public references on the validation of the proposal in industrial and real environments.

6.9. eSPEM evaluation

In [90], authors detect four problems (asynchronous events, instance features, planning support, and conguration support) which provokes that many software process cannot be modeled with SPEM2.0 [90]. In this context, they address these shortcomings and present eSPEM [90]: an extension of SPEM for enactable behavior modelling. This proposal substitutes SPEM2.0 behavior interfacing concepts by more fine-grained concepts based on the UML behavior modelling concepts and uses state machines to model process lifecycle. Below, eSPEM is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. This SPML does not introduce new improvements in the semantic richness and expressiveness of the SPEM2.0 notation. Authors just focus on defining enactable behavior modelling in SPEM2.0 as we have previously mentioned. In this context, we are going to consider that eSPEM partially supports this feature following the similar criteria used to evaluate previous SPML.

Understandability. In this case, eSPEM is designed using UML-Profiles to represent its metamodels what has made it possible to take advantage of all the semantic and visual richness of artefacts and basis models of UML. However, this notation is technical and does not include familiar metaphors for non-technical users. In addition, it has not been possible to locate public references and cites in order to evaluate and test metaphors used in eSPEM [90].

Conformity to standards. The eSPEM metamodel is defined conform to MOF and it is also an extension of SPEM2.0 (which is a standard itself).

Granularity. In this case, eSPEM does not define any additional improvement with respect to SPEM2.0 in terms of modularity or granularity. In this context, we consider it appropriate to evaluate this characteristic partially for eSPEM since eSPEM is focused on the executability of SPEM2.0 (following the same criterion previously used).

Executability & Orchestability. As mentioned above, eSPEM is an extension of SPEM, based on UML and focused on fine-grained behavior and lifecycle modeling and supports automated enactment of development processes. In this sense, authors consider the use of UML state machines for some elements of SPEM2.0 (e.g., «WorkProduct» and «TaskDefinition» among others) life-cycle modeling to be a generic, detailed, and perfectly enactable approach. However, eSPEM does not provide mechanisms for executing process models.

Measurability. This feature is not directly supported by eSPEM because it is out of scope.

Business rules. This feature is not supported by eSPEM neither SPEM2.0 yet.

Support tools. Authors have implemented the eSPEM abstract syntax using EMF and the eSPEM concrete syntax using the Graphical Modeling Framework (GMF) [92] as a basic diagram editor.

Validation in real environments. Regarding this feature, we have not found research results that evidence the application of eSPEM within real projects.

6.10. MODAL evaluation

MODAL [93,94] (Modal Oriented Development Application Language) is based on SPEM2.0 and it focuses on process models execution. However, despite the complexity of SPEM2.0 is reproduced here, MODAL aims to clarify the definition of process components to be able to construct and execute processes on-demand. Below, MODAL is evaluated taking into account each feature included in our quality model.

Semantic richness and expressiveness. Regarding this feature, we could consider that MODAL has the same evaluation than SPEM2.0. However, we are going to apply the same criteria used to evaluate previous SPMLs. That is, we are going to consider that MODAL partially supports the semantic richness and expressiveness because this proposal itself does not provide new concepts respect to SPEM2.0 for SPM.

Understandability. Although MODAL is based on SPEM2.0, authors have mainly proposed and described MODAL in a theoretical manner. In [93], it seems that MODAL models are modeled by UML-CD (using Eclipse technologies as we mention later) which has a technical notation and does not include familiar metaphors for non-technical users.

Conformity to standards. The author's proposal is based on SPEM2.0 (which is a standard itself) and also use OCL to add formal semantic constraints.

Granularity. Following the same criteria previously used in the evaluation of other SPMLs in this paper, we consider that MODAL partially supports this feature because it does not define any additional improvement with respect to SPEM2.0 in terms of modularity or granularity.

Executability & Orchestability. In [94], authors improve MODAL to execute process models. For this purpose, firstly, authors define concepts which allows defining behaviors of activities. These new concepts can be instanced in Java Cometa Framework [95] using transformation rules to obtain Cometa executable code from MODAL process model.

Measurability. This feature is not directly supported by MODAL because it is out of scope.

Business rules. This feature is not supported by MODAL neither SPEM2.0.

Support tools. At present, authors are still working on generation of executable process components deployed in a distributed environment using Eclipse technologies (EMF, Kermeta) in order to design and develop their own tool. It has not been possible to find specific references about this tool yet. However, it has been possible to find one reference [94] where authors have designed and developed a process engine prototype to execute and instance MODAL process models.

Validation in real environments. In [93], authors mention that

[7] TINA (TIme petri Net Analyzer) is a academical toolbox for the editing, execute and analysis of Petri-Nets. Website: projects.laas.fr/tina/

Table 1
Summary of the evaluation of model-based approaches.

FEATURES	APPROACHES	PLM$_4$BS	UML$_4$SPM	UML-EWM	UPME	Ferreira	SPEM 2.0	Combemale	xSPEM	eSPEM	MODAL
Semantic richness and expressiveness	Activities	2	2	2	1	2	2	1	1	1	1
	Control flow	2	2	1	1	2	2	1	1	1	1
	Exception flow	0	2	0	0	0	0	0	0	0	0
	Data flow	2	2	2	1	0	2	1	1	1	1
	Resources	1	2	2	1	0	2	1	1	1	1
	TOTAL	1,4	2	1,4	0,8	0,8	1,6	0,8	0,8	0,8	0,8
	Understandability	2	2	0	0	0	2	0	0	0	0
	Conformity to standards	2	2	2	2	2	2	2	2	2	2
	Granularity	2	2	2	2	2	2	1	1	1	1
	Executability & Orchestability	2	2	0	2	2	0	0	2	0	2
	Measurability	1	0	0	0	0	1	0	0	0	0
	Business rules	2	0	0	0	0	0	0	0	0	0
	Support tools	2	2	0	0	0	2	0	1	1	1
	Validation in Real Environments	2	1	0	0	0	2	0	1	0	1

MODAL (extension of SPEM2.0) have been applied in different experiments developing embedded systems. Authors also mention that MODAL has been used in some European project, but we have not been able to find evidences that relation this European project with MODAL.

7. Analysis and discussion

This section aims to discuss and analysis advantages and disadvantages of each studied SPML. Table 1 summaries the evaluation of each ones and it describes the scope and degree of the support of each model-based SPML per feature.

Regarding the degree of support, we evaluate each SPML with respect to quality features with «0», «1» and «2» when a feature is not supported, is partially supported and is fully supported, respectively. While being quite partial and subjective, this evaluation makes it easier for a decider to identify the language answering her expectations. However, it is important to mention that this paper provides a brief description of each evaluated SPML. Therefore, the reader can always consult the original work (using the referenced citations) to obtain more accurate information. This evaluation in detail is more necessary when several SPMLs have the same rate for a feature (e.g., «Conformity to Standard»). Moreover, the final qualification of the «Semantic Richness and Expressiveness» (macro feature) criterion is obtained after applying the arithmetic mean on all micro features of this criterion (i.e., activities, control flow, exception flow, data flow and resources).

Once applied our quality model on SPMLs that have been described in previous section, we summarize main obtained conclusions. Although our model and quality strategy systematically and methodologically allows the comparison of SPMLs, it is important to indicate that it is not possible to determine the best proposal. This choice depends on requirements and needs of each software organization.

Semantic richness and expressiveness (Fig. 3). In this case, most features are supported by each SPML as shown Fig. 3. However, exception handling is a non-supported feature by most proposals under study. UML$_4$SPM is the only one that supports exception handling. Table 1 summarizes semantic richness capabilities of each SPML. According to this summary, UML$_4$SPM followed by SPEM2.0, PLM$_4$BS and UPME are those SPMLs that provide more capabilities regarding the expressiveness requirement while the Ferreira's proposal is the less expressive one because it does not provide many mechanisms for semantic richness and expressiveness (i.e., it does not support exception and data flows, neither resource perspectives.

Understandability (Fig. 4). Most of SPMLs we have considered (i.e., PLM$_4$BS, UML$_4$SPM, UML-EWM, UPME) are based on UML-AD to represent process models except in SPEM2.0 and their extensions (i.e., MODAL, eSPEM, xSPEM). Although UML could be considered a

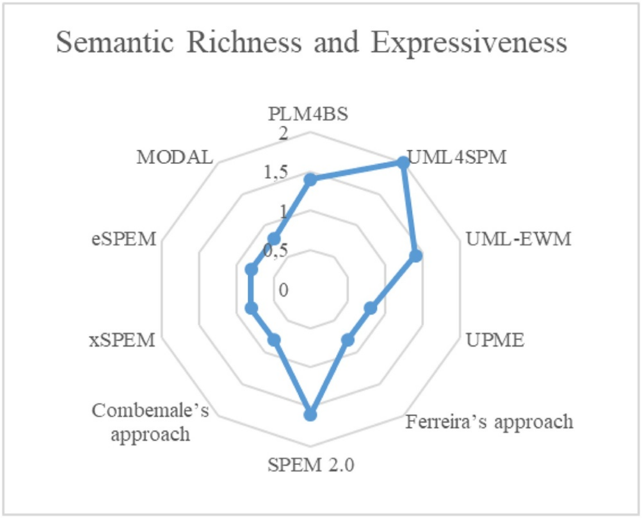

Fig. 3. Summary of the evaluation of «Semantic Richness and Expressiveness» feature.

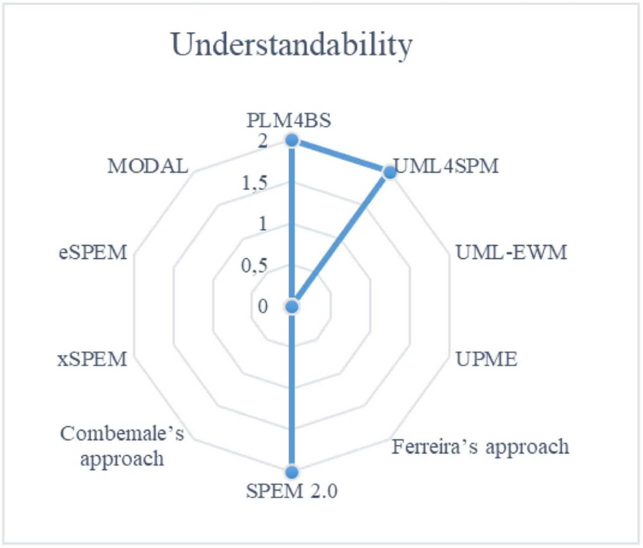

Fig. 4. Summary of the evaluation of «Understandability» feature.

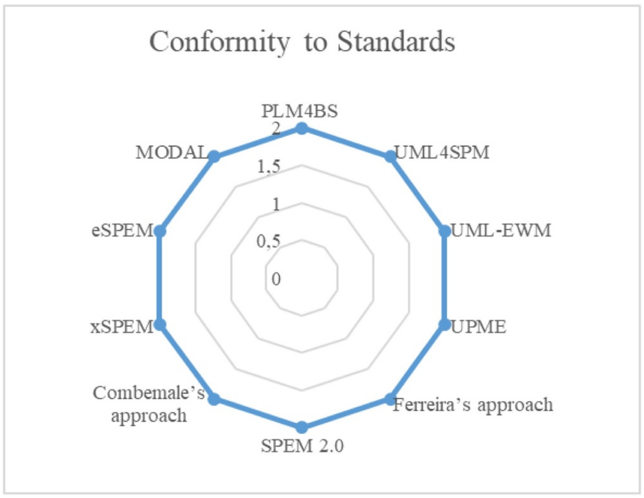

Fig. 5. Summary of the evaluation of «Conformity to Standards» feature.

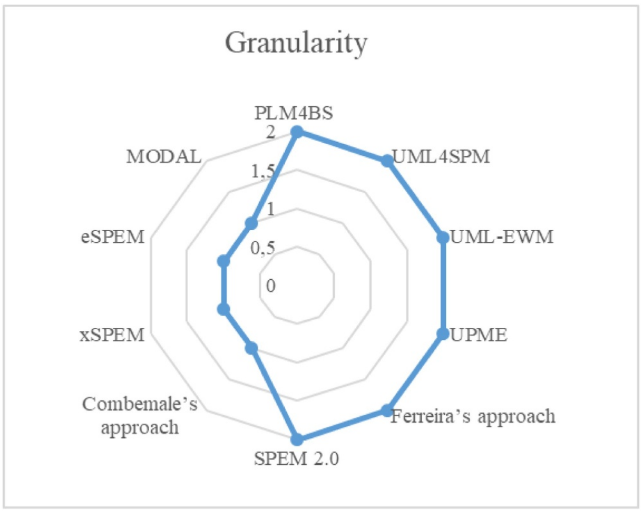

Fig. 6. Summary of the evaluation of «Granularity» feature.

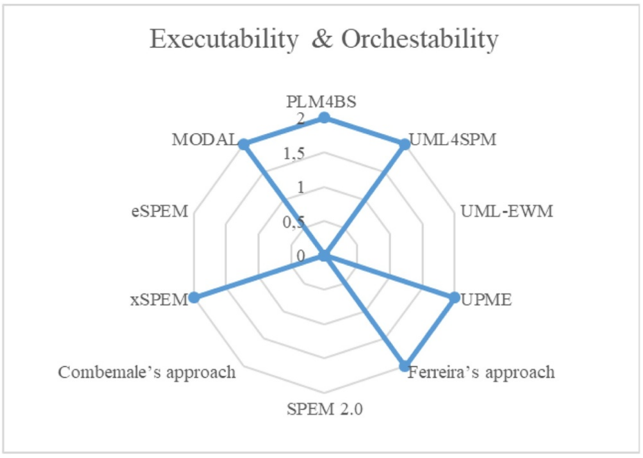

Fig. 7. Summary of «Executability & Orchestability» feature.

technical notation, UML provides semantic and visual richness of artefacts in its basis models. The UPME and Ferreira's proposals respectively use UML-CD and UML-CPD, but they do not define UML-Profiles to provide familiar metaphors when each concept of these proposal is instanced. They do not include familiar metaphors for non-technical users. However, PLM_4BS, UML_4SPM and SPEM2.0 provide familiar metaphors to model their concepts. This justifies the rate 2 in Table 1. Regarding SPEM2.0, this one does not rely on UML-AD for representing processes, but provides a behavioral diagram based on a set of proprietary notations. Finally, SPMLs based on SPEM we have considered (i.e., MODAL, eSPEM, xSPEM, and Combemale's approaches) have rate 0 in our evaluation because each one itself does not provide familiar metaphors when each concept of these proposal is instanced, neither any advantage on understandability respect to SPEM.

Conformity to standards (Fig. 5). Most of SPMLs we have discussed previously, define their own set of concepts to model software processes. Most proposals make their definition based on metamodels and some ones also define UML-Profiles instead of using UML diagrams. Anyway, each SPML conforms to UML in one way or another, which justifies the rate 2 in Table 1. In addition, some proposals also take into account other standards. For example, PLM_4BS has also been defined conforms ISO/IEC 24774: 2010.

Granularity (Fig. 6). Most of SPMLs we have evaluated provide mechanisms for defining patterns or business process models and grouping activities of a process that may be common in a particular business context. These mechanisms for reusing logical blocks are very attractive in design and management tasks because of the potential economic benefits conveyed as time-savings and qualitative improvements. Regarding the remaining SPMLs (i.e., MODAL, eSPEM, xSPEM, and Combemale's approaches), these ones support partially this feature because they are based on do SPEM2.0 and do not define any additional improvement with respect to SPEM2.0 in terms of modularity or granularity.

Executability & Orchestability (Fig. 7). Regarding executability, firstly, PLM_4BS defines transformation rules (based on MOFM2T) to obtain executable code (WS-BPEL and BPEL4People). Secondly, UML_4SPM promotes the reuse of existing WS-BPEL process engines, but it requires a configuration phase. Thirdly, UPME and the Ferreira's proposal are based on academic script language (IDL and Little-JIL, respectively).

Moreover, UML-EWM, SPEM2.0, eSPEM and Combemale's proposals do not provide mechanisms for process execution. Regarding xSPEM and MODAL (SPEM2.0 extensions), firstly, xSPEM proposes two mechanisms: (*i*) process models are transformed into Petri-net models, which are executed and simulated within TINA; and (*ii*) WS-BPEL code is generated from process models. Secondly, MODAL includes mechanisms to enact process models generating executable code (for Java Cometa Framework) from MODAL models. Anyway, it might be possible to consider that only PLM_4BS, xSPEM and UML_4SPM provide generalist mechanisms to execute the process since they are based on WS-BPEL code generation (which is a standard recognized by most BPMS).

Measurability (Fig. 8). Regarding measurability, only PLM_4BS and SPEM2.0 support this feature, but this support is partial. Although both SPML support measure definitions, they do not support the automatic calculation in runtime of these metrics/indicator yet. The rest of evaluated SPMLs do not contemplate this feature.

Business rules (Fig. 9). This feature is fully only supported by PLM_4BS using its «BusinessVar» and «checkBusinessVar» concepts. The latter represents an association class relationship between «ControlElement» metaclass and «BusinessVar» metaclass, but it is necessary to supervise that the type of the «ControlElement» metaclass is «Conditional» (this constraint is controlled by PLM_4BS with OCL). The «checkBusinessVar» also has several properties to config and build business rules (see Section 6.1). The rest of evaluated SPMLs do not contemplate this feature.

Supporting tools (Fig. 10). PLM_4BS and SPEM2.0 provide well-supported tools (both ones are on EA). This justifies the rate 2 in Table 1. Other SPMLs (i.e., UML_4SPM, xSPEM and MODAL approaches) have their own prototypes, but we have been able to access to the

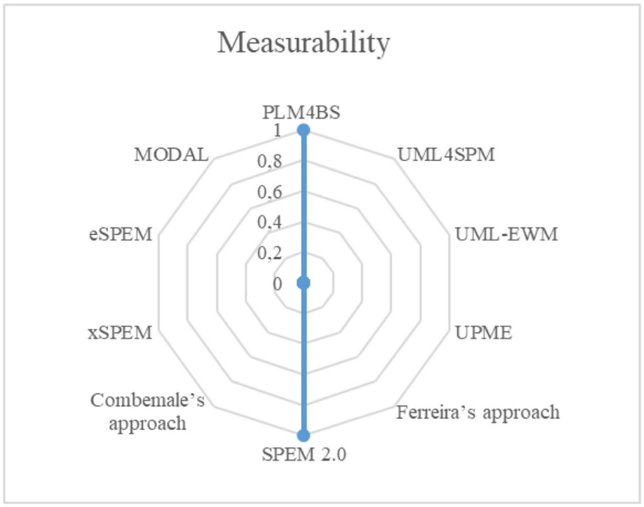

Fig. 8. Summary of the evaluation of «Measurability» feature.

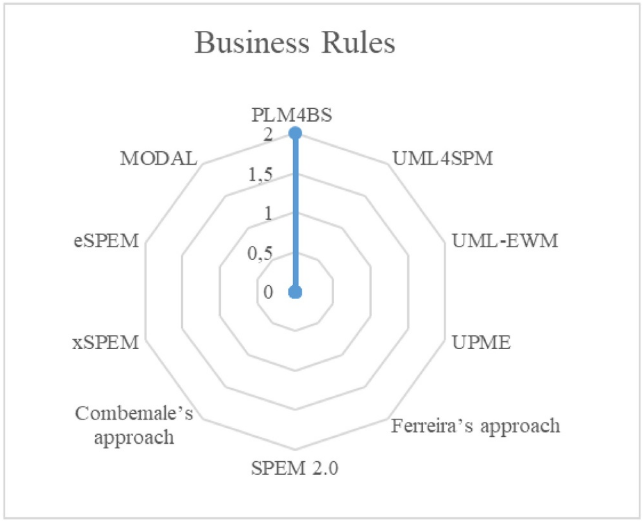

Fig. 9. Summary of the evaluation of «Business Rules» feature.

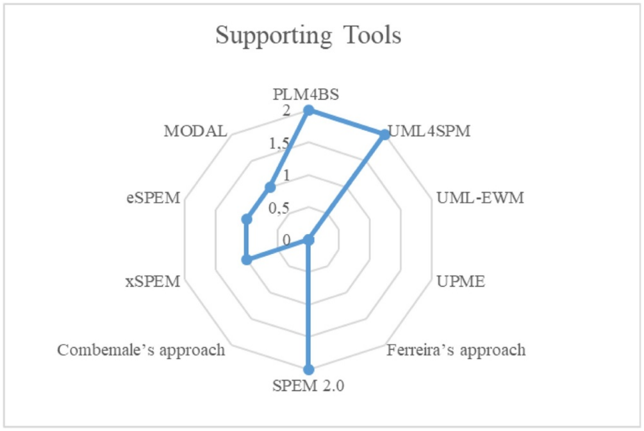

Fig. 10. Summary of the evaluation of «Supporting Tools» feature.

UML$_4$SPM process engines only. Regarding other SPMLs discussed, it has not been possible to find supporting tools.

Validation in real environments (Fig. 11). Five of the ten approaches we discussed (i.e., UML-EWM, UPME, Ferreira's and Combemale's et al. approaches, eSPEM) could be considered as theoretical

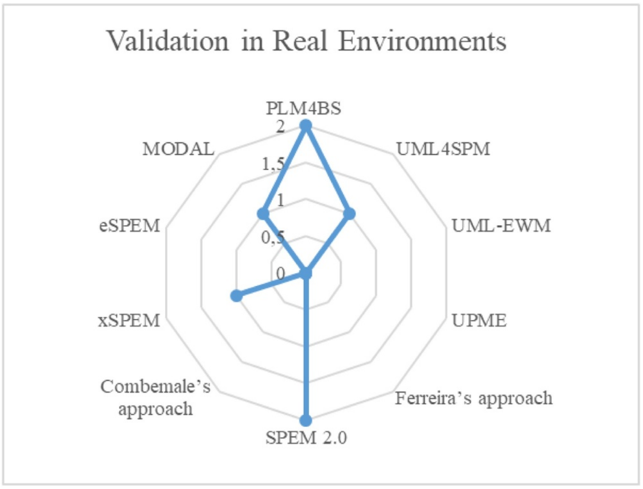

Fig. 11. Summary of the evaluation of «Validation in Real Environments» feature.

approaches because it has not been possible to find public references on the validation of the proposal in industrial and real environments. PLM$_4$BS and SPEM2.0 are the only SPMLs that provide accessible references where their practical application is explained in detail within different business contexts. For example, PLM$_4$BS has been successfully applied in healthcare and business consulting environments, whereas SPEM2.0 is applied in software projects and multi-agent systems, among others. The rest of proposals (UML$_4$SPM, xSPEM, MODAL) have been partially evaluated because it has not been possible to contrast their practical application. In the paper evaluated, authors mention that their proposals have been validated in different R&D projects, but it has not been possible to locate paper or references where that application is described.

8. Conclusions and future works

Over last decades, SPM has been a subject very investigated in Software Engineering and the problem of describing and using software processes has been addressed by many research groups around the world. In fact, different UML-based and model-based proposals have appeared along the years to model software process.

We provide an overview of model-based SPML and we establishe their scope, domain, features or benefits, among other aspects. For this purpose, we use the QuEF framework in order to define a quality model which allows comparing and evaluating this kind of modeling language.

Our quality model frames a set of features for SPML which have been used to evaluate and compare some predominant SPML. These features cover aspects such as semantic richness and expressiveness; understandability; conformity to standards; granularity; executability and orchestrability; measurability; business rules support; support tools; and validation in real environments. This characterization scheme could help to organizations to choose the approach that best fits their expectations and requirements. In this context, we highlighted advantages of each evaluated SPML as well as its drawbacks.

Either way, this paper shows that this line of research is open and booming because of the number of papers found and state-of-the-art studies in software process modelling languages. This is one of the reasons to propose, in this paper, a quality model that allows to objectively evaluate and compare SPMLs.

After carrying out this comparative study and seeing the results shown in Table 1, we can conclude that, there are many model-based proposals for SPM, but it is very difficult to establish with could be the commitment to follow. SPEM2.0 could be the solution, but its complexity and non-executability do not allow it. We have also identified some aspects not considered until now among the most proposals studied, such as validation within enterprise environments, friendly support tools, mechanisms to carry out continuous improvement, mechanisms to establish business rules and elements for software process orchestrating.

However, it can be interesting to emphasize the fact that few proposals have been applied in real environments. In fact, we have not been able to find papers where success stories are described. This situation occurs in most of the proposals mentioned above, but there are exceptions such as SPEM2.0, PLM_4BS and UML_4SPM what allows the approach between the research world and the business world. This situation is very important in order to transfer the scientific knowledge generated in academy to software industry.

As future work, we will continue studying the main features of software process to improve our quality model. Also, we will present new comparative studies to evaluate other possible SPML [96-98]. In addition, we will take our paper as initial point to study what SPML could be the best one for a kind of software company (large, small, or standard-oriented company).

Acknowledgments

This research has been supported by POLOLAS project (TIN2016-76956-C3-2-R) by the of the Spanish Ministry of Economy and Competitiveness.

References

[1] Y.B. Han, J.Y. Sun, G.L Wang, H.F. Li, A cloud-based BPM architecture with user-end distribution of non-compute-intensive activities and sensitive data, J. Comput. Sci. Tech. (2010), https://doi.org/10.1007/s11390-010-9396-z.
[2] W.M.P Van-der-AalstMaking Work Flow, On the application of petri nets to business process management, Application and Theory of Petri Nets, LNCS 2360 (2002), pp. 1–22.
[3] PMI, A Guide to the Project Management Body of Knowledge, (2008) ISBN13: 978-1933890517.
[4] ISO/IEC 9001:2008, Quality Management Systems, Requirements, International Organization for Standardization, 2008.
[5] P. Trkman, The critical success factors of business process management, Int. J. Inf. Manag. 30 (2) (2010) 125–134.
[6] ISO/IEC TR24744:2007, Software and Systems Engineering Lifecycle Management Guidelines for Process Description, (2007).
[7] A. Fuggetta, Software process: a roadmap, Proceedings of the Conference on the Future of Software Engineering, 97 2000, pp. 25–34.
[8] J. Lonchamp, A structured conceptual and terminological framework for software process engineering, Proceedings of 2nd International Conference on the Continuous Software Process Improvement, 1993, pp. 41–53.
[9] K.Z. Zamli, P.A Lee, Taxonomy of process modeling languages, ACS/IEEE International Conference on Computer Systems and Applications, 2001, pp. 435–437.
[10] K.Z. Zamli, N.M Isa, A survey and analysis of process modeling languages, Malays. J. Comput. Sci. 17 (2004) 68–89.
[11] L. García-Borgoñón, M.A. Barcelona, J.A. García-García, M. Alba, M.J. Escalona, Software process modelling languages: a systematic literature review, Inf. Softw. Syst. J. (2013), https://doi.org/10.1016/j.infsof.2013.10.001.
[12] F.J. Domínguez-Mayo, M.J. Escalona, M. Mejías, M. Ross, G Staples, A quality management based on the quality model life cycle, Computer Stand. Interfaces (2012), https://doi.org/10.1016/j.csi.2012.01.004.
[13] S.M.J. Sutton, L.J. Osterweil, The design of a next-generation process language, Software Engineering, Lecture Notes in Computer Science, 1301 1997, pp. 142–158.
[14] R. Bendraou, J.M. Jézéquel, M.P. Gervais, X. Blanc, A comparison of six UML based languages for software process modeling, IEEE Trans. Softw. Eng. 36 (2010) 662–675.
[15] F. Ruiz-González, G. Canfora, Software process: characteristics, technology and environments, SPT - Softw. Process Technol. 5 (2004) 5–10.
[16] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, H.A. Reijers, Imperative versus declarative process modeling languages: an empirical investigation, International Conference on Business Process Management, Springer, Berlin, Heidelberg, 2011, pp. 383–394.
[17] R. Lu, S Sadiq, A survey of comparative business process modeling approaches, Business Information Systems, 2007, pp. 82–94.
[18] M. Zur Muehlen, M. Indulska, Modeling languages for business processes and business rules: a representational analysis, Inf. Syst. 35 (4) (2010) 379–390.
[19] O. Kopp, D. Martin, D. Wutke, F Leymann, On the Choice Between Graph-Based and Block-Structured Business Process Modeling Languages, (2008).
[20] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS 28 (1) (2004) 75–105.
[21] K Peffers, T Tuunanen, M Rothenberger, S Chatterjee, A design science research methodology for information systems research, J. Manag. Inf. Syst. 24 (3) (2007) 45–77.
[22] M. Havey, Essential Business Process Modelling, O'Reilly Media, 2005 ISBN13: 978-0596008437.
[23] WMP Van-der-Aalst, Business process management: a personal view, Bus. Process Manag. J. 10 (2) (2004).
[24] F.J. Domínguez-Mayo, M.J. Escalona, M Mejías, QuEF (quality evaluation framework) for model-driven web methodologies, International Conference on Web Engineering, 2010, pp. 571–575.
[25] J.G. Enríquez, J.M. Sánchez-Begines, F.J. Domínguez-Mayo, J.A. García-García, M.J Escalona, An approach to characterize and evaluate the quality of product lifecycle management software systems, Comput. Stand. Interfaces (2018).
[26] J.G. Enríquez, F.J. Domínguez-Mayo, J.A. García-García, M.J. Escalona, M Risoto, a framework to manage quality of enterprise content management systems, Quality Control and Assurance-An Ancient Greek Term Re-Mastered, InTech, 2017.
[27] M.L. Jaccheri, M. Baldi, M. Divitini, Evaluating the requirements for software process modelling languages and systems, Proc. Conf. Process Support for Distributed Team-Based Software Development, 1999, pp. 570–578.
[28] P. Armenise, S. Bandinelli, C. Ghezzi, A Morzenti, A survey and assessment of software process representation formalisms, Int'l J. Softw. Eng. Knowl. Eng. 3 (3) (1993) 401–426 pp.
[29] C. Schlenoff, A. Knutilla, S Ray, Unified Process Specification Language: Requirements for Modeling Process, Unified Process Specification Language: Requirements for Modeling Process, 5910 (1996) Interagency Report.
[30] ISO 9000:2000, Quality Management Systems (Fundamentals and Vocabulary), (2018) http://www.iso.org Retrieved.
[31] ISO 9001:2008, Quality Management, (2018) http://www.iso.org Retrieved.
[32] ISO 9004:2000, Quality Management Systems (Guidelines for Performance Improvements), (2018) http://www.iso.org Retrieved.
[33] ISO/IEC 9126-1:2001, Software Engineering (Product Quality: Quality Model), (2018) http://www.iso.org Retrieved.
[34] ISO/IEC 25000:2005, Software Product Quality Properties and Evaluation, SQuaRE. (2018) http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35683 Retrieved.
[35] ISO/IEC 20000-1:2011, Information Technology, Service Management: Service Management System Properties, (2018) http://www.iso.org Retrieved.
[36] ISO/IEC 20000-2:2005, Service Management, (2018) http://www.iso.org Retrieved.
[37] ISO/IEC TR 20000-3:2009, Service Management, (2018) http://www.iso.org Retrieved.
[38] ISO/IEC TR 20000-4:2010, Service Management: Process Reference Model, (2018) http://www.iso.org Retrieved.
[39] ISO/IEC TR 20000-5:2010, Service Management: Exemplar Implementation Plan for ISO/IEC 20000-1, (2018) http://www.iso.org Retrieved.
[40] ITIL official site. Retrieved January 2018 from, http://www.itil-officialsite.com.
[41] T Morgan, Business Rules and Information Systems: Aligning IT with Business Goals, Addison-Wesley, 2002 ISBN 0-201-74391-4.
[42] B. Curtis, M.I. Kellner, J. Over, Process Modeling, Commun. ACM 35 (9) (1992) 75–90.
[43] J Krogstie, Perspectives to Process Modeling, in: M. Glykas (Ed.), Business Process Management. Studies in Computational Intelligence, 2013 https://doi.org/10.1007/978-3-642-28409-0_1.
[44] La Rosa, Marcello, ter Hofstede, H.M. Arthur, Wohed, Petia, Reijers, A. Hajo, Mendling, Jan, van der Aalst, M.P. Wil, Managing process model complexity via concrete syntax modifications, IEEE Trans. Ind. Inf. 7 (2) (2011).
[45] A.A. Abdul, G.K. Tieng Wei, G.M. Muketha, W.P Wen, Complexity metrics for measuring the understandability and maintainability of business process models using goal-question-metric, Int. J. Comput. Sci. Netw. Secur. 8 (5) (2008) 219–225.
[46] J. Mendling, H.A. Reijers, J Cardoso, What makes process models understandable? Proc. of BPM, LNCS, 4714 Springer, 2007, pp. 48–63.
[47] J. Mendling, M. Strembeck, Influence factors of understanding business process models, Proc. of BIS, LNBIP, 2008, pp. 142–153.
[48] ISO/IEC. ISO/IEC TR 24744:2007, Software and Systems Engineering Lifecycle Management Guidelines for Process Description, (2007).
[49] OMG, Meta Object Facility (MOF™) Core, (2011) http://www.omg.org/spec/MOF/ Last accessed 11/2017.
[50] B.P. Meier, S. Schnall, N. Schwarz, J.A Bargh, Embodiment in social psychology, Top. Cogn. Sci. 4 (4) (2012) 705–716.
[51] B. Frakes, Kang Kyo, Software reuse research: status and future, IEEE Trans. Softw. Eng. 31 (7) (2005) 529–536.
[52] Y. Ye, G Fischer, Supporting reuse by delivering task-relevant and personalized information, Proceedings of 24th International Conference on Software Engineering, 2002, pp. 513–523.
[53] O. Holschke, J. Rake, O. Levina, Granularity as a cognitive factor in the effectiveness of business process model reuse, Business Process Management. BPM, 2009 https://doi.org/10.1007/978-3-642-03848-8_17 2009.
[54] H.A. Reijers, J Mendling, Modularity in process models: review and effects, BPM, 5240 2008, pp. 20–35.

[55] M. Papazoglou, P. Ribbers, E-Business: Organizational and Technical Foundations, John Wiley & Sons, 2006 EditorISBN-13: 978-0470843765.
[56] G. Pedraza, J.C Estublier, Distributed orchestration versus choreography: the FOCAS approach. trustworthy software development processes, Lect. Notes Comput. Sci. 5543 (2009) 75–86 Volume.
[57] A. Meidan, J.A. García-García, M.J. Escalona, I Ramos, A survey on business processes management suites, Comput. Stand. Interfaces 51 (2016) 71–86 Volume.
[58] A Kronz, Managing of process key performance indicators as part of the aris methodology, Corp. Perform. Manag., (2006) 31–44.
[59] W. Wang, M. Indulska, S.W Sadiq, Cognitive efforts in using integrated models of business processes and rules, 2016, pp. 33–40.
[60] Van Lamsweerde, Axel. Requirements Engineering: From System Goals to UML Models to Software 10 John Wiley & Sons, Chichester, UK, 2009 Vol.
[61] DC Schmidt, Model-Driven Engineering, IEEE Comput. Comput. Soc. 39 (2) (2006) 25–31 vol.
[62] T.E. Vos, B. Marin, M.J. Escalona, A. Marchetto, A methodological framework for evaluating software testing techniques and tools, 12th International Conference on Quality Software, 2012, pp. 230–239.
[63] P. Tomas, M.J. Escalona, M Mejias, Open source tools for measuring the internal quality of Java software products. A survey, Comput. Stand. Interfaces 36 (1) (2013) 244–255.
[64] M Lang, An analysis of model-driven web engineering methodologies, Int. J. Innov. Comput. Inf. Control (2012).
[65] J.A. García-García, L. García-Borgoñón, M.J. Escalona, M Mejías, A model-based solution for process modeling in practice environments: PLM4BS, J. Softw., (2018), https://doi.org/10.1002/smr.1982.
[66] J.A. Garcia-Garcia, A. Meidan, A. Vázquez, M Mejias, A model-driven proposal to execute and Orchestrate processes: PLM4BS. Software process improvement and capability determination, 17th International Conference, 2017 https://doi.org/10.1007/978-3-319-67383-7_16.
[67] ISO/IEC 19507:2012. Object Constraint Language (OCL), formal/2012-05-09, 2012.
[68] OMG, Query/View/Transformation. Website: http://www.omg.org/spec/QVT/1.0/, (2018) Retrieved.
[69] OMG, MOF Model to Text Transformation Language (MOFM2T). Website http://www.omg.org/spec/MOFM2T/1.0/, (2018) Retrieved.
[70] OASIS, Web Services Business Process Execution Language Version 2.0, Organization for the Advancement of Structured Information Standards, 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
[71] OASIS, WS-BPEL Extension for People (BPEL4People) Specification Version 1.1, Organization for the Advancement of Structured Information Standards, 2010, http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html.
[72] www.sparxsystems.com.au, (2018) Retrieved.
[73] J.A. Garcia-Garcia, M.J. Escalona, A. Martínez-García, C. Parra, T. Wojdyński, Clinical process management: a model-driven & tool-based proposal, Information Systems Development: Transforming Healthcare through Information Systems, (2015) ISBN: 978-962-442-393-8.
[74] J.A. Garcia-Garcia, J.G. Enriquez, L. Garcia-Borgoñon, C. Arevalo, E. Morillo, A MDE-based framework to improve the process management: the EMPOWER project, IEEE 15th International Conference on Industrial Informatics, 2017, pp. 553–558 ISBN: 978-1-5386-0836-4.
[75] R. Bendraou, G. Marie-Pierre, B. Xavier, UML4SPM: a UML2.0-based metamodel for software process modelling, International Conference on Model Driven Engineering Languages and Systems, 2005.
[76] R. Bendraou, G. Marie-Pierre, B. Xavier, UML4SPM: an executable software process modeling language providing high-level abstractions, Enterprise Distributed Object Computing Conference. 10th IEEE International. IEEE, 2006.
[77] R. Bendraou, A. Sadovykh, M. Gervais, Software process modeling and execution: the UML4SPM to WS-BPEL approach, Proceedings of 33rd Conference on Software Engineering and Advanced Applications, 2007, pp. 314–321.
[78] N. Debnath, D. Riesco, G. Montejano, M.P. Cota, J. Baltasar Garcia, Perez-Schoeld, D. Romero, M. Uva, Supporting the SPEM with a UML extended workflow metamodel, IEEE International Conference on Computer Systems and Applications, 2006 2006, pp. 1151–1154.
[79] D. Riesco, M. Uva, E. Acosta, N. Debnath, G. Montejano, Including workflow concurrent modeling in an extension of the uml activity diagram metamodel, International Conference on Computer Systems and Applications, 2003.
[80] M. Wu, G. Li, J. Ying, H. Yan, A metamodel approach to software process modelling based on UML extension, IEEE International Conference on Systems, Man and Cybernetics, 6 2006, pp. 4508–4512.
[81] Ferreira, A.L., Machado, R.J., Paulk, M.C. An approach to software process design and implementation using transition rules. 37th Software Engineering and Advanced Applications Conference, pp. 330–333, 2011.
[82] A. Wise, Little-JIL 1.5 Language Report, Department of Computer Science, University of Massachusetts, Amherst, MA, 2006 UM-CS-2006-51.
[83] OMG, SPEM 2.0, Software & Systems Process Engineering Metamodel Specification, (2008) http://www.omg.org/spec/SPEM/.
[84] W. Dahhane, J. Berrich, T. Bouchentouf, M Rahmoun, SEMAT Essence's Kernel applied to O-MaSE, 5th International Conference on Multimedia Computing and Systems, 2016, pp. 799–804.
[85] JA.H. Alegria, M.C. Bastarrica, A Bergel, Analyzing the Scrum process model with AVISPA, XXIX International Conference of the, IEEE, 2010, pp. 60–65.
[86] B. Combemale, X. Cregut, A. Caplain, B. Coulette, Towards a rigorous process modelling with SPEM, 8th International Conference on Enterprise Information Systems, 2006, pp. 530–533.
[87] OMG. SPEM1.1, Software Process Engineering Metamodel, formal/2002-11-14, 2002.
[88] R. Bendraou, B. Combemale, X. Cregut, M Gervais, Definition of an executable SPEM 2.0, Software Engineering Conference, APSEC 2007 14th Asia-Pacific, 2009, pp. 390–397.
[89] Kermeta, Website https://marketplace.eclipse.org/content/kermeta, (2018) Retrieved.
[90] R. Ellner, S. Al-Hilank, J. Drexler, M. Jung, D. Kips, M Philippsen, Espem a SPEM extension for enactable behavior modelling, Model. Found. Appl. 6138 (2010) 116–131 vol.
[91] https://www.eclipse.org/modeling/emf/, (2018) Retrieved.
[92] http://www.eclipse.org/modeling/gmp/, (2018) Retrieved.
[93] A. Koudri, J.Modal Champeau, A SPEM extension to improve co-design process models, Lect. Notes Comput. Sci. 6195 (2010) 248–259 vol.pp..
[94] P.Y. Pillain, J. Champeau, H.N Tran, Towards an enactment mechanism for MODAL process models, First Workshop on Process-based approaches for Model-Driven Engineering (PMDE), 2011, p. 33 pp..
[95] OMG, UML Profile for MARTE, Beta 1, (2007) . Technical Report ptc/07-08-04.
[96] E.M. Schön, J. Thomaschewski, M.J. Escalona, Agile requirements engineering: a systematic literature review, Comput. Stand. Interfaces 49 (2017) 79–91.
[97] F. Golra, F. Dagnat, Component-oriented multi-metamodel process modeling framework (CoMProM), First Workshop on Process-based approaches for Model-Driven Engineering (PMDE 2011), 2011, p. 44.
[98] F. Aoussat, M. Oussalah, M.A. Nacer, SPEM extension with software process architectural concepts, Proceedings of the International Computer Software and Applications Conference, Munich, Germany, 2011, pp. 215–223 pp..