Received: 5 June 2020 | Revised: 21 December 2021 | Accepted: 24 January 2022

DOI: 10.1002/smr.2440

REVIEW ARTICLE

Journal of Software: Evolution and Process WILEY

Compliance checking of software processes: A systematic literature review

Julieth Patricia Castellanos Ardila | Barbara Gallina | Faiz Ul Muram

School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

Correspondence
Julieth Patricia Castellanos Ardila, School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden.
Email: julieth.castellanos@mdh.se

Funding information
ECSEL JU Project AMASS, Grant/Award Number: 692474

Abstract

The processes used to develop software need to comply with normative requirements (e.g., standards and regulations) to align with the market and the law. Manual compliance checking is challenging because there are numerous requirements with changing nature and different purposes. Despite the importance of automated techniques, there is not any systematic study in this field. This lack may hinder organizations from moving toward automated compliance checking practices. In this paper, we characterize the methods for automatic compliance checking of software processes, including used techniques, potential impacts, and challenges. For this, we undertake a systematic literature review (SLR) of studies reporting methods in this field. As a result, we identify solutions that use different techniques (e.g., anthologies and metamodels) to represent processes and their artifacts (e.g., tasks and roles). Various languages, which have diverse capabilities for managing competing and changing norms, and agile strategies, are also used to represent normative requirements. Most solutions require tool-support concretization and enhanced capabilities to handle processes and normative diversity. Our findings outline compelling areas for future research. In particular, there is a need to select suitable languages for consolidating a generic and normative-agnostic solution, increase automation levels, tool support, and boost the application in practice by improving usability aspects.

KEYWORDS
compliance checking, normative frameworks, software processes, systematic literature review

1 | INTRODUCTION

Due to its growing use, the software is becoming a public good, which quality is a concern for society. In particular, society provides the community stakeholders,[1] who strongly influence normative compliance. For example, governments and regulatory bodies request compliance with standards and policies for licensing and certification purposes. Companies also want compliance with specific regulations from their suppliers to have a standardized and transparent production.[2] Finally, knowledgeable individuals demand the use of standards to influence responsible behavior among industry practices.[3] Thus, compliance with normative frameworks is a must-do for software producers, especially when software is developed for safety-critical systems.[*]

[*]Systems whose failure could lead to unacceptable consequences, for example, death, injury, loss of property, or environmental harm.[4]

Commonly, software standards focus on the development processes (e.g., ISO/IEC/IEEE 12207[5]). Standardized software processes make development tasks more predictable, transparent, and economical.[6] Therefore, there are also standards for the assessment and improvement of such processes (e.g., ISO/IEC 15504[7] series of standards and its evolution ISO/IEC 330xx series[8]). The software located in safety-critical systems also needs to comply with specific standards. However, such standards, which cover a broad set of organizations and use cases,[1] often mean stringent compliance requirements beyond the commitment to improve the process.[9] Thus, they act as process constraints and generally omit implementation-specific details.[10]

Process-related standards provide information regarding the process elements required during software development and the mandated features. Process engineers use this information to define the sequence of tasks and the resources ascribed to such tasks, for example, personnel, work products, tools, and methods, which are also framed with essential properties. Such work can be seen as systematic, that is, methodical in procedure or plan. Thus, process compliance management has been usually supported by systematically checking that the processes used to develop software have such information at the required points.

Properly designed and developed information technology tools can support process engineers in their compliance checking tasks.[11] For this, a unifying mechanism that permits automatic reasoning between the software process models and the normative frameworks regulating them could be a solution. Several studies have approached this idea by formulating methods for automating compliance checking. However, to the best of our knowledge, there is not any systematic study in this field.

In this paper, we undertake a systematic literature review (SLR) of studies reporting solutions for automatic compliance checking of software processes. An SLR, according to Kitchenham et al,[12,13] is a secondary study used to identify, analyze, and interpret all available evidence related to a specific topic. The purpose of this SLR is summarized in four aspects. First, it provides an overview of the research regarding automated compliance checking of software processes. Second, it provides an account of the current techniques. Third, it describes their potential impacts and challenges. Finally, it outlines key areas where future research can advance to support automated compliance checking practices.

As a result, we identify *41* studies from a list of *2034* found in recognized online libraries. The selected studies provide ad hoc solutions that are interesting, applicable, and valuable contributions to the topic. Such solutions use different techniques (e.g., ontologies and metamodels) to represent processes and artifacts (e.g., tasks and roles). Various languages, which have diverse capabilities for managing competing and changing norms, and agile strategies, are also used to represent normative requirements. Thus, the artifacts described in the models vary from one solution to the other. The level of automation claimed in the studies is mainly related to the reasoning required to define compliance between software processes and the normative documents. However, current methods require human intervention, especially to implement the inputs of such a reasoning process. Most solutions are in the stage of conceptual modeling or have been materialized as proof-of-concept prototypes.

Our findings outline compelling areas where future research can advance to support companies moving towards automated compliance checking practices. First, there is a need to select suitable languages for process and normative representation because there are already too many options not being adequately exploited. Second, it is crucial to consolidate a generic and normative-agnostic solution that can handle the different concepts, structures, and scenarios provided in the standards. Third, the tool support should be concretized in the solutions to guarantee industrial adoption. Finally, implementing the methods requires improvements in compliance diagnostics, traceability, data provenance, and further support for rule formalization.

The paper is organized as follows. In Section 2, we present essential background. In Section 3, we present the research method. In Section 4, we report the results of the review. In Section 5, we discuss the findings. In Section 6, we discuss the validity of the findings. In Section 7, we discuss related work. Finally, In Section 8, we summarize the work and present future remarks.

2 | BACKGROUND

This section presents essential background required in the rest of the paper.

2.1 | Compliance checking of software processes

Compliance or conformance checking is the verification of whether a process used to develop a specific software application adheres to a determined set of normative requirements (i.e., standards, policies, or regulations).[14] The expression conformance checking is also used to determine whether a software process instance conforms to its process definition.[15] Compliance can be shown when processes comply with all requirements or when organizations perform tailoring[†] within the boundaries of the norm, which determine allowed actions and the resulting conditions.[6]

[†]Tailoring is the process of selecting applicable requirements, perform their eventual modifications, and explain their implementation.

Compliance checking requires at least two specifications:[16] the specification for executing a software process and the specification regulating such process. Automatizing this task requires that these specifications are computer-based analyzable. In particular, formal methods are of growing interest in this area[17,18] because their rigor increases confidence in the correctness and completeness of the software processes. However, the analysis of compliance is as good as the models used for such analysis.[19] Thus, precise notions,[20] which also provide an unambiguous and mathematically defined syntax and semantics,[21] are required.

Normative frameworks have two critical features. First, they have a "shall" type collection of requirements, that is, compliance requires the satisfaction of obligations.[22] Second, the requirements of reasonable regulations must be balanced with other values like the urgency of the problems in question, respect for the plurality of view of participants, values, precedents, and traditions.[23] Thus, justified exceptions are also be permitted. Accordingly, software process-based compliance requirements conform to a standard if and only if it satisfies all the obligations prescribed by the process-related requirements. Violating such requirements could introduce potential risks. However, permissions provide exceptions to obligations, indirectly affecting compliance.[24] Thus, compliance is a relationship between permissions (optional) and obligations (required).

2.2 | Compliance checking strategies

Companies commonly make use of different strategies to meet their needs regarding process compliance management. There are three main approaches towards achieving compliance.[25] The traditional one is based on retrospective reporting. Such reports are the result of audits, which are commonly conducted manually, *after the fact*, that is, the process is inspected after its runtime stage has passed. Performing compliance checks after the fact is somewhat risky because it inspects existing processes that were designed in the absence of any knowledge of specific regulations, opening possibilities for violations of such regulations. Audits of this kind can also be automatically performed in organizations with access to system logs by using techniques such as process mining.[26] A second approach is to provide automated detection at *running time*, that is, at the time the process is executed. Such monitoring capability assists in checking for compliance issues on the fly, providing some level of remediation and/or mitigation of control deficiencies. Finally, there is a preventive approach, where the set of normative requirements regulating a process are embedded into the process when the process is being designed. In such an approach, which is called *compliance by design*, the compliance check takes place in advance, before the runtime stage. According to Casanovas et al,[27] compliance by design has the advantage that subsequent proof and corrections of compliance are not required. Moreover, it is flexible as the generation can be repeated when rules are added, removed, or changed. In addition, compliance is not only detected but enforced.

2.3 | Software processes

Software processes provide a framework that defines how and when to perform activities, the roles involved, and the results to be created.[6] Explicit descriptions of the software processes servers development to proceed in a systematic way[28] increasing predictability and transparency.[6] However, development methodologies tackled the necessity of software processes in different ways. In particular, agile methodologies prioritize individuals and interactions over processes and tools[‡]. Moreover, agile follows an empirical logic, which faces fundamental challenges in regulated environments.[29] In contrast, plan-driven methodologies build mainly on the codification strategy and the definition of appropriate steps in advance. Given the successful application of agile in software projects[§] and the suitability of plan-driven methodologies in regulated environments, hybrids between them are also conceived,[30,31] for example, the Scaled Agile Framework (SAFe)[¶].

Software process models help organizations preserve, repeat, analyze and reuse process information.[32] Models also can improve the understanding of compliance needs.[33] A software process model is an abstraction whose goal is to approximate the full range of characteristics and properties of an actual software process.[34] For this, a process model should[35] (1) be described with rigorous notations, (2) be detailed enough, (3) be semantically broad, and (4) be clear and understandable to facilitate communication. The concept of the software process model is analogous to the concept of the life cycle (or lifecycle) model:[6] software life cycle models define the main steps and their sequence, while software process models provide more detailed instructions, breaking the primary steps down into substeps, and adding information about the results generated and the roles involved. According to Parnas and Clements,[36] the most advantageous form of a process description will be in terms of work products workflows.

A survey conducted by Diebold and Scherr[17] shows the characteristics of the software process models that are expected in industrial settings. In particular, it is expected to have concepts for a detailed description of the software process elements, that is, the units of work and their order, the roles performing the units of work, and the artifacts used and produced. Besides, graphical representation of the process and structured text to explain details are also desirable, mainly in projects where auditors need to assess the software process for standards compliance.

[‡]https://agilemanifesto.org/.
[§]See for instance https://stateofagile.com/.
[¶]https://www.scaledagile.com/.

Furthermore, the possibility of having different views on the software process is relevant, that is, hierarchical representation of the information, different perspectives for each role, and the usage and arrangement of compliance artifacts. Finally, artifacts and environment customization are essential aspects demanded from software process modeling tools because they can help engineers configure models according to context (or project)-specific needs.

2.4 | Software process-related normative frameworks

Normative frameworks addressing software prescribe requirements for their implementation. For example, the standard ISO/IEC/IEEE 12207[5] provides terminology to establish a common framework for software life cycle processes. The Software Process Improvement (SPI) movement started with the Capability Maturity Model Integration (CMMI)[37] as a significant innovation.[38] Then, ISO/IEC 15504[7] (also called SPICE-Software Process Improvement and Capability Determination) was also created. SPI frameworks aim at increasing product quality and reducing time-to-market and production costs.[28]

The International Organization for Standardization (ISO) has also defined the fundamentals of quality management systems, which influence the process assessment and improvement.[39] In particular, there is the ISO 9000 series,[40] for example, ISO 9001,[41] guidance for their application ISO/IEC 90003,[42] and ISO/IEC TR 29110,[43] which applies to very small entities. Additionally, the Information Technology Infrastructure Library (ITIL) framework, developed by the UK government, aims to provide a guideline for delivering quality information technology services.[39] Six Sigma, an organized methodology that guides continuous improvement on manufacturing or service processes, has also been used as a set of techniques and tools for SPI.[44]

Manufacturers of safety-critical systems have the duty of care.[#46] Consequently, ethics and regulatory regimes explicitly addressing such systems have stronger compliance requirements beyond the commitment to improve software process capability.[9] Manufacturers must establish effective software development processes based on recognized engineering principles,[47] usually found in industry standards.[48] To have legal oversight over industries, there are governing bodies that are in charge of ensuring the safety of citizens. For example, the European Commission (EC) and the United States Food and Drug Administration (FDA) enforce regulatory obligations on manufacturers of medical devices so that they are safe and fit for their intended purpose.[49] The Health and Safety Executive in England has used compliance with IEC 61508[50] as a guideline for bringing legal actions if harm is caused by safety-critical systems.[46] As compliance with safety standards has become essential evidence for a jury in a product liability action,[51] failure or inadequate compliance could lead to legal risks, that is, penalties[52] and prosecutions.[53]

In particular, prescriptive safety standards cover requirements for all software life cycle activities and exist in almost all safety-related domains, for example, ISO 26262[54] (automotive), CENELEC EN 50128[55] (railway), DO-178C[56] (avionics), and IEC 62304[57] (medical devices), to only mention some of them. Standards for software development (e.g., medical devices-IEC 62304[57] and space mission-critical software-ECSS-ST-40C[58]), risk management (e.g., ISO 14971-application of risk management to medical devices[59]), and information technology (e.g., ISO/IEC 27000[60]), are also part of the menu of standards that became the de facto regulatory frameworks subjecting the organizations to mandatory certification.

Regulations, such as EU DPD[‖62] and PIPEDA[**,63] lay down rules relating to protecting natural persons in the European Union and Canada, respectively. Regulators will likely introduce additional measures to maintain legal oversight over artificial intelligence (AI) algorithmic systems.[64] AI oversight will probably be approached from the software process perspective because it is software.[65] Thus, practitioners have to embrace software process diversity, that is, the adoption of multiple normative software process frameworks within single software processes.[66]

3 | RESEARCH METHOD

In this section, we describe the research method, which is based on the guidelines for SLR recommended by Kitchenham et al.[12,13] A SLR is a review methodology that involves three main activities:

1. *Plan the review*. This activity includes three tasks. First, we need to *identify the need for a review*. Second, we need to *specify a goal and the research questions*. Finally, we need to *design the review protocol*. The review protocol should include a search strategy, the study selection criteria, the study selection procedure and the quality assessment criteria.
2. *Conduct the review*. In this activity, the researchers apply the review protocol previously created and answer the research questions. Tasks relevant to this activity are the data collection and the data extraction.
3. *Report the results of the review*. In this activity, the researchers define the means to illustrate the findings.

[#]Legal obligation to adhere to a standard of reasonable care while performing any acts that could foreseeable harm others.[45]
[‖]European Data Protection Directive, then replaced by the General Data Protection Regulation (GDPR).[61]
[**]Personal Information Protection and Electronic Documents Act.

The activities and tasks mentioned above should, in theory, be implemented sequentially. However, in practice, it is often necessary to iterate between them and update their discovered information as the researchers' understanding of the topic deepens. In the remaining parts of this section, we describe the first two activities included in the SLR, that is, plan the review (see Section 3.1) and perform the review (see Section 3.1.3). We report the results of the review in Section 4.

3.1 | Plan the review

In this section, we describe how the SLR is planned.

3.1.1 | Identify the need for a review

Compliance checking is a challenging task. In particular, the requirements included in the standards prescribe many details regarding the software process, that is, the presence of tasks ordered in a determined way, resources ascribed to such tasks (personnel, work products, tools, and methods), and their related properties. There are also many possible ways to be compliant. For example, software process-related normative frameworks provide tailoring rules that should be applied according to specific processes' needs (which may open room for including agile methodologies). In addition, the requirements can be superseded or eliminated if explicit justifications demonstrating compliance are provided. Moreover, requirements in one part of the standard may refer to other parts of the same standard or even to different standards, making their understanding complicated. Finally, many new standards or new versions of older standards may apply to the same software process.

Properly designed and developed information technology tools has the potential to support process engineers in their compliance checking tasks. For this, a unifying mechanism that permits automatic reasoning between the software process models and the normative frameworks regulating them could be a solution. Several studies have approached this idea by formulating methods for automating compliance checking. However, to the best of our knowledge, there is not any systematic study in this field. Thus, we consider it essential to close this gap in the most possible systematic and unbiased manner by performing an SLR that permits us to recognize what exists in the current state of the art in that area.

The *scope* of the SLR is defined by taking into account the guidelines proposed by Cooper,[67] which considers aspects such as the focus (outcomes, methods, theories, and applications), the goal (integration, criticism, or identification of central issues), the reviewers' perspective (neutral representation or espousal of position), coverage (exhaustive, exhaustive with selective citation, representative, central, or pivotal), organization (historical, conceptual, or methodological), and audience (specialized scholars, general scholars, practitioners or policymakers, or the general public). Our SLR focuses on the research outcomes of the available literature addressing automated compliance checking of software processes. Our goal is to identify the specific aspects that have dominated past efforts, that is, publication trends, the characteristics of the methods, potential impact, and challenges. We consider reporting our result from a neutral representation perspective, that is, attempting to present the explicit evidence available in the literature. We aim at implementing exhaustive coverage by determining an inclusive review protocol. The SLR summary will be organized conceptually, that is, works relating to the same abstract ideas will appear together. Finally, we aim to write our SLR to target specialized scholars, practitioners, and policymakers.

3.1.2 | Specify goal and research questions

This SLR aims to characterize the research regarding automated compliance checking of software processes against the constrains associated with different normative frameworks. As presented in Section 2.1, compliance checking requires at least two specifications: the specification for executing a software process and the specification regulating such process. Thus, it is essential to know the methods used in the state of the art and their characteristics, that is, strategies, languages used, level of automation, and the mechanisms used to handle changes in the normative space. It is also crucial to learn the potential impact of the proposed methods, that is, the methods' target application domains, normative documents addressed, illustrative scenarios, and support for agile methodologies. We are also interested in knowing the evolution of the topic over time and the current challenges. Research questions are formulated in Table 1 by taking into account the research goal previously described.

3.1.3 | Design the review protocol

We present a summary of the concrete and formal plan used in the execution of the SLR.

TABLE 1 Research questions

Id	Question	Motivation
RQ1	How did research in automated compliance checking of software processes developed over time?	Identify the number of papers published, dates, the publication venues, and the active groups doing research in the topic.
RQ 2	What are the characteristics of the methods described the primary studies?	
	2.1 What are the compliance strategies used in the different methods?	Characterize the different compliance strategies used in the methods as presented in Section 2.2.
	2.2 Which are the languages used to represent software processes entities and structures?	Characterize the different alternatives used to represent the software processes concepts and their properties.
	2.3 Which are the languages used to represent the compliance requirements?	Characterize the different alternatives used to provide a representation of the requirements described in the standards.
	2.4 Which is the level of automation?	Examine the automation level described in the studies.
	2.5 What are the mechanisms, if any, used to handle standards evolution and software process reconfiguration?	Identify the characteristics used in the primary studies to manage change.
RQ 3	What is the potential impact of the proposed methods?	
	3.1 What are the application domains?	Determine the specific application domain, e.g., automotive.
	3.2 What are the types of normative documents targeted?	Determine the type of standards, policies, regulations, etc, targeted by the studies.
	3.3 What are the types of illustrative scenarios presented?	Present the scenarios used to describe the methods.
	3.4 To what extent agile methodologies are supported?	Describe whether the primary studies take into account the compliance checking in agile software processes.
RQ4	What challenges are identified in the primary studies?	Identify challenges to determine future directions in the area.

TABLE 2 Structure of the CIMO Logic

Context (C)	Intervention (I)	Mechanism (M)	Outcome (O)
Software processes.	Normative software process constraints.	Automation methods.	Results of compliance checking.

Search strategy

An SLR uses specific concepts and terms for reaching the possible amount of primary studies. In particular, the outcomes of the search should refer to factors of importance for the review. To define such factors, we consider the structure of the Context–Intervention–Mechanism–Outcome (CIMO) Logic.[68] The CIMO is a logic constructed as follows: If you have a problematic Context (C), use a special kind of Intervention (I) to invoke the generative Mechanism(s) (M) to deliver a specific Outcome (O). The context corresponds to the surroundings (external and internal environment) factors. The interventions are those factors that have the potential to do some influence. The mechanisms are the means that in a specific context are triggered by the Intervention. Finally, the outcomes are the Intervention results in their various aspects.

The factors of importance in our SLR are software processes, normative software process constraints, automation methods, and the results of compliance checking (see Table 2). Commonly, synonyms of such terms are also used in the literature. We based the selection of the synonyms on the background information gathered in Section 2. First, in Section 2.3, we found that the concept of “software process” is related to the concept of “software lifecycle” (or life cycle), “workflow,” and “software development methodology.” Second, in Section 2.4, we found sources of normative software process constraints in a “standard,” “reference model,” “framework,” “regulation,” and “policy.” Third, in Section 2.1, we see that the word “compliance” is used interchangeably with the word “conformance.” The word “checking” and “verification” could also be seen as synonyms. We are not interested in checking the compliance of specifications beyond the ones containing normative requirements. Therefore, we focus on the concept “compliance” or “conformance,” which is the current jargon, and do not strike on the concept “model checking,” which is commonly used for software verification.

We did a test search in the library Science Direct[†] to check whether the information retrieval was different between all synonyms. The word verification was showing fewer results than the searching results regarding checking. Moreover, the results were related to software (as a product) verification and not software process verification. We concluded that the word verification is not used together with the work compliance or conformance of software processes. The words “automatic,” “automated,” computer-based,” logic-based,“ and ”formal” could also be seen as synonyms. A similar test permitted us to check the difference between these three words. We found that the word “automatic” leads to more results than the word “automated.” Moreover, the results obtained with the word automated are included in the results obtained with the work

[†]https://www.sciencedirect.com/.

TABLE 3 Search string

("automatic" OR "formal") AND ("compliance checking" OR "conformance checking") AND ("software process" OR "software life cycle" OR "software lifecycle" OR "software workflow" OR "software development methodology") AND ("standard" OR "reference model" OR "framework" OR "regulation" OR "policy")

TABLE 4 Inclusion and exclusion criteria

Type	Description	
Inclusion	I1	The primary study belongs to the software engineering domain. We limit the scope of our SLR to automated compliance checking of software processes.
	I2	The primary study is about compliance/conformance checking of software processes against the constrains associated to different kind of software process-related normative frameworks.
	I3	The primary study is a peer-reviewed article written in English.
	I4	The primary study reports issues, problems, or any type of experience concerning the aspects related to process-related automated compliance checking, that is, process models, requirements formalization, and analysis of compliance.
	I5	The primary study describes solid evidence on automated compliance checking of software processes by using, for example, rigorous analysis, experiments, case studies, experience reports, field studies, and simulation.
Exclusion	E1	The primary study focus on software process aspects different from compliance checking, for example, process design, execution, or the management of workflow.
	E2	The text of the primary study is not available.
	E3	The primary study belongs to the following categories: commercials, pure opinions, gray literature (e.g., reports, working papers, white papers, and evaluations), books, tutorials, posters, and papers outside of the contexts of computer-based critical systems.
	E4	The primary study is about automatic compliance checking of processes different from software processes, for example, business processes.
	E5	The primary study is not clearly related to the research questions. The primary study does not present sufficient technical details of the topic studied.
	E6	The study is a secondary or tertiary study.
	E7	The primary study does not undergo a peer-review process, such as nonreviewed journal, magazine, conference papers, master theses, and books (in order to ensure a minimum level of quality).

automatic. Thus, the word automated is not included in the final search string. The results obtained with the words computer-based and logic-based were very few. Moreover, such results were included in the search that included the word automatic. Thus, computer-based and logic-based are not used in the final search string. Instead, the word formal yielded relevant new results. Thus, the word formal is included in the final search string. Finally, we tested the plurals software processes, software workflows, software development methodologies, standards, reference models, frameworks, regulations, and policies. There were no new results by using such plurals. Based on the analysis and the combinations of the terms previously defined, we specify our search string (see Table 3).

Study selection criteria

Primary studies are searched on popular scientific online digital libraries that are widely used in computer science and software engineering research, as reported in Zhang et al.:[69] (1) ACM Digital Library,[‡‡] (2) IEEE Xplore Digital Library,[§§] (3) Springer Link,[¶¶] and (4) Google Scholar.[1] We also include the results we gathered during our search string test in the library Science Direct. The search time frame is not restricted to a specific interval because we also want to see the evolution of the topic over time. The inclusion and exclusion criteria is presented in Table 4.

Study selection procedure

The search string defined in Table 3 is applied to the electronic databases selected in the study selection criteria. Different filtering levels are then applied to the retrieved studies to find the relevant ones for this research. Initially, we perform a title screening on the initial set of retrieved publications. In this phase, we also remove the duplicates that can be found in different databases. Then, we perform an abstract screening, from which we select the papers that would be thoroughly read. After, we perform a snowballing,[70] which is a technique that aims at reaching more relevant primary studies. First, we perform backward snowballing, which refers to searching relevant studies by considering the reference list of an initial

[‡‡]https://dl.acm.org/,
[§§]https://ieeexplore.ieee.org/Xplore/home.jsp.
[¶¶]https://link.springer.com/.
[1]https://scholar.google.com/.

TABLE 5 Data extraction criteria

Extracted data	Used for
Author information and study title	Study overview
Year, publication types venues, and research groups	Study overview and RQ1
Compliance strategies	RQ2.1
Languages for representing software processes	RQ2.2
Languages for representing requirements mandated by standards	RQ2.3
Level of automation	RQ2.4
Mechanisms for handling variability, if any	RQ2.5
Application domains	RQ3.1
Normative frameworks addressed	RQ3.2
Illustrative scenarios	RQ3.3
Support for agile, if any	RQ3.4
Challenges	RQ4

set of primary studies. This procedure is repeated until no more papers are found. Then, we perform forward snowballing, which aims at identifying more relevant studies based on those papers citing the paper being examined. For the forward snowballing, we use Google Scholar due to its convenient facilities for finding referring papers. Again all relevant papers are added until no more papers could be found. The number of papers resulting from this selection procedure were fully processed in the SLR. The first author (who is a Ph.D. student) does the paper's search and selection. During every phase, the second and third authors perform quality controls. To record the data for later analysis and correlation, we used spreadsheets. In particular, we focused on the data presented in Table 5.

Quality assessment criteria

We developed a checklist for the quantitative and qualitative assessment of the selected research articles (see Table 6), based on criteria formulated by Kitchenham et al.[71] For Items QA1 to QA7, the scoring procedure has only three optional answers: Yes = 1, Partially = 0.5, or No = 0. For a given study, its quality score is computed by summing up the scores of the answers to the quality assessment questions.

TABLE 6 Quality assessment criteria

Item	Assessment Criteria	Score	Description
QA1	Does the study includes a clear statement of the goal?	0	No. The goal is not described.
		0.5	Partially. The goal is described, but unclearly.
		1	Yes. The goal is well described and clear.
QA2	Does the selected primary study discuss their results?	0	No. The results are not explicitly discussed in a discussion (or similar) section.
		0.5	Partially. There is a discussion section, but not clearly discussed.
		1	Yes. The results are well discussed.
QA3	Is the paper based on research (or it is merely a "lessons learned" report based on expert opinion)?	0	The paper is a report based on expert opinion.
		0.5	Partially. It is not completely clear the research validity of the study.
		1	Yes. The paper is based on research.
QA4	Does the selected primary study completely addresses the topic of automated compliance checking of software processes?	0	No. The paper is not completely addressing the topic of the research.
		0.5	Partially. The study partially address the topic of the research.
		1	Yes. The paper completely addresses the topic of research.

TABLE 6 (Continued)

Item	Assessment Criteria	Score	Description
QA5	Is there an adequate description of the context in which the research was carried out?	0	No. The paper is not describing an adequate context of the research.
		0.5	Partially. The study partially describes the context of the research.
		1	Yes. The paper is describing an adequate context of the research.
QA6	Is there a clear statement of findings?	0	No. The paper is not having a clear statement of the findings.
		0.5	Partially. The study partially describes the findings of the research.
		1	Yes. The paper is having a clear statement of the findings.
QA7	Are the results in accordance with the goal of the selected primary study?	0	No. The results are not in accordance with the goal.
		0.5	Partially. The study partially describes the findings of the research.
		1	Yes. The results are in accordance with the goal.

3.2 | Perform the review

In this section, we present the details regarding how we perform the review.

3.2.1 | Data collection

We apply the search string defined in Table 3 to the different databases included in the study selection criteria without trunking the dates of the search. Our search was performed between February 22 to March 15, 2021. The databases Springer Link, ACM (in which we took the option "Expand our search to The ACM Guide to Computing Literature"), and IEEE Xplore accepted all the words included in the search string. Instead, in Google Scholar, we needed to divide the search string in two strings and do two different searches. As presented in Figure 1, from these searches, we got 153, 71, 1, and 1601 studies, respectively. We also added the 208 primary studies that we found in the search string test that we performed in Science Direct. In total, our search resulted in 2034 hits. Then, we perform the title screening. In this step, we selected papers that match at least one of the criteria we defined in the search string but do not match any exclusion criteria. For example, the paper is selected if the title has the word process and conformance checking. However, if the title has the expression business process, it is immediately discarded.

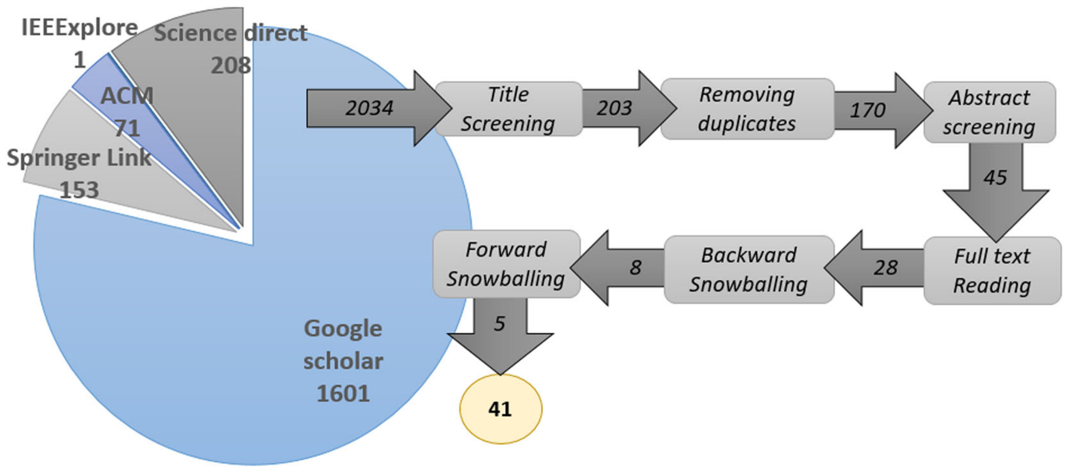


FIGURE 1 Paper selection process

We did this to have a more accurate filter of useful material from the first phase of our SLR. Given this strategy, we selected 68 primary studies in Springer Link, 17 in ACM, 1 in IEEE Xplore, 106 in Google Scholar, and 11 in Science Direct. The total of primary studies after title screening was 203. Then, we discarded the duplicates found in different databases, resulting in 170 possible relevant studies. Then, we performed abstract screening and selected 45 primary studies. We fully read the 45 studies and apply to them the quality criteria. We decided to select the studies that got 6 of 7 in the quality criteria. As a result, 28 articles are selected. We performed the snowballing process to the 28 articles previously selected. As a result we got eight new primary studies in the backward snowballing and five new primary studies in the forward snowballing. The complete set of primary study that we have included in our SLR is 41. The selection criteria grading is presented in Appendix A.

3.2.2 | Data extraction

The select 41 papers are presented in Table 7. We record the data in Excel spreadsheets for analysis and correlation. The data extracted for answering the research questions is presented in Appendix B.

TABLE 7 Selected primary studies

ID	Title	Year	Type
S1	Tailoring and conformance testing of software processes: The ProcePT approach[72]	1995	Conference
S2	Managing standards compliance[73]	1999	Journal
S3	Managing process compliance[74]	2003	Journal
S4	Compliance flow—Managing the compliance of dynamic and complex processes[75]	2008	Journal
S5	An automatic compliance checking approach for software processes[76]	2009	Conference
S6	Supporting qualification-safety standard compliant process planning and monitoring[77]	2010	Conference
S7	Defining software process model constraints with rules using OWL and SWRL[78]	2010	Journal
S8	A model-driven engineering approach to support the verification of compliance to safety standards[79]	2011	Conference
S9	NOVA Workflow: A workflow management tool targeting health services delivery[80]	2012	Journal
S10	Towards a process for legally compliant software[81]	2013	Worshop
S11	Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation[82]	2013	Journal
S12	A framework to formally verify conformance of a software process to a software method[83]	2015	Conference
S13	Cybersecurity policy verification with declarative programming[84]	2016	Journal
S14	Representing software process in Description Logics: An ontology approach for software process reasoning and verification[85]	2016	Conference
S15	How to assure correctness and safety of medical software: The hemodialysis machine case study[86]	2016	Conference
S16	A framework for safety-critical process management in engineering projects[87]	2017	Conference
S17	Applying process mining techniques in software process appraisals[88]	2017	Journal
S18	Continuous process compliance using model driven engineering[89]	2017	Conference
S19	Towards increased efficiency and confidence in process compliance[90]	2017	Conference
S20	Automated legal compliance checking by security policy analysis[91]	2017	Conference
S21	A formalization of the ISO/IEC 15504: Enabling automatic inference of capability levels[92]	2017	Conference
S22	Security analysis and legal compliance checking for the design of privacy-friendly information systems[93]	2017	Conference
S23	Towards efficiently checking compliance against automotive security and safety standards[94]	2017	Workshop
S24	An axiom based metamodel for software process formalisation: An ontology approach[95]	2018	Journal
S25	Enabling compliance checking against safety standards from SPEM 2.0 process models[96]	2018	Conference
S26	Ensuring conformance to process standards through formal verification[97]	2018	Conference
S27	Integrating formal methods into medical software development: The ASM approach[98]	2018	Journal
S28	Transforming SPEM 2.0-compatible process models into models checkable for compliance[99]	2018	Conference
S29	Compliance of agilized (software) development processes with safety standards: A vision[14]	2018	Conference
S30	Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3—Enabling automatic inference of maturity and capability levels[100]	2018	Journal

TABLE 7 (Continued)

ID	Title	Year	Type
S31	Fast compliance checking in an OWL2 fragment[101]	2018	Conference
S32	Developing medical devices from Abstract State Machines to embedded systems: A smart pill box case study[102]	2018	Journal
S33	Facilitating automated compliance checking in the safety-critical context[103]	2019	Journal
S34	Formalising process assessment and capability determination: An ontology approach[104]	2019	Conference
S35	Using models to enable compliance checking against the GDPR: An experience report[105]	2019	Conference
S36	A life cycle for authorization systems development in the GDPR perspective[106]	2020	Conference
S37	Co-engineering of safety and security life cycles for engineering of automotive systems[107]	2020	Journal
S38	Separation of concerns in process compliance checking: Divide-and-conquer[108]	2020	Conference
S39	Reusing (safety-oriented) compliance artifacts while recertifying[109]	2021	Conference
S40	Compliance-aware engineering process plans: The case of space software engineering processes[2]	2021	Journal
S41	Supporting quality assurance with automated process-centric quality constraints checking[110]	2021	Conference

4 | RESULTS

In this section, we report the results of the SLR.

4.1 | Summary of the primary studies

In this section, we summarize the primary studies selected in the SLR. The studies are categorized by taking into account the type of approach used to model the software process concepts. As Figure 2 depicts, there are two categories that include the most significant amount of studies, which consider the modeling of process-related elements from specific standards concepts (34%) and those that take as a base consolidated process modeling languages to which a layer of analysis using formal languages is added (32%). Those studies are summarized in Sections 4.1.1 and 4.1.2. The first category shows that there is an interest to be faithful to the process concepts described in the requirements of the standards as they are, that is, the researchers do not consider it essential to model information that is not mentioned in the standards. The second category shows an interest in busting already consolidated software process modeling languages, which is an action towards standardization of methods and interoperability between methods. The minority of the studies found are distributed as follows: 5% of the methods take into account the documents workflow, 14% of the methods take into account access rights given to the roles in a process, and the final 15% of the methods have other types of proposals, for example, process mining, databases, and workflow modeling. These last three categories, which are summarized in Sections 4.1.3–4.1.5, restrict compliance analysis to only one particular element of the development process, that is, roles, documents, and workflows or tasks, respectively, narrowing its applicability.

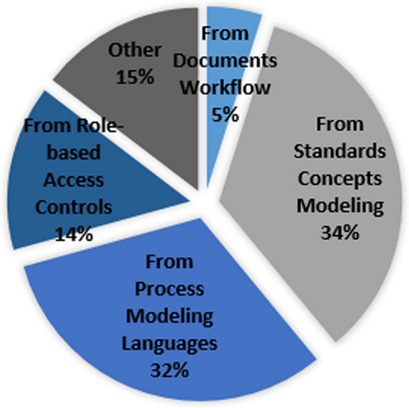


FIGURE 2 Approaches

TABLE 8 Standards' concepts modeling methods: Limitations and validation

ID	Limitations	Validation
S3	The metamodel uses reduced definitions that cover several concepts.	There is no proper evaluation.
S4		The method is illustrated with a case study.
S5	The patterns are defined in terms of the process concepts. Thus, the checking results are not clear in terms of the standards requirements.	For evaluation, three different projects from a software organization have been selected as experimental subjects.
S8 S11	Constraints are translated into process elements concepts, which do not permit to have literal explanations of the requirements.	Case study: IEC 61508[50] profile applied to the subsea control domain.
S10	The study only describes roles in the process.	Illustrative example of a personal health information privacy system.
S12	The resource is a concept used to model all the elements required in a task.	Evaluation on an SCRUM[111] Software Process Model.
S14	Constraints derived from the standards requirements are translated into process elements concepts, which does not permit to have literal explanations of the requirements.	Examples are given to justify the feasibility of this proposed method.
S24		Use cases from ISO/IEC 29110[43] and ISO/IEC 24744[112] standards.
S26		Illustrative example of a Moodle e-learning system.
S34		Illustrative example of a software requirements analysis (SRA) process.
S18	Every standard requires its own metamodel.	Application is presented with the standard ECSS-E-ST-40C.[58]
S21 S30	Every standard requires its own ontology	Inference of the capability levels in five organizations.

4.1.1 | Compliance checking from standards concepts modeling

Modeling the process elements required by specific standards is an approach widely used in the selected studies. In this category, we found 14 studies. In Table 8, we present the limitations of these studies and the validation performed by the authors.

In S3, the method considers the Models of Standards (MoS), a knowledge base containing the process concepts defined in the standard. It also considers the User-Defined Process (UdP), which is an instance of a process. An intelligent agent, called the *Inspector*, performs compliance checks by comparing the MoS and UdP during process design and runtime. S4 extends S3 by providing more detailed information regarding preconditions and postconditions, recommendations, and capabilities (skill, technique, method, knowledge, or any attribute that a task agent requires to perform a task). In S5, the method requires the interpretation of quality standards as process patterns. Compliance checking is a comparison between the patterns and the UdP. The process deviation is measured during process enactment, providing information about absent or skipped elements, correct/erroneous reverse order, absent iterations, and inconsistent control.

In S8, a conceptual model of standards (in terms of activities, artifacts produced and required, techniques, and roles) is created and used as a form that gives shape to the instances of the particular standards. In S11, the method described in S8 is evaluated by considering experts' opinions. In S10, the author presents a Governance Analysis Tool (GAT) for information privacy. GAT is a Unified Method Language (UML)-based metamodel that contains a Governance Analysis Model (GAM) and a Governance Analysis Language (GAL). GAM captures information domain, that is, process activities and roles, organizational information, and general information regarding the legal entity. In S12, the authors propose a library of best practices related to four popular software development methods, namely: The Unified Process (OpenUP[2]), Extreme Programming,[113] Scrum,[111] and Kanban.[114] Process engineers can instantiate such libraries to create a process model according to the project constraints and context.

In S14, S24, S26, and S34, the authors present the evolution of a framework for software process assessment and capability determination. In particular, in S14, the method focuses on translating software process standards into a knowledge base that represents properties of the process (i.e., title, purpose, outcomes, activities, and tasks). In S24, the method is augmented with an axiom metamodel containing specific metaconcepts (i.e., Achievable, Doable, Tangible, and Assessable). Such concepts are extracted from the software implementation process ISO/IEC 29110.[43] In S26, the method extends S14 and S24 by including a formal approach to software process analysis and verification using Description Logic (DL).[3] In this case, the DL axioms represent the process reference model (PRM) defined by ISO/IEC 15504-5.[7] Finally, in S34, the authors include DL axioms related to formalizing the process capability dimension of the process assessment model (PAM).

[2]https://www.utm.mx/caff/doc/OpenUPWeb/.
[3]https://dl.kr.org/.

TABLE 9 Process modeling language methods: Limitations and validation

ID	Limitations	Validation
S6	The metamodel translation loses information in the process.	Example in the automotive domain: the concept and system design phases.
S7		There is no any type of validation.
S16		Industry scenario: railway automation unit.
S19	The checking result are not back propagated into the process modeling environment.	Illustrative example in the automotive domain: the software unit design.
S23		
S25		
S28		Illustrative example in the railway domain: the software unit design.
S29		Examples from three different hybrid software development process models.
S33		There is no any type of validation.
S38		Illustrative example in the railway domain: architecture and design phase.
S39		Illustrative example in the medical domain: risk analysis phase.
S40		Illustrative example in the space domain: software engineering process.
S37	There are no textual representation of the standards requirements.	Case study: Fleet of autonomous (model) cars.

In S18, the authors present a metamodel defining two layers of abstraction. The abstract level defines the abstract notions of process design, and the concrete level defines the corresponding concrete implementations. The notion of a contract is used to bind the activities and specifying constraints. In S21 and S30, the authors present the evolution of a framework for enabling inference of maturity and capability levels of software processes. Concepts related to processes and work products and the compliance requirements are included in the method. The difference between S21 and S30 is the amount of standards concepts modeled, that is, S21 contains only the ISO/IEC 1550-4 concepts, while S30 is augmented with the SEI CMMI v1.3.

4.1.2 | Compliance checking from process modeling languages

In some methods, process modeling languages are used to create the process elements and their interactions. We found 13 studies in this category. In Table 9, we present the limitations of these studies and the validation performed by the authors.

In S6, the authors propose a framework in which an ontology is used to formalize domain standards and further domain knowledge required to understand processes. The information in the ontology is constrained with DL rules and transformed into the Software and Systems Process Engineering Metamodel (SPEM) 2.0[115] process models. In S7, the authors present an approach in which software processes are implemented in SPEM 2.0 and then translated into ontologies to permit the application of constraints derived from SPI frameworks. In S16, the authors propose a framework for compliance checking that includes the formalization of process in Business Process Management and Notation (BPMN)[116] and then transformed into timed Petri nets. Compliance constraints are represented in a constraint language able to retrieve information from Resource Description Framework (RDF).[4]

In S19, S23, S25, S28, S29, S33, S38, S39, and S40, the authors present the evolution of a safety-centered planning-time framework for compliance checking of safety-related processes. Process plans are modeled with a reference implementation of SPEM 2.0, which permits the representation of tasks, roles, work products, guidance, tools, and workflows. In S19 and S23, standards requirements are modeled in Defeasible Logic (Def-L)[117] and an approach for the management of safety-oriented process lines. In S25 and S28, the authors include Formal Contract Logic (FCL),[118] which is an evolution of Def-L that combines concepts and temporal knowledge representation deontic characteristics (obligations, prohibitions, and permissions). In S33, the authors augment the framework with process patterns extracted from the automotive standard ISO 26262. In S38, the authors include process compliance hints, which are based on dividing requirements in terms of the elements they target and the specific properties defined for each element. In S39, the framework adds variability management by implementing feature models that constraint the process derived according to different versions of the standards. In S40, the authors evaluate the framework and present a case

[4]https://www.w3.org/RDF/.

TABLE 10 Documents workflow methods: Limitations and validation

ID	Limitations	Validation
S1	The management of personnel, which is considered important by the authors, is not considered. Moreover, mandatory activities, which do not have associated work products cannot be checked.	The feasibility of the method application is illustrated by checking compliance between the German process model VORGEHENSMODELL (short GV-Moclel[119]) and ISO 9001.[41]
S2	There is no interest in the dynamics of the process. An UML class diagram has to be created for every standard.	The method is partially illustrated with the Space standard PSS-05.[120]

study considering the standard ECSS-E-ST-40. In S29, the static compliance checking capabilities provided by the previously defined method are analyzed in the context of agile environments.

In S37, the authors propose a method where the process elements are manually selected according to one specific standard and modeled in SPEM 2.0. Then, a standard of the same family, that is, a standard with similar characteristics, is selected and manually compared with the initial one. Such comparison, which highlights the common and variable process aspects mandated by the standards, is used to model a process line. Finally, the compliance checking is done using a feature tree-like models tool, in which basic rules constrain the selection of process elements.

4.1.3 | Compliance checking from documents workflow

Document-centered approaches are considered by some authors as a well-founded model to assess software process-related normative compliance. We have found two studies in this category. In Table 10, we present the limitations of these studies and the validation performed by the authors.

In S1, the method focuses on checking document compliance to quality constraints during the development process. For this, it is considered the state of the document, that is, planned, in use, submitted, ready for quality assurance (QA), accepted, and present. Thus, compliant activities are derived from the compliant documents because a document has a state in a determined activity. In S2, the method uses policies that trigger the appropriate checks whenever an event (open, close, and update) occurs in a document. Policies act in different modes (error, warning, and guideline), differentiated by the compulsory nature of the check (mandatory in the first two and advisable in the last). The check result can be of three types, that is, a list of noncompliant states, statistical analysis of the noncompliant states, or a filtered document with highlighted non-compliant states.

4.1.4 | Compliance checking from role-based access controls

Role-centered approaches are also considered by some authors as a sufficient model to assess process-related normative compliance. We have found six studies in this category. In Table 11, we present the limitations of these studies and the validation performed by the authors.

TABLE 11 Role-based access controls methods: Limitations and validation

ID	Limitations	Validation
S13	The use of Answer Set Programming (ASP[121]) requires highly skilled developers. The order of tasks is not considered important for the compliance analysis.	The applicability of the method is illustrated with a real world application related to a hiring systems.
S20 S22	It only considers the attributes that define a role to assign permits for actions. The visualization of the checking shows a set of regions that grows exponentially according to the number of roles involved in the checking.	Experimental results of a prototype for checking compliance of a system for managing the salary slips of employees against the European Data Protection Directive (EU DPD[122]). Additional evaluation on e-Health services.
S35	An individual conceptual model should be done for tailoring the conceptual model to a specialized context. Constraints that involve variability are incomplete in the initial model and should be updated or redefined based in every application.	Validation sessions with legal experts where performed during the modeling part of the work. For this, the legals experts were trained in UML class diagrams notation.
S31	Subsumption queries are a restricted kind of constrains that limited the reasoning process respect the normative provisions of the standards.	Two examples of artifacts collecting personal information are presented.
S36	This method does not utilizes compliance checking to show the validation of the policies. Instead, it is a method for deriving test cases that could enforce the policies at testing time.	Fictional example used to demonstrate the applicability of the method.

In S13, the method permits to address the verification of security policies applied to enterprise software. The automated security policy verification is based on the separation of duty security policy in role-based access control. In S20 and S22, the authors based their approach on the premise: "access rights are permitted or denied depending on the security characteristics of the entities involved in the access control." The process is described as a purpose-aware access control model concretized with message sequence charts (MSCs). The message chart, which represents the interaction between roles, specifies how an organization performs a particular process. Access rights to certain information have to be granted to the roles taking into account the types of permitted actions.

In S35, the authors propose a two-tier conceptual representation of the General Data Protection Regulation (GDPR[123]). The generic tier captures the concepts and principles of the GDPR that apply to all contexts. In contrast, the specialized tier describes the specific tailoring of the generic tier to a given context. In S31, the authors illustrate the formalization of data usage policies in an ontology that can be used to encode a company's data protection policy, data subjects' consent to data processing, and part of the GDPR. With this formalization, a company's policy can be checked for compliance with data subjects' consent and with part of the GDPR through subsumption queries, i.e., the inference that permits knowing that one class is a subclass of another. In S36, control policies are represented as user histories, for example, As a [Data Subject], I want [to access my Personal Data and all the information (e.g., purpose and categories)], so that [I can be aware about my privacy]. Then, such policies are translated into machine-interpretable statements. As a result, a list of policies encoding the GDPR's provisions is defined. The list of policies is instantiated with actual attributes. An access control tool uses the derived attribute classification for mapping them into the user histories and enforce policies.

4.1.5 | Other methods

Other methods are used to check compliance. We found six studies in this category. In Table 12, we present the limitations of these studies and the validation performed by the authors.

In S9, the authors present a workflow management system, which is not explicitly defined for compliance checking of software processes but can be used for that purpose. The approach uses timed Petri nets to create units of work (or tasks) that are translated into the language of the DiVinE model checker, which is based on Linear Temporal Logic (LTL). There is a small ontology representing the facts and rules found in health care policies.

In S15, S27, and S32, the authors propose an incremental life cycle model for medical software development based on model refinement, including essential software engineering activities. The method is based on a transition system that extends finite states machines with the domain of objects with functions and predicates. Such a system uses a modeling technique that integrates dynamic (operational) and static (declarative) descriptions, as well as an analysis technique that combines validation (by simulation and testing) and verification methods at any desired level of detail.

In S17, the authors present a method for discovering actual software process models based on event logs and check conformance with the SCAMPI Model (Standard CMMI Appraisal Method for Process Improvement v1.3b included in the CMMI-DEV model[37]). For this, an event log is used to construct a Petri net explaining the log's behavior automatically. The conformance checking process aims to verify the discovered process with the "assessable" elements of the development life cycle model proposed by CMMI-DEV, modeled by using LTL. The result is a report presenting if specific properties of such model hold in a log.

TABLE 12 Other methods: Limitations and validation

ID	Limitations	Validation
S9	The method is based on the correct by construction approach, which could result inflexible in cases where tailoring of requirements is permitted.	Case study of a monitor system following the guidelines for the management of cancer related pain in adults.
S15	The method is base on manually performed refinements. The initial model should not have ambiguities to the initial requirements.	Case study: Hemodialysis machine.
S27		
S32		Case study: Smart Pill Box.
S17	The checking results only shows inconsistencies between the event log and the model of the rules by highlighting the inconsistent activity in a specific color. In the absence of data logs, the methods is not useful.	Two case studies are performed to test the method and check whether the requirements for the method and its adequacy of use are met. Expert review to check is the limitations of the SCAMPI process are properly addressed.
S41	The method is too flexible and tolerates inconsistencies.	The evaluation was performed on two use cases, that is, an agile process, a safety-critical system in the air traffic management domain.

In S41, the method focuses on an approach that relies on passive process execution, that is, tracking process steps via monitoring engineering artifacts, complemented by constant evaluation. This approach tolerates inconsistencies, permitting process engineers to deviate from the accepted practices in an informed fashion.

4.2 | Analysis

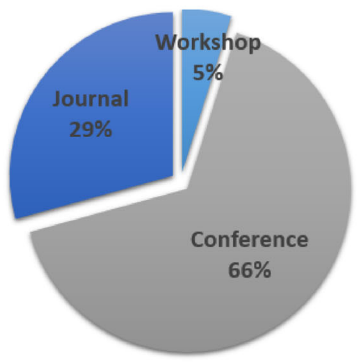
In this section, we present the analysis of the results in relation to the addressed research questions presented in Table 1.

4.2.1 | RQ1. Publications distribution

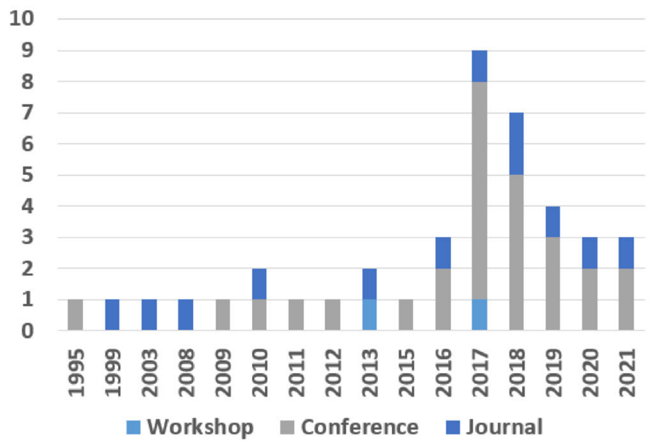
The distribution of the studies in terms of publication types and the years when they were published is presented in Figure 3.

As Figure 3A depicts, most primary studies were published in conferences (66%), while journals (29%) and workshops (5%) were the sources of fewer studies. In Figure 3B, we are presenting the distribution of the studies according to the year of publication. In particular, in the first years (1995 to 2009), only one or no publications were discovered. The distribution of the publications presents one peak in 2017, where nine papers were found. Then, in 2018, the publication of papers descent again to seven papers and continue in descending mode until 2021. We also could see that most of the studies have been found after 2017 (26 out of 41 studies 63%). However, the literature revision during 2021 only included the first 3 months. Thus, the trend could change during this year.

Concerning the active research groups, we looked at the selected primary studies' affiliation details. The assignment of contributed studies of each active research group is based on the affiliations given in these studies to the first author. Table 13 presents the active research groups (with at least two publications on the mentioned topic) and the corresponding number of contributed studies. The results depict that the Mälardalen University is the leading organization in terms of the number of publications, followed by Griffith University. Then, Charles University,



(A) Venues Types.



(B) Years.

FIGURE 3 Publications distribution

TABLE 13 Active research groups

Affiliations	Primary studies	Total
Mälardalen University	S19, S23, S25, S28, S29, S33, S37, S38, S39, S40	10
Griffith University	S14, S24, S26, S34	4
Charles University	S15, S27	2
Loughborough University	S3, S4	2
Universidade de Lisboa	S21, S30	2
University of Oslo	S8, S11	2
Other universities/centers	S1, S2, S5, S6, S7, S9, S10, S12, S13, S16, S17, S18, S20, S22, S31, S32, S35, S36, S41	19

Loughborough University, Universidade de Lisboa, and the University of Oslo appear with two publications. The rest of the universities and centers only have one publication (in total, 19). Thus, there are research groups around the world doing research in this topic.

We did not set a lower boundary for the year of publication in our search process because, to the best of our knowledge, there is no precise date where the concept (or the topic) was coined, as it happens in other subject areas. However, as Figure 3B depicts, the time frame identified the first primary study on the topic back in the 1990s. Previous to this year, we did not find primary studies. Thus, we could consider the 1990s the initiation of this topic's work. This result corroborates with the publication of the seminal paper "Software Processes are Software Too"[124] back in 1987, where the author, Leon Osterweil, discussed the nature of software processes concluding that processes are in the category of software programs and therefore, they can be programmed. The conceptualization of "software process programming" by Osterweil[124] could be the source of interest for work related to the formalization of processes and their normative constraints. However, after analyzing the general temporal view of the studies, we can conclude that the number of studies about automated compliance checking of software processes has been rare through the years. Although the apparent increase in the number of primary studies found in 2017, this result corroborates that the topic has been somewhat neglected. However, some groups, especially in Europe and Australia, continue advancing the research on the topic.

4.2.2 | RQ2. Methods' characteristics

In this section, we present the characteristics of the methods described in the primary studies by answering questions RQ2.1, RQ2.2, RQ2.3. RQ2.4, and RQ2.5.

RQ2.1. Compliance strategies

In the studies, compliance checking is performed by using different strategies, that is, planning time, running time, and after the fact. As Figure 4 depicts, there is a tendency to perform compliance checking at design time (studies S3, S4, S6, S7, S8, S10, S11, S12, S14, S15, S18, S19, S20, S21, S23, S24, S25, S26, S28, S29, S30, S31, S33, S34, S35, S36, S37, S38, S39, and S40). One reason for this situation is that the majority of the normative documents require compliant process plans to support the execution of the actual development process activities. In addition, contracts between partners require initial agreements on the development activities. Compliance checking at a running time is also greatly considered in the methods studied (studies S1, S2, S3, S4, S5, S9, S13, S14, S15, S18, S20, S22, S24, S26, S27, S32, and S41). This strategy permits two things. First, to guarantee that process plans are executed (as in studies S3, S4, S14, S15, S18, S24, and S26, which combines design time with execution time). Second, it warns process users regarding the possible uncompliant actions on the fly. Compliance after the fact, which is represented with a lower number of studies (S3, S4, S16, S17, and S41), is an alternative that could reconstruct a process in the absence of a predetermined model. Compliance after the fact produces the actual process used in the software developments, which can be uncompliant and challenging to repair.

RQ2.2. Languages used to represent software processes

With well-defined processes, organizations and third parties (such as customers and regulatory bodies) are informed about what is expected in a certain project and from whom. For this reason, the granularity in which a process is represented has significant importance in compliance checking. As Figure 5A depicts, the methods use 14 different languages to represent the processes. In particular, S15, S27, and S32 use Abstract State Machines (ASM);[125] S13 uses Answer Set Programming (ASP);[121] S14 uses Composition Tree (CT) notation;[126] S9 uses Time Compensable Workflow Modeling Language (CWMLT);[127] S12 uses Foundational Subset for Executable UML Models (fUML);[128] S22 uses MSC;[93] S21, S24, S26, S30, S31, and S34 use Web Ontology Language (OWL);[129] S17 uses Petri net; S16 uses Business Process Management Notation (BPMN);[116] S20 uses Satisfiability Modulo Theories (SMT);[130] S1 uses Process Programming & Tailoring (ProcePT);[72] S6, S7, S19, S23, S25, S28, S29, S33, S37, S38, S39, and S40 use SPEM 2.0;[115] S2, S3, S4, S5, S8, S10, S11, S18, S35, and S41 use UML;[131] and finally, S36 uses OASIS eXtensible

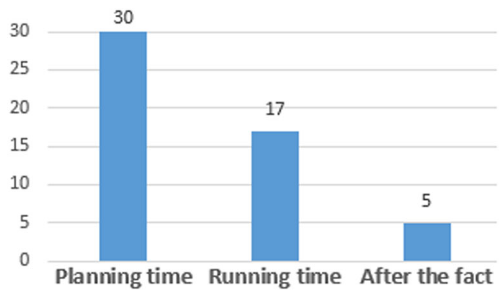
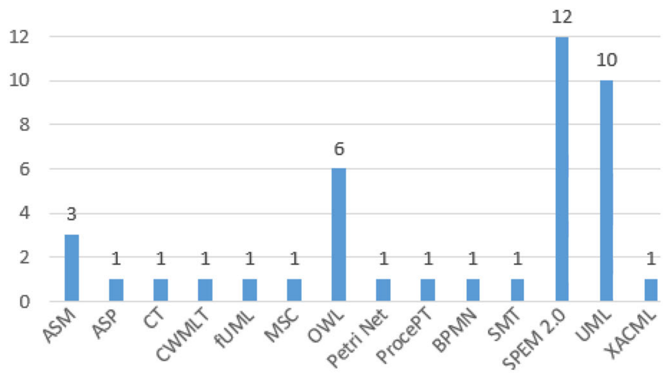
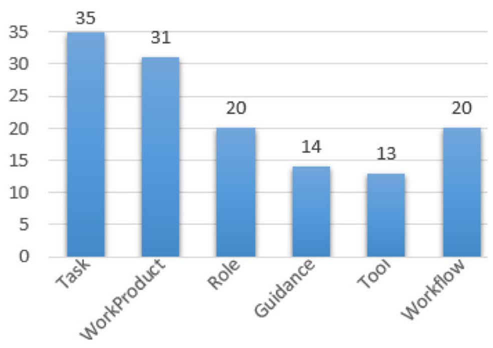


FIGURE 4 Compliance strategy



(A) Languages for Modeling Process.



(B) Process Elements Represented.

FIGURE 5 Process modeling languages characteristics

Access Control Markup Language (XACML).[132] It is important to note that models created in OWL and UML could also be considered as new languages. As presented in Figure 5B, some process elements have more importance than others in the modeling languages created/reused, as each normative framework regulates different process elements. In particular, significant attention is given to the tasks and work products.

Researchers use different types of approaches and methodologies to represent the software process to be used for automatic compliance checking. The purpose of the primary studies is to model the specific concepts provided in particular standards. In most cases, the standards only prescribe the sequence of tasks (process behavior) and process outcomes (defined in the work products). Only a few primary studies provide the possibility of modeling several process elements rather than only tasks and work products. As a result, new languages with limited scope have been created. The continuous creation of ad hoc software process-related modeling solutions could be a disadvantage, especially when well-defined process modeling languages (SPEM 2.0 and BPMN) can be used and extended according to specific needs.

RQ2.3. Languages used to represent compliance requirements

Compliance checking of software processes builds on the capabilities provided by logic-based languages for representing the requirements prescribed by the normative frameworks. The selection of languages in the primary studies was very diverse, showing a similar trend to languages used to represent software processes. In particular, in the analysis of the selected primary studies (see Figure 6), we found that a wide range of studies uses FCL,[118] that is, S25, S28, S29, S33, S38, S39, and S40, to formalize the requirements prescribed by the standard. In the second place, the preferred languages are LTL,[133] that is, S9, S12, S15, S17, and S27, and DL, that is, S14, S24, S26, S31, and S34. In the third place, the selected language is First Order Logic (FOL), that is, S1, S2, S20, and S22. Other languages, such as UML, Object Constraint Language (OCL),[134] and Semantic Web Rule Language (SWRL),[135] are used in three studies, respectively. There are also languages in many other flavors that the researchers prefer to represent the requirements prescribed by the standards. We found Answer Set Logic (ASL), Basic Constraint Language (BCL), Computational Tree Logic (CTL), Def-L, GAL, Process Pattern Structure Tree (PPST), Shapes Constraint Language (SHACL), XACML, Extensible Stylesheet Language Transformations (XSLT), and a database approach (not specified).

In general, every formal method has its strengths and limitations as its formal approaches and semantics. The coverage, readability characteristics, and tool support also vary from one formal language to the other. Thus, it is crucial to find the correct balance between all the capabilities

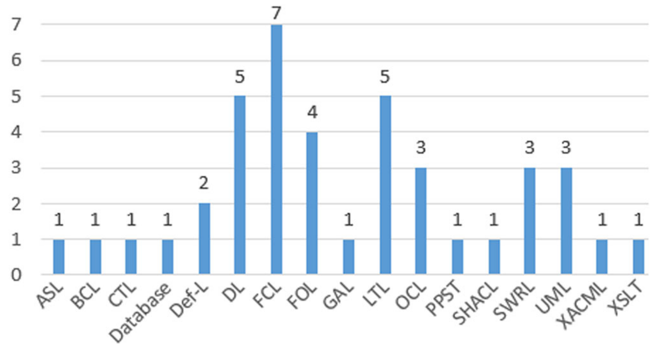


FIGURE 6 Norms modeling languages

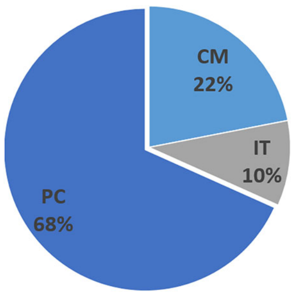


FIGURE 7 Tooling state

included in the languages used to check compliance to achieve the best fit for the problem at hand. For example, FOL has a standard interpretation in terms of state individual with properties that are exemplified timelessly or a temporal instant. Conversely, processes are temporal developments that can be analyzed as temporarily structured sequences of stages. We could think that alethic (the logic of necessity and possibility) temporal logic like LTL is the best fit to express behavioral specifications of process models. However, LTL and other logics such as DL, UML, OCL, SWRL (which combines OWL with Datalog), BCL, CTL, GAL (which are based on UML), PPST (which is a workflow graph that is based on propositional logic), XSLT (a language designed primarily for transforming XML documents into other XML documents), and the database approaches are declarative languages. In declarative languages, the conclusion is determined after the truth of the conditions provided in the rules.

The previously mentioned languages (see the complete list in Appendix C) provide different syntax that could be more or less beneficial for compliance tasks. However, it is common to find incomplete and contradictory information in norms. Moreover, some cases support compliance (e.g., tailoring) even though the normative requirement are not strictly fulfilled. For these cases, declarative languages are insufficient. Therefore, other formal languages are more suitable. For example, in S13, the method uses ASL, which is based on autoepistemic logic and default logic that makes a distinction between a strong (or traditional) negation and negation as failure (negation derived from incomplete information). Thus, ASL allows contradictory conditions, but it does not resolve the conflict if such conditions hold simultaneously. In S16 and S38, the authors selected SHACL and XACML, respectively. SHACL and XACML are declarative, but they have advantageous characteristics. SHACL has a specific set of constraints (SPARQL[5]) that permit a more precise selection of elements from a set. XACML is a language used to describe policies and rules. It also provides the concept of obligations explicitly provided and the function SHALL, which is used to describe requirements. However, the formalization in those two languages tends to be complicated, and they are not defeasible languages. Instead, Def-L provides superiority relations that prioritize rules that conflict. Furthermore, FCL, which is derived from Def-L, also includes deontic elements. Deontic elements explicitly describe the concepts of obligation, permission, and prohibitions, which are the typical expression found in a normative document (as explained in Section 2.1).

RQ2.4. Level of automation

The automation part claimed in the studies is related to the automatic comparison between the process and the normative documents. Frameworks composed of chained tools also automatically transform the information between such tools. However, the formalization of requirements is a task that should be done manually. In some cases, the methods provide patterns. For example, process patterns are supported in S5 and S7, while normative patterns are described in S33 and S40. Templates for modeling requirements are provided in S38, S39, and S40. Figure 7 presents the distribution of the state of the tools support. As the figure depicts, 68% of the methods, namely, S1, S2, S4, S5, S6, S7, S8, S9, S11, S14, S15, S16, S22, S24, S25, S26, S27, S28, S29, S32, S33, S34, S35, S37, S38, S39, S40, and S41, are prototypes that are used as a proof of concept (i.e., PC); 22% of the methods, namely, S3, S13, S18, S19, S21, S23, S30, S31, and S36, are conceptual models (i.e., CM); and only 10% of the methods, namely, S10, S12, S17, S20, and S22, are fully implemented tools (i.e., IT). Essentially, the surveyed methods require human intervention, especially to implement the inputs of the reasoning process. Furthermore, the manual mapping or formalization of requirements as constraints requires specialized knowledge of the formal techniques.

Formal approaches are often not easy to use for many process engineers. Given this aspect, there is a need to automate the transformation of normative requirements into formal representation, or at least, to provide editors, which could facilitate the formalization part. In addition, it is challenging to promote methods for automatic compliance checking in the industry when the tool support is nonexistent.

RQ2.5. Evolution handling

As presented in Figure 8, only 39% of the primary studies, namely, S1, S6, S9, S19, S20, S23, S24, S25, S28, S29, S33, S35, S37, S38, S39, and S40, present explicit means for addressing software process reconfiguration in the light of standards evolution (i.e., the release of a new version of standards), tailoring (i.e., the selection, eventual modification, and implementation rationale), and process diversity (application of several

[5]https://www.w3.org/TR/rdf-sparql-query/.

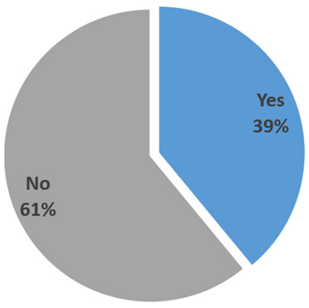


FIGURE 8 Evolution handling

standards in the same project). In S1, for example, there are specific structures, such as the definition of integrity levels prescribed by safety standards, which permit the deletion and modification of work products and activities according to the project's characteristics. In S6 and S9, there is a tailoring step when creating processes models, which is in charge of transferring only those requirements, methods, and activities, which are relevant according to the system's safety integrity levels. In addition, in S9, a monitor system follows the guidelines for managing constraints and permit workflows that are correct by construction, preventing the incorrect composition of tasks. In S19, S23, S37, and S39, the methods use process lines methodologies, permitting the management of commonalities and variabilities that appears in process diversity. They also use feature tress, which permits to constrain the selection of the elements according to variation points. In S20, the method can compare two different versions of a policy by considering the evolution of the rules (called a refinement). If the two rules agree, then the rules are compliant. In S24, it is used a mechanism called powertype, which is a pattern for modeling that combines instantiation and generalization semantics in process metamodeling. In S25, S28, S29, S33, S38, and S40, the change management is based on the extension capabilities reuse and traceability provided by the process modeling language SPEM 2.0. In S35, the variation points describe specific situations where the generic representation needs can be adapted to a given domain or organizational context. A variability table was used to enable the legal expert to verify that the list of extracted variation points was complete and precise.

Evolution handling is a crucial aspect of process-related compliance management. However, the results of the SLR show that this aspect has been somewhat neglected. Moreover, the rules, in almost all the methods, are hard coded, which is a situation that could lessen their extensibility, generality, and application scope. Therefore, there is a need to provide change management means that permit process engineers to understand, plan, implement, and communicate the change due to the evolution of the standards, tailoring, and process diversity.

4.2.3 | RQ3. Potential impact

In this section, we present the potential impact of the studies by answering questions RQ3.1, RQ3.2, RQ3.3, and RQ3.4.

RQ3.1. Application domains

Several application domains, that is, safety-critical systems, SPI and process quality, data protection, software process verification, cybersecurity and health care, are addressed in the primary studies. Figure 9 presents the distributions of the application domains. The most representative application domain is the safety critical, with 51% of the studies tackling this sector, that is, S3, S4, S6, S8, S11, S15, S16, S18, S19, S23, S25, S27, S28, S29, S32, S33, S37, S38, S39, S40, and S41. In safety-critical context, compliance with standards, specifically safety standards is imposed, either by law or by consumers. Thus, it is not uncommon that efforts that help to simplify this task are on the mind of current researchers. Then, we find that the researchers are interested in SPI and quality (22%), that is, S1, S2, S5, S14, S17, S21, S26, S30, and S34, and data protection (15%), that is, S10, S20, S22, S31, S35, and S36. Other application domains are also represented in less quantity, that is, software process verification (7%), that is, S7, S12, and S24; cybersecurity (2%), that is, S13; and health care (2%), that is, S9.

RQ3.2. Normative documents targeted

Different standards have been modeled and used in the experimentations or illustration results provided in the primary studies. As depicted in Figure 10, the standards more used are ISO 26262[54] (15%), that is, S6, S19, S23, S25, S29, S33, and S37; IEC 61508[50] (9%), that is, S3, S4, S8, and S11; and ISO/IEC 15504[7] (9%), that is, S21, S26, S30, and S34.

To a lesser extent, the primary studies used GDPR[123] (6%), IEC 62304[57] (6%), SAE J3062[136] (4%), FDA principles for software development (4%), software process guidelines (4%), internal guidelines (4%), ECSS-E-ST-40C[58] (4%), DO-178C[56] (4%), and CMMI[37] (4%). Other standards were also used, representing 19% of the studies (i.e., ISO 12207,[5] ISO 14971,[137] ISO 9001,[40] ISO/IEC 29110,[43] ISO/IEC 90003,[42] ISO/IEC TS 33053,[8] PIPEDA,[63] ASPICE,[138] and EN 50126).[139]

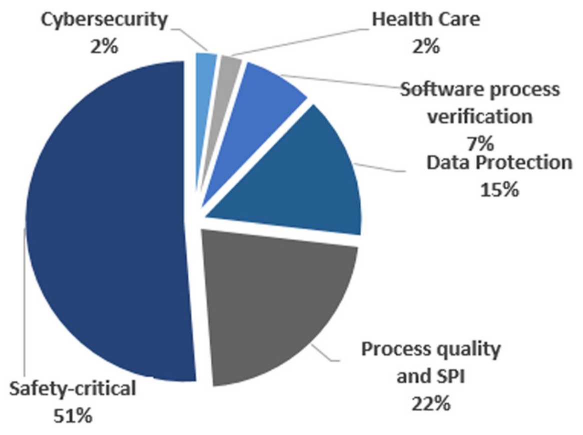


FIGURE 9 Application domains

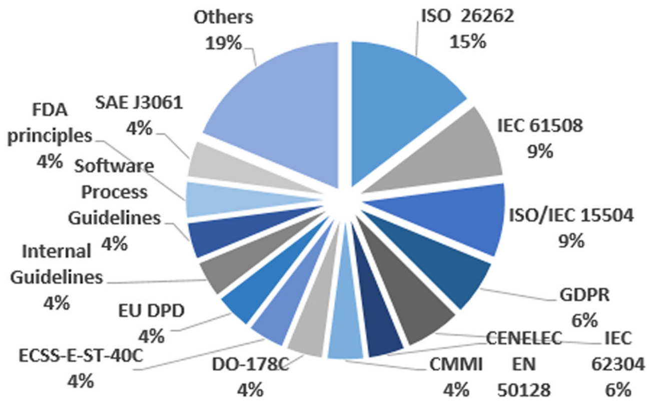


FIGURE 10 Normative frameworks addressed

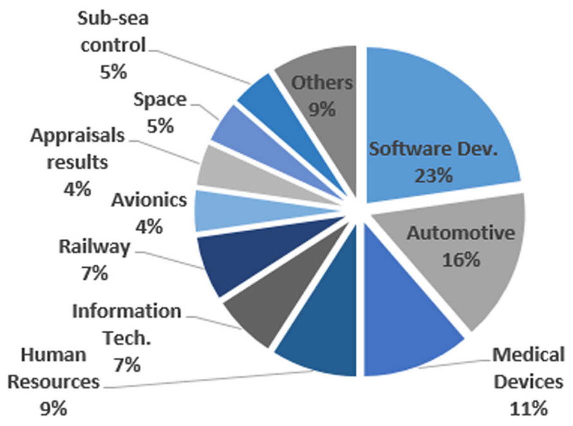


FIGURE 11 Illustrative scenarios

A particular problem of the normative documents targeted is that, in almost all the studies, they are addressed in isolation, reducing the possibility to generalize the results.

RQ3.3. Illustrative scenarios

As presented in Figure 11, the studies focused primarily in general aspects of software development (23%), that is, the GV-Model in S1, Case PSS-05 in S2, testing procedures, and scrum processes; automotive examples (16%), that is, S6, S19, S23, S25, S33, and S37; and medical devices

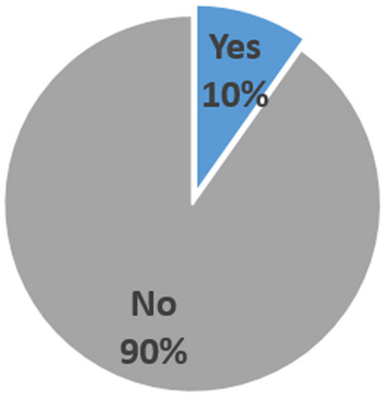


FIGURE 12 Agile support

development (11%), that is, S17 and S19 with the hemodialysis machine, S32 with the smart pill box, S39 with a general risks analysis for medical devices, and S40 with a wearable fitness appliance. Representative examples were also found in human resources systems (9%), that is, S13, S14, S20, and S22; general applications in information technology (7%), that is, S10, S17, and S35; railway (7%), that is, S16, S28, and S28; avionics (4%), that is, S29 and S41; space (5%), that is, S18 and S49; subsea control (5%), that is, S22 and S36; and appraisals results (4%), that is, S31 and S34. Other illustrative scenarios have a 9% of representation (i.e., agilized environments, programable electronic systems, recommendation handling systems, and services delivery).

In total, 18 of the 41 studies (approximately, 44%) used data extracted from industrial settings to evaluate their methods, that is, S4, S5, S8, S9, S10, S11, S13, S14, S15, S16, S20, S21, S22, S27, S30, S31, S32, and S41. The validation used by the studies is presented in detail in Tables 8 to 12. This result shows that there is no consistent use of data from industry, limiting the validation and evaluation of the studies.

RQ3.4. Agile support

Commonly, normative documents that are used in software development do not mandate a specific kind of life cycle. However, documents state the tasks that should be carry out, the artifacts to be produced, the roles required, and the recommended techniques and tools to be used. Normally, plan-driven development processes are more suitable in this environments, but industries are becoming more agile and the need to comply with standards is still needed. In most of the methods used, constraints derived from the standard requirements are defined strictly. Therefore, there is not room for methodologies beyond plan driven. However, there is some explicit support for agile in the studies selected (only 10% of the studies showed some agile-related information, as presented in Figure 12).

The information related to agile compliance is having different characteristics in different studies. For example, in studies S7 and S12, the techniques apply for compliance checking with the SCRUM framework, but there are no direct observations regarding compliance checking with a regulatory text. In S29, the support is provided to agilized environments, that is, environments that result from the combination of agile and plan-based development processes, especially applicable to regulated contexts. In S35, the support is presented by providing normative requirements formalization templates in the form of user stories. Finally, in S41, the framework uses mining techniques to extract the developers' performed work. This technique is restricted to process executions and reconstruction of compliance after the fact. Thus, some support for agile methodologies exists. However, there is much room for improving this aspect. Correctly combined with other techniques, agile methodologies in compliance with normative frameworks could be better supported.

5 | DISCUSSION

In this section, we answer RQ4 by highlighting the challenges found in the selected studies. Such challenges are discussed based on the results found in Section 4 and the authors' interpretation of such results.

5.1 | The use of software process modeling languages

There is a first evident limitation regarding the expressivity of the software process models. In particular, significant attention is paid to process tasks, and work products (as presented in Figure 5B) because these aspects are the control baseline defined by most normative frameworks. However, it is essential to provide process models that describe the whole process to support the management perspective. For example, a process description that does not indicate roles in charge of tasks is not likely to be of much value in supporting reasoning about how to improve team coordination. Therefore, as most methods focus on the pure compliance problem, they are opening a breach with the companies' needs regarding process management.

Some contributions consider a more comprehensive range of concepts for describing software processes. However, definitions are ambiguous. For example, in S3 and S4, a performer (the person required to do a task) should be modeled under the concept capability. In S8 and S11, a performer is called an agent and could be of two types, that is, organization and individual. In S31 and S36, the data subject is the primary role to which some permissions are granted, based on different attributes of the data that the role manages. However, there is no information regarding the subject itself, that is, its capabilities or properties. In S12, the term resource is used to map the roles involved in the process and work products produced.

Notably, a software product with desirable attributes (e.g., safety, quality, and reliability) results from a well-defined process where several artifacts supplement each other and actors perform on them with specialized techniques and tools. Consequently, it is essential to describe all concepts and structures included in a software process and their more relevant properties. In the last years, researchers tend to use consolidated process modeling languages, such as SPEM 2.0 and BPMN (see Figure 5A). Such modeling languages provide all the process elements required for compliance checking and process management. In particular, SPEM 2.0 has additional characteristics, for example, extensibility and reuse capabilities, that make it very suitable for compliance management. In addition, most of the consolidated process modeling languages already offer tool support and user guidelines, which makes their use easier. Thus, we consider that new research efforts in automatic compliance checking, specifically for software processes, could consider existing process modeling languages to accelerate results in the topic and standardize the techniques and tool support.

5.2 | Language suitability for addressing normative requirements

The languages used in the primary studies (see Figure 6) provide exploitable characteristics. For example, FCL explicitly describes the compliance concepts, that is, obligations, prohibitions, and permissions. FCL and Def-L provide defeasibility, that is, means for attacking previous conclusions and rule superiority. Such characteristics also allow reasoning with contradictory and ambiguous information. ASL provides a clear distinction between strong (or traditional) negation to represent a negation derived from evidence and negation as failure admitting reasoning with incomplete information.

The remaining languages consider the requirements as constraints that restrict the processes' scope of action. In other words, requirements are defined as the obligations that the process or the process elements should fulfill or the prohibitions that should avoid. Thus, they can cope with the concept of obligation (or prohibition) very well, even though such a concept is not explicitly defined. However, the possibility to handle contradictions and incomplete information are not provided. Such reduced semantics lead to reduced reasoning capabilities, decreasing the scope of the methods used for compliance checking.

First and foremost, compliance is a relationship between permissions (what you are allowed to do), obligations (what you have to do), and prohibitions (what you should avoid) (see Section 2.1). In the case of compliance with standards, the concept of tailoring is also relevant. Tailoring allows organizations to adapt normative requirements to specific project conditions. In the tailoring process, justifications (called rationales) are a mandatory element to legitimize changes. Tailoring can be seen as a sort of justified exception handling in software process compliance checking. Thus, the language selected to represent normative frameworks should facilitate the description of the mentioned concepts because they are necessary to tackle the compliance checking problem of software processes.

5.3 | Towards a generic and domain-agnostic method

Formalizing normative requirements and software processes permits the discovery of errors. For example, in S1, the authors said that “By formalizing the GV-Model, we discovered mistakes in its informal description.” This is true with every formalization enterprise because for doing this task, an in-depth analysis is required. Thus, an important aspect that needs to be covered in compliance checking is the management of such contradictions. In addition, when the normative documents are issued, their authors considered them as applicable from a specific point in time. However, new versions are released to improve concepts or cope with new aspects. Thus, there should be a mechanism in the normative formalization that permits the management of changes. In general, most of the languages analyzed (see the answer to RQ2.3 in Section 4.2.2.3) are able to perform inferences in deductive reasoning, which conclusions can only be attacked on their premises. However, rules are hard coded and cannot be defeated with new information. Therefore, such models do not permit an easy evolution when the normative document changes or contradictions are discovered.

In the studies, there are different approaches used to represent the information contained in the software processes (see Figure 2) and the compliance checking strategies (see Figure 4). Some of these methods describe the requirements provided in the norms in terms of software process artifacts (see Section 4.1.1). Providing a model in such a way can be an improvement towards the practical use of standards because it minimizes the interpretation and ambiguity. However, they may limit their usability because every standard would require its own model. For this reason, a generic mechanism, which separately captures a wider range of concepts related to the normative documents and the process, is required.

There are some languages in the studies explicitly created for dealing with normative reasoning. For example, in S10, the authors use the language GAL, which can express many types of legal and organizational requirements. However, in its actual form, GAL only considers constraints that apply to roles in a process. We find that FCL is generic. In particular, it provides deontic concepts and rule prioritization (which can be used for exception handling), which permits a semantic correlation of information in terms of specific and clear concepts associated with normative documents. Moreover, it permits the inclusion of new rules that can challenge the preestablished ones. Semantic correlation and defeasibility are essential properties of a language intended to be used for compliance checking.

In general, most of the methods aim at seeking compliance at planing time (as presented in Figure 4). As such, compliance checking is able to demonstrate intentional compliance, that is, distribution of responsibilities, such that if every actor fulfills its goals, then the compliance is ensured.[140] However, intentional compliance can only permit, not guarantee, any quality attribute of the process. Nonconformance between process design and execution can put the software development at risk in realizing the compliance required. Thus, combinations between compliance of software process plans and follow-ups during process execution should be made available (as studies S3, S4, S14, S15, S18, S24, and S26 provide). In our opinion, the results of the methods surveyed in this study could fertilize each other towards the consolidation of a more holistic, generic, and normative-agnostic solution that can tackle, for example, quality, SPI, safety, and cybersecurity. A resulting method with such characteristics could be more attractive to organizations, and industrial applications could be made on a larger scale.

5.4 | Increase the level of automation and tool support

It is difficult to guarantee industrial adoption when there is nonexistent or loosely coupled tool support (as seen in Figure 7). Thus, it is crucial to provide adequate and complete tool support for automatically perform compliance checking. This aspect can be facilitated by integrating existing development tools like Rational Method Composer, which is are already used in industry. It is also essential to increase the automation means to ease the creation of rules, that is, rule editors and process models, because formalizing requirements still need human intervention.

A good aspect is that the research arena moves towards automatic means to model the process after the fact, namely, process mining approaches. These approaches suit agile/agilized environments very well if automation is used during the development process stages. However, where there are no process logs available, the approach is not very suitable. Besides, mining techniques could extract the information too late in the development process, and compliance may be challenging to fix. In our opinion, process mining techniques can be included in a framework for facilitate the compliance checking life cycle but not as a standalone technique to guarantee compliance.

Finally, there is the issue of information visualization, which is often an aspect associated with the usability of the tool support. In some of the studies collected, we find good examples of such visualization. For example, in S9, the browser presents a hierarchical representation of data in a mind map. In S41, a web-based process dashboard makes process progress and quality constraints evaluation results continuously available to software engineers. Those methods that use SPEM 2.0-like artifacts are also showing the information in a suitable form. Unfortunately, the tool support is still not concretized (as answered in RQ2.4).

5.5 | Application in practice

The methods provided in the primary studies represent a set of engaging, applicable, and useful aspects contributing to the automation of compliance checking of software processes. In general, there are diverse application domains (see Figure 9), different standards targeted (see Figure 10), and illustrative scenarios (see Figure 11). From such scenarios, valuable lessons learned and practical insights have also been collected. However, in most cases, normative documents have been considered in isolation resulting in ad hoc solutions. In addition, the use of case studies from industry, even though it is good (44%), should be increased in order to provide real setting insights. Moreover, there is no explicit support for agile in most of the methods (see Figure 12). In reality, manufacturers have to deal with software process diversity, tailoring, and standards evolution. Moreover, software organizations are moving towards agile, even in heavily regulated domains, such as the safety critical. Thus, the narrow focus of the methods reported, the poor support for agile environments, and the nonconcretized tool support (which is the common aspect) may be factors that also hinder their application in practice.

Some other aspects that should be considered for the applications in practice of the methods are listed below.

Compliance diagnostic: It is important to provide process engineers with solutions that guide them in establishing a sound relationship between the software process models and the normative imperatives. However, in most of the methods, the compliance results regard to be only high-level

diagnostic, that is, they mainly signal the problem (or the violation of a requirement). Such diagnostics should be interpreted as a warning for which further analysis should be carried out. Therefore, concrete reparation policies that suggest how to solve the compliance violations are required.
Transparency: Article 22 of the GDPR[123] stipulates that whenever a decision that legally or significantly affects an individual relies solely on automated processing, the right to contest the decision must be guaranteed. Thus, there is a need to clearly explain the automatic compliance checking results that guarantee rights for organizations and individuals. Consequently, means for transparency have to build in the methods. Transparency can be achieved by implementing data provenance and traceability mechanisms. Data provenance is associated with data regarding origin, changes, and details supporting confidence or validity. Traceability is related to the relationships between compliance results, software process elements, and normative frameworks. Most of the studies consider traceability in the models in one another way. However, not all the studies consider the inclusion of textual explanation in the formal representations. Thus, we also consider that informal explanations should always accompany formal specifications to clarify the rules' meaning and place them in context. In that way, if problems arise with released software products, transparent, traceable, and fully documented compliance checking results could show that the prescribed procedure was applied.
Further support for rule formalization: Assuming that the compliance checking algorithms and the tool support are correctly designed, sound results may be expected. However, correct answers depend on the quality of the inputs that the tool receives. Unfortunately, the use of mathematical methods for compliance checking, as presented in the studies, are not a guarantee of correctness because humans apply them. Cognitive biases, which are deviations from the rational way we expect our brains to work, may appear when we formalize normative documents. Therefore, there must be a layer of trust in the methods, which guarantees no requirement poisoning, that is, rules incorrectly derived from the normative frameworks. In that way, we could fight the lack of trust between organizations participating in global software development governance and the utilization of automated means for compliance checking.

6 | VALIDITY OF THE RESULTS

The research method used in this work intends to capture all studies addressing automatic compliance of software processes. Therefore, we followed strictly the guidelines recommended in Kitchenham et al.[12,13] However, some threats could undermine the validity of the results obtained in this systematic review. This section addresses potential threats regarding publication bias, identification of primary studies, and data extraction consistency.

Publication bias. Refers to the problem that positive results are more likely to be published than negative results. We designed a review protocol by following the steps proposed by the guidelines described in Section 3. The first author prepared the protocol while the second and third authors (who have previously participated in research involving SLR; see for instance Carlan et al[141] and Ul Muram et al.[141]) ensure appropriateness by performing an exhaustive review and assessment. We also pay careful attention to external reviewers' critical comments on an earlier versions of this paper (see the report previously published in Castellanos Ardila et al,[143] which was also included in the Ph.D Dissertation[144]). Their observations lead to an increase in the clarity of the review protocol. We also include Google Scholar to avoid limiting information sources to specific publishers, journals, or conferences. In order to accumulate reliable information, we decided not to restrict the search dates and avoid the inclusion of technical reports, works in progress, unpublished paper, or non-peer-reviewed publications.

Identification of primary studies. Refers to the strategy used to collect all possible studies. We aimed at ensuring that the search addresses our review intentions. For this, we performed a careful characterization of the topic (see Section 2) in an attempt to discover all the possible concepts and their respective synonyms. We additionally tested such concepts in a known digital library. With such a result, we concretized our search string as presented in Table 3. We are aware that the search strategy is not sufficient to capture all the possible studies. For this reason, we carry out the snowballing process to mitigate this threat. Consequently, we manually scanned and analyzed the references used primary studies retrieved from the automated search (backward snowballing) and the citations such studies get in Google Scholar (forward snowballing). The main goal was to ensure that our SLR also covers follow-up works that might exist but have not been included in the search. The process of identifying primary studies was performed by the first author, who is a Ph.D. student. The prospective primary studies were evaluated and cross-validated by the second and third authors, who are experienced researchers.

Data extraction consistency. Refers to the strategy to extract all data required to address the review questions. We based our data extraction strategy on the data extraction criteria presented in Table 5. The first author prepared the selection criteria by considering the quality criteria presented in Table 6, and the research questions we intend to answer in the SLR, presented in Table 1. We checked the data extraction table's consistency by conducting a data extraction pilot on a set of primary studies. After that test, we refine the data extraction table by aggregating parametrization. For instance, we defined parameters for the information regarding automation levels of the studies surveyed (CM, PC, and IT). The data were distributed in two tables. The first table contains selection criteria and articles identification (see Appendix A). The second table contains 16 columns aimed at recording the information corresponding to the research questions (see Appendix B). All this information was recorded and analyzed by using Excel spreadsheets. We consider that the adopted data extraction strategy could help to reduce threats regarding the data extraction consistency.

7 | RELATED WORK

SLRs regarding process-based compliance checking have been conducted primarily in business-related areas. In particular, the work included in Becker et al[145] presents a classification of approaches for compliance checking at design time (processes are checked at the moment they are created) based on business-related compliance patterns and the use of different techniques for modeling processes. The work of Ly et al[146] provides a systematic comparison of existing approaches for monitoring compliance rules over business processes during run time (compliance is checked during process execution). Hashmi and Governatori[147] evaluate selected frameworks regarding the modeling of different compliance requirements and their link with the business process. Hashmi et al.[26] and Kharbili et al.[148] present an evaluation of compliance management strategies at different times of the compliance life cycle, that is, design time, run time, and auditing time (compliance is checked after the process has been executed). Hashmi et al. also review how control flow structures and norms are modeled. Like the previous SLRs, our work also found that different formal approaches are used to model processes and normative frameworks. However, any of these SLRs include compliance checking of software processes, which is our focus. Moreover, in our work, we found that the concepts used to describe processes are modeled according to the specific standard's needs. Instead, the business context reviews found that it is more common to model artifacts in existing business-oriented process modeling languages. Besides, none of the previous SLRs review the concepts required for modeling complete process specifications, according to software process needs, that is, the definition of roles, work products, guidance, and tools. Only the review presented by Hashmi et al[26] considers the data management at run time but only from the perspective of norms definition.

In engineering contexts, we find the work of Boella et al[149] and, more recently, Akhigbe et al,[150] whose focus is surveying the representation of knowledge for legal and regulatory requirements engineering. On the one hand, the work of Boella et al. focuses on norms representation. On the other hand, Akhigbe et al. focus on studying the uses and main claimed benefits and drawbacks of goal-oriented and non-goal-oriented modeling methods for legal and regulatory compliance. Instead, we focus on characterizing compliance checking as a whole. For this reason, we include the languages used to model the normative frameworks and the processes used to engineer the software.

There are works targeting software processes from different perspectives. For example, the work done by von Wangenheim et al.[151] is an SLR that focuses on software process capability/maturity models. In addition, the work done by Yan et al.[152] presents a systematic mapping study on quality assessment models. Our work, instead, focuses on all the models that can be derived from normative frameworks applied to software processes, which include quality and SPI.

The work done by García-Borgoñon et al[153] focuses on the identification of software process modeling languages. We do similar research, but we also include the models for normative frameworks required for compliance analysis. Finally, in the context of safety-related compliance management, we find the work of Nair et al,[154] which focuses on the characterization of compliance artifacts, including the importance of providing process-based compliance checks. However, it is not covering how such checking is done.

8 | CONCLUSIONS AND FUTURE WORK

The world is permeated by software applications, many of them acting in safety-critical environments. Organizations doing software solutions also have to implement processes, often mandated by normative frameworks, that is, standards, regulations, laws, and guidance. For this reason, software process compliance is not an option. However, software process compliance checking is challenging due to the numerous normative frameworks to which organizations need to comply. Automated techniques are more objective than manual techniques because the result obtained from the checking algorithms (they are well designed) are deterministic, which show the same results when repeated with the same parameters and input data.[88] In the research arena, several studies have tackled the compliance checking problem of software processes from diverse perspectives. In this paper, we characterized the state of the art by performing a SLR on the topic.

In our opinion, the primary studies selected provide a set of ad hoc solutions that are interesting, applicable, and valuable contributions to the topic. There is also diversity regarding process modeling languages and the types of artifacts described. Most of the languages used for representing requirements primarily cover the concept of obligations and prohibitions (what should be done and what should be avoided) but leave aside other considerations, such as the permitted actions that could indirectly affect compliance, for example, requirements tailoring. The level of automation claimed is related to the compliance reasoning required to compare processes and the normative documents and tool-chain information integration. Essentially, the surveyed methods require human intervention to implement the inputs of the reasoning process. Tool support is still an issue because most of the approaches are in the stage of conceptual modeling or have been materialized as proof-of-concept prototypes. In addition, few of the methods contemplate agile environments and standards evolution.

In the future, we will consider possible solutions for the challenges discovered in this SLR (see Section 5). First, new research efforts in automatic compliance checking, specifically for software processes, need to consider existing process modeling languages to accelerate the topic's results and standardize the techniques and tool support. In particular, it is essential to promote well-defined software process modeling languages, such as SPEM 2.0, to avoid repetition in creating process-related modeling resources. For this, we could perform comparative studies between existing process modeling languages and case studies showing their capabilities. Second, researchers need to find appropriate means for using logical approaches for the representation of normative frameworks. In that sense, we will continue investigating how to combine existing languages. The goal is to contribute with a well-defined (set of) logical structure(s) that harmoniously work in all the aspects required for software process-related compliance checking: reasoning capabilities, means for variability management, support for agile environments, and process execution conformance. However, we need to avoid the case of a new person feeling confused and frustrated when using formal methods. In particular, it could be interesting to develop short, straightforward expressions (i.e., syntactic sugar) that make it easier to read or to express normative frameworks, especially when the complexity (and size) of the compliance checking tasks grows. Third, we believe that existing studies could be combined to achieve a generic and normative-agnostic method. Fourth, it is also vital to increase the automation level by defining mechanisms that support the formalization of rules and reuse. It is also essential to concretize the tool support and increase the use of data derived from industrial-related software processes to evaluate the methods. Fifth, we also mentioned incorporating a trust layer to guarantee that rules are correctly derived from the normative frameworks. For addressing this issue, we could contact standardization/regulatory bodies to investigate the possibility of releasing process models and formal representations of the requirements within the release of new versions of the standards. With this strategy, we could reduce undesired room for interpretation of the normative texts. Moreover, formalizing requirements could help to improve the coherence of the normative frameworks. Finally, we could complement our research by implementing a Multivocal Literature Review (MLR), which is a form of a SLR that includes gray literature in addition to the published (formal) literature to provide insight into the “state of the practice.”[155]

ACKNOWLEDGMENTS

The authors of this paper are grateful to the anonymous reviewers and the editor for the thorough feedback on earlier versions of this paper. Their comments and suggestions were of great help. This work is partly supported by the EU and VINNOVA via the ECSEL JU Project AMASS (No. 692474).[156]

ORCID

Julieth Patricia Castellanos Ardila https://orcid.org/0000-0001-9970-7580
Barbara Gallina https://orcid.org/0000-0002-6952-1053

REFERENCES

1. Usman M, Felderer M, Unterkalmsteiner M, Klotins E, Mendez D, Alégroth E. Compliance requirements in large-scale software development: an industrial case study. In: International Conference on Product-Focused Software Process Improvement. Springer; 2020:385-401.
2. Castellanos Ardila JP, Gallina B, Governatori G. Compliance-aware engineering process plans: the case of space software engineering processes. *Artif Intell law*. 2021;29:587-627.
3. Rahim MM, Idowu SO. *Social Audit Regulation: Development, Challenges and Opportunities*, CSR, Sustainability, Ethics & Governance: Springer; 2015. ISBN 978-3-319-15838-9.
4. Leveson N. Safety: why, what, and how. *ACM Comput Surv (CSUR)*. 1986;18(2):125-163.
5. ISO/IEC/IEEE 12207—Systems and Software Engineering—Software Life Cycle Processes; 2017.
6. Kneuper R. *Software Processes and Life Cycle Models: An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes*: Springer; 2018. ISBN 978-3-319-98845-0.
7. ISO/IEC 15504—Information Technology—Process Assessment; 2012.
8. ISO/IEC 330XX—Information Technology—Process Assessment—Concepts and Terminology; 2015.
9. Biro M. Open services for software process compliance engineering. In: International Conference on Current Trends in Theory and Practice of Informatics. Springer; 2014:1-6.
10. Alexander IF. A taxonomy of stakeholders: human roles in system development. *Int J Technol Human Interaction (IJTHI)*. 2005;1(1):23-59.
11. Kerrigan S, Law KH. Logic-based regulation compliance-assistance. In: 9th International Conference on Artificial Intelligence and Law. ACM; 2003: 126-135.
12. Kitchenham B, Charters S. Guidelines for performing systematic literature reviews in software engineering. Technical report, Ver. 2.3 EBSE Technical Report, EBSE.
13. Kitchenham B, Brereton P. A systematic review of systematic review process research in software engineering. *Inform Softw Technol*. 2013;55(12): 2049-2075.
14. Gallina B, Ul Muram F, Castellanos Ardila JP. Compliance of agilized (software) development processes with safety standards: a vision. In: 4th International Workshop on Agile Development of Safety-Critical Software. ACM; 2018:1-6.
15. Song W, Jacobsen H-A, Zhang C, Ma X. Dependence-based data-aware process conformance checking. *IEEE Trans Serv Comput*. 2018;14(3): 654-667.
16. De La Vara JL, Ruiz A, Attwood K, et al. Model-based specification of safety compliance needs for critical systems: a holistic generic metamodel. *Inform Softw Technol*. 2016;72(C):16-30.
17. Diebold P, Scherr S. Software process models vs descriptions: what do practitioners use and need? *J Softw Evol Process*. 2017;29(11):1-13.
18. Vilkomir S, Bowen J, Ghose A. Formalization and assessment of regulatory requirements for safety-critical software. *Innov Syst Softw Eng*. 2006; 2(3-4):165-178.
19. Munoz-Gama J. Conformance checking and its challenges. *Conformance Checking and Diagnosis in Process Mining: Comparing Observed and Modeled Processes*: Springer; 2016:11-18.

20. Lúcio L, Rahman S, Cheng C-H, Mavin A. Just formal enough? automated analysis of ears requirements. In: NASA Formal Methods Symposium. Springer; 2017:427-434.
21. Brown D, Delseny H, Hayhurst K, Wiels V. Guidance for using formal methods in a certification context. *ERTS2 2010, Embedded Real Time Software & Systems*: HAL—Open Science; 2010.
22. Harju H, Lahtinen J, Ranta J, Nevalainen R, Johansson M. Software safety standards for the basis of certification in the nuclear domain. In: 7th International Conference on the Quality of Information and Communications Technology. IEEE Xplore; 2010:54-62.
23. Jääskinen N. Better regulation programs: some critical remarks. In: International Conference on Legislative Studies in Helsinki. National Research Institute of legal Research Communications; 2008:29-33.
24. Dinesh N, Joshi A, Lee I, Sokolsky O. Checking traces for regulatory conformance. In: International Workshop on Runtime Verification. Springer; 2008:86-103.
25. Governatori G, Sadiq S. The journey to business process compliance. *Handbook of Research on Business Process Modeling*: IGI Global; 2009:426-454.
26. Hashmi M, Governatori G, Lam H, Wynn M. Are we done with business process compliance: state of the art and challenges ahead. *Knowl Inform Syst*. 2018;57(1):79-133.
27. Casanovas P, González-Conejero J, de Koker L. Legal compliance by design (LCbD) and through design (LCtD): preliminary survey. In: 1st Workshop on Technologies for Regulatory Compliance (TERECOM). CEUR—Workshop Proceedings; 2017.
28. Cugola G, Ghezzi C. Software processes: a retrospective and a path to the future. *Softw Process: Improvement Pract*. 1998;4(3):101-123.
29. Fitzgerald B, Stol KJ, O'Sullivan R, O'Brien D. Scaling agile methods to regulated environments: an industry case study. In: 35th International Conference on Software Engineering (ICSE). IEEE Computer Society; 2013:863-872.
30. Kuhrmann M, Diebold P, Münch J, et al. Hybrid software and system development in practice: waterfall, scrum, and beyond. In: International Conference on Software and System Process. ACM; 2017:30-39.
31. Kuhrmann M, Diebold P, Munch J, et al. Hybrid software development approaches in practice: a European perspective. *IEEE Softw*. 2018;36(4):20-31.
32. Clarke LA, Osterweil LJ, Avrunin GS. Supporting human-intensive systems. In: FSE/SDP Workshop on Future of Software Engineering Research. ACM; 2010:87-92.
33. De La Vara JL, Marín B, Ayora C, Giachetti G. An empirical evaluation of the use of models to improve the understanding of safety compliance needs. *Inform Softw Technol*. 2020;126:106351.
34. Cha S, Taylor RN, Kang K. *Handbook of Software Engineering*. 1st ed.: Springer; 2019. ISBN 978-9-812-38971-8.
35. Osterweil LJ. Formalisms to support the definition of processes. *J Comput Sci Technol*. 2009;24(2):198-211.
36. Parnas DL, Clements PC. A rational design process: how and why to fake it. *IEEE Trans Softw Eng*. 1986;2:251-257.
37. Software Engineering Institute—Carnegie Mellon University. CMMI for Development Version 1.3—Capability Maturity Model Integration; 2011.
38. Biro M. Open services for software process compliance engineering. *SOFSEM 2014: Theory and Practice of Computer Science*: Springer International Publishing; 2014:1-6.
39. Khan AA, Keung J, Niazi M, Hussain S, Ahmad A. Systematic literature review and empirical investigation of barriers to process improvement in global software development: client-vendor perspective. *Inform Softw Technol*. 2017;87:180-205.
40. ISO 9000—Quality Management Systems—Fundamentals and Vocabulary; 2005.
41. ISO 9001-3—Quality Management and Quality Assurance Standards—Part 3; 1991.
42. ISO/IEC 90003:2004—Software Engineering—Guidelines for the Application of ISO 9001:2000 to Computer Software; 2004.
43. ISO/IEC TR 29110-5-1-2—Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs): Management and Engineering Guide: Generic Profile Group: Basic Profile; 2011.
44. Tonini AC, Mesquita Spinola MD, Barbin Laurindo FJ. Six Sigma and software development process: DMAIC improvements. In: PICMET 2006 Conference. IEEE Xplore; 2006:2815-2823.
45. Icheku V. *Understanding Ethics and Ethical Decision-Making*. 1st ed.: Xlibris Corporation; 2011. ISBN 978-1-4653-5131-9.
46. Ladkin PB. Duty of care and engineering functional-safety standards. *Digital Evidence Elec Sig L Rev*. 2019;16:51.
47. Regan G, Biro M, Mc Caffery F, Mc Daid K, Flood D. A traceability process assessment model for the medical device domain. In: European Conference on Software Process Improvement. Springer; 2014:206-216.
48. Generowicz M. *The Easy Path to Functional Safety Compliance*: I&E Systems Pty Ltda; 2013;1-3. https://www.iesystems.com.au/wp-content/uploads/2015/04/Duty-of-Care-Article.pdf, Accessed March 30, 2021.
49. Carroll N, Richardson I. Software-as-a-medical device: demystifying connected health regulations. *J Syst Inform Technol*. 2016;18(2):186-215.
50. International Electrotechnical Commission. IEC 61508—Functional safety of electric/electronic/programmable electronic safety-related systems; 1998.
51. Schwartz A. Statutory interpretation, capture, and tort law: the regulatory compliance defense. *Am Law Econ Rev*. 2000;2(1):1-57.
52. Cusumano MA. Who is liable for bugs and security flaws in software? *Commun ACM*. 2004;47(3):25-27.
53. Ingolfo S, Siena A, Mylopoulos J. Establishing regulatory compliance for software requirements. In: International Conference on Conceptual Modeling. Springer; 2011:47-61.
54. International Organization for Standardization—Technical Committee: ISO/TC 22/SC 32. ISO 26262: Road Vehicles Functional Safety; 2018.
55. CENELEC—EN 50128. Railway Applications—Communication, Signaling and Processing Systems Software for Railway Control and Protection Systems; 2011.
56. European Organisation for Civil Aviation Equipment & European Organisation for Civil Aviation Equipment. RTCA/DO-178C—Software Considerations in Airborne Systems and Equipment Certification; 2011.
57. Internation Organization for Standardization - Technical Committee 210. IEC 62304—Medical device software—Software life cycle processes; 2006.
58. ECSS-E-ST-40C—Space Engineering Software. https://ecss.nl/standard/ecss-e-st-40c-software-general-requirements/; 2009.
59. ISO 14971:2019—Application of risk management to medical devices; 2019.
60. ISO/IEC 27000—Information Technology; 2018.
61. General Data Protection Regulation (GDPR); 2016.
62. EU DPD—European Data Protection Directive; 1995.

63. PIPEDA—Personal Information Protection and Electronic Documents Act; 2000.
64. Bauer K, Hinz O, van der Aalst W, Weinhardt C. Expl(AI)n it to me—explainable AI and information systems research. *Business Inform Syst Eng*. 2021; 63:79-82.
65. Vakkuri V, Jantunen M, Halme E, et al. Time for AI (ethics) maturity model is now. arXiv preprint arXiv:210112701; 2021.
66. Ramasubbu N, Bharadwaj A, Tayi GK. Software process diversity: conceptualization, measurement, and analysis of impact on project performance. *MIS Quart*. 2015;39(4):787-808. https://www.jstor.org/stable/26628652
67. Cooper HM. Organizing knowledge syntheses: a taxonomy of literature reviews. *Knowl Soc*. 1988;1(1):104-126.
68. Denyer D, Tranfield D, Van Aken JE. Developing design propositions through research synthesis. *Org Studies*. 2008;29(3):393-413.
69. Zhang H, Ali-Babar M, Tell P. Identifying relevant studies in software engineering. *Inform Softw Technol*. 2011;53(6):625-637.
70. Wohlin C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: 18th International Conference on Evaluation and Assessment in Software Engineering. ACM; 2014:1-10.
71. Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Linkman S. Systematic literature reviews in software engineering—a systematic literature review. *Inform Softw Technol*. 2009;51(1):7-15.
72. Welzel D, Walter H, Schmidt W. Tailoring and conformance testing of software processes: the ProcePT approach. In: Software Engineering Standards Symposium. IEEE; 1995:41-49.
73. Emmerich W, Finkelstein A, Montangero C, Antonelli S, Armitage S, Stevens R. Managing standards compliance. *IEEE Trans Softw Eng*. 1999;25(6): 836-851.
74. Cheung L, Chung P, Dawson R. Managing process compliance. *Information Management: Support Systems & Multimedia Technology*: IGI Global; 2003: 48-62.
75. Chung P, Cheung L, Machin C. Compliance flow: managing the compliance of dynamic and Complex processes. *Knowl-Based Syst*. 2008;21(4): 332-354.
76. He X, Guo J, Wang Y, Guo Y. An automatic compliance checking approach for software processes. In: Asia-Pacific Software Engineering Conference. IEEE; 2009:467-474.
77. Jost H, Hahn A, Häusler S, et al. Supporting qualification: safety standard compliant process planning and monitoring. In: Symposium on Product Compliance Engineering. IEEE Xplore; 2010:1-6.
78. Rodriguez D, Garcia E, Sanchez S, Nuzzi CR-S. Defining software process model constraints with rules using OWL and SWRL. *Int J Softw Eng Knowl Eng*. 2010;20(4):533-548.
79. Panesar-Walawege R, Sabetzadeh M, Briand L. A model-driven engineering approach to support the verification of compliance to safety standards. In: International Symposium on Software Reliability Engineering. IEEE Xplore; 2011:30-39.
80. Maccaull W, Rabbi F. NOVA Workflow: a workflow management tool targeting health services delivery. In: International Symposium on Foundations of Health Informatics Engineering and Systems. Springer; 2012:75-92.
81. Hassan W, Logrippo L. Towards a process for legally compliant software. In: 6th International Workshop on Requirements Engineering and Law (RELAW). IEEE; 2013:44-52.
82. Panesar-Walawege R, Sabetzadeh M, Briand L. Supporting the verification of compliance to safety standards via model-driven engineering: approach, tool-support and empirical validation. *Inform Softw Technol*. 2013;55(5):836-864. https://doi.org/10.1016/j.infsof.2012.11.009
83. Khelladi D-E, Bendraou R, Baarir S, Laurent Y, Gervais M-P. A framework to formally verify conformance of a coftware process to a software method. In: 30th Annual ACM Symposium on Applied Computing. ACM; 2015:1518-1525.
84. Hewett R, Kijsanayothin P, Bak S, Galbrei M. Cybersecurity policy verification with declarative programming. *Appl Intelligence*. 2016;45(1):83-95.
85. Kabaale E, Wen L, Wang Z, Rout T. Representing software process in Description Logics: an ontology approach for software process reasoning and verification. In: Software Process Improvement and Capability Determination Conference. Springer; 2016:362-376.
86. Arcaini P, Bonfanti S, Gargantini A, Riccobene E. How to assure correctness and safety of medical software: the hemodialysis machine case study. In: International Conference on Abstract State Machines. Springer; 2016:344-359.
87. Bala S, Cabanillas C, Haselböck A, et al. A framework for safety-critical process management in engineering projects. In: International Symposium on Data-Driven Process Discovery and Analysis. Springer; 2017:1-27.
88. Valle AM, Santos EAP, Loures ER. Applying process mining techniques in software process appraisals. *Inform Softw Technol*. 2017;87:19-31.
89. Golra FR, Dagnat F, Bendraou R, Beugnard A. Continuous process compliance using model-driven engineering. In: International Conference on Model and Data Engineering. Springer; 2017:42-56.
90. Castellanos Ardila JP, Gallina B. Towards increased efficiency and confidence in process compliance. In: The 24th European & Asian Systems, Software & Service Process Improvement & Innovation (EuroAsiaSPI) Conference. Springer; 2017.
91. Ranise S, Siswantoro H. Automated legal compliance checking by security policy analysis. In: International Conference on Computer Safety, Reliability, and Security. ACM; 2017:361-372.
92. Proença D, Borbinha J. A formalization of the ISO/IEC 15504: enabling automatic inference of capability levels. In: International Conference on Software Process Improvement and Capability Determination. Springer; 2017:197-210.
93. Guarda P, Ranise S. Security analysis and legal compliance checking for the design of privacy-friendly information systems. In: Symposium on Access Control Models and Technologies. ACM; 2017:247-254.
94. Castellanos Ardila JP, Gallina B. Towards efficiently checking compliance against automotive security and safety standards. In: 7th IEEE International Workshop on Software Certification (WoSoCer). IEEE Xplore; 2017.
95. Kabaale E, Wen L, Wang Z, Rout T. An axiom-based metamodel for software process formalisation: an ontology approach. In: International Conference on Software Process Improvement and Capability Determination (SPICE). Springer; 2017:226-240.
96. Castellanos Ardila JP, Gallina B, Ul Muram F. Enabling compliance checking against safety standards from SPEM 2.0 process models. In: Euromicro Conference on Software Engineering and Advanced Applications. IEEE; 2018:45-49.
97. Kabaale E, Wen L, Wang Z, Rout T. Ensuring conformance to process standards through formal verification. In: International Conference on Software Process Improvement and Capability Determination (SPICE). Springer; 2018:248-262.
98. Arcaini P, Bonfanti S, Gargantini A, Mashkoor A. Integrating formal methods into medical software development: the ASM approach. *Sci Comput Programm*. 2018;158:148-167.

99. Castellanos Ardila JP, Gallina B, Ul Muram F. Transforming SPEM 2.0-compatible process models into models checkable for compliance. In: International Conference on Software Process Improvement and Capability Determination (SPICE). Springer; 2018.
100. Proença D, Borbinha J. Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3—enabling automatic inference of maturity and capability levels. *Comput Standards Interfaces*. 2018;60:13-25.
101. Bonatti P. Fast compliance checking in an OWL2 fragment. In: 27th International Joint Conferences on Artificial Intelligence Organization. IJCAI; 2018:1746-1752.
102. Bombarda A, Bonfanti S, Gargantini A. Developing medical devices from Abstract State Machines to embedded systems: a smart pill box case study. In: International Conference on Objects, Components, Models and Patterns. Springer; 2019:89-103.
103. Castellanos Ardila JP, Gallina B, Ul Muram F. Facilitating automated compliance checking of processes in the safety-critical context. *Electronic Commun EASST*. 2019;078:1-20.
104. Kabaale E, Wen L, Wang Z, Rout T. Formalising process assessment and capability determination: an ontology approach. In: European Conference on Software Process Improvement (SPICE). Springer; 2019:594-605.
105. Torre D, Soltana G, Sabetzadeh M, Briand L, Auffinger Y, Goes P. Using models to enable compliance checking against the GDPR: an experience report. In: 22nd International Conference on Model Driven Engineering Languages and Systems. IEEE Xplore; 2019:1-11.
106. Daoudagh S, Marchetti E. A life cycle for authorization systems development in the GDPR perspective. In: ITASEC. Open Portal; 2020:128-140.
107. Bramberger R, Martin H, Gallina B, Schmittner C. Co-engineering of safety and security life cycles for the engineering of automotive systems. *ACM SIGAda Ada Lett*. 2020;39(2):41-48.
108. Castellanos Ardila JP, Gallina B. Separation of concerns in process compliance checking: divide-and-conquer. In: European Conference on Software Process Improvement (EuroAsiaSPI). Springer; 2020:135-147.
109. Castellanos Ardila JP, Gallina B. Reusing (safety-oriented) compliance artifacts while recertifying. In: 9th International Conference on Model-Driven Engineering and Software Development—Volume 1: MODELSWARD. INSTICC. SciTePress; 2021:53-64.
110. Mayr-Dorn C, Vierhauser M, Bichler S, et al. Supporting quality assurance with automated process-centric quality constraints checking. In: 43rd International Conference on Software Engineering (ICSE). IEEE/ACM; 2021.
111. Schwaber K, Beedle M. *Agile Software Development with Scrum*. 1st ed.: Prentice Hall; 2002. ISBN 978-0-1320-7489-6.
112. ISO/IEC 24744—Software Engineering—Metamodel for Development Methodologies; 2007.
113. Beck K. *Extreme Programming Explained: Embrace Change*, XP Series: Addison-Wesley Professional; 2000. ISBN 201-61641-6.
114. Kniberg H, Skarin M. *Kanban and Scrum-Making the Most of Both*. 1: InfoQ; 2010. ISBN 978-0-557-13832-6.
115. Software & Systems Process Engineering Meta-Model Specification. V.2.0 (SPEM 2.0). https://www.omg.org/spec/SPEM/2.0/; 2008.
116. Business Process Model and Notation. http://www.bpmn.org/; 1997.
117. Antoniou G, Billington D, Governatori G, Maher MJ. Representation results for defeasible logic. *ACM Trans Comput Logic*. 2000;2(2):255-287.
118. Governatori G. Representing business contracts in RuleML. *Int J Cooperative Inform Syst*. 2005;14(2n03):181-216.
119. General Directive 250: Software Development Standard for the German Federal Armed Forces, V-model, Software Lifecycle Process Model; 1992.
120. ESA PSS-05-0 Software Engineering Standards - Issue 2; 1991.
121. Lifschitz V. What is Answer Set Programming. In: Twenty-Third AAAI Conference on Artificial Intelligence. AAI.org; 2008:1594-1597.
122. Directive 95/46/EC of the European Parliament and of the Council; 1995.
123. GDPR—General Data Protection Regulation; 2016.
124. Osterweil L. Software processes are software too. In: 9th International Conference on Software Engineering (ICSE 1987). IEEE; 1987.
125. Börger E, Stark R. *Abstract State Machines: A Method for High-Level System Design and Analysis*. 1st ed.: Springer-Verlag; 2003. 978-3-642-18216-7.
126. Wen L, Tuffley D, Rout T. Using Composition Trees to model and compare software processes. In: European Conference on Software Process Improvement (SPICE). 2011. CCIS. Springer; 2011:1-15.
127. Mashiyat A, Rabbi F, Maccaull W. Modeling and verifying timed compensable workflows and an application to health care. In: International Workshop on Formal Methods for Industrial Critical Systems. Springer; 2011:244-259.
128. FUML—Semantics of a Foundational Subset for Executable UML Models. https://www.omg.org/spec/FUML/1.5/About-FUML/; 2021.
129. Web Ontology Language (OWL). https://www.w3.org/OWL/; 2012.
130. Barrett C, Tinelli C. Satisfiability Modulo Theories. *Handbook Model Checking*: Springer; 2018:305-335.
131. Unified Modeling Language Specification Version 2.5.1. https://www.omg.org/spec/UML/; 2017.
132. OASIS eXtensible Access Control Markup Language (XACML) TC. https://www.oasis-open.org/committees/tc_home.php?w_abbrev=xacml; 2003.
133. Pnueli A. The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science. IEEE; 1977:46-57.
134. Object Constraint Language. Version 2.4. https://www.omg.org/spec/OCL/2.4/PDF; 2014.
135. SWRL: a Semantic Web Rule Language combining OWL and RuleML. https://www.w3.org/Submission/SWR; 2004.
136. SAE J3061—Cybersecurity Guidebook for Cyber-Physical Vehicle Systems; 2016.
137. ISO 14971:2000—Application of risk management to medical devices; 2000.
138. Automotive SPICE V.3.0—Process Assessment/Reference Model; 2015.
139. CENELEC-EN 50126. Railway Applications—The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS); 2017.
140. Siena A, Mylopoulos J, Perini A, Susi A. From laws to requirements. *Requirements Engineering and Law (RELAW)*: IEEE Xplorer; 2008:6-10.
141. Carlan C, Gallina B, Soima L. Safety case maintenance: a systematic literature review. In: 40th International Conference on Computer Safety, Reliability and Security (SAFECOMP). Springer; 2021.
142. Ul Muram F, Tran H, Zdun U. Systematic review of software behavioral model consistency checking. *ACM Comput Surv (CSUR)*. 2017;50(2):1-39.
143. Castellanos Ardila JP, Gallina B, Muram FU. Systematic literature review of compliance checking approaches for software processes; 2021. Technical report.
144. Castellanos Ardila JP. A safety-centered planning-time framework for automated process compliance checking. *Ph.D. Thesis*: Mälardalen University; 2021. http://www.es.mdh.se/publications/6320-
145. Becker J, Delfmann P, Eggert M, Schwittay S. Generalizability and applicability of model-based business process compliance-checking approaches: a state-of-the-art analysis and research roadmap. *Business Res*. 2012;5(2):221-247.

146. Ly L, Maggi F, Montali M, Rinderle-Ma S, van der Aalst WM. Compliance monitoring in business processes: functionalities, application, and tool-support. *Inform Syst*. 2015;54:209-234.
147. Hashmi M, Governatori G. A methodological evaluation of business process compliance management frameworks. *Asia Pacific Business Process Management (AP-BPM)*, Lecture Notes in Business Information Processing, vol. 159: Springer; 2013:106-115.
148. Kharbili M, Medeiros AKA, Stein S, van der Aalst WM. Business process compliance checking: current state and future challenges. In: Modellierung betrieblicher Informationssysteme (MobIS). Gesellschaft für Informatik eV; 2008:107-113.
149. Boella G, Humphreys L, Muthuri R, Rossi P, van der Torre L. A critical analysis of legal requirements engineering from the perspective of legal practice. In: 7th International Workshop on Requirements Engineering and Law. IEEE Xplore; 2014:14-21.
150. Akhigbe O, Amyot D, Richards G. A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory compliance. *Requirements Eng*. 2019;24(4):459-481. https://doi.org/10.1007/s00766-018-0294-1
151. von Wangenheim CG, Hauck JCR, Salviano CF, von Wangenheim A. Systematic literature review of software process capability/maturity models. In: International Conference on Software Process Improvement and Capability Determination (SPICE). Springer; 2010.
152. Yan M, Xia X, Zhang X, Xu L, Yang D, Li S. Software quality assessment model: a systematic mapping study. *Sci China Inform Sci*. 2019;62(9):1-18.
153. García-Borgoñon L, Barcelona MA, García-García JA, Alba M, Escalona MJ. Software process modeling languages: a systematic literature review. *Inform Softw Technol*. 2014;56(2):103-116.
154. Nair S, De La Vara J, Sabetzadeh M, Briand L. An extended systematic literature review on provision of evidence for safety certification. *Inform Softw Technol*. 2014;56(7):689-717.
155. Garousi V, Felderer M, Mäntylä MV. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In: 20th International Conference on Evaluation and Assessment in Software Engineering. ACM; 2016:1-6.
156. Architecture-Driven, Multi-Concern and Seamless Assurance and Certification of Cyber-Physical Systems—AMASS. http://www.amass-ecsel.eu/
157. Schmidt W. *Prädikative spezifikation und analyse des vorgehensmodells*: Forschungszentrum Informationstechnik; 1996. http://publica.fraunhofer.de/documents/n-42675.html. ISBN: 3-88457-283-0.
158. Vanhatalo J, Völzer H, Koehler J. The refined process structure tree. *Data Knowl Eng*. 2009;68(9):793-818.
159. Rabbi F, Wang H, MacCaull W. Compensable workflow nets. In: International Conference on Formal Engineering Methods. Springer; 2010:122-137.

How to cite this article: Castellanos Ardila JP, Gallina B, Ul Muram F. Compliance checking of software processes: A systematic literature review. *J Softw Evol Proc*. 2022;e2440. doi:10.1002/smr.2440

APPENDIX A: LIST OF SELECTED STUDIES

TABLE A1 Selected primary studies using SLR

	Quality assessment criteria							
ID	QE1	QE2	QE3	QE4	QE5	QE6	QE7	Score
S1[72]	1	0	1	1	1	1	1	6
S2[73]	1	0	1	1	1	1	1	6
S3[74]	1	1	1	1	1	1	1	7
S4[75]	1	1	1	1	1	1	1	7
S5[76]	1	1	1	1	1	1	1	7
S6[77]	1	0	1	1	1	1	1	6
S7[78]	1	0	1	1	1	1	1	6
S8[79]	1	0	1	1	1	1	1	6
S9[80]	1	0	1	1	1	1	1	6
S10[81]	1	0	1	1	1	1	1	6
S11[82]	1	1	1	1	1	1	1	7
S12[83]	1	1	1	1	1	1	1	7
S13[84]	1	0	1	1	1	1	1	6
S14[85]	1	0	1	1	1	1	1	6
S15[86]	1	0	1	1	1	1	1	5
S16[87]	1	0	1	1	1	1	1	5
S17[88]	1	1	1	1	1	1	1	7
S18[89]	1	0	1	1	1	1	1	6
S19[90]	1	1	1	1	1	1	1	7
S20[91]	1	0	1	1	1	1	1	6
S21[92]	1	0	1	1	1	1	1	6
S22[93]	1	1	1	1	1	1	1	7
S23[94]	1	0	1	1	1	1	1	6
S24[95]	1	0	1	1	1	1	1	6
S25[96]	1	0	1	1	1	1	1	6
S26[97]	1	0	1	1	1	1	1	6
S27[98]	1	0	1	1	1	1	1	6
S28[99]	1	1	1	1	1	1	1	7
S29[14]	1	0	1	1	1	1	1	6
S30[100]	1	0	1	1	1	1	1	6
S31[101]	1	1	1	0.5	1	1	1	6.5
S32[102]	1	1	1	1	1	1	1	6
S33[103]	1	0	1	1	0.5	1	1	5.5
S34[104]	1	0	1	1	1	1	1	6
S35[105]	1	1	1	1	1	1	1	7
S36[106]	1	0	1	0.5	1	1	1	5.5
S37[107]	1	1	1	1	1	1	1	7
S38[108]	1	1	1	0.5	1	1	1	5.5
S39[109]	1	1	1	1	1	1	1	7
S40[2]	1	1	1	1	1	1	1	7
S41[110]	1	1	1	1	1	1	1	7

APPENDIX B: SUMMARY OF THE REVIEWED STUDIES

TABLE B1 Summary of the reviewed studies

ID	Process representation	Tasks	Work products	Roles	Guidance	Tools	Workflow	Requirements representation	Level of automation	Evolution handling?	Illustrative scenarios	Industrial settings?	Standards targeted	Support agile?	Application domains
S1	ProcePT[157]	✓	✓					FOL (Prolog)	PC	✓	Software development		ISO 9001[41]		Quality
S2	UML		✓					FOL	PC		Software development		PSS-05[120]		Quality
S3	UML	✓	✓	✓	✓	✓	✓	UML	CM		Recommendation handling	✓	IEC 61508[50]		Safety critical
S4	UML	✓	✓	✓	✓	✓	✓	UML	PC		Programmable electronic systems	✓	IEC 61508		Safety critical
S5	UML	✓	✓					PPST[158]	PC		Software development		ISO/ IEC90003		Quality
S6	SPEM 2.0	✓						XSLT	PC	✓	Automotive system design		ISO 26262		Safety critical
S7	SPEM 2.0	✓	✓	✓				SWRL	PC		Software development		Software Process Guidelines	✓	Process verification
S8	UML	✓	✓	✓	✓			OCL	PC		Subsea control	✓	IEC 61508.		Safety critical
S9	CWMLT	✓					✓	LTL	PC	✓	Services delivery	✓	Internal Guidelines.		Health care
S10	UML	✓	✓					GAL	IT		Information privacy	✓	PIPEDA.		Data protection
S11	UML	✓	✓	✓	✓			OCL	PC		Subsea control	✓	IEC 61508		Safety critical
S12	fUML	✓	✓	✓			✓	LTL	IT		Software development		Software Process Guidelines	✓	Process verification
S13	ASP	✓		✓			✓	ASL	CM		Human resources	✓	RBAC policy		Cybersecurity
S14	CT	✓	✓					DL	PC		Human resources	✓	ISO/IEC TS 33053		Process verification
S15	ASM	✓					✓	LTL	PC		Medical devices	✓	IEC 62304		Safety critical
S16	BPMN	✓	✓					SHACL	PC		Railway	✓	EN 50126		Safety critical
S17	Petri net	✓					✓	LTL	IT		Information technology		CMMI-DEV v1.3		Process verification

(Continues)

TABLE B1 (Continued)

ID	Process representation	Tasks	Work products	Roles	Guidance	Tools	Workflow	Requirements representation	Level of automation	Evolution handling?	Illustrative scenarios	Industrial settings?	Standards targeted	Support agile?	Application domains
S18	UML	✓	✓			✓		Mapping	CM		Space		ECSS-ST-40C		Safety critical
S19	SPEM 2.0	✓	✓	✓	✓	✓	✓	Def-L	CM	✓	Automotive		ISO 26262 ASPICE		Safety critical
S20	SMT		✓					FOL	IT	✓	Human resources	✓	EU DPD		Data protection
S21	OWL	✓	✓					SWRL	CM		Process assessments	✓	ISO/IEC 15504		SPI
S22	MSC		✓					FOL	PC		Human resources	✓	EU DPD		Data protection
S23	SPEM 2.0	✓	✓	✓	✓	✓	✓	Def-L	CM	✓	Automotive		ISO 26262 SAE J3061.		Safety critical
S24	OWL	✓	✓					DL	PC	✓	Software development		ISO/IEC 29110		Process verification
S25	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL	PC	✓	Automotive		ISO 26262		Safety critical
S26	OWL	✓	✓					DL	PC		Software requirements analysis		ISO/IEC 15504		SPI
S27	ASM	✓					✓	LTL	PC		Medical devices	✓	IEC 62304 FDA general principles		Safety critical
S28	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL	PC	✓	Railway		CENELEC EN 50128		Safety critical
S29	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL	PC		Agilized environments		Safety Standards	✓	Safety critical
S30	OWL	✓	✓					SWRL	CM		Companies appraisals results	✓	ISO/ IEC15504 CMMI v1.3		SPI
S31	OWL			✓				DL	CM		Medical device	✓	GDPR		Data protection
S32	ASM	✓					✓	CTL	PC		Medical device	✓	IEC 62304 FDA general principles		Safety critical

TABLE B1 (Continued)

ID	Process representation	Tasks	Work products	Roles	Guidance	Tools	Workflow	Requirements representation	Level of automation	Evolution handling?	Illustrative scenarios	Industrial settings?	Standards targeted	Support agile?	Application domains
S33	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL	PC	✓	Automotive		ISO 26262		Safety critical
S34	OWL	✓	✓					DL	PC		Software development		ISO/IEC 15504		SPI
S35	UML			✓				OCL	PC	✓	Information technology		GDPR		Data protection
S36	XACML			✓				XACML	CM		Authorization systems		GDPR	✓	Data protection
S37	SPEM 2.0	✓	✓	✓	✓	✓	✓	BCL	PC	✓	Automotive		ISO 26262 SAE J3061		Safety critical
S38	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL	PC	✓	Railway		CENELEC EN 50128.		Safety critical
S39	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL, BCL	PC	✓	Medical devices		ISO 14971		Safety critical
S40	SPEM 2.0	✓	✓	✓	✓	✓	✓	FCL	PC	✓	Space		ECSS-E-ST-40C		Safety critical
S41	UML	✓	✓				✓	Declarative	PC	✓	Avionics	✓	DO-178C	✓	Safety critical

Note: ✓, supported. Reference to the languages used in the studies is presented in Appendix C.
Abbreviations: CM, conceptual model, IT, implemented tool; PC, proof-of-concept prototype.

APPENDIX C: LANGUAGES USED IN THE STUDIES

TABLE C1 Languages used in the studies

Abbreviation	Name
UML	Unified Modeling Language[a]
OWL	Web Ontology Language[b]
SPEM 2.0	Software & Systems Process Engineering Metamodel[c]
CWMLT	Compensable Workflow Modeling Language[159]
fUML	Foundational Subset for Executable UML Models[d]
ASM	Abstract State Machines125, [125]
ASP & (ASL)	Answer Set Programming & Answer Set Logic[121]
BPMN	Business Process Management Notation[e]
SMT	Satisfiability Modulo Theories[130]
XACML	eXtensible Access Control Markup Language[f]
FOL	First Order Logic
PPST	Process Pattern Structure Tree[158]
CT	Composition Trees
XSLT	Extensible Stylesheet Language Transformations[g]
SWRL	Semantic Web Rule Language[135]
OCL	Object Constraint Language[h]
LTL	Linear Temporal Logic[133]
GAL	Governance Analysis Language[81]
DL	Description Logic[i]
SHACL	Shapes Constraint Language[j]
Def-L	Defeasible Logic[k]
FCL	Formal Contract Logic[118]
CTL	Computational Tree Logic
BCL	Basic Constraint Language

[a]https://www.omg.org/spec/UML/.
[b]https://www.w3.org/OWL/.
[c]https://www.omg.org/spec/SPEM.
[d]https://www.omg.org/spec/FUML.
[e]https://www.bpmn.org/.
[f]https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
[g]https://www.w3.org/TR/2017/REC-xslt-30-20170608/.
[h]https://www.omg.org/spec/OCL/.
[i]https://dl.kr.org/.
[j]https://www.w3.org/TR/shacl/.
[k]https://www.defeasible.org/.