Software process modeling languages: A systematic literature review

L. García-Borgoñón [a,b], M.A. Barcelona [a,b], J.A. García-García [b], M. Alba [b], M.J. Escalona [b]

[a] Aragón Institute of Technology, Zaragoza, Spain
[b] IWT2 Research Group, University of Seville, Seville, Spain

Keywords:
Software process modeling
Software process language
Systematic literature review

A B S T R A C T

Context: Organizations working in software development are aware that processes are very important assets as well as they are very conscious of the need to deploy well-defined processes with the goal of improving software product development and, particularly, quality. Software process modeling languages are an important support for describing and managing software processes in software-intensive organizations.
Objective: This paper seeks to identify what software process modeling languages have been defined in last decade, the relationships and dependencies among them and, starting from the current state, to define directions for future research.
Method: A systematic literature review was developed. 1929 papers were retrieved by a manual search in 9 databases and 46 primary studies were finally included.
Results: Since 2000 more than 40 languages have been first reported, each of which with a concrete purpose. We show that different base technologies have been used to define software process modeling languages. We provide a scheme where each language is registered together with the year it was created, the base technology used to define it and whether it is considered a starting point for later languages. This scheme is used to illustrate the trend in software process modeling languages. Finally, we present directions for future research.
Conclusion: This review presents the different software process modeling languages that have been developed in the last ten years, showing the relevant fact that model-based SPMLs (Software Process Modeling Languages) are being considered as a current trend. Each one of these languages has been designed with a particular motivation, to solve problems which had been detected. However, there are still several problems to face, which have become evident in this review. This let us provide researchers with some guidelines for future research on this topic.

Contents

1. Introduction

Today, software applications are used extensively to support day to day business activities in every kind of company. Organizations working in software development are aware that software processes are very important assets as well as they are very conscious of the need to deploy well-defined processes with the goal of improving the software products development and, of course, quality [3]. Due to their importance, a challenging goal for software development organizations is to find the best way of describing and managing software processes.

A software process may be defined as the set of partially ordered steps, with subsets of related artifacts, human and computerized resources, and organizational structures and constraints intended to produce and maintain the requested software deliverables [7]. This definition shows the large amount of factors that may influence software development processes. Software processes are more complex and unpredictable than typical production processes [2] as they depend on people and circumstances.

After Osterweils' statement: ''Software processes are software too'' [8], many process languages and models have emerged. Since the late 1980s, the software community has shown a growing interest in finding the best way to describe software processes to be understood and institutionalized within organizations, becoming an important research topic in Software Engineering. It achieved such a relevance that sometimes people talked about several generations of software process modeling languages [13].

Software process languages have been created from different sources, such as programming-based languages, Petri net-based languages or rule-based languages. There was no clear trend to select one as a foundation. The most recent literature offers some possibilities, such as UML, as a basis to cope with this possible de facto standardization. Therefore, as software-intensive organizations (defined as private or public organizations extensively using or developing software [4]) have not yet adopted any of these proposals in a practical sense, it may not be determined which technology should work as basis or standard.

Despite the status of the topic, the state-of-the art is not yet very clear, and the different approaches are widely spread in the literature. Previous reviews and surveys [11] [12] have been found, although they are not very recent, as well as comparisons among some specific work [2] [5]. Nevertheless, a complete view of the last years is not available. Consequently, the general aim of this systematic review is to capture the current state-of-the art in software modeling languages and identify needs and opportunities for future research work.

This systematic literature review contributes to ongoing research in the field of software process modeling in four ways: (1) by reviewing and showing all software process modeling languages that have been created since 2000, as well as their description and motivation, (2) by summarizing what problems have been addressed in Software Process Engineering through software process modeling languages, (3) by creating a new taxonomy for software process modeling languages in order to classify them by their base technology, and (4) by offering directions for future research.

This paper is structured as follows. Section 2 summarizes related work. In Section 3, the method used for the systematic review is introduced. Section 4 shows the results of the review, and then, Section 5 offers discussions on these results. In Section 6, directions for future research are pointed out and finally, Section 7 states conclusions and future work based on the findings obtained.

2. Related work

Although this topic has been discussed and studied for many years, we have found too few literature reviews and surveys in this area. We have also sought some comparison between several languages as a starting point for our review. The next paragraphs summarize the most relevant work we have found in our searches.

Zamli [11] conducted a survey about the state-of-the art of the second generation PMLs (Process Modeling Languages). Sutton and Osterweil [9] stated that a PML belongs to the second generation, if published after 1996. This work proposes a simple classification based on process enactment support. They classified a PML as non-enactable, simulated or enactable. Non-enactable PMLs only support process modeling, but not process enactment. Simulated PMLs enable a high-level simulation, but do not provide fine-grained control of the software process. Finally, enactable PMLs permit the process model to be enacted to control a software process. Besides this classification, they introduced five areas that a PML must support: modeling support, enactment support, evaluation support and human dimension support, and they tabulated support areas in accordance with the aforementioned classification in three groups.

Some years later Zamli et al. [12] studied in depth the classification of the PMLs by using the taxonomy and support areas described in their previous work, and listed features covered by each PML in a specific support area.

A comparison of six UML-based languages for software process modeling was presented in [2]. The authors selected the most predominant and common ones from the many requirements identified in the literature related to process modeling languages. Then, they evaluated those six languages against requirements and compared them.

Henderson-Sellers et al. [5] examined critically four metamodels that have been constructed to underpin and formalize methodologies. Based on this analysis, the authors proposed a new metamodel that supported both the software development and capability assessment, which was used in ISO 24744 standard.

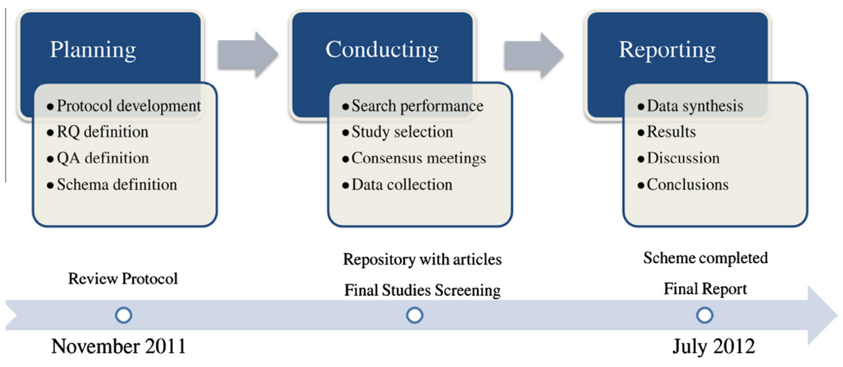

Fig. 1. Activities in the systematic review.

In summary, too few studies have conducted a systematic review of SPMLs, and those existing are neither very recent and have limited coverage. Therefore, we consider our work to be an important starting point to find which SPMLs have been proposed.

3. Method

Systematic literature review has been the method chosen to achieve the goal described in Section 1. It allows identifying, evaluating and interpreting all available research data relevant to a particular research question in a specific investigation area. The guidelines proposed by Kitchenham [6], which are among the most widely accepted in software engineering, have been followed to carry out this work.

These guidelines establish that a review should comprise three phases: planning, conducting and reporting. The planning activity deals with developing the review protocol as well as deciding how the researchers should work and interact to conduct the review. This protocol prescribes a controlled procedure for executing the review and includes research questions, search and evaluation strategies, inclusion/exclusion criteria, quality assessment, data collection form and methods of analysis. The second phase focuses on executing the protocol as it has been defined. Finally, the last phase describes how the final report has been elaborated.

Fig. 1 outlines all the activities included in each phase. They will be described in detail in the following sub-sections.

3.1. Research questions

Defining and describing the software process is not a new problem. Over the years, different approaches, languages and metamodels have been proposed to describe processes with a unique goal: to find the best way of representing the software process to obtain a successful application in industry. Due to this goal concerning understanding the existing research proposals within software process modeling, a general research question (RQ) was defined: “What is the state-of-the art in software process modeling languages?”.

As this question was too general, it was reformulated into these three more specific questions, which guided this research work:

- RQ1. What software process modeling languages have been defined? Why?
- RQ2. What is the current trend when selecting a base technology to define an SPML?
- RQ3. What are the limitations of the current research?

3.2. Search strategy

An exhaustive search for papers was carried out to answer these research questions. It was mainly focused on major digital libraries since journal articles, workshop and conference papers were the objective to be covered.

First of all, the keywords for the search were selected. As this selection was known to be relevant for the quality of results, general terms were used with the aim of confirming that most of the research papers were included in the study.

After some pilot testing, the final search strings consisted of the following Boolean expression “*A* **AND** *(B1* **OR** *B2* **OR** *B3)* **AND** *(C1* **OR** *C2)*” where search expressions are represented in Table 1 as follows:

The search was carried out through the following consulted databases: ACM Digital Library, Ei Compendex, IEEE Xplore, ISI Web of Knowledge, Science Direct, SCOPUS and Wiley InterScience Journal Finder. Both, an excel file to store the completed searches and Jabref [1], a reference manager tool which helped us to manage the references and perform the systematic search, were used to manage results.

3.3. Study selection and the inclusion and exclusion criteria

The study selection process was performed in 6 phases as outlined in Table 2. The goal of such a selection process was to identify the relevant articles that may match the objectives of the systematic literature review. The search strings were too general and it was expected that not all studies found would be chosen for the final phase in the selection process. JabRef [1] supported the study selection, which allowed the team to manage more efficiently the duplicate references among databases and generate an integrated file with the first version of results.

In this process, there were two consensus meetings where all researchers jointly decided whether the studies were considered relevant for the work. The first one focused on the abstracts, keywords and titles, whereas the second was based on the full text. The guidelines of a systematic review strongly recommend the participation of several researchers in the process, and this was evidenced by less subjective decisions and the fact that researchers jointly made decisions according to the agreed criteria. The systematic review protocol selected explicitly defines inclusion and exclusion criteria, as it is shown in Table 3.

Table 1
Search expressions.

A. Software process	B1. Description	C1. Metamodel
	B2. Definition	C2. Language
	B3. Modeling	

Table 2
Inclusion phases.

Phase	Relevance analysis phase description	Involvement
P1	Search-based studies selection	Leading researcher
P2	Screening: exclusion based on date	Leading researcher
P3	Screening: exclusion based on titles, abstracts and keywords	Two researchers
P4	Consensus meeting	All researchers
P5	Relevance analysis: exclusion based on full text	All researchers
P6	Consensus meeting	All researchers

Table 3
Inclusion and exclusion criteria per phase.

Phase	Inclusion/exclusion criteria
P1	Not duplicated
	Only published work
	Contains the search strings
P2	Date of publication after 2000
P3	Not editorials, prefaces, discussions
P4	Not summaries of tutorials, workshops or panels
P5	Only English
	Full text obtained
P6	New language or a language modification proposed
	Neither surveys nor reviews

3.4. Quality assessment

A questionnaire, which had to be filled in for each included paper, was elaborated with the purpose of assessing the quality of the obtained studies. Three possible answers could be chosen for each question, *yes*, *no* or *partially*.

Table 4 shows both the quality assessment questions and the criteria described to evaluate them.

3.5. Data collection and analysis

A data collection form was defined to extract the most relevant information from the selected studies and facilitate the process of analyzing the compiled data. From now on, this form will be referred as schema, as it is shown in Table 5. In a further review, all researches evaluated the selected studies and completed the schema. After that, all participants discussed and agreed on all issues in the final data collection.

4. Results

This section presents the results of the systematic literature review. On the one hand, the search results are analyzed and on the other hand, quality evaluation results are shown.

4.1. Search results

Once the protocol was defined, it was executed. First of all, the selection process was performed with the aim of identifying the relevant papers for the systematic review. As mentioned above, the search strings were so general that they influenced the search results.

As there is no standard way of conducting searches for all search engines, we executed a set of manual queries and integrated the results in each search engine. Fig. 2 represents firstly, the papers that were retrieved from each search engine after all queries were executed and secondly, the number of different studies that were collected from each search engine after removing duplicated entries for the same search engine. Finally, we show the number of papers that were included in this work following the inclusion criteria defined.

Fig. 3 shows the number of studies finally included in our analysis and retrieved from the search engine divided by the number of studies selected from all search engines. It can be observed that Ei Compendex and SCOPUS provided us with more than 60% of the studies. The second value represents the studies finally included in our analysis and retrieved from the search engine divided by all different studies retrieved from the same search engine. It shows that most of them include 10% of the results approximately.

Those studies that were included and found in various search engines have been also included in Figs. 2 and 3,, to avoid penalizing any search engine. Fig. 4 shows the number of search engines

Table 4
Quality assessment questionnaire.

Quality question – score
QA1: Did the study make a review of previous research for the topic?
• Yes: it either extensively compared previous research with a contextual situation or mentioned different generations in software process languages
• Partially: it only mentioned a few previous researchers and did not establish a clear background for the topic
• No: it did not mention any previous research
QA2: Did the study mention a base technology for its proposal?
• Yes: it explained which was the starting point and why
• Partially: it commented on the support base, but did not explain why
• No: it started from scratch without any justification
QA3: Did the study show the further continuous research?
• Yes: it showed future research in the field of the software process modeling language
• Partially: it showed only future work on its proposal
• No: no future research work was shown

Table 5
Data schema.

Basic info	It means title and author
Publication info	It refers to book, journal, conference or technical paper where the study was published
Year	It makes reference to the year when the study was published
Language defined	It deals with the name of the language defined in the study, the problem addressed, its description and motivation
Based-on	It refers to the technology or reference language on which this new language is based and the advantages and disadvantages of other alternatives
Related work	It lists the name of studies cited and whether the study has a state-of-the art section
Cited by	It identifies the study and whether it is referenced explicitly in other selected studies
Future work	It proposes future work and challenges shown in the study related to the research questions

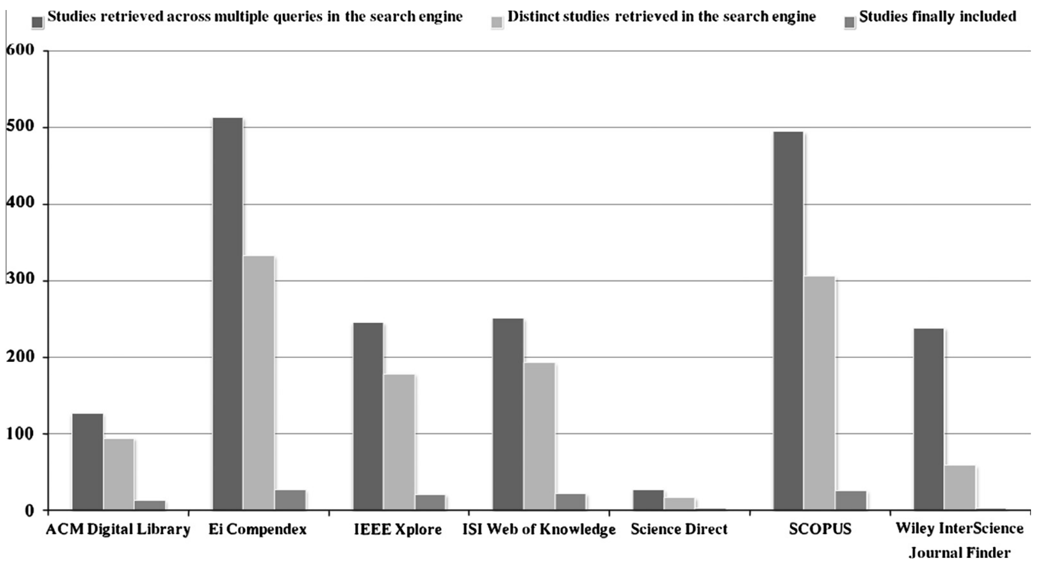

Fig. 2. Studies retrieved through search engines.

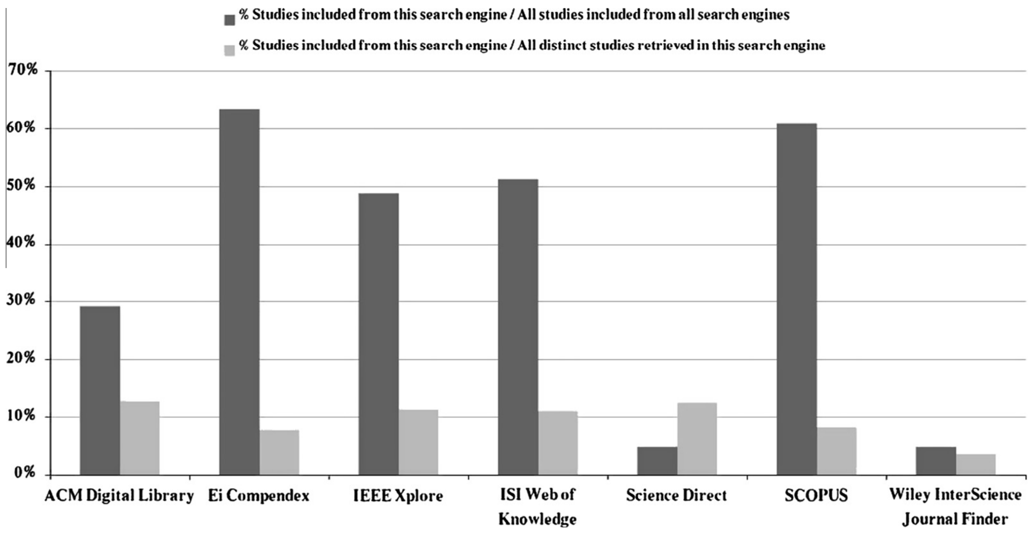

Fig. 3. Analysis of retrieved results from search engines regarding the total studies included.

that found studies included in our analysis. It can be observed that 56% of the studies were found in one or two search engines, and the remaining 44% has been obtained from three or more of them. None were found in more than 5 search engines.

The process was carried out according to the six phases previously presented, whereby the number of selected papers decreased. Fig. 5 and Table 6 show the number of included and excluded studies per phase. All included references are listed in the References section. Excluded studies are not mentioned due to space restrictions.

4.2. Quality evaluation

The included studies underwent a quality evaluation by means of the questionnaire presented in Table 4, with the aim of measuring their degree of representativeness that may enable obtaining relevant conclusions. The three quality questions were rated for each included study in accordance with the criteria established in the quality questionnaire. After that, responses to the quality questions were discussed in order to find their degree of coverage.

Fig. 6 shows the coverage of every QA in the studies included. It shows that QA1 and QA2 were covered in a rate higher than 80% by *Yes* or *Partially* answers. In contrast, QA3 has less coverage. All of them were covered at least at 70% by *Yes* or *Partially* answers.

5. Discussion

This section will discuss one by one the answers to the three research questions defined as the target to fulfill this systematic literature review. Then, the weaknesses of this study will be pointed out.

5.1. RQ1. What software process modeling languages have been defined? Why?

Fig. 7 is a summary of the SPMLs defined since 2000. Thus, every year, it includes the languages that have been reported.

After this first snapshot, in which all relevant SPMLs have been identified, the following Tables 7–10 offer a short description of each SPML and the motivation for which it was created.

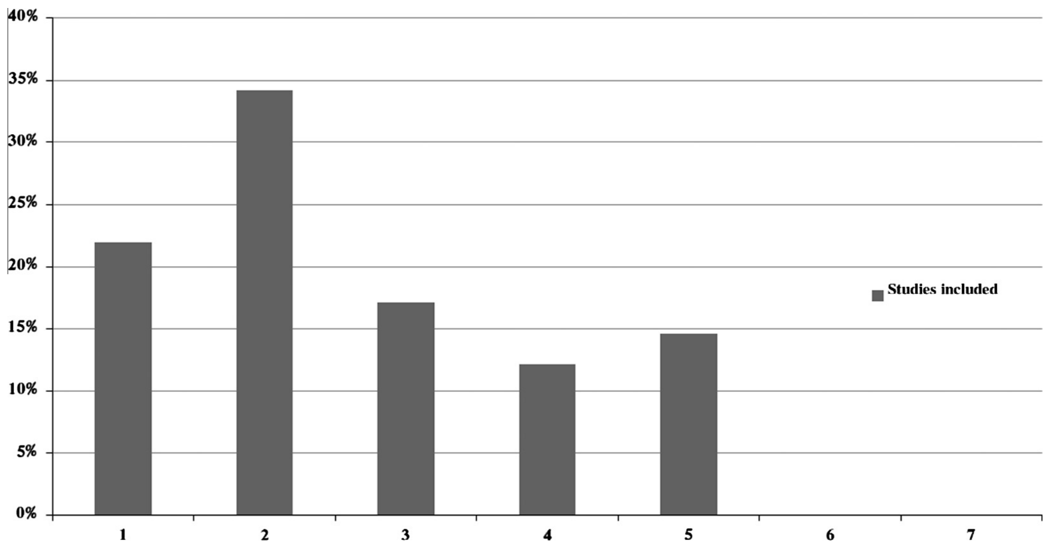

Fig. 4. How many search engines found the studies included in our analysis?

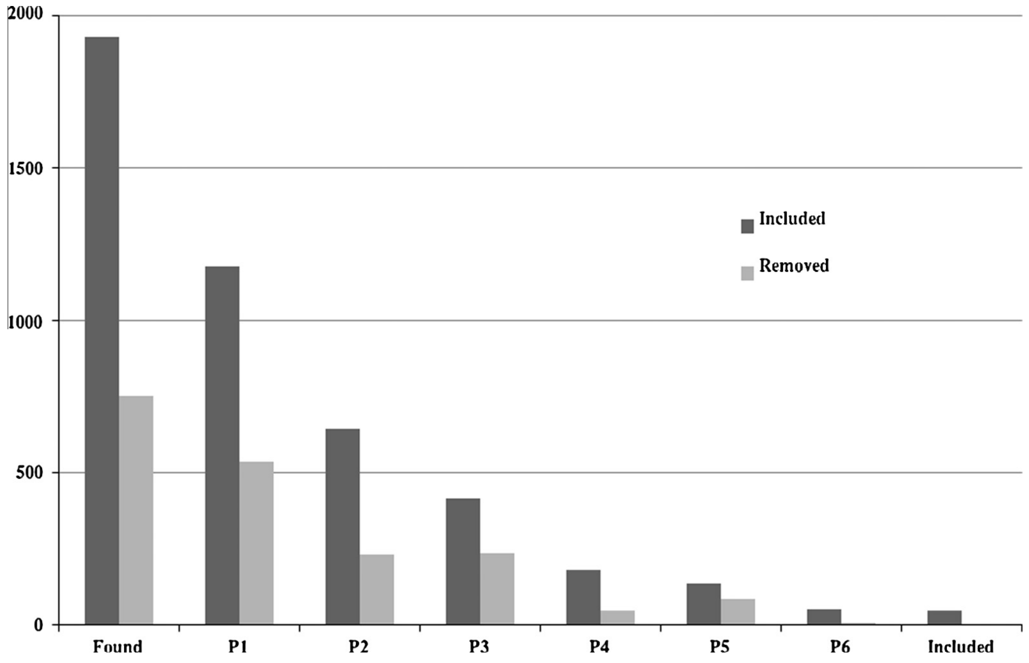

Fig. 5. Studies removed in the revision phase.

Table 6
Included and excluded studies per phase.

Phase	Included	Removed
Found	1929	752
P1	1177	534
P2	643	230
P3	413	233
P4	180	48
P5	132	82
P6	50	4
Included	46	0

Table 11 summarizes the main problems faced to answer the first research question. As this table shows, problems appointed in these papers can be grouped in four major topics: support for static process documentation, support for analysis and management of processes before their enactment, dynamic process support (automation and flexibility) and support tools and environments for software processes.

5.2. RQ2. What is the current trend when selecting base technology to define a SPML?

We aim to find how SPMLs have been classified in the literature to answer this question. Historically, some SPML generations have been considered as a way to classify them [2]. The first generation relates to those software process modeling languages which are Petri nets-based, rule-based or programming language-based. These focus on process execution and formality, which make them become complex, inflexible and difficult to understand. The second generation coincides with the moment when UML became mature as a standard language in the software industry. The idea is to

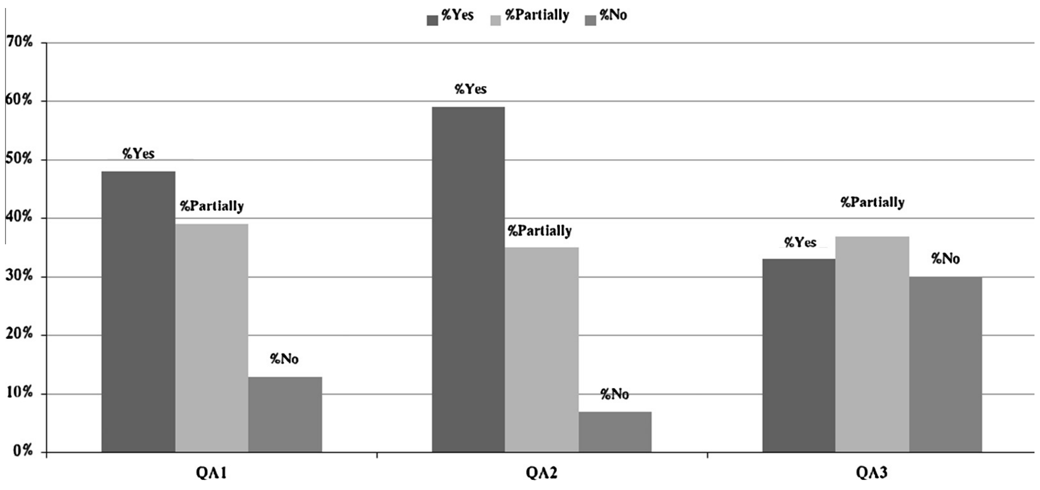

Fig. 6. Quality assessment results per question and type of assessment response.

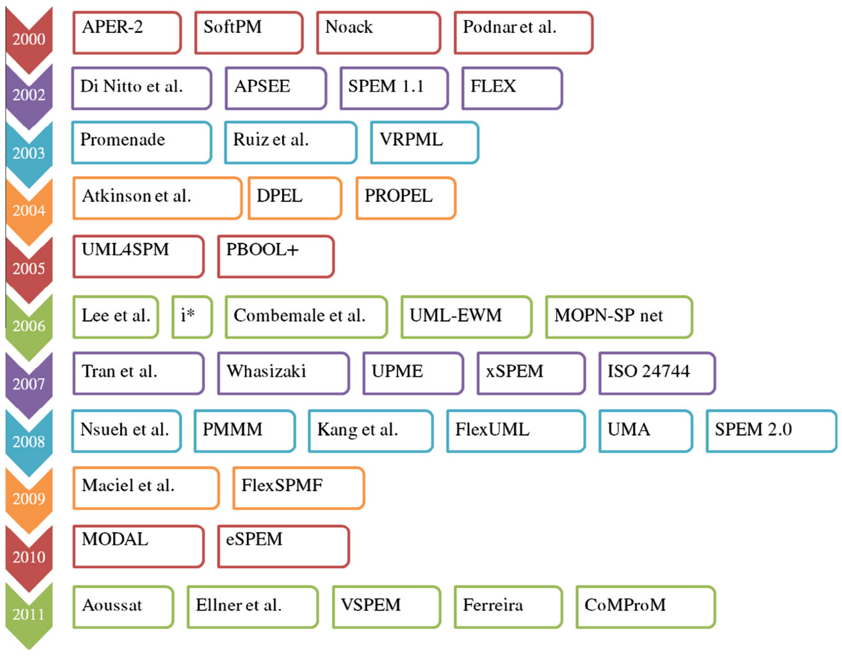

Fig. 7. Summary of software process modeling languages.

explore the possibility of using it as a process modeling language. Some UML-based languages or approaches have appeared, mainly characterized by expressiveness and lack of formality, which have influenced the possibility of running (enacting) the process models.

However, by checking the results on Tables 7–9, we conclude that to talk about generations of languages is not particularly appropriate, since although more second-generation proposals have been launched from some years ago up to date, there are recent proposals which can be classified as first-generation languages.

Thus, a new taxonomy for SPMLs has been established. For this, we have considered the base technology used in the SPML development. We have structured SPMLs in three groups according to the results of the review, as can be seen in Fig. 8. The first group of grammar-based languages includes all studies that focus on formal languages, mathematical and programming, by means of rules or restrictions. A second group contains several versions of UML-based SPMLs, and finally, the last group includes metamodel-based SPMLs or DSLs.

Fig. 9 outlines the correspondence among each SPML obtained in RQ1 and the aforementioned groups. Following the proposed taxonomy defined in Fig. 8, grammar-based SPMLs are shown with no background color or frame, UML-based SPMLs are shown with background color and model-based SPMLs are shown with a frame.

A relevant issue for us was to identify why a base technology was chosen in each SPML. Thus, Tables 12 and 13 summarize the main advantages and disadvantages the authors highlighted regarding the base technologies.

In light of this, it can be concluded that UML has been considered a suitable base technology, since it constitutes a standard in Software Engineering. However, its weakness is the lack of capability to execute processes, although some UML diagrams can be executed (simulated) if an execution semantics are assigned to them. Considering this fact and the temporal view of SPMLs, the trend

Table 7
Description of existing software process modeling languages.

Year	SPML [Ref.]	Based on	Description	Motivation
2000	APER-2 [25]	PML	It is a developer-centered (people-centered) language to facilitate the calculation of the workload and model multiple concurrent activities	There were only product-centered and activity-centered languages
2000	SoftPM [49]	Petri-Nets	The paper proposes a PSEE (process-centered software engineering environment) based on a high-level Petri net formalism for process modeling and exploits a multi-level modeling mechanism for software processes representation	There was no PSEE that has different ways to represent processes (easy to do and not ambiguous) and in addition to preserving formality
2000	Noock [50]	UML	The language proposed in the paper is a UML-centered approach to manage knowledge in a large software development organization	There was a gap regarding modeling a process that integrates engineering and management practices, in order to get manageable tasks, exchangeable components and comparable projects
2000	Podnar et al. [52]	PML	The paper proposes a specification and description language-based approach (a formal object-oriented language for communicating systems) for software process modeling	Software processes had similar characteristics to real-time systems (since a timely response to external stimuli is required), so that an ITU-T standard was used
2002	FLEX [24]	PML	It is an object-oriented, rule and constraint-based language that supports semantic richness, simplicity of use, flexibility, scalability, re-usefulness and distribution. It is also analyzable, executable and evolutionary	Existing process models were too fine-grained, so their understandability and reuse were difficult
2002	Di Nitto et al. [29]	UML	The papers proposes a language that offers the possibility of applying a subset of UML as an executable PML by transforming UML models to enactable workflow models	UML was conceived as non-executable, semi-formal language, but it was very popular, so the authors tried to achieve process enactment through UML
2002	APSEE [43] [53]	Graph PML	It is a graph grammar-based PML that provides support for dynamic changes (flexibility) in enacting processes	Existing PSEE did not offer flexibility for process enactment and integrated support for tools and services to automate process analysis and improvement (i.e. simulation and reuse facilities)
2003	PROMENADE [33]	UML	The language proposed in the paper covers process reuse (the ability to construct new processes by assembling those which have been already built) and process harvesting (the ability to build generic processes that may be further reused from existing ones)	Existing SPMLs had some limitations: expressiveness, standardization, modularity and flexibility and the authors tried to improve them
2003	Ruiz et al. [54]	UML	It proposes a language for using MOF and XMI to represent software processes	There was no concrete technological proposal applied to SPML
2003	VRPML [59]	Graph PML	It is a control-flow PML based on graph theory focused on visual modeling	Existing SPMLs lacked dynamic creation and assignment of tasks and resources, support for awareness and visualization issues (to have enactable PMLs)

Table 8
Description of existing software process modeling languages (cont).

Year	SPML [Ref.]	Based on	Description	Motivation
2004	Atkinson et al. [15] [16]	PML	It is a control-based PML that tries to cover simplicity, flexibility, expressiveness and enactability	The authors tried to support evolutionary model development and verification
2004	DPEL [26]	PML	The paper proposes a decentralized process enactment model to build a distributed PSEE	Existing PSEEs were centralized, so that they presented a single point of failure and a bottleneck related to process enactment
2004	PROPEL [37] [19]	UML	The paper proposes a language for providing concepts attending to a semiformal description and relation among process patterns	The authors tried to support flexible processes through process patterns
2005	UML4SPM [18] [19] [22] [20] [21]	UML	The language proposed in the paper is a MOF-compliant metamodel that has four objectives: expressiveness, understandability, precision and executability	The authors tried to cover the limitations of OMG SPEM 1.1 which had not yet reached the required level for the specification of executable models
2005	PBOOL+ [55]	Graph PML	It is a graph PML focused on reusing software processes through process components and process patterns	The authors studied the problem of characterizing reusable process components
2006	Lee et al. [42]	UML	The language proposed in the paper is a UML-based metamodel approach for task assignment policy in software process	There was no proposal regarding task assignment policy in software process
2006	i*	PML [23]	The paper proposes a language for using both an agent-oriented language to model processes and a goal-driven procedure to design them	There was no proposal regarding agent-oriented technologies applied in software process modeling
2006	Combemale et al. [27]	SPEM 1.1	The language proposed in the paper is a SPEM 1.1 extension specialized on semantics	The authors tried to cover the limitations of OMG SPEM 1.1 which only partly formalized the semantics and did not help building a process model
2006	UML-EWM [28]	UML	The paper proposes a language to cover workflows processes modeling	There was no proposal that established a mapping between a development methodology and a workflow process
2006	MOPN-SP-net [34] [35]	Petri-Nets	The language proposed in the paper is a multi-view software process model-based on multi-object Petri nets	The authors argued that both Petri-nets and the multiple views of a software process seemed to be similar
2007	Tran et at [56]	UML	The language proposed in the paper allows an explicit representation of process patterns in process models	The authors tried to cover building and also improving process models
2007	Washizaki [57]	SPEM 1.1	The paper proposes a language that deals with stating clearly the commonality and variability in process workflows when they are modeled as UML activity diagrams	Existing proposals were not always oriented toward overall optimization and did not lead to generally applicable process model structures

Table 9
Description of existing software process modeling languages (cont).

Year	SPML [Ref.]	Based on	Description	Motivation
2007	UPME [58]	UML	The language proposed in the paper allows modeling software processes in three steps: metamodel, model instantiation, and model compilation in order to translate them into object-oriented code skeleton (s) for process enactment	The authors tried to cover complexity reduction through process reuse by using metamodeling
2007	xSPEM [17]	SPEM 2.0	The language proposed in the paper tries to extend SPEM2.0 to allow process models to be checked through a mapping with Petri nets and monitored through a transformation into BPEL	The authors provided a definition of an Executable SPEM 2.0
2008	Hsueh et at [38]	UML	The paper proposes a language to define, verify, and validate an organization's process	The authors proposed an approach to validate processes in a simulation environment
2008	PMMM [39]	Powertypes	The paper proposes a language for Domain-Specific Process Management	The authors tried to separate general process modeling principles from domain specific languages
2008	Kang et at [40]	XML	It is an evolutionary process component description language	There was no systematic method for describing a software evolution process component
2008	FlexUML [47]	UML	The language proposes a two-step approach to model controlled flexibility in software processes	There was a need of process participants being able to express and control the amount of flexibility allowed in those processes
2009	Maciel et at [44]	SPEM 2.0	The paper proposes an integrated approach for MDA process modeling and enactment based on some SPEM 2.0 concepts specialization	There was a lack of standard terminology and notation addressing design aspects of a MDA process because tools were only focused on defining and executing model transformations
2009	FlexSPMF [48]	SPEM 2.0	The paper proposes a framework for modeling and learning flexibility in software processes	The authors tried to improve process flexibility when dynamic activities that had to evolve to cope with changes occur
2010	MODAL [41,51]	SPEM 2.0	The language proposed in the paper is a SPEM extension to improve co-design process models	There was a lack in current PML to achieve integration of MBE (Model Based Engineering) into system and software process models
2010	eSPEM [31]	SPEM 2.0	The language proposed in the paper is a SPEM Extension for Enactable Behavior Modeling	There was a lack in SPEM because it covered a rather coarse description of processes behavior. Support for a more fine-grained behavior model and life-cycle modeling was missing

Table 10
Description of existing software process modeling languages (cont).

Year	SPML [Ref.]	Based on	Description	Motivation
2011	Aoussat [14]	SPEM 2.0	The paper proposes a SPEM extension that defines explicit software process connectors that facilitate, adapt and control software process interactions	The authors tried to cover the limitations of OMG SPEM 1.1 regarding architectural concepts for software process modeling based on software architectures
2011	Ellner et al. [30]	OMG's FUML	The paper proposes a FUML-based distributed execution machine for enacting software process models	FUML was insufficient to execute software process models to drive realistic projects with large and geographically spread teams
2011	Ferreira [32]	UML	The paper is a modeling approach for software process design and implementation dealing with increasing the size and complexity of large process systems	There was no proposal regarding the incremental size and complexity of processes
2011	vSPEM [45] [46]	SPEM 2.0	The paper proposes a language that models variability in software processes	There was no variant-rich SPML so that a process can be customizable to specific project goals and environments
2011	CoMProM [36]	UML	The paper proposes a component-based language to automate the software development processes allowing the flexibility to take processes as components	There were many current approaches offering processes as components but they failed to provide the execution semantics for these process components

deals with using metamodels as base technology when generating new SPMLs.

5.3. RQ3. What are the limitations of current research?

The aim of this question is to reflect on the state of the current research in software process modeling language area, and this is going to be revised from two points of view. We are going to consider data collected from "Future work and Conclusions" sections appearing in papers included in this study. Table 14 summarizes them, grouping those that are common to several papers. Table 15 shows the open issues to be addressed in the future. We have decided to group them into related subjects, as issues to take into account when considering new proposals: to integrate software process models with decision support tools, to improve process evolution, to verify usefulness in practice, to reach consensus on what topics should be covered in software process modeling and how technologies could be integrated into software process modeling.

A proposal that may allow establishing both, a modeling and execution environment, maintaining suitable levels of understandability, would result in an important alternative in this area, also entailing the possibility of being applied in software-intensive organizations.

5.4. Limitations of our study

The systematic literature review guidelines proposed by Kitchenham [6] have been followed in order to carry out this work which has been supported by a pre-defined study protocol and continuous interaction among authors. Nevertheless, it has some limitations.

5.4.1. Search strategy

The first important activity to undertake this work was to define the search strategy, that is, where we must look for and what keywords must be used. Clearly, the set of search engines used determined the overall number of studies on which then we applied the

Table 11
Description of problems dealt with existing SPMLs.

Problem description	Refs.
To document processes in order to improve communication among people in real situations	[49]
To manage knowledge in a large software development organization	[50]
To provide a high-level representation to non-experts, so that it can be easy to use	[24]
To provide an adequate process representation that is vital for software organizations: understandable for people and formalized to avoid ambiguity	[49]
To define, verify and validate an organization's process	[38]
To create a formalism to represent and exchange software processes	[21]
To extend the abstraction level to improve model understandability and reduce model maintenance effort	[38]
To allow software process analysis and simulation	[52]
To control the human participation in software development activities	[43,53]
To support a more fine-grained behavior model and life cycle modeling	[31]
To model a process that integrates engineering and management practices	[50]
To exchange and subcontract results and components	[50,52]
To reuse software processes	[24,37,55,56,58]
To validate a process before enactment	[15,16]
To facilitate software process management automation	[43,53]
To apply the idea of software product lines to software processes	[57]
To include architectural concepts in software process modeling	[14]
To model controlled flexibility in software process	[47]
To deal with the increasing size and complexity of large systems of processes	[32]
To support software process tailoring to meet the specific goals and characteristics demanded by organizations and projects	[45,46]
Existing MDA tools are focused on code generation but other activities in a software process are usually not considered	[44]
To achieve integration of MBE into system and software process models	[41]
To support enactment by exploiting the full potential of MDE through the use of models	[51,36]
To create a de facto standard to model and enact software process	[59]
To define a standard terminology and notation addressing design aspects of an MDA-based process	[44]

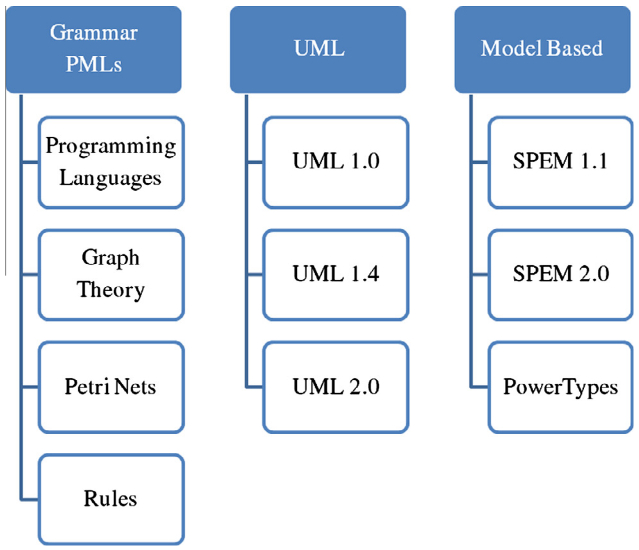

Fig. 8. Classification of formalisms for SPML.

inclusion and exclusion criteria. For this reason, although we have included seven search engines (ACM Digital Library, Ei Compendex, IEEE Xplore, ISI Web of Knowledge, Science Direct, SCOPUS and Wiley InterScience Journal Finder) that we considered relevant to this topic, they are not exhaustive and therefore, they limit the work performed. In this sense, although increasing the number of search engines does not imply an improvement on the basis of publications retrieved, the more search engines we selected, the bigger the number of initial papers we identified. A further improvement would be to conduct an analysis to determine which search engines best fit the field of software process modeling languages.

In addition, once a search engine has been selected, the results were determined by the combination of keywords and fields which can be used in the search. In this sense we have tried to maximize the search by not discarding any relevant publication at an early stage.

A special way to expand the search process would be to go from the included studies either backwards using the reference lists of these publications or forward by looking at citations of these publications, following a snowballing approach proposed in [10].

5.4.2. Study selection

Another activity that may limit the results is related to the process defined to choose which items were relevant to our study. First, we decided that only work published in English would be included. Consequently, other publications (such as any technical report, editorial preface, discussion, summary, tutorial or panel), studied out of the scientific scope or published in languages other than English, were excluded from this study. Expanding the search towards general search engines could give us more publications. Nevertheless, we concluded that due to the number of retrieved studies and attending to the goals of this work, they should have no effect on the results.

Furthermore, we established post-2000 as inclusion criterion. We knew that Zamli [10] [11] had already conducted a survey about the state-of-the art of PMLs in 2001, that is the reason why we considered that all SPMLs that were created before 2000 were included in this work. If this criterion would have been expanded, more languages could have been included. In our opinion, older publications are not so relevant for answering the research questions, specially "Which is the current trend when selecting a base techonology to define a SPML?", because some paradigms did not exist before 2000.

5.4.3. Quality assessment

We defined a Quality Assessment Question for each research question with the aim of assessing whether the papers included give us enough information to draw conclusions regarding the defined research questions. For each question, we evaluated a *Yes*, *Partially* or *No* answer and we have concluded that at least 70% of the QAs were covered by *Yes* or *Partially* values. This is a subjective method of evaluation, as it depends on each evaluator, therefore consensual decisions were reached on values. For this reason,

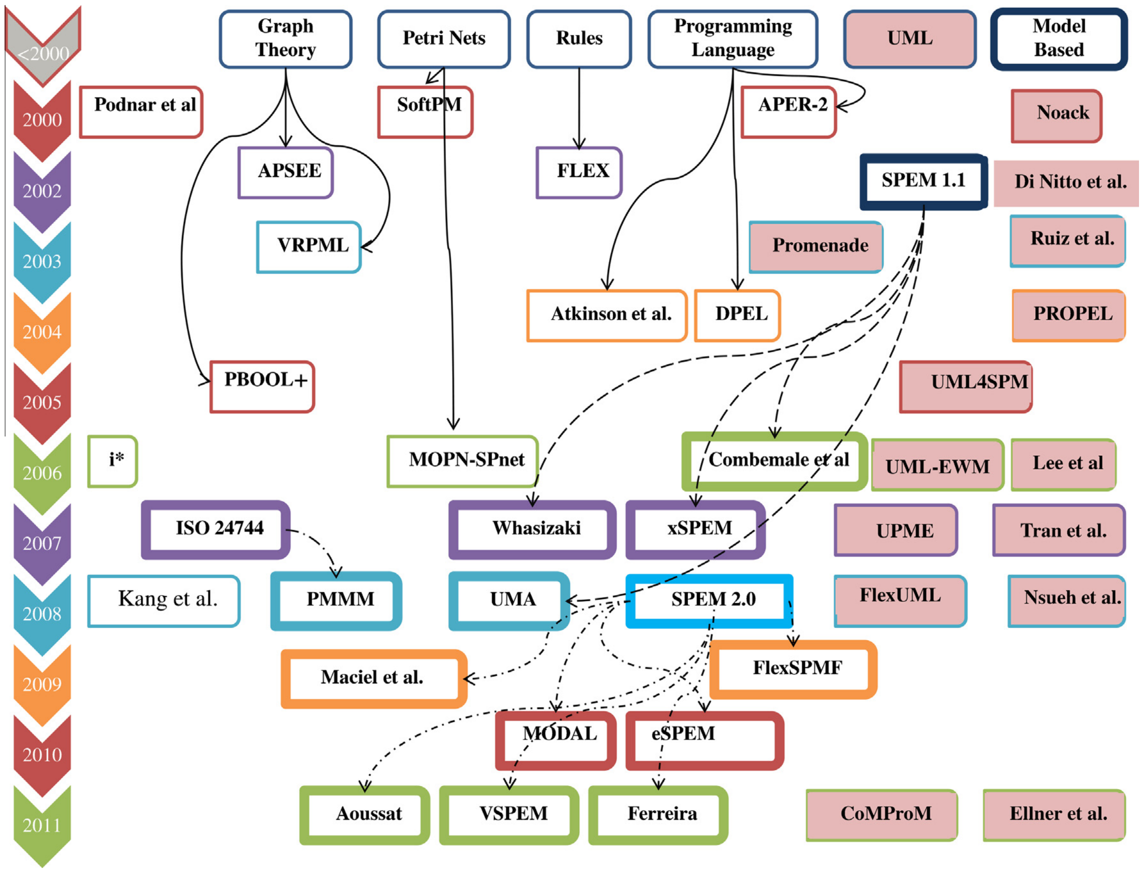

Fig. 9. Relations and base technology of existing software process modeling languages.

Table 12
Advantages and disadvantages of base technologies to create SPMLs.

Paradigm	Advantages	Disadvantages
Grammar PML	• Precise syntax and semantics [52]	• Difficult to use for modeling a software process [49] • Executable semantics missing [24] • No global functional and behavioral view of process model [24] • Difficult to use by humans [29] • No support for distributed processes [29]
Petri Net	• Graphical representation [49] [52] • Powerful means of representing the static structure and the dynamic properties of software process [49]	• Only understandable for simple processes [49,52] • Inadequate for describing model entities [52] • Difficult for people to use it [29] • No support for distributed processes [29] • Too complex for a general process enactment role of a software development unit [38]
Rules	• Precise syntax [52]	• Difficult to use for people [29] • No support for distributed processes [29] • Order of tasks in the process cannot be controlled [15]

a Partially score may occasionally represent a form of consensus involving different points of view.

Defining a larger number of Quality Assessment Questions associated with each research question would provide a more realistic background on the quality of the selected papers in relation to those questions. We realized that it limited the scope of the work, but without altering the results or conclusions of the study.

5.4.4. Data extraction and author bias

Finally we noticed that in our review process we aimed to have a larger number of primary studies in a first phase, not to rule out any study that might be relevant. We gathered more than 1000 studies to be reviewed, then, to be able to approach this task, the first criterion of exclusion (P3) was based on titles, abstracts and keywords. In this sense, if an initially retrieved study was unrelated to the Software Process Modeling Language topic in its title, abstract or keywords, it would be left out. A way of improving this procedure would consist of analyzing all retrieved articles based on their full text. It is also important to note that papers were evaluated by people who, based on their knowledge, rated each of them with the defined schema. Although we arranged peer reviews and consensus meetings, author bias is certainly an associated risk that

Table 13
Advantages and disadvantages of base technologies to create SPMLs (cont).

Paradigm	Advantages	Disadvantages
UML	• Popularity and attractiveness for software engineers (not another formalism to be learned) [29] [33] [58] • The most widely used modeling language [18] as a standard [29] • Graphical representation [29] • Extensibility [29] [38] • Supported by tools [29] [18] • Easy to comprehend [29] [58] • Supports model-based approaches [58] • Different views to present a process [38] • OCL supports model verification [38]	• Non-executability [29] [21] • Lack of formal semantics [29] • Not interpreted by machines (neither sufficiently precise nor detailed) [29]
Metamodel-based	• Separation between general process modeling principles and domain specific languages [39] • Modeling patterns implementable (powertypes and type/usage concept) [39]	

Table 14
Future work in studies.

SPM future work	Study future work	Refs.
To get executable UML models	To use the approach to model other processes such as RUP	[29]
To verify the usefulness of a model	To develop a scheduling engine	[42]
To reuse a software process	To use on heterogeneous enactment and modeling systems	[53]
To develop a standard SPML	UML4SPM model execution, ATL or QVT-based model transformations	[18,19,22,17]
To enact a process model described in SPEM	To describe operational semantics for metamodels	[27]
To build process models by reusing process patterns	To implement the proposed metamodel as a UML profile	[56]
To create process modeling applications in a distributed environment	To develop CMMI-related rules to verify an organization's processes	[38]
To provide round-trip process flexibility evolution	To integrate FlexEPFC to Jazz process engine	[47]
To allow a process definition to be modified at runtime	To keep on using Kermeta and aspect-oriented modeling techniques	[20,21]
To integrate with an approach for model-driven testing	To develop new case studies to perform further evaluations involving quantitative assessment	[44]
To implement process components effectively	To allow customization of existing process components	[41]
To evaluate empirically computer-aided process enactment impact on real development projects	To fully implement eSPEM's concrete syntax and additional tooling	[31]
To support process evolution	Develop the operational semantics in the FUML extension	[30]
To achieve process institutionalization (tailoring a process before representation)	To verify process usability and applicability to real processes	[45]
To handle the verification of process intention achievement	To develop a method to capture and manage snapshots of process instances to enable analysis and possibly process mining	[51]

Table 15
Description of open issues for future research.

Problem description	Refs.
SPM as a decision support tool (SPM and simulation technologies integration)	[52]
To integrate SPModels with support tools	[31]
To improve traceability support for artifacts	[31]
To improve process flexibility through process patterns support	[37]
To improve process evolution	[31]
To provide round-trip process flexibility evolution	[47]
To improve process flexibility through reflection support in SPMLs	[18]
To verify usefulness by measuring software development quality and productivity in practice	[57]
To develop quantitative studies and analysis in the area of SPM	[52]
To verify usefulness by measuring variability mechanisms and understandability in practice	[45,46]
The effect of using the formalism was not empirically evaluated	[52]
To develop empirical validation with industrial usage	[23]
To verify usefulness by measuring process diagram understandability in practice	[45,46]
To evaluate empirically the impact of computer-aided process enactment on real development projects	[31]
To achieve process institutionalization (tailoring a process before representation)	[45,46]
To get a deeper understanding of the process variability requirements in organizations and their fulfillment by process variability mechanisms	[45,46]
To develop PSEE for MDA software processes	[44]
There is no overall consensus on what topics should be covered in SPM	[50]
There is no overall consensus on how to describe and integrate technologies into a SPModel	[50]
There is no concrete approach for a semantically rich PML	[51,36]

can be avoided when assessing the contributions of each article. Involving a larger number of researchers in the review of each study could improve this work.

6. Directions for future research

There are several opportunities for further work. We suggest some lines to keep on working on this issue, according to the literature review and the answers to RQs, specially if we take into account that some problems are still unsolved.

Firstly, a topic that can be identified in almost all reviewed papers is the need for a larger number of empirical studies. In such studies, it would be necessary to put into practice the proposals in software-intensive organizations. This way, we would be able to know the results of usage and the main barriers as well as obtain an important feedback about these organizations' needs. This kind of work is called a longitudinal in-depth study.

Secondly, it would be very interesting to address transversal studies, by exploring the relationship among SPMLs with the rest of the fields in the Software Process Engineering area like software process improvement, simulation and orchestration.

Finally, but very closely connected to the previous issue, it would be useful to create an integrated environment where languages and tools were incorporated, so that software-intensive organizations would have the capacity for developing and executing software processes.

7. Conclusions and future work

Software process modeling is a well-known topic in the Software Engineering research that has been studied during the last twenty years. Researchers in this area have addressed the problem of describing and using it, focusing on the idea that a software process is a specific kind of software.

Different work proposals on this purpose, based on different technologies, have appeared along the years. Petri nets, rules, programming languages, graph theory, and more recently UML, have been used as support to define software process modeling languages. In this review we presented software process modeling languages that were recently developed, in order to know what the state-of-the art is, the motivation for developing these languages and the problems they address. We also aimed at learning whether there was a particular trend to create new languages in the last ten years. Now, after analyzing them, we can conclude that those model-based languages have been considered as the path to follow.

Despite several problems being tackled, some others still need to be addressed, such as integrating software process models with decision support tools, improving process evolution, verifying usefulness in practice, reaching consensus on what topics should be covered in software process modeling and deciding how to integrate the use of software tools into software process modeling.

Some directions for future research are offered as guidelines in order to cope with these problems: increasing the number of empirical studies in software-intensive organizations, studying it transversally against other software process engineering topics and, finally, developing environments where a SPML and support tools are integrated in order to be defined, modeled, orchestrated and executed.

Due to the large number of potential SPML users, such as software process engineers, project managers, software engineers, system engineers and customer manager, it is difficult to establish the best language to be used. Their individual information needs and expertise make widely diverging demands on a process modeling language. Thus, as future work, we are currently working on a practical case in which a SPEM 2.0 subset is applied in a software organization. We aim to search the SPML requirements to be practically applied in software-intensive organizations. With this knowledge, together with those requirements that have been described by some authors in their research proposals, we will develop a framework that would allow someone to evaluate a SPML against some requirements and characteristics and know which SPML best fits with an organization's particular needs.

To conclude, it must be pointed out that we are also working on a process environment definition, where a software process language will be included to allow tools orchestration to be easily applied in software-intensive organizations.

Acknowledgements

This research has been supported by the Tempros project (TIN2010-20057-C03-02) and Red CaSA (TIN 2010-12312-E) of the Ministerio de Economía y Competitividad, Spain and the NDTQ-Framework project of the Junta de Andalucía, Spain (TIC-5789).

References

[1] M.O. Alver, JabRef reference Manager, <http://jabref.sourceforge.net/, 2012> (accessed 08.07.2013).

[2] R. Bendraou, J.M. Jézéquel, M.P. Gervais, X. Blanc, A comparison of six UML-based languages for software process modeling, IEEE Transactions on Software Engineering 36 (2010) 662–675.

[3] A. Fuggetta, Software process: a roadmap, in: Proceedings of the Conference on The Future of Software Engineering 97, 2000, pp. 25–34.

[4] Y. Hauge, C. Ayala, R. Conradi, Adoption of open source software in software-intensive organizations: a systematic literature review, Information and Software Technology 52 (2010) 1133–1154.

[5] B. Henderson-Sellers, C. Gonzalez-Perez, A comparison of four process metamodels and the creation of a new generic standard, Information and Software Technology 47 (2005) 49–65.

[6] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[7] J. Lonchamp, A structured conceptual and terminological framework for software process engineering, in: Proceedings of the 2nd International Conference on the Continuous Software Process Improvement, IEEE Comput. Soc. Press, 1993, pp. 41–53.

[8] L. Osterweil, Software processes are software too, in: Proceedings of the 9th International Conference on Software Engineering, ICSE '87, vol. 3, IEEE Computer Society Press, 1987, pp. 2–13.

[9] S.M.J. Sutton, L.J. Osterweil, The design of a next-generation process language, in: Software Engineering – ESEC/FSE'97, Lecture Notes in Computer Science, 1301, Springer, Berlin Heidelberg, 1997, pp. 142–158.

[10] C. Wohlin, R. Prikladniki, Systematic literature reviews in software engineering, Information and Software Technology (2013).

[11] K.Z. Zamli, Process modeling languages: a literature review, Malaysian Journal of Computer Science 14 (2001) 26–37.

[12] K.Z. Zamli, N.M. Isa, A survey and analysis of process modeling languages, Malaysian Journal of Computer Science 17 (2004) 68–89.

[13] K.Z. Zamli, P.A. Lee, Taxonomy of process modeling languages, in: ACS/IEEE International Conference on Computer Systems and Applications, IEEE, 2001, pp. 435–437.

[14] F. Aoussat, M. Oussalah, M.A. Nacer, SPEM extension with software process architectural concepts, in: Proceedings of the International Computer Software and Applications Conference, Munich, Germany, 2011, pp. 215–223.

[15] D. Atkinson, D. Weeks, J. Noll, The design of evolutionary process modeling languages, in: Proceedings of the 11th Asia-Pacific Software Engineering Conference, 2004, pp. 73–82.

[16] N. Atkinson, Weeks, tool support for iterative software process modeling, Information and Software Technology 49 (2007) 493–514.

[17] R. Bendraou, B. Combemale, X. Crégut, M. Gervais, Definition of an Executable SPEM 2.0, in: Proceedings of the 14th Asia-Pacific Software Engineering Conference, 2007 (APSEC 2007), IEEE, 2007, pp. 390–397.

[18] R. Bendraou, M.P. Gervais, X. Blanc, UML4SPM: A UML2.0-based metamodel for software process modelling, in: Model Driven Engineering Languages and Systems, Springer, 2005, pp. 17–38.

[19] R. Bendraou, M.P. Gervais, X. Blanc, Uml4spm: an executable software process modeling language providing high-level abstractions, in: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, 2006 (EDOC'06), IEEE, 2006, pp. 297–306.

[20] R. Bendraou, J.M. Jezequel, F. Fleurey, Combining aspect and model-driven engineering approaches for software process modeling and execution, in: Trustworthy Software Development Processes, Springer, 2009, pp. 148–160.

[21] R. Bendraou, J.M. Jezequel, F. Fleurey, Achieving process modeling and execution through the combination of aspect and model-driven engineering approaches, Journal of Software Maintenance and Evolution: Research and Practice (2010).
[22] R. Bendraou, A. Sadovykh, M.P. Gervais, X. Blanc, Software process modeling and execution: the UML4SPM to WS-BPEL approach, in: Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007), Lubeck, Germany, 2007, pp. 314–321.
[23] C. Cares, E. Mayol, X. Franch, E. Alvarez, Goal-driven agent-oriented software processes, in: Proceedings of the 32nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA, Cavtat/Dubrovnik, Croatia, 2006, pp. 336–343.
[24] C. Chen, B.J. Shen, Y.Q. Gu, Flexible and formalized process modeling language, Journal of Software 13 (2002) 1374–1381.
[25] J. Chen, S. Chou, W. Liu, APER-2: a developer-centered, object-oriented process language, in: Proceedings of the International Symposium on Multimedia Software Engineering, IEEE Comp Soc; Tamkang Univ; Taiwan Minist Educ; Taiwan Natl Sci Council, 2000, pp. 297–303.
[26] S.C. Chou, DPEM: a decentralized software process enactment model, Information and Software Technology 46 (2004) 383–395.
[27] B. Combemale, X. Cregut, A. Caplain, B. Coulette, Towards a rigorous process modeling with SPEM, in: Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS 2006), vol. ISAS, Paphos, Cyprus, 2006, pp. 530–533.
[28] N. Debnath, D. Riesco, G. Montejano, M.P. Cota, J. Baltasar Garcia, Perez-Schofield, D. Romero, M. Uva, Supporting the SPEM with a UML extended workflow metamodel, in: IEEE International Conference on Computer Systems and Applications, vol. 2006, 2006, pp. 1151–1154.
[29] E. Di Nitto, L. Lavazza, M. Schiavoni, E. Tracanella, M. Trombetta, Deriving executable process descriptions from UML, in: Proceedings of the 24rd International Conference on Software Engineering (ICSE2002), Orlando, FL, United States, 2002, pp. 155–165.
[30] R. Ellner, S. Al-Hilank, J. Drexler, M. Jung, D. Kips, Philippsen, A FUML-based distributed execution machine for enacting software process models, in: Modelling Foundations and Applications, Springer, 2011, pp. 19–34.
[31] R. Ellner, S. Al-Hilank, J. Drexler, M. Jung, D. Kips, M. Philippsen, eSPEM – a SPEM extension for enactable behavior modeling, in: Modelling Foundations and Applications, Springer, 2010, pp. 116–131.
[32] A.L. Ferreira, R.J. Machado, M.C. Paulk, An approach to software process design and implementation using transition rules, in: Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2011, pp. 330–333.
[33] X. Franch, J.M. Ribó, A UML-based approach to enhance reuse within process technology, in: Software Process Technology, Springer, 2003, pp. 74–93.
[34] J. Ge, H. Hu, Q. Gu, J. Lu, Modeling multi-view software process with object Petri nets, in: Proceedings of the International Conference on Software Engineering Advances (ICSEA'06), IEEE, 2006, pp. 41–41.
[35] J. Ge, H. Hu, J. Lu, Order constraints for multi-view software process model, in: Proceedings of the International Conference on Computer Science and Software Engineering, vol. 2, 2008, pp. 639–642.
[36] F. Golra, F. Dagnat, Component-oriented multi-metamodel process modeling framework (CoMProM), in: First Workshop on Process-based approaches for Model-Driven Engineering (PMDE 2011), 2011, p. 44.
[37] G.V. Hagen, M., Towards flexible software processes by using process patterns, in: Proceedings of the 8th IASTED International Conference on Software Engineering and Applications, 2004, pp. 436–441.
[38] N.L. Hsueh, W.H. Shen, Z.W. Yang, D.L. Yang, Applying UML and software simulation for process definition, verification, and validation, Information and Software Technology 50 (2008) 897–911.
[39] S. Jablonski, B. Volz, S. Dornstauder, A meta modeling framework for domain specific process management, in: Proceedings of the 32nd Annual IEEE International Computer Software and Applications (COMPSAC '08), 2008, pp. 1011–1016.
[40] H. Kang, F. Dai, B. Huang, Evolution process component description language, in: Proceedings of the International Conference on MultiMedia and Information Technology (MMIT 2008), Three Gorges, China, 2008, pp. 306–309.
[41] A. Koudri, J. Champeau, MODAL: a SPEM extension to improve co-design process models, in: New Modeling Concepts for Today's Software Processes, Springer, 2010, pp. 248–259.
[42] S. Lee, J. Shim, C. Wu, A meta model approach using UML for task assignment policy in software process, in: Proceedings of the 9th Asia-Pacific Software Engineering Conference, 2002, pp. 376–382.
[43] C. Lima Reis, R. Quites Reis, M. Abreu, H. Schlebbe, D. Nunes, Flexible software process enactment support in the APSEE model, in: Proceedings of the IEEE Symposia on Human Centric Computing Languages and Environments, 2002, pp. 112–121.
[44] R. Maciel, B. da Silva, P. Magalhaes, N. Rosa, An integrated approach for model driven process modeling and enactment, in: Proceedings of the 23rd Brazilian Symposium on Software Engineering (SBES '09), 2009, pp. 104–114.
[45] T. Martinez-Ruiz, F. Garcia, M. Piattini, J. Munch, Applying AOSE concepts to model crosscutting variability in variant-rich processes, in: Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2011, pp. 334–338.
[46] T. Martinez-Ruiz, F. Garcia, M. Piattini, J. Munch, Modelling software process variability: an empirical study, Software, IET, vol. 5, IET, 2011, pp. 172–187.
[47] R. Martinho, J. Varajao, D. Domingos, A two-step approach for modelling flexibility in software processes, in: Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), L'Aquila, Italy, pp. 427–430.
[48] R. Martinho, J. Varajao, D. Domingos, FlexSPMF: a framework for modelling and learning flexibility in software processes, in: Visioning and Engineering the Knowledge Society, A Web Science Perspective, Springer, 2009, pp. 78–87.
[49] S.Y. Min, H.D. Lee, D.H. Bae, SoftPM: a software process management system reconciling formalism with easiness, Information and Software Technology 42 (2000) 1–16.
[50] J. Noack, Extending the software development process with a toolkit of UML-centered techniques, in: Proceedings of the International Conference on Software Methods and Tools, 2000, pp. 87–96.
[51] P.Y. Pillain, J. Champeau, H.N. Tran, Towards an enactment mechanism for MODAL process models, in: Proceedings of the 1st Workshop on Process-based approaches for Model-Driven Engineering (PMDE-2011), 2011, p. 33.
[52] I. Podnar, B. Mikac, A. Caric, SDL based approach to software process modeling, in: R. Conradi (Ed.), Software Process Technology, Lecture Notes in Computer Science, vol. 1780, 2000, pp. 190–202.
[53] R.Q. Reis, C.A. Reis, H. Schlebbe, D.J. Nunes, Early experiences on promoting explicit separation of details to improve software processes reusability, in: Proceedings of the IEEE International Computer Software and Applications Conference, Oxford, United kingdom, 2002, pp. 373–378.
[54] F. Ruiz, A. Vizcaino, F. Garcia, M. Piattini, Using XMI and MOF for representation and interchange of software processes, in: Proceedings of the 14th International Workshop on Database and Expert Systems Applications, 2003, pp. 739–744.
[55] T.D. Thu, T.H. Nhi, D.T.B. Thuy, B. Coulette, X. Cregut, Topological properties for characterizing well-formedness of process components, Software Process Improvement and Practice, vol. 10, 2005, pp. 217–247.
[56] H.N. Tran, B. Coulette, B.T. Dong, Modeling process patterns and their application, in: Proceedings of the 2nd International Conference on Software Engineering Advances – ICSEA 2007, Cap Esterel, France, 2007.
[57] H. Washizaki, Deriving project-specific processes from process line architecture with commonality and variability, in: Proceedings of the IEEE International Conference on Industrial Informatics (INDIN'06), Singapore, 2007, pp. 1301–1306.
[58] M. Wu, G. Li, J. Ying, H. Yan, A metamodel approach to software process modeling based on UML extension, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 6, Taipei, Taiwan, 2007, pp. 4508–4512.
[59] K. Zamli, P. Lee, Modeling and enacting software processes using VRPML, in: Proceedings of the 10th Asia-Pacific Software Engineering Conference, 2003, pp. 243–252.