A model-based solution for process modeling in practice environments: PLM$_4$BS

Julián Alberto García-García | Laura García-Borgoñón | María José Escalona | Manuel Mejías

Escuela Técnica Superior de Ingeniería Informática Web Engineering and Early Testing (IWT2) Group, University of Seville, Sevilla, Spain

Correspondence
Julián Alberto García-García, Escuela Técnica Superior de Ingeniería Informática, Web Engineering and Early Testing (IWT2) Group, University of Seville, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain.
Email: julian.garcia@iwt2.org

Funding information
Spanish Ministry of Economy and Competitiveness, Grant/Award Numbers: TIN2015-71938-REDT and TIN2016-76956-C3-2-R

Abstract
Today's world economic situation is ruled by issues such as reducing cost, improving quality, maximizing profit, and improving and optimizing processes at organizations. In this context, business process management can be an essential strategy, but it is not usually consolidated at software organizations because software process properties involve a complex business process management application on software lifecycle. Consequently, software organizations often focus on Software Process Modeling (SPM), and each involved role performs process execution and orchestration independently and manually. This fact makes software processes maintenance, monitoring, and measurement become difficult tasks. This paper proposes a model-based approach for SPM taking into account concepts related to process execution, orchestration, and monitoring. It is framed into a model-driven engineering-based and tool-based framework: Process Lifecycle Management for Business Software (PLM$_4$BS). We present a SPM metamodel and its concrete syntax (through Unified Modeling Language profiles) that lays the foundation for extending PLM$_4$BS. Its underlying metamodel allows managing processes automatically. Furthermore, PLM$_4$BS improves current state-of-the-art proposals in 6 dimensions: expressiveness, understandability, granularity, measurability, orchestrability, and business variables and rules. Also, PLM$_4$BS has been evaluated in a multiple-case study, in which the 6 mentioned dimensions were already validated.

KEYWORDS
business process management, business process modeling, model-based approach, PLM$_4$BS, software engineering

1 | INTRODUCTION

Today's world economic situation is ruled by issues such as globalization, which involves relocation of companies, constant search for lower costs to maximize profits by continuous mergers and acquisitions, and constant motivation for improving quality and optimizing all Business Processes (BP). Thus, organizations should strengthen their mechanisms to be more competitive in this economic context.

In light of this, this paper starts with one general hypothesis: *At any organization, a proper Business Process Management (BPM) can contribute to achieve the organizational objectives outlined above*. For this purpose, organizations should implement their BPs with mature management models to maintain and/or achieve an edge over their competitors using different technologies such as cloud solutions,[1] among others.

Business process management could be considered as a management strategy that includes methods, techniques, and tools to support the BP lifecycle, which includes design, enactment, management, and analysis of operational BPs.[2] Business process management aims to reduce costs and improve BPs (through a cycle of continuous improvement) in many organizations.[3] In fact, some studies such as ISO/IEC.ISO 9001:2008[4] conclude that the reasons for adopting BPM can be grouped into 3 main needs: (i) understand and assimilate the intrinsic knowledge of business processes, (ii) know the employees' performance during the execution processes, and (iii) monitor and measure processes. Controlling these needs improves return on investment parameter through reducing production costs.[5]

There are also 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.[6-8] Such parameters have been followed by a large number of companies in all areas of business.

However, BPM in software process engineering context is not as simple as it seems to be because of the inherent properties of software processes. Martínez-Ruiz et al identify and describe properties that characterize software processes in comparison with other processes (eg, industrial processes).[7] These characteristics are mainly the following: (i) They are constantly evolving, they usually incorporate new lifecycles and new technologies, they frequently comprise several iterations, and they often produce different software product versions; (ii) they are complex because they are strongly influenced by many unpredictable circumstances and numerous 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.

The reasons above enumerated have led software companies to usually focus on defining their processes, but each involved role performs orchestration (ie, centralized and coordinated management of events during the process execution) and execution processes in an independent and manual way. For this reason, software processes maintenance, evolution, monitoring, and measurement (among other aspects) become complex and hard tasks.

Anyway, Software Process Modeling (SPM) is a well-known topic in the software engineering research that has been studied during the last 20 years. Different work proposals on this purpose have appeared along the years, and many research studies have identified them (eg, García-Borgoñón et al[9] and Zamli et al,[10] among others). These above-cited references analyze the most recent proposals related to SPM and identifies a great amount of them that use different techniques, such cash petri-nets, models, or programming languages (the motivation for developing these kinds of languages and the problems they address are described in García-Borgoñón et al,[9] Zamli and Isa,[10] and Zamli and Lee[11]). This fact confirms that there is no ideal language for process modeling. After analyzing these kinds of SPM, those model-based SPMs have been considered as the path to follow.[9]

However, at present, there are recurring problems in most of model-based SPM (such as execution, measuring, and choreography). In addition, some little proposals offer mechanisms to execute the process into ad hoc systems. That makes harder their application in real environments because companies already use specific process engines. It is interesting to underline that none of these proposals mention mechanisms to support the process orchestration (see Section 7 for more details). For these reason, we have proposed a new SPM language. The fact of making a new contribution allows that our proposal is independent of other SPMLs that could be understood to be more transversal/universal. In addition, our choice also allows to simplify and focus our SPML in the areas of interest, without adherences or features that do not add value (these features could be included when using an existing proposal).

The present study aims to (i) lay the theoretical foundations of a framework based on the Model-Driven Engineering (MDE) paradigm to apply BPM in real environments and (ii) propose a flexible, easily extensible (if necessary), practical, but highly semantic, model-based proposal for SPM. Our main objective is to propose a SPML which is as simple as possible (but complete) as well as easy to learn (to decrease the learning curve of users), use, and understand by most people and which is not cumbersome to be learnt. These features have allowed us the rapid application of our proposal in real environments, and, especially, with non-Information and Communication Technology (ICT) users.

Anyway, we therefore ask ourselves the following question: Which of the elements in large PMLs (such as Business Process Modeling Notation (BPMN)) are most used in practice and how frequently? The studies presented in Muehlen and Recker[12] and Recker et al[13] provide valuable answers to this question. In Recker,[14] authors show the frequency distribution of the individual BPMN constructs ranked by overall frequency. It can be noticed only 4 constructs being common to more than 50% of the diagrams.[15] Also, studies demonstrate that the level of errors in process modeling with BPMN is still very high.[16] This justification is valid also for any other language; it is not something specific or concrete of BPMN.

In this context, our objective is not to propose another general purpose SPML that includes all the aspects that other languages already have (such as Software Process Engineering Metamodel (SPEM) or BPMN) to software process modeling. Our objective is to propose a language simple, clear, and concise.

The framework we propose is named process lifecycle management for business software (PLM_4BS) and, currently, it is a tool-based and MDE-based framework for *(a) process modeling* and *(b) process execution and orchestration* in real business environments. Nonetheless, this paper will focus on presenting how PLM_4BS supports process modeling. The support of process execution and orchestration is published in García-García et al.[17] Thus, PLM_4BS opens lots of opportunities to improve MDE-based process management (eg, simulation, monitoring, and continuous improvement of processes, among other aspects). These elements are being already addressed in different lines of work.

Moreover, while designing PLM_4BS, we also kept in mind its validation in real environments. In this sense, our research group, IWT2[*] (Web Engineering and Early Testing) of the University of Seville (Spain), has and is having extensive experience in carrying through R&D projects in liaison with important public and private companies in Spain.[†] This context offers us opportunities to apply and validate PLM_4BS. For instance, PLM_4BS has been validated in 2 ways. On the one hand, the theoretical basis of PLM_4BS has enabled the development of EMPOWER platform,[18] which is the result of a research and innovation project carried out by Servinform Company[‡] together with our research group. This platform

[*]IWT2 research group. Website: http://www.iwt2.org

[†]IWT2 research group. Website (section of «Projects»): http://www.iwt2.org/en/projects/

integrates the theoretical foundations of PLM_4BS to apply advanced MDE techniques to improve the application of BPM in any business environments. It must be pointed out that it has been successfully applied to real situations (Andalusian Health Service, Department of Andalusian Public Works, and Servinform Company, among others). On the other hand, the usefulness of PLM_4BS has been validated during the definition and application of NDTQ-Framework (Navigational Development Techniques and Quality Framework)[§],[19] which offers a suitable and a global process-based solution for real applications of Navigational Development Techniques (NDT) methodology[20] in R&D projects. Thanks to PLM_4BS, NDTQ Framework has been improved to include monitoring elements and execution information in NDT processes. These real projects have been used to evaluate PLM_4BS, such as is presented in Section 6.

Finally, after this introduction, this paper is organized as follows: Section 2 describes our research question, which is followed by the research method that tries to approach it. Then, Section 3 explains the theoretical principles of PLM_4BS and its MDE-based architecture, which is also centered on a formal process improvement lifecycle. Section 4 defines a model-based solution for SPM, its semantic, and its concrete syntax using a Unified Modeling Language (UML) profile. Later, Section 5 analyzes how PLM_4BS has been technologically designed and developed. Subsequently, Section 6 presents the evaluation of our approach, and Section 7 discusses the related work. Finally, Section 8 draws the conclusions from this research and outlines our future lines of work.

2 | RESEARCH QUESTION AND METHODS

After presenting the context of this paper and our motivating scenario in the previous section, it is important to establish our hypothesis and research question to define our proposal rigorously. Therefore, we formulate the following research questions: "*Is it possible to bring BPM towards software organizations and successfully combine MDE and BPM?*" and "*Is it possible to define a SPM language that contains concepts related to processes execution, orchestration, and monitoring?*."

To answer these questions, this paper proposes a solution following methodological principles, as suggested by Hevner et al.[21] Specifically, this paper follows the Design Science Research Methodology[22] that establishes 5 phases:

2.1 | Phase 1: problem identification and motivation

We approach this phase from 2 different points of view. On the one hand, we carry out a thorough systematic literature review (SLR) to collect existing proposals related to our research question (ie, SPM proposals that have been suggested in the last decade). On the other hand, we propose a characterization scheme to assess the advantages and disadvantages of such model-based proposals.

Firstly, our SLR is described in García-Borgoñón et al.[9] The conclusion we reach is that, at present, there is a great amount of proposals that provide Process Modeling Languages (PMLs) to define software processes using different techniques such as petri-nets, models, or programming languages. This makes us reflect on the idea that there is no perfect language for process modeling. Each language has strong and weak points depending on the requirements of each organization at a specific time (no modeling language is better than another at all times). In fact, the main challenge for an organization is to choose the best and most suitable PML to meet its requirements. This is an important decision, and it should be critical because once a specific PML is chosen, a very close dependency is created. If the company requires using another modeling language, it is necessary to migrate the notation of each process model, that constitutes a complex and costly task in itself.

Secondly, once the large number of proposals is known (particularly, model-based proposals, because it is the topic related to our research question), it is interesting to establish a characterization scheme that allows us to evaluate and compare model-based proposals objectively. This scheme is compulsory because preliminary conclusions from our previous SLR confirm that many model-based PMLs do not provide mechanisms to support process execution, orchestration, and monitoring. Section 7 describes this characterization scheme and how it has been working on the main model-based proposals found out in literature.

2.2 | Phase 2: objective of the solution

This phase aims to develop a proposal with its own graphical notation for SPM, and it provides mechanisms to support process execution, orchestration, and monitoring. In this sense, we propose a flexible, easily extensible, scalable, practical, but highly semantic, model-based language to software modeling. Our PML is the basis of the PLM_4BS framework. Process lifecycle management for business software is a tool-based and MDE-based framework for (i) *process modeling* and (ii) process execution and orchestration in real enterprise environments.

However, because of this paper's limited extension, it just focuses on presenting how PLM_4BS supports process modeling, as how it supports process execution and orchestration has already been published in García-García et al.[17] Process lifecycle management for business software

[‡]Servinform Company. Website: http://www.servinform.es/

[§]NDTQ Framework[19] defines around 30 processes to apply NDT methodology in real software environments. These processes are grouped into six categories: software processes related to software development, testing, maintenance, quality management, security management, and software project management.

opens lots of opportunities to improve MDE-based process management (eg, simulation, monitoring, and continuous improvement of processes, among other aspects) that are already being addressed in different future lines of works).

Finally, according to the results of the previous phase (ie, *Phase 1: problem identification and motivation*), our PML should be expressive, traceable to business process, amenable to automation, precise, platform-independent, and comprehensible for all stakeholders. These features are crucial to successfully apply PLM$_4$BS in real environments.

2.3 | Phase 3: design and development

This phase involves 2 steps: first, designing a novel SPM metamodel and its associated semantic constraints as well as MDE-based theoretical bases of PLM$_4$BS framework and secondly, developing PLM$_4$BS suite, which is the tool to back up the application of our SPM metamodel and to automate parts of PLM$_4$BS framework.

2.4 | Phase 4: demonstration

This phase comprises the development of a PLM$_4$BS suite software prototype that effectively shows that MDE mechanisms developed in PLM$_4$BS are amenable to friendly defined processes. In addition, the development of PLM$_4$BS suite to define processes is the key point to identify other important features like process simulation, monitoring or execution, and orchestration, among others. Finally, it is worth mentioning that our software solution is platform-independent because it is developed using standards such as UML.

2.5 | Phase 5: evaluation

We carry out different multiple-case studies. These enable researchers to conduct an evaluation of PLM$_4$BS based on 3 study cases (ie, 3 real projects with different companies) and taking into account several dimensions. In addition, these dimensions also allow us to compare PLM$_4$BS with other SPMLs in Section 7.

However, before proposing our dimensions, we have taken into account other comparative studies to establish appropriate criteria. Many requirements related to PMLs have been identified in the literature.[11,23-26] There is variability of criteria: from facilitating human understanding, to analyzing processes, or to providing an automated execution support. After taking into account this references, we have selected most predominant and common criteria (expressiveness, understandability, granularity), and we have included other ones (measurability, orchestrability, and business rule definition) as core dimensions. In this context, we propose to use 6 dimensions whose mean is the following:

- Expressiveness. It establishes how much modeling constructs (human activities, machine activities, conditional, branches, parallel branches, exception handling, products, and roles) are offered by a specific SPML.
- Understandability. One of the challenges in the field of BPM concerns the management of process models and its understandability,[27] which is considered, in this paper, as the ability to easily manage, read process flow without additional explanation. This evaluation is based on the use of familiar metaphors, number of symbols, number of kind of structures used, and unstructured statements.
- Granularity. Many research studies demonstrate that reuse of concepts is a basic human psychological mechanism,[28] which is also present within software and technological organizations[29] where the application of reuse mechanisms is very attractive in design and management tasks. The reuse of proven conceptual models, such as process models, may provide an efficient means for successfully exerting that control.[30]
- Executability and orchestability. Although BPM has been successfully applied to many kinds of organizations, there are difficulties in many companies when they have to be executed.[31] Therefore, this feature seems 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.
- Measurability. Once the process is deployed and is being executed into a process engine, it is time to evaluate its effectiveness. This evaluation provides a granular view of the overall productivity of each process, and it is based on the definition of key performance indicators (KPIs). However, collecting and defining process-related KPIs are the first prerequisites for holistic process management and form the basis for consistent and continuous process optimization.[32] For this purposes, the model-based PML should support the definition of indicators or metrics that help the process engineer to measure processes.
- Business rules. Process models mainly focus on the modeling of activities without taking into account aspects very important to support for the remaining stages of the BPM lifecycle. One of these forgotten aspects is the definition of business rules when process engineer models his/her processes.[33] In practice, business rules play an indispensable role in the design and implementation of process models because they could be extracted from laws, policies, procedures, etc. Business rules can be represented in an integrated manner (ie, using graphical mechanisms in a process model; such as graphical links or text annotations) or in a separated manner (ie, 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.[33]

A fact that reinforces the positive feedback obtained from our evaluation is that we have recently carried out a R&D project[34] (named EMPOWER) whose goal is to deploy PLM$_4$BS and its supporting tool in production to improve process management using MDE paradigm.

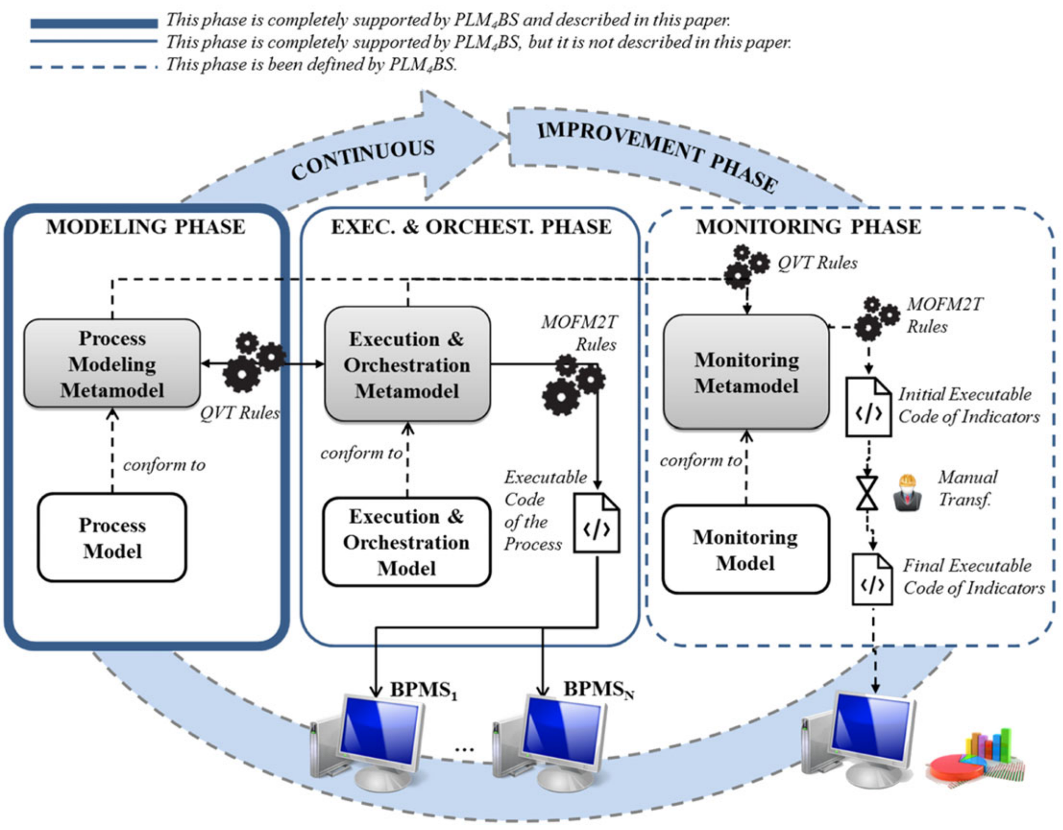

This phase is completely supported by PLM4BS and described in this paper.
This phase is completely supported by PLM4BS, but it is not described in this paper.
This phase is been defined by PLM4BS.
CONTINUOUS
IMPROVEMENT PHASE
MODELING PHASE
EXEC. & ORCHEST. PHASE
MONITORING PHASE
Process Modeling Metamodel
QVT Rules
conform to
Process Model
Execution & Orchestration Metamodel
MOFM2T Rules
conform to
Execution & Orchestration Model
Executable Code of the Process
QVT Rules
Monitoring Metamodel
MOFM2T Rules
Initial Executable Code of Indicators
conform to
Manual Transf.
Monitoring Model
Final Executable Code of Indicators
BPMS1
BPMSN

FIGURE 1 Theoretical architecture of process lifecycle management for business software based on model-driven engineering and a process improvement lifecycle

Thanks to this project, PLM$_4$BS has been successfully applied to numerous real situations developed in different bodies such as Andalusian Health Service, Department of Andalusian public works, and Servinform company, among others).

3 | THEORETICAL FOUNDATIONS OF PLM$_4$BS

Business process management can be considered a process-oriented management strategy with a clear multidisciplinary nature that has conditioned the appearance of different views, definitions, and perspectives of BP lifecycles and their continuous improvement.[35-38] In this case, process orchestration[¶] is an aspect that has not been clearly defined, although the trend has been to simplify the definition and number of phases so as to facilitate their application to different contexts. Nevertheless, this aspect is relevant because over the last decade, more companies (especially, software companies) use different interconnected tools to run their business processes.[39] Therefore, it is deemed necessary and important to support, in a theoretical way, process orchestration in BPM continuous improvement lifecycle.

Considering the aforementioned arguments, the architecture of PLM$_4$BS is based on BP lifecycle which comprises 4 phases[#] as follows: (i) modeling, (ii) execution and orchestration, (iii) monitoring, and (iv) continuous improvement. Such phases are also integrated into PLM$_4$BS using MDE to take advantage of the MDE benefits.[40] Figure 1 conceptually shows the lifecycle of PLM$_4$BS as well as the phases that are completely and incompletely defined at this moment. The former is represented using continuous line (ie, (i) modeling and (ii) execution and orchestration), whereas the latter are represented using dashed lines (ie, (iii) monitoring and (iv) continuous improvement). However, it is important to state again that this paper is focused on describing the first phase of the lifecycle. Even though the second phase (execution and orchestration) is already fully supported by PLM$_4$BS,[17] it is not going to be explained in this paper because it may become too extensive. The third and fourth phases constitute future work in which we are working at present. Below, the phases of our process improvement lifecycle are further described:

1. Modeling phase. At this phase, the process engineer can model and describe his/her BP in a structured manner (ie, identifying roles, activities, deliverables generated during the process, or deliverables required, for instance). Process lifecycle management for business software

[¶]Process orchestration is understood in this paper as the centralized coordination of events that allows conditioning the evolution and execution of process flow.

[#]This paper offers a particular view of a lifecycle that is complementary to the others.[35-38]

proposes a metamodel to support this phase. This metamodel is explained in Section 4 and takes the form of a Meta-Object Facility (MOF) compliant** metamodel. It also follows the guidelines of ISO/IEC TR 24744 standard††.[8]

2. Execution and orchestration phase.[17] Nowadays, this phase is critical and essential because companies are being driven by the need to extensively automate their processes to execute and orchestrate them with enterprise management systems. At this moment, the process defined by the process engineer at previous phase must be executed and orchestrated in a process engine (known as business process management suite (BPMS)[41]). For this purpose, the process engineer must detail execution parameters as well as parameters for the communication and integration with external systems.

Nevertheless, most BPMSs have inflexible PMLs, so these tools do not allow executing processes that have been defined conformed to another PML.[41] To solve this situation, PLM$_4$BS provides MDE mechanisms to enable the execution of processes. Taking into account this procedure, the process engineer can detail execution and orchestration parameters.

On the one hand, PLM$_4$BS defines an execution and orchestration metamodel. It also comes in the form of a MOF compliant metamodel that complies with ISO/IEC TR 24744, too. This metamodel allows defining execution parameters to run the process into a BPMS, and it is systematically obtained using model-to-model (M2M) transformation rules from the process modeling metamodel. These rules have been also formalized using Query/View/Transformation (QVT).[42]

On the other hand, a systematic and automatic transformation protocol has been defined to generate executable code from the cited execution metamodel. This protocol is based on model-to-text (M2T) transformation rules using MOFM2T.[34] The process engineer should be able to instance and run processes into any process engine when the process execution context is defined.

The execution metamodel and our transformation protocol will not be explained here, because they are out of the scope of this paper and it would become too extensive. However, it is possible to find further information in García-García et al.[17]

3. Monitoring phase. Once the process is deployed into a BPMS, it is time to evaluate its effectiveness. This evaluation provides a granular view of the overall productivity of each process, and it is based on the definition of key performance indicators

In this case, PLM$_4$BS provides 2 types of mechanisms to support this phase. Firstly, the process modeling metamodel includes concepts (such as metric and indicator) that help the process engineer to measure processes. Indicators are defined during the modeling phase. Secondly, and after identifying each indicator, PLM$_4$BS defines a monitoring metamodel that includes elements to permit the process measurement. Process lifecycle management for business software plans to generate this monitoring metamodel from metamodels defined into previous phases. Finally, a set of M2T transformation rules is also defined in PLM$_4$BS to generate a measurement database and code scripts to manage each defined indicator.

Although this phase is not supported by PLM$_4$BS yet, at present, we are working in this phase as future work. Basically, we are defining a theoretical MDE-based framework to support the monitoring phase of PLM$_4$BS.[43]

4. Continuous improvement phase. Finally, after evaluating process performance (through assessment indicators and metrics), an organization should start an internal improvement process to achieve higher quality, efficiency, effectiveness, and performance levels during process execution. If necessary, the organization can iterate over our BPM lifecycle as many times as necessary to achieve business goals.

As explained above, this paper only solves the first phase (software process modeling phase). The remaining phases will be supported after carrying out our future works, as it was mentioned in Section 1. Consequently, our SPM must take into account concrete concepts to make easier process execution, orchestration, and monitoring, once modeled. In light of this, this paper proposes an initial mechanism to define indicators and execution parameters.

4 | A MODEL-BASED SOLUTION TO PROCESS MODELING: PLM$_4$BS

This section presents our model-based SPM language (Section 4.1), which also conforms to ISO/IEC 24744:2007. Then, a concrete syntax is defined by means of UML profiles to use our language (Section 4.2) in real environments.

Our proposal is a simple, flexible, and highly semantic model-based language to support the modeling phase shown in Figure 1. It could be considered minimalist, but this feature could be interesting to apply and validate PLM$_4$BS into different business environments (especially to software organizations). Once PLM$_4$BS is checked, it can be extended to include more semantic power in our PML. Besides, our research group is

**Meta-Object Facility (MOF) is a set of standard interfaces that can be used to define and manipulate a group of interoperable metamodels and the corresponding models.

††ISO/IEC TR 24744 presents guidelines that help to define process models so as to improve consistency and uniformity in their definition.

experienced and currently working in transferring scientific knowledge from the university to companies of different business contexts. These opportunities and capabilities allow us to get valuable feedback to extend our proposal as well as consider new concepts in the process definition. Additionally, we offer a simple and flexible language with several goals: (i) facilitate the application of MDE-based mechanisms to PLM$_4$BS, (ii) reduce users' cognitive overload and decrease her/his learning curve when they are utilizing our SPM language, and (iii) allow the rapid application of PLM$_4$BS in real environments, and, especially, with non-ICT users.

From a MDE perspective, this simplicity helps us to successfully apply our proposal to enterprise environments and open new research lines related to testing and simulating running processes. This simplicity is not seen as a drawback because our proposal has extension mechanisms. Moreover, our experience in transferring knowledge to companies confirms that naive MDE-based solutions are more likely to succeed because they enable designing and implementing transformation rules. For this reason, a simple language can help to generate an execution and orchestration process model.

In relation to usability, there are studies based on well-established theories[44] that have analyzed the cognitive overload of several languages (eg, i*,[45] BPMN,[46] UML[47]). These studies justify that complex modeling languages increase user's cognitive overload and make the defined model difficult to understand. This usability aspect has a major influence on users' efficiency when using a language.

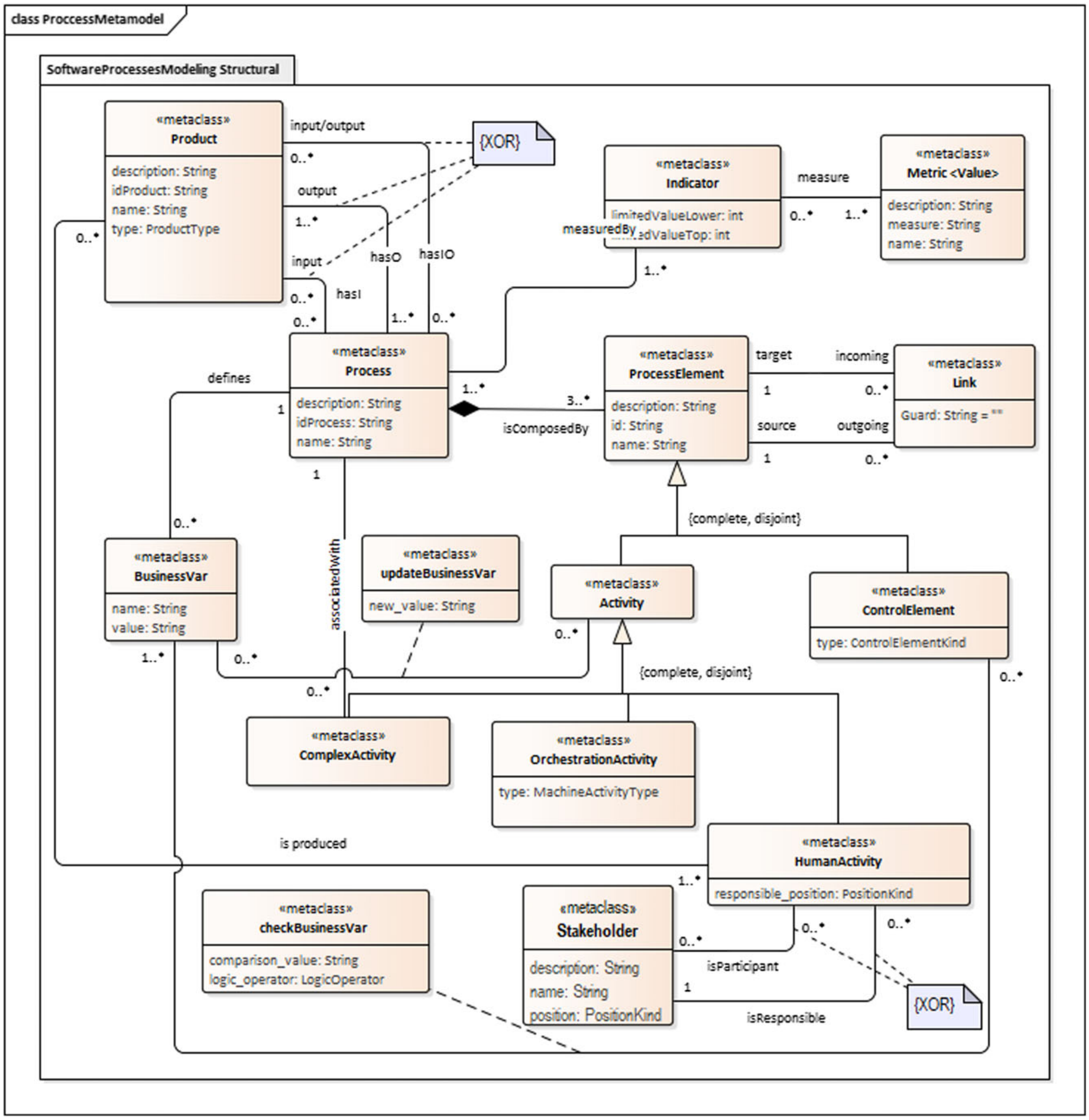


FIGURE 2 Software process modeling metamodel of process lifecycle management for business software

4.1 | Defining a metamodel for SPM

Figure 2 shows our SPM language, once the preliminary aspects have been commented. It takes the form of a MOF compliant metamodel and follows the guidelines defined in ISO/IEC 24744. Before going further, it is worth clarifying that the syntax used is not enough to semantically define our metamodel. Consequently, we use Object Constraint Language (OCL)[48] to add formal constraints that validate process models. All our constraints will not be explained because of the scope of this paper, but they are described in García-García.[49] Nevertheless, as an illustrative example, several constraints are explained in detail as follows.

The *Process metaclass* is the main metaclass in our software process modeling metamodel. It can have associated business variables and a set of ordered *ProcessElements* to produce results or products.

On the one hand, a process can have business variables. This concept is a key element to select the next activity to execute, because it provides indirect data flow paths when a *Conditional* node is reached and it is made up of building blocks to define business rules in the process. Business variables are metamodeled using *BusinessVar metaclass*, which allows defining and initializing variables associated with 1 process (this relationship is established through aggregation relationships between Process metaclass and BusinessVar metaclass).

On the other hand, a process is composed of a set of ordered ProcessElements. *ProcessElement metaclass* represents any element of the process workflow. This metaclass has been specialized in 2 metaclasses: *ControlElement* and *Activity*. The former defines the process control elements, which allow establishing the process structure and its different paths from the start node to the end node. The latter represents an action that should be executed to develop the process. It can update the value of any business variable defined in the process when *Activity* has been completed.

ControlElement metaclass includes *type* property that allows to distinguish among diverse kinds of control elements. In this sense, we consider (i) *InitialElement*, which is the first activity of the process and the entry point to the sequence of remaining activities (it is crucial because in runtime, the initial activity is treated differently than other activities, and it is assigned a special behavior, as it is considered the current context of the process); (ii) *FinalElement*, which is the last activity of the process and the exit point; (iii) *Conditional*, which enables creating disjoint branches of the workflow; and (iv) *Fork* or *Join*, which allows starting and ending parallel branches on the workflow, respectively. Users can model considerable complex processes from this set of concepts.

A ProcessElement must be linked to another ProcessElement so as to build the process workflow. This association is named *Link* in our metamodel. However, the syntax used in this association is not enough to semantically define semantic constraints during process modeling. Consequently, defining OCL constraints fills semantic gaps. Table 1 shows an example to control the number of links that an instance of *ControlElement* metaclass can have.

Activity metaclass represents an action that should be executed to develop the process and is specialized in 3 metaclasses: (i) *OrchestrationActivity*, which represents an orchestration activity (ie, an activity performed by a machine); (ii) *ComplexActivity*, which allows including a process within another process; and (iii) *HumanActivity*, which represents an activity that someone performs. These metaclasses contain ProcessElement metaclass properties, although they include other properties. *HumanActivity metaclass* sets the position of someone who carries out the activity (*responsible position* property). *OrchestrationActivity metaclass* has *URI*, which reveals the identifier of the resource or service accessed by the activity, and *type*, which sets the type of orchestration activity (types are defined in *MachineActivityType* enumeration). OrchestrationActivity metaclass provides a mechanism to indicate script execution, service call, and messaging.

To include a process within another process, it is necessary to define a *ComplexActivity* in the process and associate both of them (*associatedWith* metaclass association). The aforementioned granularity goal is supported through this relationship, because it allows us to launch and invoke a process from another process.

TABLE 1 Object Constraint Language constraint of *ControlElement* metaclass

```
context ControlElement inv:
if (self.type = Conditional) then
self.oclAsType(ProcessElement).incoming➔size() >= 1 and
self.oclAsType(ProcessElement).outgoing➔size() >= 2
else if (self.type = Fork) then
self.oclAsType(ProcessElement).incoming➔size() >= 1 and
self.oclAsType(ProcessElement).outgoing➔size() >= 1
else if (self.type = Join) then
self.oclAsType(ProcessElement).incoming➔size() >= 1 and
self.oclAsType(ProcessElement).outgoing➔size() = 1
else if (self.type = InitialElement) then
self.oclAsType(ProcessElement).incoming➔size() = 0 and
self.oclAsType(ProcessElement).outgoing➔size() = 1
else
self.oclAsType(ProcessElement).incoming➔size() > 0 and
self.oclAsType(ProcessElement).outgoing➔size() = 0
endif endif endif endif
```

As previously indicated, business variables can be updated in any process activity. This requirement is met through the *updateBusinessVar* association, which has a single attribute.

The use of variables provides indirect data flow paths from the point where a value is written in the variable to all the points where the value is read. In consequence, the *Conditional* node (ControlElement metaclass) is the node where variables are checked to select the next action to be performed. It is possible to check [1..*] status variables in 1 Conditional node. The checking action is represented by an association (named *checkBusinessVar*) class relationship between ControlElement metaclass and BusinessVar metaclass, but it is necessary to supervise that the ControlElement metaclass has Conditional type. This constraint is reviewed through several OCL constraints.

Anyway, *comparison_value* property and *logic_operator* property must be set at checkBusinessVar association. Nowadays, the available logic operators are *equal* and *nonequal*. 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:

$$Result\ (C) = \wedge_{i=1}^{n}\{C_i(initValueVar)\ C_i(logicOperator)\ C_i(comparisonValue)\}$$

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 *nonequal*.
- C_i*(comparisonValue)* is the value used to compare the initial value of C_i variable.

Moreover, 1 stakeholder must at least be connected to HumanActivity to perform it. *Stakeholder metaclass* represents a system or a role that can be either participant or responsible, but cannot perform both roles (our metamodel shows this constraint using *XOR* logical operator between *isParticipant* and *isResponsible* associations). We have considered these 2 kinds of stakeholders involved in running an activity, because in software companies, there are many work teams including a large amount of members, and each of them cooperate in developing activities depending on the involvement degree. They differ in that participants can 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 to continue the process actions sequence.

Product metaclass represents the product obtained as a result when *Process* is executed, or as an entry product. In this sense, *Product* can be typed as an input, an output, or an input/output product of Process. Our metamodel shows this constraint using *XOR* logical operator among *hasI*, *hasO*, and *hasIO* associations. In addition, Process has at least 1 Product (this constraint is described within our metamodel through cardinality between Process and Product metaclasses).

Product is associated with Process, even though it may be generated by an activity, if completed. Moreover, the number of output Products linked to Process must coincide with the total number of Products generated from all activities conforming Process. In addition, HumanActivity that is part of Process must produce Product. This constraint cannot be expressed within our metamodel because the syntax used is not sufficient to semantically define this constraint. Thus, we use OCL to describe it (see Table 2).

Finally, establishing preliminary mechanisms for measuring some properties of a process is essential to continuously improve a software process. As it will be mentioned later, enhancing this aspect of our lifecycle is 1 of our next future lines of work. Meanwhile, we have included metric and indicator concepts in our metamodel to achieve this goal. *Metric metaclass* defines the quantitative measure of the extent to which a process

TABLE 2 Object Constraint Language constraint of the *Process* metaclass

```
context Process inv:
let totalProducts: Integer = self.output➔size()
let activitiesColl: SortedSet =
self.ProcessElement➔select(oclIsTypeOf(HumanActivity))

--Each product must have been generated by one of
--the HumanActivities that is part of the Process

self.output->forAll (p1: Product|
activitiesColl➔exist (a: HumanActivity|
a.Product➔exist (p2: Product|p1 = p2)))
and
--The number of output Products of the Process must match
--the total number of Products generated from all
-- HumanActivities that are part of the Process

totalProducts =
activitiesColl➔iterate(a: HumanActivity; acc:Integer=0|
acc + a.Product->size())
```

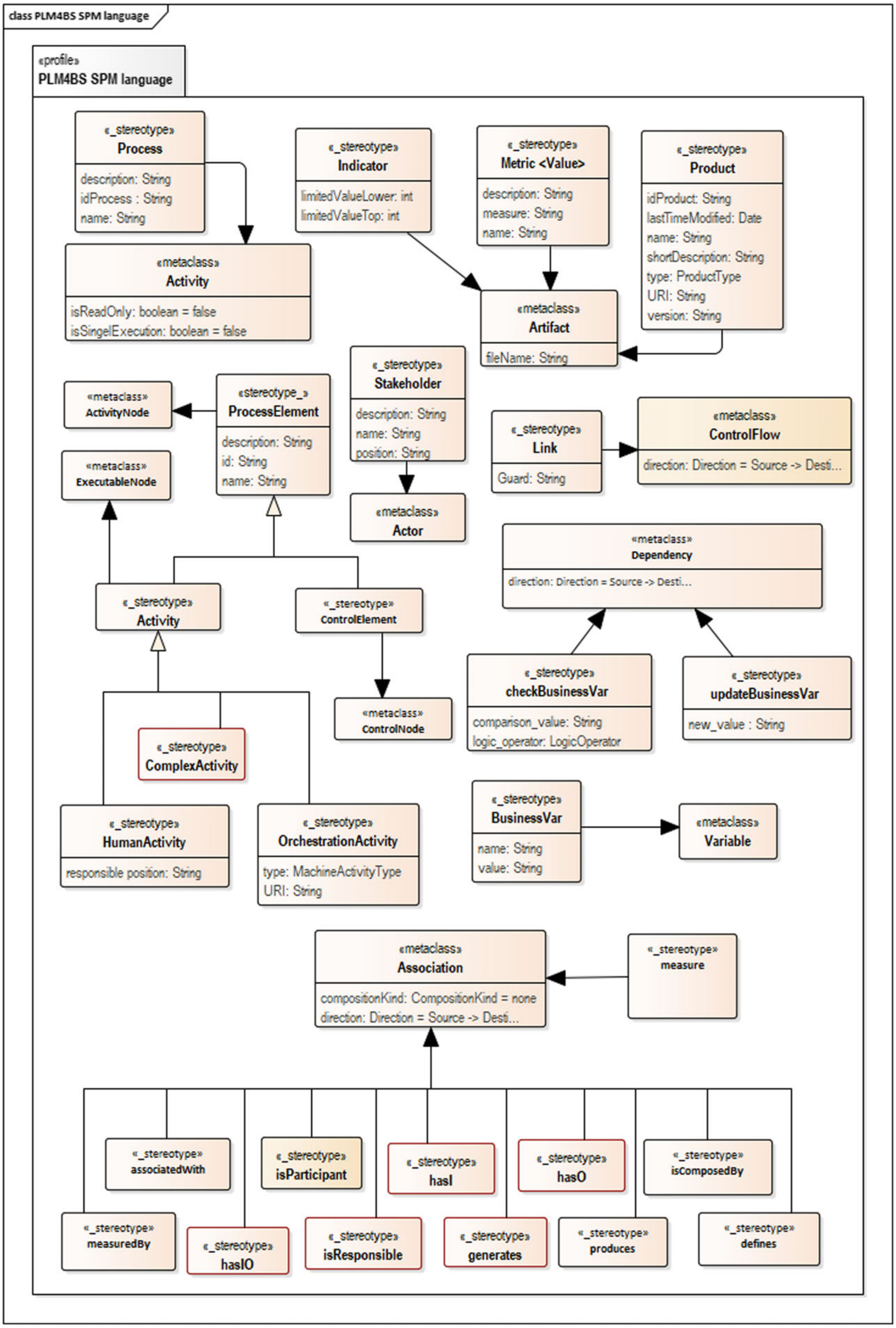


FIGURE 3 Unified Modeling Language profile of process lifecycle management for business software metamodel for software process modeling

contains a given attribute (ie, it is a handle or guess about a given attribute). *Indicator metaclass* is used to compare a metric (or measure) with a target value. It enables measuring a concrete metric objectively, and it constitutes a basis for a scorecard. This metaclass defines the value interval on which the metric is evaluated.

4.2 | Defining a concrete syntax

Using a metamodel for process modeling guarantees uniformity, formalized terminology, and correct definition. However, if a suitable tool is not defined, maintenance can result to complex and expensive in enterprise environments.[50]

Consequently, it is required to offer a tool-based mechanism and start with wondering how user can instance our software process modeling metamodel. In this sense, Object Management Group proposes to define concrete syntaxes in 2 perspectives,[51] either to define a new language (UML alternative) or to extend UML. Each of these alternatives hides advantages and disadvantages. Defining a new ad hoc language allows greater expressiveness and correspondence with concepts of particular application domain, like getting a tailored suit. Nevertheless, even if these new ad hoc languages are described with MOF, the failure to comply with the standard UML complicates the application of these languages and the management of their concepts in a natural way.

We have chosen to define concrete syntax through UML profiles because they provide a flexible and usable mechanism for adapting our metamodel to enterprise environments. In fact, UML is a well-known standard with many successful cases.[52] In addition, using UML profiles provides great flexibility, expressiveness, and a generic extension mechanism for building UML models in particular domains. This option also allows defining a collection of UML extensions that together describe some particular modeling problems and facilitate modeling constructions in that domain. Unified Modeling Language extension mechanisms are based on (i) *stereotype*, defining the elements of a specific domain to extend UML metaclasses from them; (ii) *tagged value*, adding extra meta-attributes to any stereotyped element defined in the profile; and (iii) *constraint*, identifying the conditions of the stereotyped elements to create well-defined models and extrapolate constraints (defined in the metamodel) to the profile. Figure 3 shows the UML profile associated with our metamodel (2 types of triangles are used: the dark triangle denotes UML extension mechanism, and the clear triangle represents UML inheritance mechanism).

To define our UML profile, we have defined 1 stereotype per element of our SPM metamodel and we have used different UML2.5 metaclasses to extend each of them (Table 3).

5 | SUPPORTING TOOL: PLM$_4$BS SUITE

One of the most important aspects to ensure the applicability of PLM$_4$BS in real environments is to design and develop software tools that back up our theoretical framework in a friendly way. These tools should allow using our UML profile as well as automatic verifications of our OCL semantic constraints in runtime.

TABLE 3 Relationship between stereotypes and UML2.5 metaclass

UML2.5 Metaclass	Stereotypes	Comments
“Association”	“updateBusinessVar,” “checkBusinessVar,” “isParticipant,” “isResponsible,” “generates,” “hasI,” “hasO,” “defines,” “produces,” “hasIO,” “generates,” “isComposedBy,” “measuredBy,” “associatedWith”	It specifies a semantic relationship that can occur among typed instances and confirms that instances of the associated types can be linked.
“Artifact”	“Product,” “Metric,” and “Indicator”	It represents an item of information that is either used or produced by a software development process or by a system operation.
“Activity”	“Process”	It is a *Behavior* specified as a sequence of subordinated units, using control and data flow model.
“ActivityNodes”	“ProcessElement”	They are used to model the individual steps in the behavior specified by an *Activity* of UML.
“ControlNodes”	“ControlElement”	They act as traffic switches managing the information flow across *ActivityEdges*” (UML metaclass).
“ExecutableNodes”	“Activity”	They carry out the desired behavior of an “Activity” of UML.
“ControlFlow”	“Link”	It is used to explicitly sequence the execution of *ActivityNodes*. In consequence, we use this metaclass because (i) it bridges the flow between “ActivityNode” nodes and (ii) it can define Guard (ie, a condition that must be true before control passes along that activity edge).
“Variable”	“BusinessVar”	It allows supporting indirect data flow paths from the point at which a value is written in the *Variable* to all the points at which the value is read from the “Variable.”
“Actor”	“Stakeholder”	It specifies a role played by a user or any other system that interacts with the subject.

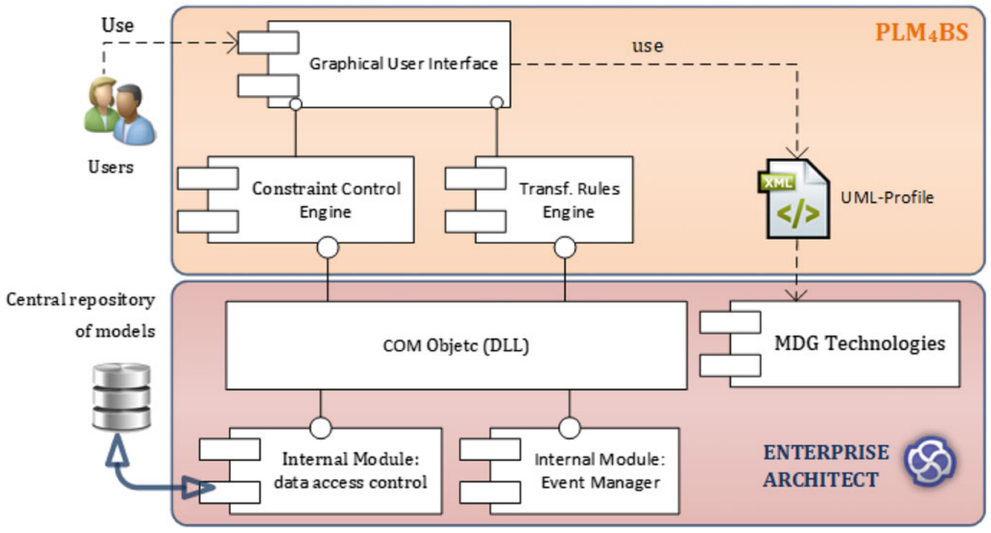


FIGURE 4 Software components of process lifecycle management for business software

The main issue to design and develop the supporting tool of PLM$_4$BS is to decide whether we need to develop our own modeling tool or we need to use any other modeling tool as reference (such as Enterprise Architect[53] (EA) or START UML,[54] among others) where the functionality of PLM$_4$BS is deployed.

Definitively, EA is chosen to integrate itself the functionality of PLM$_4$BS to provide a professional working space. Enterprise Architect is a well-known tool in business environments, so PLM$_4$BS could be more likely to be accepted and used in companies, if it is well integrated into EA.

However, making this decision has not been an easy task. We carried out several market studies[‡‡] in collaboration with the Andalusian Regional Ministry of Culture, Education and Sport (Spain). These studies compared 9 modeling tools to identify the best one to conduct our research about technology transfer issues. This comparative considered (i) UML2.0 extension mechanisms (ie, a tool must allow developing and designing UML profiles), (ii) MDE mechanisms (ie, a tool must support mechanisms or algorithms to systematically generate models from other models), (iii) automatic and flexible generation of documentation, (iv) management of big project lifecycles, (v) usefulness for the entire software lifecycle, and (vi) compatibility with UML2.0.

Once finished, we concluded that EA offered the best value for money in the evaluated points. Another reason that led us towards using this tool instead of another was the fact that EA is widely used in a large number of IWT2 projects,[19,55,56] and it is well known by most customers and partners. All these reasons made us finally choose EA. Nonetheless, it is important to emphasize that PLM$_4$BS can be implemented by any modeling tool that manages UML and extension mechanisms as it is possible to migrate models among modeling tools (whenever they provide import and export options in the XMI format).

Therefore, and as introduced above, PLM$_4$BS is designed and developed using plugins on EA. In this sense, PLM$_4$BS is composed of 4 software components (see Figure 4).

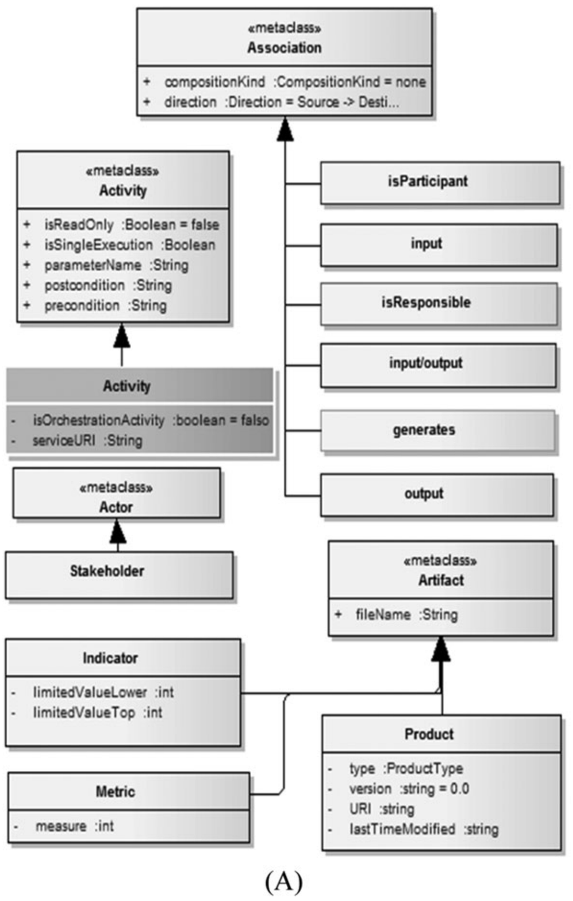
The first one is *graphical user interface*, which comprises specific user interfaces to friendly instance each concept of our SPM metamodel (see Figure 2) using the UML profile shown in the previous section. This *UML profile for SPM* is the second component of PLM$_4$BS. The theoretical definition of this UML profile was described in the previous section, but it is necessary to implement it in a specific tool to instance each concept of our SPM metamodel. For this purpose, EA provides mechanisms (named MDG Technologies[§§]) for defining UML profile based on XMI format. Figure 5A shows part of our UML profile that was implemented using EA. The complete diagram does not appear because of its dimension and the fact that it neither provides any relevant nor new information related to Figure 3. In addition, Figure 5B shows the final toolbox[¶¶] of PLM$_4$BS suite in EA.

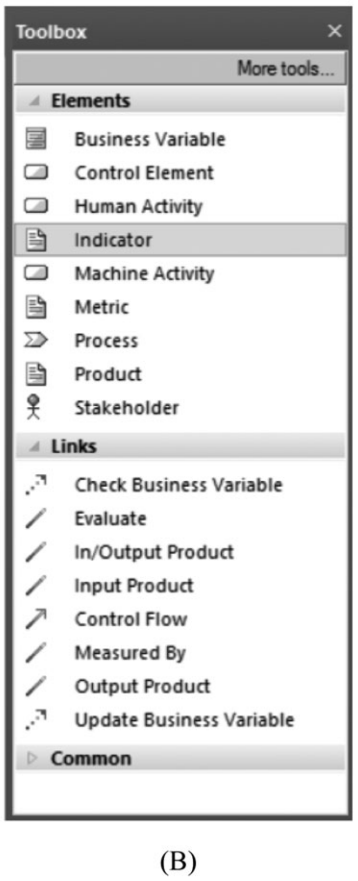
The third and fourth components of PLM$_4$BS are *Constraint Control Engine* and *Transformation Rules Engine*, which execute each OCL semantic constraints to ensure building well-formed models and execute each M2M and M2T transformation rules among models, respectively. Regarding the fourth component, it is important to remember that this paper is focused on the first phase of our process lifecycle (modeling phase; Figure 1) and does not define transformation rules, which are identified in the execution and orchestration phase[17] (the second phase of our process lifecycle; see Section 3). Thus, it is relevant to show the Transformation Rules Engine component to present all the software components of PLM$_4$BS.

[‡‡]These market studies were written in Spanish. Even though they were not published, they can be consulted in www.iwt2.org.

[§§]The mechanism offered by EA to define UML profiles is named MDG Technologies. This functionality allows users to extend EA's modeling capabilities to specific domains and notations. MDG Technologies seamlessly plug into EA to provide additional toolboxes, UML profiles, patterns, templates, and other modeling resources.

[¶¶]Toolbox is a panel of icons used to create elements and connectors on a diagram into EA. Within the Toolbox, related elements and connectors are organized into pages, which in turn contain the elements or connectors used for a particular type of diagram. When a diagram is opened (for instance, a diagram of PLM$_4$BS suite), the Toolbox automatically provides the element and relationship pages corresponding to the type of diagram.




Toolbox
More tools...
Elements
Business Variable
Control Element
Human Activity
Indicator
Machine Activity
Metric
Process
Product
Stakeholder
Links
Check Business Variable
Evaluate
In/Output Product
Input Product
Control Flow
Measured By
Output Product
Update Business Variable
Common

(B)

FIGURE 5 Part of our Unified Modeling Language profile implemented using Enterprise Architect

6 | PLM$_4$BS EVALUATION

To assess the applicability of PLM$_4$BS and its features, we carried out a multiple-case study with 3 cases, that is presented in the following subsections. This section does not aim to conduct a full experiment, and it does not pretend to be exhaustive, because it is out of the scope of this paper, but introduce insights, issues, and ideas for an evaluation with statistical significance. Our purpose is to present a multiple-case study, gather initial data to validate our proposal, and obtain initial evidence to demonstrate the benefits of the approach.

6.1 | Case study research process

The evaluation of PLM$_4$BS has been carried out for 2 years and following the case study research method proposed in Runeson et al.[57] This method is presented in Figure 6 and defines 3 phases: (i) design case study; (ii) plan, collect, and analyze; and (iii) joint analysis and report.

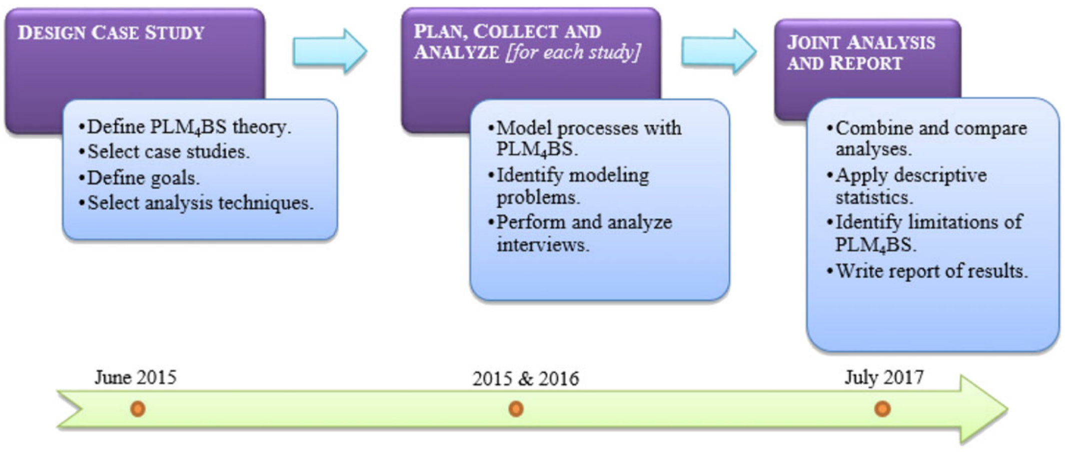

DESIGN CASE STUDY
•Define PLM4BS theory.
•Select case studies.
•Define goals.
•Select analysis techniques.
PLAN, COLLECT AND ANALYZE [for each study]
•Model processes with PLM4BS.
•Identify modeling problems.
•Perform and analyze interviews.
JOINT ANALYSIS AND REPORT
•Combine and compare analyses.
•Apply descriptive statistics.
•Identify limitations of PLM4BS.
•Write report of results.
June 2015
2015 & 2016
July 2017

FIGURE 6 Method of our multiple-case study

On the one hand, it is necessary to define our evaluation context and design our case study. In this sense, this case study aims to evaluate PLM$_4$BS for SPM and its supporting tool within real environments. In this context, together with the theoretical framework described in previous sections, the objective of this case study is to empirically review PLM$_4$BS for SPM in expressiveness, understandability, granularity, measurability, orchestrability, and business rules definition. This objective is further divided into several research questions that are presented in Section 6.2.

About case selection, we initially selected 3 cases: (i) neurological service of Andalusian Health and Social Welfare Service (AHSWS) in Spain, where PLM$_4$BS was applied to adapt the e-Health Web platform of AHSWS to a software architecture based on processes and clinical international standards; (ii) Public Works Agency (PWA) of the Andalusian Regional Government (Spain), where PLM$_4$BS was applied to define administrative processes related to public works contracts and maintenance services of infrastructures[58]; and (iii) Web Engineering and Early Testing (IWT2) group of the University of Seville (Spain), which used PLM$_4$BS within the NDTQ Framework (NDTQF) project[19] to define 6 groups of software processes (related to software development, testing, maintenance, quality management, security management, and software project management).

These study cases were selected for 3 main reasons: (i) evaluation of PLM$_4$BS in different domains, (ii) interest of organizations in either improving processes and management for their involvement in a quality certification process or adopting widely acknowledge good practices, and (iii) availability of data sources and subjects' willingness to cooperate. As a result, the data collection protocol was designed by defining the desired data to be collected and the type of analysis to be performed and establishing a plan to address the following steps of the study.

As the second phase of our case study research method is concerned, it was necessary to *plan*, *collect*, and *analyze* the results. For this purpose, 3 steps were followed. Firstly, it was necessary to *plan data collection*, that implies different activities depending on the case. Such activities included (i) identification of the archival data available that could be provided to the researchers for analysis, (ii) the preparation of material, and (iii) subsequent training of 1 of the cases' participants for the definition of process models using PLM$_4$BS and its supporting tool for process modeling. Besides, the second phase was also required to *collect evidences*. In this sense, we considered the use of data collection techniques from the 3 degrees defined in Lethbridge et al[59]: direct methods consisting of interviews and independent analysis of already available work artifacts. To store the data collected, we used spreadsheets to facilitate their posterior analysis. Then, we carried out a deductive *analysis of collected data*. The activities performed involved different types of qualitative analyses, inspired by the process for qualitative data analysis introduced in Runeson.[57] They were complemented by some quantitative analyses based on descriptive statistics. There were 3 researchers who conducted the analyses, to reduce bias by individual researchers (2 researchers analyze data, and then, their individual results were merged and discussed by both of them together with an additional one).

Finally, the last phase of our case study research method was to provide a *joint analysis* and *report* where we described conclusions, results, and limitations of PLM$_4$BS as well as elaborated the final report.

6.2 | Design the case study

The established objective for our multiple-case study was conceived into different research questions (RQ) as follows:

- RQ1 (expressiveness) To what extent is PLM$_4$BS capable for modeling users' processes and their needs in activities (human, complex, and orchestration activities[##]), conditional and parallel branches, exception handling, products, and roles? Were there some processes that could not be modeled by PLM$_4$BS?
- RQ2 (granularity) Can users model their processes with the level of detail they need?
- RQ3 (understandability) How well or easily can users understand (read and model) their processes using PLM$_4$BS? Does PLM$_4$BS offer familiar metaphors to represent modeling concepts? What are the experiences and user feedback?
- RQ4 (measurability) In what way does PLM$_4$BS recommend users that they should use key perform indicators associated with the process or activities of the process?
- RQ5 (orchestrability) To what extent is PLM$_4$BS capable for expressing automatic events or calls to other software components (such as services or code scripts, for instance)?
- RQ6 (business rules) How well can users indicate business rules to condition indirect data flow paths during the process modeling taking into account, for instance, results of an orchestration activity?

Regarding the 3 selected cases, Table 4 summarizes the context and scope of each study case in involved subjects and evaluated objects. This information was collected after doing different previous interviews with the staff responsible of each project. On the one hand, the involved subjects in each study case have been categorized as manager role (MR, who is responsible for overseeing and leading the work of a group of people in the evaluated project) and technical tole (TR, who has usually business and technical knowledge within the organization). On the other hand, evaluated objects are related to the processes that have been considered in our study. Especially, we show number of evaluated processes, number of gateways, and number of activities included in each process (both ones are metrics of the size and complexity of each process).

[##]Section 4.1 describes these concepts and their differences.

TABLE 4 Context of our case studies

		AHSWS	PWA	NDTQF	
Evaluated objects	# Total processes	2	4	28	34
	# Total activities	64	132	351	423
	# Total gateways	17	33	49	99
Involved subjects	MR	2	2	1	8
	TR	1	2	1	

Abbreviations: MR, manager role; TR, technical role.

Process lifecycle management for business software was within the soaADAPT project[60] in the case of the neurological service of AHSWS. It aimed to adapt the e-Health Web platform of AHSWS to SOA-based process architecture to allow better modularity, independence, maintainability, and reusability of developed services. It mainly consisted in establishing MDE-based mechanisms to combine a clinical process model and health care data standards.[61,62] Process lifecycle management for business softwarewas used to model 2 health care processes within the soaADAPT project: managing the emergency telephone service and medical visits of patients with spinal cord injury. Both processes amounted 64 activities in total (among human, complex and orchestration activities) and 17 gateways. This study case was composed by 3 involved subjects: 2 managers involved in this case were linked to end users (managers are responsible for continuous improvement) and 1 technician in charge of modeling both healthcare processes using PLM$_4$BS.

Moreover, PLM$_4$BS was also applied within the context of the THOT project[58] in the PWA of the Andalusian Regional Government. It aimed to define a MDE-based environment for managing agile documentary records (associated with contracting services for transport and infrastructure constructions) and improve the administration of these records as well as their integration into enterprise content management systems. For this purpose, it was necessary to model 4 administrative processes related to concessions of public contracts in public infrastructure, assignments of services in public infrastructures, and concessions of public construction contracts. These processes consisted of 132 activities and 33 gateways. This study case was composed by 4 involved subjects: 2 manager, who was responsible for the project and its execution, and 2 technicians who were in charge of modeling, using PLM$_4$BS, the 4 administrative processes of the THOT project.

Similarly, PLM$_4$BS was used within the NDTQF project[19] that was carried out to offer a MDE-based solution to improve the application of NDT methodology[20] in R&D projects. For this purpose, PLM$_4$BS was used to define 28 software processes related to software development, software testing, software maintenance, software quality management, security management, and software project management. These processes comprised 351 activities and 49 gateways. This study case was composed by 2 involved subjects: 1 manager role and 1 technical role.

As data collection protocol is concerned, interviews and some available archival data were the initially defined data sources, according to the information available at that moment. Later on, they were explained and extended during the study cases, as detailed in the following section.

6.3 | Plan and data collection

Planning data collection and actual collection were performed by means of iterative cycles. We applied data collection techniques from the 3 degrees defined in Lethbridge et al.[59] Below, we describe them in data type. The degree is specified in parenthesis. In addition, it is important to mention that 2 researchers (PLM$_4$BS experts) carried out the data collection to minimize the possible subjectivity of this manual task.

6.3.1 | Process models previously defined (third degree)

Andalusian Health and Social Welfare Service was the unique organization that uses BPMN to model some of its processes. Especially, AHSWS only uses BPMN to model administrative processes and not health care processes because BPMN was very complicated to understand by doctors according to indications of manager in that organization. In addition, it was very hard to connect and combine BPMN with health care data models (which complied with different clinical international standards[61,62]).

Consequently, these health care processes were only used in a documentary way and not in everyday life. Public Works Agency and NDTQF modeled their processes (previously to this paper) using nonstandard notations like ad hoc table-based notations written in a natural language. All these available data were collected, and the process models were remodeled using PLM$_4$BS, if necessary.

6.3.2 | Process models defined using PLM$_4$BS (second and third degrees)

The definition of the provided process models using PLM$_4$BS involved different roles depending on the unit. In AHSWS, the 2 health care processes were modeled by the technician with the supervision and support of the member responsible for processes and continuous improvement. In PWA, the technicians modeled the 4 administrative processes using PLM$_4$BS.

For this purpose, the 2 PLM$_4$BS experts had to explain some BPM principles to each technician throughout several previous meetings. In NDTQF, 1 technician was in charge of modeling 28 software processes taking into account initial lessons concerning the use of PLM$_4$BS offered by experts.

TABLE 5 Summary of the characteristics of process models of each case study

	AHSWS	PWA	NDTQF	
# Human activities	37	95	287	419
# Complex activities	2	37	7	46
# Orchestration activities	25	5	57	87
# Paths	≈ 34	≈ 21	≈ 112	≈ 151
# Business variables	7	3	13	23
# Business rules	3	5	17	25
# Indicators	2	0	84	86

6.3.3 | Interviews (first degree)

Direct methods for collecting data were used in 3 of the cases. During this phase, interviews were conducted for other researcher (PLM$_4$BS experts) than one who conducted the data collection. In this context, the researcher presented the final process models during interviews in different levels (firstly with managers and later with the technician of each project). This was a semistructured interview, similar to a discussion. The most interesting facts and important answers were written down by 1 of the researchers in the form of notes that were later sent to the interviewer for validation. They allowed us to know expressiveness-related limitations and understandability-related weaknesses in PLM$_4$BS.

Although interviews were similar to a discussion, these ones are semistructured to not introduce subjective information into the analysis of the results. Below, the interview guidelines used by PLM$_4$BS are described in detail.

On the one hand, interviews contain questions about statistical information related to the different concepts used when the metamodel of PLM$_4$BS is instanced. This information has been categorized in (i) number of human, complex, and orchestration activities; (ii) number of different paths within process models; (iii) number of business variables, which are used to build business rules; (iv) number of business rules; and (v) number of metrics and/or indicators, which are used to measure the process or its activities.

This information is obtained after modeling each process in each study case. This modeling is carried out by each technical role involved. Modeling mistakes (erroneous or ambiguous definitions) introduced by technicians along their processes have to be reviewed. Specifically, PLM$_4$BS experts note each mistake with the PLM$_4$BS concepts that were incorrectly used in the model. This information together with the results of interviews was used to identify the notation constructs that presented more problems in relation to understandability and correctness.

On the other hand, the changes undergone by the process in the graphically defined process models are analyzed during the interviews regarding our 6 aforementioned dimensions (expressiveness, understandability, granularity, measurability, orchestrability, and business rules definition).

6.4 | Analysis of collected data

We carried out a deductive analysis to conduct this study. This entails that categories of analysis are imposed prior to data collection. The main categories of analysis in our multiple-case study (expressiveness, understandability, granularity, measurability, orchestrability, and business rules definition) match with the 6 research questions raised (see Section 6.2). Next, the specific actions accomplished during this analysis are detailed. We have tried to publish the models of each case study. However, we have found inconveniences on each organism because of the confidentiality of the data.*** Anyway, below, we describe the main characteristics of these processes to present our conclusions, results, and limitations in Section 6.5.

Firstly, the whole set of process models of the different cases was reviewed to check whether all of them could be defined or not with PLM$_4$BS. For those process models that could not be defined with PLM$_4$BS, the reasons that prevented their definition were identified and categorized according to limitations of PLM$_4$BS and missing or ambiguous information. Whenever possible, clarification and provision of missing information were required to the roles involved in process modeling.

Secondly, descriptive statistics (previously described at the ending of Section 6.3) were applied to obtain information related to the different concepts used when the metamodel of PLM$_4$BS is instanced. Table 5 shows this information following the used interview guide (which is described in Section 6.3).

Finally, each process model has been analyzed by means of the information obtained from the evolution of the 3 case studies. The data collected during the interviews were analyzed to draw conclusions regarding our 6 aforementioned dimensions (which have been previously described). Particularly, 2 researchers (PLM$_4$BS experts) analyzed the interviews and coded them according to these categories. The coded data were stored in tables together with references to the data source to ensure full traceability and maintenance of a chain of evidences. These tables were then used to identify results across data sources and cases, merging the results obtained by the 2 researchers and discussing them with the

***If reader wants to see these models, s/he could request it to authors who will request the permissions to the organism to be able to send it.

third researcher. In addition, the preliminary results from the study, including the interview, were presented back to each role involved in each case study. Their opinions were collected, and any misinterpretation was corrected. Final conclusions were based on all the gathered information.

6.5 | Results and limitations

This section presents the results identified after the analysis of the collected data. We structure the findings according to the 6 categories corresponding to the research questions posed in Section 6.2.

6.5.1 | RQ1 (expressiveness)

After the analysis of the collected data, only 1 limitation related to the creation of exception paths into process models was found in PLM$_4$BS. Process lifecycle management for business software does not support this functionality yet, but we are already working on this topic to extend our SPM metamodel and PLM$_4$BS suite. However, technicians involved in each case study tried modeling exception paths combining normal path (ie, ControlElement metaclasses) and business rules (which were not well defined after being reviewed by the PLM$_4$BS experts involved). In fact, the rate of exception paths was 15.5%, 35.8%, and 9% in AHSWS, PWA, and NDTQF, respectively. These figures highlight the need to provide efficient, friendly, and effective mechanisms to define exception paths in process models. This implementation is 1 of our priority future works.

6.5.2 | RQ2 (granularity)

Each technician who could define processes with the level of detail required was able to successfully overcome this aspect. However, a third of the interviewed technicians responded that the graphical notation to represent a subprocess was not the best one as it seemed to be another task of the process. This aspect is addressed in the next paragraph (see Section 6.5.3.).

6.5.3 | RQ3 (understandability)

The first applications of PLM$_4$BS revealed a couple of drawbacks related to the extent to which users understand (read and model) their processes using PLM$_4$BS or if PLM$_4$BS offers familiar metaphors to represent its modeling concepts. Firstly, most of interviewed experts commented that it could be interesting to improve the graphical notation of Indicator metaclass of our metamodel (see Figure 2) when an Indicator is instanced. At present, the graphical notation of this metaclass is very similar to that of an UML class, making it difficult to understand these process models at a glance. Secondly, some manager roles do not know how to distinguish each type of orchestration activity (ie, script execution, service call, and messaging; see Section 4.1) because, at present, PLM$_4$BS uses the same graphical notation to represent any kind of orchestration activity (a rectangle with the image of a computer in a corner to symbolize that it is an activity executed by a computer). This fact makes it difficult to distinguish between kinds of orchestration at a glance. Finally, the third restriction has been mentioned above (see Section 6.5.2.).

In these cases, PLM$_4$BS was improved to include graphical notations closer to their purpose. For example, Indicator metaclass is graphically represented with the icon of a bevel to symbolize a measurement. After these implementations, users (ie, organizations' employees and researchers) were able to read and validate their process models reported during the interviews. Furthermore, more than 82% considered that these graphical improvements were right.

6.5.4 | RQ4 (measurability)

After the analysis of the collected data, a couple of weak points were uncovered regarding how PLM$_4$BS supports the definition of key perform indicators (KPI). At present, PLM$_4$BS allows defining KPI in a textual way and later associating them to processes or activities using the measuredBy of our metamodel (see Figure 2). Nowadays, the main problem of PLM$_4$BS (which is commented by each interviewed role) is that this textual definition of KPI is not enough to automatically execute, understand, and maintain indicators. This definition of KPI is the first step to generate an executable model from this textual definition, but it is necessary to provide mechanisms for defining other parameters, such as mathematical formulas, to apply each KPI or units of measure, for instance. Overcoming this limitation of PLM$_4$BS is important for us, being in fact the objective of the third phase of our process lifecycle (monitoring phase; see Figure 1). At present, a PhD student, whose doctoral thesis is defining a theoretical MDE-based framework to support the monitoring phase of PLM$_4$BS,[43] is carrying on with the intended study of PLM$_4$BS.

6.5.5 | RQ5 (orchestrability)

This RQ is related to how PLM$_4$BS allows expressing automatic events or calls to other software components (such as services or code scripts, among others). This aspect was partially overcome by each technician because they were only able to indicate the name and description of each orchestration activity of their process models. More than 60% technicians were missing how they must define the execution context of their orchestration activities (eg, URL of services or input data). We explained them that this comment is not really a limitation of PLM$_4$BS because the evaluation performed in this paper aims to validate and evaluate the process modeling language of PLM$_4$BS. The definition of the execution context of each orchestration activity has been carried out in the second phase of our process lifecycle (execution and orchestration phase; see Figure 1) which is described in detail in García-García et al.[17] After carrying out interviews with managers and technicians and keeping in mind that the execution and orchestration phase of PLM$_4$BS is out of the scope of this paper, we explained this second phase to everyone and they were very interested in evaluating it. As a result, we are already planning and designing an experiment as a new forthcoming work.

6.5.6 | RQ6 (business rules)

After the analysis of the collected data, a couple of limitations were detected in PLM$_4$BS and the notation was consequently improved.

On the one hand, the first problem was related to the creation of variables and building business rules (using existent variables) because most managers and technicians do not understand suitably the visual representation (ie, syntaxes of this representation) implemented in PLM$_4$BS suite and our UML profile. At the beginning, the updateBusinessVar and checkBusinessVar metaclasses were defined as simple UML associations between instances of BusinessVar and Activity, and BusinessVar and ControlElement metaclasses, respectively (with their specific properties according to our SPM metamodel). This kind of graphical notation (direct associations among elements) proves to be confusing when the process diagram is drawn. It is also very impractical when PLM$_4$BS is widely used to continuously define large processes. Therefore, PLM$_4$BS was improved for users to facilitate the creation of variables and the elaboration of business rules. For this purpose, new intuitive graphical user interfaces were designed and developed in PLM$_4$BS similar to assistant wizards. Today, users can create variables or business rules with wizards, and PLM$_4$BS is internally responsible for producing and exemplifying each necessary metaclass according to the metamodel shown in Figure 2.

On the other hand, the second problem reported by technician was related to the available logic operators to define business rules. This limitation was mainly reported by technicians involved in AHSWS case. As mentioned in Section 4.1, PLM$_4$BS uses equal and nonequal to compare the value of a business variable with another value. This catalog of operators does not seem to be enough to model complex business rules, as it happened in AHSWS (for example, some managers considered it necessary to have mathematical operators such as minor, major, or minor than). This improvement will constitute 1 of our priority future works.

6.6 | Threats to validity

In this section, we discuss possible threats to our evaluation of PLM$_4$BS (see Figure 6). There are always limitations and threats to the validity to all studies, and so also for our case study. To discuss some possible threats, we organize this section in several sections: (a) threats to selection of case studies, (b) threats to data extraction, (c) threats to generalization of results, and (d) threats to conclusions.

6.6.1 | Threats to selection of case studies

Threats to the selection of case studies deal with possible limitations of the case studies themselves. In the beginning, we only evaluate our proposal in a business context: the NDTQF project. Two subjects (1 manager role and 1 technical role) were involved in this study case, and, in addition, they had software background. This situation could cause an important bias in the use of PLM$_4$BS, so we evaluated many software processes in the NDTQF project to minimize the impact of this situation on our evaluation. However, we thought it was not enough because most of these software processes could be categorized within the same kind of process (ie, methodological processes).

To mitigate this threat, we selected 2 new projects (PWA and AHSWS) to include them in the evaluation of our proposal. This decision was performed by us for several reasons: (i) evaluation of PLM$_4$BS in different business domains where involved subjects have not software engineering background, (ii) incorporate in our study another kind of processes of different complexity and size (for instance, the processes of PWA are administrative processes whereas the processes of AHSWS are health care processes), and (iii) availability of data sources and subjects' willingness to cooperate.

6.6.2 | Threats to data extraction

Threats to data extraction deal with possible problems that might arise in the data collection phase. The most common threat during this phase is the subjectivity of the researcher who performs the data collection. To mitigate this threat, we developed a data collection validation mechanism that involved at least 2 researchers (PLM$_4$BS experts), as it is presented in Section 6.3. In addition to that, during this phase, interviews performed in plan and data collection phase were directed for other researcher (PLM$_4$BS experts) than one who conducted the data collection.

6.6.3 | Threats to conclusions

Threats to conclusion validity are factors that can lead to incorrect conclusions. In this category, we have been able to identify that we measure research intensity with the number of case studies and number of processes like metrics. However, to mitigate this threat, such attempt would introduce subjective criteria into the analysis of the results. For this reason, we preferred to use an objective (directly measurable) criterion for characterizing the use of PLM$_4$BS in each case study. These criteria are described in Section 6.4.

7 | RELATED WORK

Over the last years, some theoretical solutions have been proposed to document and model processes. Accordingly, we have conducted a study[9] to determine the degree of maturity of this research topic. That study analyzes the most recent proposals related to SPM and identifies a great amount of them that use different techniques, such cash petri-nets, models, or programming languages. This fact confirms that there is no ideal language for process modeling.

TABLE 6 Coverage of the related work analyzed

	Expressiveness							Understandability	Granularity	Measurability	Orchestrability	Business Rules
	Human Activities	Machine Activities	Conditional Branches	Parallel Branches	Exception Handling	Products	Roles					
Chou[66]	√			√	√	√	√	√	√			
Di Nitto[67]	√	≈	√	√		√	√	√	√		√	
UML-EWM[30]	√		√			√	√	√	√			
UML4SPM[4,5,7]	√	√	√	√	≈	√	√	√	√			
Combemale[55]	√		√			√	√	√	√			
UPME[8,40]	√	√				√	√		√			
FlexUML[18]	√		√	√		√	√	√				
Ferreira[39]	√	√							√		√	
SPEM2.0[68]	√		√	√		√	√	√	√	≈		
xSPEM[6]	√	√	√	√		√	√	√	√			
eSPEM[35]	√		√	√		√	√	√	√			
MODAL[34]	√		√	√		√	√	√	√	≈		
PLM$_4$BS	√	√	√	√		√	√	√	√	≈	√	√

Cells with √ symbol represent that the feature is supported by the model-based proposal, whereas ≈ symbol and empty cells mean that the feature is partially or not supported, respectively.

Although this topic has been discussed and studied for many years, there are some papers focused on literature reviews and surveys in PMLs,[9-11] but we have found too few on comparative studies in this area,[63-65] and most of them includes general criteria to compare SPML.

In this context, we have decided to use the same dimensions described in Section 2 to briefly compare PLM$_4$BS and other SPML proposals. As mentioned and justified above in Section 2, these dimensions have been selected from the most predominant and common requirements in the literature. Finally, Table 6 summarizes and compares some the most important model-based SPML to know their advantages and disadvantages, together with the degree of maturity this research topic has achieved (PLM$_4$BS is also included). We just consider model-based proposal exclusively, as it is the technique used in PLM$_4$BS.

As a result of this comparison, it is possible to conclude that, even though there are many proposals for SPM, none of them is mature enough to comply with the commitment we follow. SPEM2.0 could be the solution, but its complexity and nonexecutability make us discard it for our goal. Regarding executability, many researches have been conducted to solve the main drawback it involves.[69] If a software process language is aimed at succeeding, it will be mandatory that it can fill in the gap between the process description and its execution in a real environment.

We have also identified some elements that were hidden until now in most of the studied proposals, such as usability in enterprise environments, mechanisms to carry out continuous improvement, mechanisms to establish business rules, and elements for process orchestration. All these aspects have been supported by our proposal.

Moreover, despite all the existing proposals, 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 which has been used in many projects of various areas, especially multiagent systems, software product lines, agile software development, unified process, and embedded real-time systems, among others. Also, SPEM is supported by tools, eg, EA.[53]

The reader might argue that this fact may have been caused for 2 key reasons: (i) BPMN[6] is the most popular business process modeling language among BPMS and (ii) these tools do not support any other modeling language. Thus, he/she is right. Nevertheless, some papers conclude that BPMN and UML ActivityDiagram (AD)-based proposals allow modeling equivalent models.[70] In fact, BPMN and UML AD evaluations conducted by means of workflow patterns, as defined by Russell et al,[71] revealed that both business process modeling languages provide similar solutions for most of the patterns. Consequently, it is possible to establish rule mapping BPMN and UML AD[70] without excessive cost.

This way, bringing closer the research world and the business world is very important to transfer the scientific knowledge generated in academy to software industry. In addition, our research group has broad experience in technology transfer projects. For this reason, we intend to apply our proposal to real projects, and once validated in these business environments, we will extend our metamodel with new concepts, for instance, mechanisms for defining exception paths that have not been supported by PLM$_4$BS yet.

8 | CONCLUSIONS AND FUTURE WORKS

An efficient BPM is crucial in any company to reduce costs, maximize profits, and increase productivity and competitiveness. However, applying BPM is not a simple task because management and software development processes are characterized, among others, by the following aspects: (i) they are constantly evolving; (ii) they usually incorporate new lifecycles and new technologies, they frequently comprise several iterations, and they often produce different versions of software products; (iii) they are complex because they are strongly influenced by many unpredictable circumstances and many work teams; and (iv) they often rely on communication, coordination, and cooperation of different frameworks and development technologies, as well as different roles.

For these reasons, software companies usually focus on defining their processes, although orchestration and execution processes are performed in an independent and manual way by each involved role. This fact makes software processes maintenance, evolution, monitoring, and measurement become a hard task to execute.

This paper proposes a MDE-based solution that would help to manage the conceptual complexity of the process engineering area. For this purpose, it presents a model-based approach for SPM that is framed into a new continuous improvement process lifecycle (Figure 1) that has allowed us to define the PLM$_4$BS framework.

Regarding our SPM metamodel, it is simple enough to enable using PLM$_4$BS in real environments, but allows defining processes of considerable complexity. We have suggested a straightforward language for several reasons. Firstly, our language is simple to apply PLM$_4$BS to different business environments and then check and validate it. This provides us with valuable feedback to improve our proposal. Therefore, we have successfully applied PLM$_4$BS into different real case studies to evaluate and validate PLM$_4$BS. These results let us open new lines of work to further investigate and improve the application of process management using the MDE paradigm. Secondly, we propose a simple language to facilitate the insertion of MDE-based mechanisms to PLM$_4$BS, decrease the learning curve, and reduce user's cognitive overload when users use our SPM language (this usability aspect has a major influence on user's efficiency, and, especially, with non-ICT users).

Moreover, as mentioned above, this paper supports the first phase of a process lifecycle, but we have already scheduled different future lines of work to support the remaining phases. They are explained as follows.

To begin with, we will define MDE mechanisms for measuring BP (monitoring and continuous improvement phases; see Figure 1) to provide support for the process verification and validation in PLM$_4$BS. For this purpose, papers related to the definition of software metrics[72] will be analyzed and taken into account to define the proposal and integrate it into PLM$_4$BS. At present, a PhD student is extending it as a topic of her doctoral thesis. She is defining a theoretical MDE-based framework to support the monitoring phase of PLM_4BS[43] to assess the execution of software processes.

Then, we aim to analyze how simulation mechanisms can be included in PLM_4BS so as to support decision-making procedures related to resource allocation or deadlock identification, among other aspects.

Finally, as explained previously in Section 6.5, we will continue improving aspects of PLM_4BS related to the expressiveness and understandability of process models.

ACKNOWLEDGEMENTS

This research has been supported by the POLOLAS project (TIN2016-76956-C3-2-R) and by the SoftPLM Network (TIN2015-71938-REDT) of the Spanish Ministry of Economy and Competitiveness. To conclude, I would like to show my gratitude to Dra. M.J. Escalona (University of Seville, Spain) for her valuable comments.

ORCID

Julián Alberto García-García http://orcid.org/0000-0003-2680-1327

REFERENCES

1. Han YB, Sun JY, Wang GL, Li HF. A cloud-based BPM architecture with user-end distribution of non-compute-intensive activities and sensitive data. *J Comput Sci Technol*. 2010;25(6):1157-1167. https://doi.org/10.1007/s11390-010-9396-z
2. Van-der-Aalst WMP. Making work flow: on the application of petri nets to business process management. LNCS, Application and Theory of Petri Nets. 2002;2360:1-22.
3. PMI. *A Guide to the Project Management Body of Knowledge*. ISBN13; 2008. ISBN:978-1933890517.
4. ISO/IEC.ISO 9001:2008. Quality management systems, requirements. International Organization for Standardization. 2008.
5. Trkman P. The critical success factors of business process management. *Int J Inform Manag*. 2010;30(2):125-134.
6. OMG. BPMN, business process modelling notation, Version 2.0 Object Management Group. 2011.
7. Martínez-Ruiz T, García F, Piattini M. Towards a SPEM v2.0 extension to define process lines variability mechanisms. In: *Software engineering research, management and applications*; 2008:115-130.
8. ISO/IEC.ISO/IEC TR24744:2007 Software and systems engineering lifecycle management guidelines for process description. International Organization for Standardization. 2007.
9. García-Borgoñón L, Barcelona MA, García-García JA, Alba M, Escalona MJ. Software process modelling languages: a systematic literature review. *Inform Softw Syst J*. 2013;56(2):103-116. https://doi.org/10.1016/j.infsof.2013.10.001
10. Zamli KZ, Isa NM. A survey and analysis of process modeling languages. *Malays J Comput Sci*. 2004;17:68-89.
11. Zamli KZ, Lee PA. Taxonomy of process modeling languages. ACS/IEEE International Conference on Computer Systems and Applications, IEEE, pp. 435–437, 2001.
12. zur Muehlen M, Recker JC. How much language is enough? theoretical and practical use of the business process modeling notation. *CAiSE Lect Notes Comput Sci*. 2008;5074:465-479.
13. Recker J, Indulska M, Rosemann M, Green P. How good is bpmn really? insights from theory and practice. In 14th European Conference on Information Systems, pages 1582 1593, 2006.
14. Recker J. Opportunities and constraints: the current struggle with BPMN. *Bus Process Manag J*. 2010;16(1):181-201.
15. Istoan PA. Methodology for the derivation of product behaviour in a Software Product Line. Tesis doctoral, Rennes 1. 2014.
16. Recker J. BPMN Research: What We Know and What We Don t Know. Springer 2012.
17. García-García JA, Maidan A, Vázquez A, Mejías M. A model-driven proposal to execute and orchestrate processes: PLM4BS. *Int Conf Softw Process Improv Capability Determ*. 2017;211-225.
18. Garcia-Garcia JA, Enriquez JG, Garcia-Borgoñon L, Arevalo C, Morillo E. A MDE-based framework to improve the process management: the EMPOWER project. IEEE 15th International Conference on Industrial Informatics. ISBN: 978-1-5386-0836-4. pp. 553-558.2017.
19. Ponce J, García-Borgoñon L, García-García JA, et al. Model-driven approach for business process management. *Covenant J Eng Technol*. 2013;1(2):32-52.
20. Escalona MJ, Aragón G. NDT: a model-driven approach for web requirements. *IEEE Trans Softw Eng*. 2008;34(3):370-390.
21. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. *MIS Q*. 2004;28(1):75-105.
22. Peffers K, Tuunanen T, Rothenberger M, Chatterjee S. A design science research methodology for information systems research. *J Manag Inform Syst*. 2007;24(3):45-77.
23. Curtis B, Kellner MI, Over J. Process modeling. *Commun ACM*. 1992;35(9):75-90.
24. Jaccheri ML, Baldi M, Divitini M. Evaluating the requirements for software process modelling languages and systems. Process support for Distributed Team-based Software Development (PDTSD'99), Florida, USA. 1999;570-578.
25. Armenise P, Bandinelli S, Ghezzi C. A survey and assessment of software process representation formalisms. *Int J Soft Eng Knowl Eng*. 1993;3(3):401-426.
26. Schlenoff C, Knutilla A, Ray S. *Unified Process Specification Language: Requirements for Modeling Process*. Interagency Report, vol 5910, 1996.
27. La Rosa M, ter Hofstede AHM, Wohed P, Reijers HA, Mendling J, van der Aalst WMP. Managing process model complexity via concrete syntax modifications. *IEEE Trans Ind Informat*. 2011;7(2).

28. Meier BP, Schnall S, Schwarz N, Bargh JA. Embodiment in social psychology. *Top Cognit Sci*. 2012;4(4):705-716.

29. Frakes William B, Kyo K. Software reuse research: status and future. *IEEE Trans Soft Eng*. 2005;31(7):529-536.

30. Holschke O, Rake J, Levina O. Granularity as a cognitive factor in the effectiveness of business process model reuse. In Business Process Management. BPM 2009. Lecture Notes in Computer Science, DOI: https://doi.org/10.1007/978-3-642-03848-8_17, vol 5701. Springer. 2009.

31. Ruiz-González F, Canfora G. Software process: characteristics, technology and environments. *SPT - Softw Process Technol*. 2004;5:5-10.

32. Kronz A. Managing of process key performance indicators as part of the aris methodology. In: *Corporate performance management*; 2006:31-44.

33. Wang W, Indulska M, Sadiq SW. Cognitive efforts in using integrated models of business processes and rules. In: *CAiSE Forum*; 2016:33-40.

34. OMG. MOF model to text transformation language (MOFM2T). Website: http://www.omg.org/spec/MOFM2T/1.0/ 2017.

35. Havey M. *Essential Business Process Modelling*. O'Reilly Media, ISBN13: 978-0596008437, 2005.

36. Hill JB, Sinur J, Flint D, Melenovsky MJ. Gartner's position on business process management. Business Issues Gartner, Stamford, CT, 2006.

37. Shewhart WA. Statistical method from the viewpoint of quality control. Department of Agriculture. Dover, p45. 1986.

38. Van-der-Aalst WMP. Business process management: a personal view. *Bus Process Manag J*. 2004;10(2).

39. Bosch, J. From software product lines to software ecosystems. 2009;111-119.

40. Schmidt DC. Model-driven engineering. *IEEE Comput Compu Soc*. 2006;39(2):25-31.

41. Meidan A, García-García JA, Escalona MJ, Ramos I. A survey on business processes management suites. *Comput Stand Interfac*. 2016;51:71-86.

42. OMG. Query/view/transformation. 2017 Website: www.omg.org/spec/QVT/1.0/.

43. Meidan A., García-García JA, Ramos I, Escalona MJ, Arevalo C. A model-based proposal for integrating the measures lifecycle within the process lifecycle. Jornadas de Ingeniería del Software y Bases de Datos (JISBD). Handle: 11705/JISBD/2017/053.2017.

44. Goodman N. *Languages of Art: An Approach to a Theory of Symbols*. Bobbs-Merrill; 1968.

45. Moody DL, Heymans P, Matulevicius R. Improving the effectiveness of visual representations in requirements engineering: an evaluation of i* visual syntax. 17th IEEE International Requirements Engineering Conference. 2009a;171–180.

46. Genon N, Heymans P, Amyot D. Analysing the cognitive effectiveness of the BPMN 2.0 visual notation. *Softw Language Eng LNCS*. 2011;6563:377-396.

47. Moody DL, van Hillegersberg J. Evaluating the visual syntax of UML: an analysis of the cognitive effectiveness of the UML family of diagrams. LNCS. 2009;5452:16–34. Springer.

48. ISO/IEC. ISO/IEC 19507:2012 Information technology—Object Management Group Object Constraint Language (OCL). International Organization for Standardization, formal/2012-05-09. 2012.

49. García-García JA. Una propuesta para el uso del paradigma guiado por modelos (MDE) para la definición y ejecución de procesos de negocios. PhD thesis. University of Seville. http://hdl.handle.net/11441/26740.2015.

50. Van Lamsweerde A. *Requirements Engineering: From System Goals to UML Models to Software*. 10 Chichester, UK: John Wiley & Sons; 2009.

51. Richters M. *A Precise Approach to Validating UML Models and OCL Constraints*. Logos; 2002.

52. Dzidek WJ, Arisholm E, Briand LC. A realistic empirical evaluation of the costs and benefits of UML in software maintenance. *IEEE Trans Softw Eng*. 2008;34(3):407-432.

53. SparxSystems. Enterprise architect. Website: www.sparxsystems.com.au. 2017.

54. StarUML 2. Sophisticated software modeler. Website: www.staruml.io 2017.

55. Cutilla CR, Garcia-Garcia JA, Alba M, Escalona MJ, Ponce J, Rodriguez L. Model-driven engineering applied in functional testing: the practical experience of AQUA-WS. In: *Information Systems and Technologies*; 2011.

56. Escalona MJ, García García JA, Dominguez-Mayo FJ, Ramos I. Technical tool surveys and comparative studies: a systematical approach. Information Systems Development: Transforming Organisations and Society through Information Systems. ISBN: 978-953-6071-43-2.2014.

57. Runeson P, Host M, Rainer A, Regnell B. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons; 2012.

58. Escalona MJ, Domínguez-Mayo FJ, García-García JA, Sánchez N, Ponce J. Evaluating enterprise content management tools in a real context. *J Softw Eng Appl*. 2015;8(08):431-453.

59. Lethbridge TC, Sim SE, Singer J. Studying software engineers: data collection techniques for software field studies. *Empir Softw Eng*. 2005;10(3):311-341.

60. García-García JA, Escalona MJ, Martínez-García A, Parra C, Wojdyński T. Clinical process management: a model-driven & tool-based proposal. Information Systems development: transforming healthcare through information systems. In: *ISBN: 978-962-442-393-8*; 2015.

61. ISO. ISO 13606:2008, Health informatics, electronic health record communication. International Organization for Standardization, 2008.

62. UNE. UNE-EN 13940-1:2007, Health informatics, system of concepts to support continuity of care. AENOR. 2007.

63. Sutton SMJ, Osterweil LJ. The design of a next-generation process language, in: Software Engineering – ESEC/FSE'97, Lecture Notes in Computer Science, 1301, Springer, Berlin Heidelberg. 1997;142–158.

64. Bendraou R, Jézéquel JM, Gervais MP, Blanc X. A comparison of six UML based languages for software process modeling. *IEEE Trans Softw Eng*. 2010;36(5):662-675. Lu R, Sadiq S. A survey of comparative business process modeling approaches. In: *Business information systems*. Springer Berlin, Heidelberg; 2007:82-94.

65. Zur Muehlen M, Indulska M. Modeling languages for business processes and business rules: a representational analysis. *Inform Syst*. 2010;35(4):379-390.

66. Chou S-C. A process modeling language consisting of high level UML-based Diagrams and low level process language. *J Object Technol*. 2002;1(4):137-163.

67. Di Nitto E, Lavazza L, Schiavoni M, Tracanella E, Trombetta M. Deriving executable process descriptions from UML. Proceedings of the 24th International Conference on Software Engineering ICSE 2002, 2002, vol. Compendex, pp. 155-165. 2002.

68. MG. SPEM, Software & systems process engineering metamodel specification. Object Management Group. 2008.

69. Bendraou R, Combemale B, Crégut X, Gervais MP. Definition of executable SPEM 2.0. *Softw Eng Conf*. 2007;390-397.

70. Geambaşu CV. BPMN vs. UML activity diagram for business process modelling. *Account Manag Inform Syst*. 2012;11(4):637-651.

71. Russell N, van der Aalst WM, Ter Hofstede AH. Exception Handling Patterns in Process-Aware Information Systems. BPM Center Report BPM-06-04. 2006.

72. Kumar Chhabra J, Gupta VJ. A survey of dynamic software metrics. *J Comput Sci Technol*. 2010;25(5):1016-1029. https://doi.org/10.1007/s113nse90-010-9384-3