# Systematic Literature Review of Compliance Checking Approaches for Software Processes

Julieth Patricia Castellanos Ardila, Barbara Gallina, and Faiz Ul Muram

IDT, Mälardalen University, Västerås, Sweden
{julieth.castellanos, barbara.gallina, faiz.ul.muram}@mdh.se

**Abstract. Context:** Software processes have increased demands coming from normative requirements. Organizations developing software comply with such demands to be in line with the market and the law. The state-of-the-art provides means to automatically check whether a software process complies with a set of normative requirements. However, no comprehensive and systematic review has been conducted to characterize such works. **Objective:** We characterize the current research on this topic, including an account of the used techniques, their potential impacts, and challenges. **Method:** We undertake a Systematic Literature Review (SLR) of primary studies reporting techniques for automated compliance checking of software processes. **Results:** We identify *41* papers reporting solutions focused on limited normative frameworks. Such solutions use specific languages for the processes and normative representation. Thus, the artifacts represented vary from one solution to the other. The level of automation, which in most methods requires tool-support concretization, focuses mostly on the reasoning process and requires human intervention, e.g., for creating the inputs for such reasoning. In addition, only a few contemplate agile environments and standards evolution. **Conclusions:** Our findings outline compelling areas for future research. In particular, there is a need to consolidate existing languages for process and normative representation, compile efforts in a generic and normative-agnostic solution, increase automation and tool support, and incorporate a layer of trust to guarantee that rules are correctly derived from the normative requirements.

**Keywords:** Automated compliance checking · software processes · normative frameworks · Systematic Literature Review.

## 1 Introduction

Many applications and infrastructures rely on software, including the internet, warning systems, and medical and financial information systems [59]. Due to its growing use, the software is becoming a public good, and its quality is a concern for society [133]. In particular, there is a group of stakeholders, called community stakeholders [144], including governments, regulatory bodies, and companies or individuals, who make a strong influence on normative compliance.

Governments and regulatory bodies demand compliance with standards and policies for licensing and certification purposes. Companies acting as customers in a production chain commonly demand compliance with specific regulations from their suppliers to have a standardized and transparent production [32]. There are also knowledgeable individuals demanding the use of standards to influence responsible behavior among industry practices [126]. Thus, compliance with normative frameworks is a must-do for software development organizations, especially when software is developed for safety-critical systems[1].

The software engineering community has observed that standardized software processes make development tasks more predictable, transparent, and economical [41, 139, 100]. Standardized software processes are referenced in international standards, e.g., ISO/IEC 12207 [84] for software, and ISO/IEC 15504 [80] series of standards - and its evolution ISO/IEC 330xx series [82] for assessment and improvement processes. It is also common to find standards and regulations in the safety-critical context that follow a prescriptive approach, i.e., they mandate a rigorous process for software development [92]. However, such standards mean stringent compliance requirements beyond the commitment to improve process capability [14]. In general, requirements regulating software aim at covering a broad set of organization and use cases [144]. For this reason, they act as process constraints and generally omit implementation-specific details [4].

Standards commonly provide information regarding the process elements required during software development and the mandated features. When seeking compliance, process engineers use this information to include the sequence of tasks mandated (i.e., the process behavior) and the resources ascribed to such tasks, e.g., personnel, work products, tools, and methods, which are also framed with essential properties (i.e., the process structure). Such work can be seen as systematic, i.e, methodical in procedure or plan[2]. 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 has the potential to support process engineers in their compliance checking tasks [93]. 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 this task. However, to the best of our knowledge, no comprehensive and systematic review has been conducted to characterize them.

In this paper, we undertake an SLR (Systematic Literature Review) of primary studies reporting techniques for automated compliance checking of software processes. An SLR, according to Kitchenham and others [97, 99], is a secondary study used to identify, analyze, and interpret all available evidence related to a specific topic. Briefly, the purposes of this SLR are as follows: 1) provide an overview regarding the evolution of the research regarding automated compli-

[1] Safety-critical systems are those whose failure could lead to unacceptable consequences, e.g., death, injury, loss of property, or environmental harm [105]

[2] https://www.merriam-webster.com/dictionary/systematic

ance checking of software processes; 2) provide an account of the current techniques; 3) describe their potential impacts and challenges; and, 4) outline key areas where future research can advance to support companies moving towards automated compliance checking practices.

As a result, we identify *41* primary studies from a list of *2033* found in recognized online libraries. The selected primary studies provide a set of ad hoc solutions that are interesting, applicable, and valuable contributions to the topic. However, such solutions use specific languages for the processes and normative representation. Thus, the artifacts represented vary from one solution to the other. Most of the languages used for representing requirements primarily define obligations (the mandatory requirements) but leave aside other considerations, such as the permitted actions that could indirectly affect compliance, e.g., exceptional cases surrogated by requirements tailoring. 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. Tool support still needs concretization since most of the approaches are in the stage of conceptual modeling or have been materialized as proof-of-concept prototypes. In addition, only a few methods contemplate agile environments and standards evolution.

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 consolidate existing languages for process and normative representation since there are already too many options not being adequately exploited. In our opinion, it is also crucial to consolidate a generic and normative-agnostic solution that can handle the different concepts, structures, and scenarios provided in the standards. Such a solution could be more attractive to organizations. It is also crucial to increase automation for easing the creation of rules, i.e., rule editors, since formalizing requirements still need human intervention. It is also essential to provide concrete and stable tools that can support the compliance checking process. Finally, a layer of trust should be incorporated in the methods for compliance checking to guarantee that rules are correctly derived from the normative frameworks.

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

Software process compliance aims to ensure the fidelity of the processes used to engineer software products to a selected normative framework, usually in the form of an industry standard [127]. For this, organizations show either full adherence, by complying with all requirements set out by the applicable standard or perform requirements tailoring. Tailoring requires selecting applicable requirements, performing their eventual modifications, and explaining their implementation according to the project's particular circumstances. A tailoring process should also ensure consistency to the defined normative framework, which determines allowed actions and the resulting conditions [100]. Traceability is also a mandated requirement [129]. Normative frameworks commonly prescribe requirements that include the tasks to be performed and resources ascribed to such tasks, i.e., personnel, work products, tools, and methods, which are also framed with essential properties [30]. Given these features, compliance management can be supported by checking that the process used to engineer software systems fulfill the properties set down by standards at given points.

Compliance checking requires at least two sources of information [45]. One is the normative document to be complied with, and the other is the process for which compliance is desired. Automatizing this task requires that these specifications are computer-based analyzable. Formal methods, which are a set of domain theorems that are amenable to formal proving through reasoning, are of growing interest for compliance checking [47, 148]. Using formal methods provides rigorous methodologies that increase 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 [114]. Moreover, the translation of requirements written in natural language is complex. For this reason, precise notions are required [109]. Moreover, to be formal in a certification context, a model must have an unambiguous, mathematically defined syntax and semantics [21].

A formal language should be expressive enough to cover the properties described by the models under consideration. Commonly, there are two critical features expressed in the normative frameworks. First, software process reference models prescribed by the standards are a "shall" type collections of requirements, i.e., compliance requires the satisfaction of all requirements, and precise documentation documented along with the reasoning behind the requirements [64]. 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 [85]. 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 [48]. Thus, compliance is a relationship between permissions (optional) and obligations (required).

### 2.2 Software Processes

Software developers perform processes, which are often defined to various levels of detail [118]. According to Parnas et al. [121], the most advantageous form of a process description will be in terms of work products workflow. Lonchamp [108] highlighted the importance of organizational structures. Fuggetta [56] concretize the definition by including the involvement of constraints governing the conception, development, deployment, and maintenance. Software processes have also been considered analogous to other kinds of processes [132, 69]. However, the software process definition goes further since the software has a unique characteristic, i.e., it is a pure information product that requires high abstraction levels [117]. According to Armour [8], the authentic product of the software development is the knowledge contained in the software. Thus, the software process's primary goal is to solve an application data processing problem [117] by performing a knowledge acquisition activity [8].

Explicit descriptions of the software processes servers development to proceed in a systematic way [41], increases predictability and transparency [100]. Software process descriptions are also commonly used to convince third parties, such as customers or regulatory bodies, regarding the quality of the software [56, 100]. However, different development methodologies tackled the necessity of software processes in different ways. In particular, agile methodologies prioritize individuals and interactions over processes and tools[3]. Moreover, agile follows an empirical logic. In regulated environments, a defined logic is more desirable. Thus, agile is faced with some fundamental challenges in regulated environments [55]. In contrast, plan-driven methodologies build mainly on the codification strategy and the definition of appropriate steps in advance, making it more suitable for regulated environments. Given the successful application of agile in software projects[4] and the suitability of plan-driven methodologies in regulated environments, hybrids between them are also conceived [101, 102], e.g., the Scaled Agile Framework (SAFe)[5].

Software process models help organizations preserve, repeat, analyze and reuse process information [36]. Models also can improve the understanding of compliance needs [44]. A software process model is an abstraction whose goal is to approximate the full range of characteristics and properties of an actual software process [33]. For this, a process model should [118]: 1) be described with rigorous notations; 2) be detailed enough; 3) be semantically broad; and 4) be clear and understandable to facilitate communication. 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.

The concept of the software process model is analogous to the concept of the life cycle (or lifecycle) model [100]: software life cycle models define the main steps and their sequence, while software process models provide more detailed in-

[3] https://agilemanifesto.org/
[4] See for instance: https://stateofagile.com/
[5] https://www.scaledagile.com/

structions, breaking the main steps down into sub-steps, and adding information about the results generated and the roles involved.

A recent survey conducted by Diebold and Scherr [47] shows the most expected characteristics of the software process models in industrial settings. In particular, it is expected to have concepts that permit the creation of a detailed description of the software process elements, i.e., 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. The possibility of having different views on the software process is relevant, i.e., 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 since they can help engineers configure models according to context (or project)-specific needs.

### 2.3 Software Process-related Normative Frameworks

Normative frameworks addressing software processes prescribe requirements for their implementation. Organizations follow these documents, which are also called prescriptive [106], to facilitate the process standardization, evaluation, and improvement [54, 103]. For example, the standard ISO/IEC/IEEE 12207 [84] 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) [136] as a significant innovation [15]. Then, the Software Process Improvement and Capability Determination-SPICE (ISO/IEC 15504 [80]) was also created. SPI frameworks, which mainly impose a plan-based development paradigm, aim at increasing product quality but also to reduce time-to-market and production costs [41].

Several SPI context-specific frameworks exist [149], e.g., Automotive SPICE (or ASPICE) [9], the medical devices MDevSPICEe [37] and the Object-Oriented Software Process OOSPICE [138]. Recently, SPICE has been revised and replaced with the ISO/IEC 330XX series [82], e.g., ISO/IEC TS 33053 [2], which defines a process assessment, and process reference model (PRM) for quality management.

The International Organization for Standardization (ISO) has also defined the fundamentals of quality management systems, which influence the process assessment and improvement [94]. In particular, there is the ISO 9000 series [78], e.g., ISO 9001 [77], guidance for their application ISO/IEC 90003 [73], and ISO/IEC TR 29110 [79], which applies to very small entities. Additionally, the information technology infrastructure Library (ITIL) framework, which the UK government has developed, aims to provide a guideline for delivering quality information technology services [94]. 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 [142].

Manufacturers of safety-critical systems have the duty of care[6] [104]. Consequently, ethics and regulatory regimes explicitly addressing such systems have stronger compliance requirements beyond the commitment to improve software process capability [14]. Manufacturers then must establish effective software development processes based on recognized engineering principles [129], usually found in industry standards [59]. There are governing bodies that are in charge of ensuring the safety of citizens. For example, e.g., 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 [24]. The Health and Safety Executive in England has used compliance with IEC 61508 [76] as a guideline for bringing legal actions if harm is caused by safety-critical systems [104]. As compliance with safety standards has become essential evidence for a jury in a product liability action [134], failure or inadequate compliance could lead to legal risks, i.e., penalties [42] and prosecutions [72].

In particular, prescriptive safety standards cover requirements for all software life-cycle activities, and exist in almost all safety-related domains, e.g., ISO 26262 [83] (automotive), CENELEC EN 50128 [50] and EN 50126 [51] (railway), DO-178C [1] (avionics), and IEC 62304 [75] (medical devices), to only mention some of them. Cybersecurity handbooks (e.g., cyber-physical vehicle systems SAE J3061 [131]), standard for software development (e.g., medical devices-IEC 62304 [75] and space mission-critical software-ECSS-ST-40C [53]), risk management (e.g., ISO 14971-application of risk management to medical devices [74]), and information technology (e.g., ISO/IEC 27000 [81]), are also part of the menu of standards that became de facto regulatory frameworks subjecting the organizations to mandatory certification.

We also find explicitly defined regulations. For instance, the European Data Protection Directive (EU DPD) [140], then replaced by the General Data Protection Regulation (GDPR) [52], and PIPEDA (Personal Information Protection and Electronic Documents Act) [61]. Both regulations 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 [12]. Since AI is still software, its needs will probably be approached from the software process perspective [145]. Thus, practitioners have to embrace software process diversity, i.e., the adoption of multiple normative software process frameworks within single software processes [127].

## 3 Research Method

This section describes our research method, which is based on the guidelines for Systematic Literature Review (SLR) recommended by Kitchenham and oth-

[6] In tort law, a duty of care is a legal obligation which is imposed on an individual requiring adherence to a standard of reasonable care while performing any acts that could foreseeably harm others [71].

ers [97, 99]. A SLR is a rigorous review methodology that involves three main activities.

1. **Plan the review**. This is a pre-review activity, which includes three tasks.
   (a) *Identify the need for a review.* This tasks permits to identify the reasons for undertaking the review and its scope.
   (b) *Specify goal and research question.* The goal and the research questions that aim at guiding the review are specified.
   (c) *Design the review protocol.* The review protocol should include a search strategy, which contains the search terms and resources to be searched, e.g., digital libraries. It also includes the study selection criteria that are used to determine which studies are included and excluded. Moreover, it contains the study selection procedure, which describes how the selection criteria will be applied. Finally, it includes the quality assessment criteria used to determine the rigorousness and credibility of the used research methods and the relevance of the studies.
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, including the SLR results and analysis.

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, i.e., plan and conduct the review, while we report the results of the review in Section 4.

### 3.1 Plan the Review

This section presents the pre-review activities, i.e., identify the need for a review, specify the goal and research questions, and design the review protocol.

**Identify the Need for a Review** As recalled in Section 2.2 the primary goal of a software process is to solve an application data processing problem by performing a knowledge acquisition activity. As such, software processes are valuable informational assets, which have increasing demands regarding the inclusion of requirements associated with normative frameworks (as recalled in Section 2.3). Organizations understand that they have to adhere to such demands because it is implicitly or explicitly dictated by both, the market and the law. However, those organizations that want to move towards greater agility may find it challenging since normative frameworks are commonly prescribed in a plan-based development paradigm.

Techniques for software process compliance checking could be helpful for organizations. Such techniques permit organizations to verify whether a software

process complies (or conforms) with the applicable normative requirements (as recalled in Section 2.1). However, the software process compliance checking tends to be complex. We present some factors that add complexity to the compliance checking tasks, as follows.

- The requirements included in the standards prescribe many details regarding the process structure (the presence of tasks ordered in a determined way, and resources ascribed to such tasks, i.e., personnel, work products, tools, and methods), and the properties of the process elements;
- There are many possible ways to be compliant. In particular, 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);
- The requirements can be superseded or eliminated if assessed rationales, i.e., explicit justifications demonstrating compliance, are provided;
- 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:
- Many standards or new versions of older standards may apply to the same software process.

For this reason, the automation of the tasks involved in compliance checking of software process is an area of research of increasing interest. Such task is considered to provide benefits in terms of efficiency and confidence to managers and process engineers, who require to (re)configure their software processes according to applicable software process-related normative frameworks. Efficiency could be reached by leaving the repetitive work of checking the requirements to a machine. Confidence could instead be reached when, after providing appropriate representations of both standards and processes, proofs of compliance can be derived. Several methods for compliance checking of software processes against different kinds of normative frameworks have been proposed in the literature. However, to the best of our knowledge, no comprehensive and systematic review has been conducted to characterize them. 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.

***Scope:*** The scope of an SLR can be defined by taking into account the guidelines proposed by Cooper et al. [40]. In particular, it is essential to determine the focus (outcomes, methods, theories, 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, the general public).

In particular, we focus on the research outcomes of the available literature addressing automated compliance checking of software process. Our goal is to

identify the specific aspects that have dominated past efforts regarding such topic, i.e., publication trends, the characteristics of the methods, potential impact, and challenges. We consider reporting our result from a neutral representation perspective, i.e., 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, i.e., works relating to the same abstract ideas will appear together. Finally, we aim to write our SLR to target specialized scholars, practitioners, and policymakers.

**Specify Goal and Research Questions** In this section, we describe the main goal of our SLR and the research questions.

***Goal:*** Based on the need identified in Section 3.1, this SLR goal is to characterize the current state-of-the-art regarding automated compliance checking of software processes against the constrains associated to different kind of software process-related normative frameworks (as recalled in Section 2.3). As presented in Section 2.1, compliance checking requires at least two sources of information, the normative document to be complied with and the process for which compliance is desired. To automatize this task, such sources of information should be computer-based analyzable. Thus, it is essential to know the methods used in the state-of-the-art to represent such specifications as well as the individual concepts used to describe the features included in the specifications. Moreover, it is important to identify the status of the tool-support provided and the mechanisms used to handle changes in the normative space, e.g., recertification. It is also crucial to learn the methods' target application, i.e., 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:*** Research questions are formulated by taking into account the research goal previously described (see Table 1).

**Design the Review Protocol** We present a summary of the concrete and formal plan used in the execution of the SLR.

***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 [46]. 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

Table 1: Research Questions

| Id | Question | Motivation |
|---|---|---|
| RQ 1 | How did research in automated compliance checking of software processes developed over time? | Identify the publication trend (i.e., number of papers published, dates and the publication venues), and the active groups doing research in the context of automated compliance checking of software processes. |
| RQ 2 | What are the characteristics of the methods described the primary studies? | |
| | 2.1 Which are the languages used to represent software processes entities and structures? | Characterize the different alternatives used to represent the software processes entities (units of work, roles, tools, and guidance) and their properties, as well as structures such as workflows, which are required for automated compliance checking described in the primary studies. |
| | 2.2 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. |
| | 2.3 Which is the level of automation? | Examine the automation level described in the studies. |
| | 2.4 What are the mechanisms, if any, used to handle standards evolution and software process reconfiguration? | Identify the characteristics used in the primary studies to address 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 standards in the same project). |
| 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, general software purposes, etc. |
| | 3.2 What are the types of normative documents targeted? | Describe the type of standards, policies, regulations, reference models or frameworks that target the studies. |
| | 3.3 What are the types of illustrative scenarios presented? | Extract information regarding the examples, illustrations, validation or use cases that describe the methods/frameworks/techniques. |
| | 3.3 To what extent agile methodologies are supported? | Describe whether the primary studies take into account the compliance checking in agile software processes. |
| RQ 4 | What challenges are identified in the primary studies? | Identify the challenges in current research or open problems, which can be used to determine future directions in this area. |

Table 2: Structure of the CIMO Logic.

| CIMO criteria | Factors |
|---|---|
| Context (C) | Software processes |
| Intervention (I) | Normative software process constraints |
| Mechanism (M) | Automation methods |
| Outcome (O) | Results of compliance checking |

the means that in a specific context are triggered by the intervention. Finally, the outcomes are the intervention results in its various aspects.

As presented in Table 2, the factors of importance in our SLR are software processes, normative software process constraints, automation methods, the re-

sults of compliance checking. 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 2.2, we found that the concept of "software process" is related to the concept of "software lifecycle" (or life cycle), "software workflow," and "software development methodology." Second, in Section 2.3, we found sources of "normative software process constraints" in a "standard", "reference model", "framework", "regulation", "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. Actually, using the mentioned words, we find articles containing techniques for compliance checking by means of model checking technology (See S9, S12, S15, and S27), So, the not inclusion of the concept did not limit the selections of the corresponding studies.

We did a test search in the library Science Direct[7] 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 automatic. Thus, the word automated is not included in the final search string. The results obtained with the word 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).

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") |

[7] https://www.sciencedirect.com/

***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 [153]: 1) ACM Digital Library[8], 2) IEEE Xplore Digital Library[9], 3) Springer Link[10], and 4) Google Scholar[11]. 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 since we also want to see the evolution of the topic over time. The inclusion and exclusion criteria is presented in Table 4.

Table 4: Inclusion and Exclusion Criteria.

| **Type** | **Description** | |
|---|---|---|
| **Inclusion** | I1 | The primary study belongs to the software engineering domain. We are only interested in automated compliance checking of software processes. |
| | I2 | The primary study is about compliance/conformance checking of software processes against the constraints associated to different kind of software process-related standards and reference frameworks |
| | I3 | The primary study included in the selection is a peer-reviewed article (i.e., scientific journal, conference, symposium, or workshop) written in English related to automatic compliance checking of software process. |
| | I4 | The primary study reports issues, problems, or any type of experience concerning the aspects related to process-related automated compliance checking, i.e., process models, requirements formalization, analysis of compliance. |
| | I5 | The primary study describes solid evidence on automated compliance checking of software processes by using, e.g., 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, e.g., process design, execution, the management of workflow, or adherence of a software process plan with the execution, or is does not does not present sufficient technical details regarding automated compliance checking of software processes. |
| | E2 | The text of the primary study is not available. |
| | E3 | The primary study belongs to the following categories: commercials, pure opinions, grey 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, e.g., business processes, building processes, etc. |
| | E5 | The primary study is not clearly related to at least one aspect of the specified research questions. |
| | E6 | The study is a secondary or tertiary study. |
| | E7 | The primary study did not undergo a peer-review process, such as non-reviewed journal, magazine, or conference papers, master theses and books (in order to ensure a minimum level of quality). |

***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

[8] https://dl.acm.org/
[9] https://ieeexplore.ieee.org/Xplore/home.jsp
[10] https://link.springer.com/
[11] https://scholar.google.com/

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 [151], which is a technique that aims at reaching more relevant primary studies. Backward snowballing refers to searching relevant studies by considering the reference list of an initial set of primary studies. Forward snowballing 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. The number of papers resulting from this selection procedure were be 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.

Table 5: Data Extraction Criteria.

| **Extracted data** | **Used for** |
|---|---|
| Author information, Study title | Study overview |
| Year, Publication types venues, and research groups | Study overview and RQ1 |
| Languages for representing software processes | RQ2.1 |
| Languages for representing requirements mandated by standards | RQ2.2 |
| Level of automation (fully automated, semi-automated) | RQ2.3 |
| Mechanisms for handling variability, if any | RQ2.4 |
| Validation/illustration/exemplification scenarios | RQ3.1 |
| Standards /policies/regulations/frameworks addressed | RQ3.2 |
| Support for agile, if any | RQ3.3 |
| Application domain | RQ3.4 |
| Challenges | RQ4 |

***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 and others [98]. For each item 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.

### 3.2 Perform the Review

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

**Data Collection** We apply the review protocol described in Section 3.1. In particular, we applied the search string defined in Table 3 to the different databases included in the study selection criteria without trunking the dates

Table 6: Study 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 are well described and clear |
| QA2 | Does the selected primary study discuss their results? | 0 | No. The results are not explicitly discussed in a discussion section (or a similar section) |
| | | 0,5 | Partially. There is a discussion section (or something similar), but results are not completely and 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. |
| 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. |

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 IEEExplore accepted all the words included in the search string. From these searches, we got 153, 71, and 1 possible primary studies, respectively. Instead, in Google scholar, we needed to divide the search string in two. The first one was ("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 "regulation" OR "policy") and the second one was (automatic OR formal) AND ("compliance checking" OR "conformance checking") AND ("software pro-

cess" OR "software life cycle" OR "software lifecycle" OR "software workflow" OR "software development methodology") AND ("reference model" OR "framework"). We obtained 762 and 839 possible primary studies, respectively (a total of 1601 primary studies, many of them were repeated). 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. 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 IEE Explore, 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 8 new primary studies in the backward snowballing and 5 new primary studies in the forward snowballing. The complete set of primary study that we have included in our SLR is 41. We illustrate the search process and the number of primary studies identified at each stage in Figure 1.

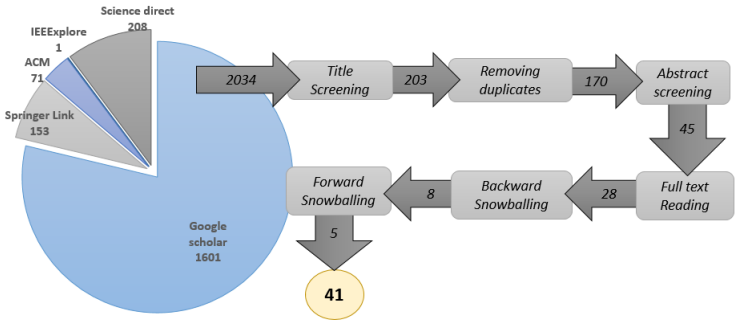



Fig. 1: Paper Selection Process.

**Data Extraction** The selected 41 papers were carefully read, evaluated with the quality criteria (presented in Table 6), and compiled in Table 7. Then, we did a summary of every approach, which we present in Section 4. We also collect relevant data that could help to answer our research questions (presented in Table 1). To record the data for later analysis and correlation, we used Excel spreadsheets. In particular, we focused on the data presented in Table 5.

Table 7: Selected primary studies using SLR.

| ID | Study Description | Year | Type | Quality Assessment Criteria | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | QE1 | QE2 | QE3 | QE4 | QE5 | QE6 | QE7 | Score |
| S1 | Tailoring and Conformance Testing of Software Processes: The ProcePT approach [150] | 1995 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S2 | Managing Standards compliance [49] | 1999 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S3 | Managing Process compliance [34] | 2003 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S4 | Compliance flow - managing the compliance of dynamic and complex processes [35] | 2008 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S5 | An Automatic Compliance Checking Approach for Software Processes [68] | 2009 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S6 | Supporting Qualification- Safety Standard Compliant Process Planning and Monitoring [87] | 2010 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S7 | Defining Software process model constraints with rules using OWL and SWRL [130] | 2010 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S8 | A model-driven engineering approach to support the verification of compliance to safety standards [119] | 2011 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S9 | NOVA Workflow: A Workflow Management Tool Targeting Health Services Delivery [111] | 2012 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S10 | Towards a process for legally compliant software [67] | 2013 | Worshop | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S11 | Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation [120] | 2013 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S12 | A framework to formally verify conformance of a software process to a software method [96] | 2015 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S13 | Cybersecurity policy verification with declarative programming [70] | 2016 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S14 | Representing Software Process in Description Logics: An Ontology Approach for Software Process Reasoning and Verification [88] | 2016 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S15 | How to Assure Correctness and Safety of Medical Software: The Hemodialysis Machine Case Study [7] | 2016 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5 |
| S16 | A Framework for Safety-Critical Process Management in Engineering Projects [10] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5 |
| S17 | Applying process mining techniques in software process appraisals [146] | 2017 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |

**Table 7 Continued:** Selected Primary Studies.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S18 | Continuous process compliance using model driven engineering [60] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S19 | Towards Increased Efficiency and Confidence in Process Compliance [29] | 2017 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S20 | Automated legal compliance checking by security policy analysis [128] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S21 | A formalization of the ISO/IEC 15504: enabling automatic inference of capability levels [123] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S22 | Security Analysis and Legal Compliance Checking for the Design of Privacy-friendly Information Systems [63] | 2017 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S23 | Towards efficiently checking compliance against automotive security and safety standards [26] | 2017 | Workshop | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S24 | An Axiom Based Metamodel for Software Process Formalisation: An Ontology Approach [89] | 2018 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S25 | Enabling compliance checking against safety standards from SPEM 2.0 process models [27] | 2018 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S26 | Ensuring Conformance to Process Standards Through Formal Verification [90] | 2018 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S27 | Integrating formal methods into medical software development: The ASM approach [6] | 2018 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S28 | Transforming SPEM 2.0-Compatible Process Models into Models Checkable for Compliance [28] | 2018 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S29 | Compliance of agilized (software) development processes with safety standards: a vision [57] | 2018 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S30 | Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3 – Enabling automatic inference of maturity and capability levels [124] | 2018 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S31 | Fast Compliance Checking in an OWL2 Fragment [18] | 2018 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S32 | Developing Medical Devices from Abstract State Machines to Embedded Systems: A Smart Pill Box Case Study [17] | 2018 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| S33 | Facilitating Automated Compliance Checking in the Safety-critical Context [25] | 2019 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S34 | Formalising Process Assessment and Capability Determination: An Ontology Approach [91] | 2019 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |

**Table 7 Continued:** Selected Primary Studies.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S35 | Using Models to Enable Compliance Checking against the GDPR : An Experience Report [143] | 2019 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S36 | A Life Cycle for Authorization Systems Development in the GDPR Perspective [43] | 2020 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S37 | Co-engineering of safety and security life cycles for engineering of automotive systems [20] | 2020 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S38 | Separation of Concerns in Process Compliance Checking: Divide-and-Conquer [30] | 2020 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S39 | Reusing (Safety-oriented) Compliance Artifacts while Recertifying [31] | 2021 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S40 | Compliance-aware Engineering Process Plans: The Case of Space Software Engineering Processes [31] | 2021 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S41 | Supporting Quality Assurance with Automated Process-Centric Quality Constraints Checking [113] | 2021 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |

# 4 Results

In this section, we report the results of the SLR. In Section 4.1, we present the a summary of the primary studies selected. Finally, in section 4.2, we present the analysis of the results in relation to the research questions of the study.

## 4.1 Summary of the Primary Studies

In this section, we summarize the main results obtained in the SLR. Specifically, the selected 41 papers (see Table 7) are categorized into 5 groups, according to the type of approach (see Table 8). As the table shows, most of the primary studies aim at performing compliance checking from the modeling of standards concepts (15). The second-largest group of primary studies belongs to the category in which compliance checking is performed from process modeling languages (13). Compliance checking from role-based access controls has 5 primary studies, compliance checking from documents workflow has 2 primary studies, and other approaches have 6 primary studies. Table 9, presents as summary of the main characteristics of the 41 studies. In the remaining part of this section, we present a summary of the studies according to the types of approaches previously described.

Table 8: Type of Approaches.

| **Type** | **Primary Studies** | **Total** |
|---|---|---|
| Compliance Checking from Documents Workflow | S1, S2 | 2 |
| Compliance Checking from Standards Concepts Modeling | S3, S4, S5, S8, S10, S11, S12, S14, S18, S21, S24, S26, S30, S31, S34 | 15 |
| Compliance Checking from Process Modeling Languages | S6, S7, S16, S19, S23, S25, S28, S29, S33, S37, S38, S39, S40 | 13 |
| Compliance checking from Role-based Access Controls | S13, S20, S22, S35, S36 | 5 |
| Other Methods | S9, S15, S17, S27, S32, S41 | 6 |

**Compliance Checking from Documents Workflow** Initial approaches for process-based compliance checking with quality standards are based on the verification of document evolution. In S1, the authors describe the process model as the specification of documents flow for all necessary tasks during standards assessments. The check is performed by reviewing whether the activities can occur by verifying the conditions for documents' existence. The rules that condition the specification of documents are provided in PROLOG III [39], which is a language that has its roots in FOL. The conformance checking is done with a PROLOG-based tool called ProcePT (Process Programming & Tailoring) and exemplified with the German process model VORGEHENSMODELL (short GV-Model) [22], which can be tailored against different kinds of quality standards such as ISO 9001. Such tailoring is done by removing activities and documents if they are not required in the selected standard. The approach only takes tasks and

documents under the GV model. Process elements such as persons and means to activities should be passed on experience from previous projects.

The authors of S2 consider that process compliance is represented in the documents produced during the engineering process. For this, a specification of a document schema in UML is proposed. The properties of the documents prescribed by the standards are formalized in FOL. Checks are performed when there is an attempt to read or write documents during process enactment. The environment is based on DOORS (Dynamic Object Oriented Requirements System) for managing the documents. DOORS has a Dynamic eXtension Language (DXL) that can be used to automate tasks. The checking of FOL rules is done with AP5 [38], which is an extension of Common Lisp. The standard used for exemplifying the approach is called PSS-05. Both tools in S1 and S2 are proof of concept prototypes.

**Compliance Checking from Standards Concepts Modeling** Several approaches consider the modeling of process-related elements from specific standards concepts provided by the standards. In S3, the authors conceive a workflow manager, in which a given standard, in this case, the standard IEC 61508, is represented as a Model of Standards, which acts as a knowledge-base to provide the required information. The standard's meta-model is created in UML. An intelligent compliance agent, called the Inspector, performs compliance checks by comparing the model of standards and the User-Defined Process (UDP). The approach is illustrated with a recommendation handling example. In S4, the same approach than in S3 is enhanced with more details. In particular, the model of standards is presented as an activity-based ontology where task execution is constrained by the pre-and post-conditions, with an added type of precondition being the technique that has to be used to carry out a task. A task agent performs a task, and the progress of the task's execution is represented through several states. A capability is a skill, technique, method, knowledge, or any attribute that a task agent requires to perform a task. Compliance checks are for checking a UDP against the Model to identify compliance errors and assist the user in specifying a process that meets a selected standard's requirements. The approach is evaluated with the light guard development project, an application for assuring programmable electronic systems.

In S5, the authors present an approach for compliance checking with quality standards (such as ISO/IEC 90003) in which the process-based requirements of a standard are represented as process patterns. The process elements normally found in standards, i.e., activities, roles, and work products, are defined as UML classes. The compliance checking is a measure of the process deviation (absent or skipped element, or reverse order of the implemented tasks represent a non-compliant process model) during enactment using feature diagrams called PPST (Process Pattern Structure Tree). PPST is based on the idea of Structured Activity Node in UML Activity Diagram and PST (Process Structure tree) [147]. The approach is exemplified with a general software development process.

In S8, the authors propose the use of UML metamodel for creating process models and the conceptual models of the safety standard. The UML model of the process concepts includes activities, artifacts produced and required, techniques and roles. UML profiles are created to describe instances of the standards, in this case, the standard IEC 61508. The profile is augmented with verifiable constraints written in OCL. The compliance checking is automated. Compliance rules have to be manually created, as well as the process model. The approach is tool-supported, i.e., by using Rational Software Architect[12]. The application UML profile is done in a case study related to the construction of a domain model for sub-sea control systems in compliance with IEC 61508. In S11, the same approach as in S8 is evaluated by taking into account experts opinions, which found the approach easy to use and with a good acceptance.

In S10, the author present a Governance Analysis Tool (GAT) for information privacy. GAT is a UML-based metamodel that contains a Governance Analysis Model(GAM) and a Governance Analysis Language (GAL). GAM captures information domain, i.e., process activities and roles, as well as organizational information and general information regarding the legal entity. GAL is capable of expressing many types of legal and organizational requirements. The MIT's logic analyzer Alloy[13] is the engine on which GAT runs. For this, GAL information is translated into assertions in Alloy's language (which uses predicate logic) and the Alloy tool can find counterexamples indicating situations of non-compliance. A case related to compliance checking of a personal health information agains PIPEDA (the Canada's Personal Information Protection and Electronic Documents Act) is presented for illustration purposes.

In S12, the authors propose a general software process reasoning and verification tool by using fUML, a language that defines precisely the execution semantics for a subset of UML Activity Diagram. The formalization of constraints included in software process reference frameworks such as OPENUP, extreme programming, scrum and Kanban is done by using Linear Temporal Logic (LTL). The tools, which is graphical-based, is developed as an Eclipse EMF plug-in. Modelling of process and the formalization of constraints is manual but assisted with the tool and by a specific template-based constraint language. Evaluation is presented on a Scrum-based process.

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 authors present an approach for software process verification and reasoning, which permits the translating of process models represented in composition tree notations into DL. The knowledge of the process models contains the title, purpose, outcomes, activities and task. The resulting knowledge base representing properties of the process elements that can be constrained with software process standards such as ISO/IEC 12207, and ISO/IEC/29110. The approach is illustrated with a case study related to the Human Resource Management Process. In S24, the authors present an ontological approach (defined as an ax-

[12] https://www.ibm.com/developerworks/downloads/r/architect/index.html
[13] https://alloytools.org/

iom metamodel) in OWL-DL. Such ontology contains 4 main concepts which are originally selected form the standard ISO/IEC 29110 is presented. In S26, the authors built on top of the previous work S14 and S24, which is related to the creation of the process model, including a formal approach to software process analysis and verification using DL-based ontology. In this case the DL axioms represent the based practices or the process reference model (PRM) defined by ISO/IEC 15504-5. To illustrate the process verification approach and the inferencing services offered by ontologies, Protegé[14] is used. The case study selected is related to the development of Moodle[15]. In S34, the authors include DL axioms related to the formalisation of the process capability dimension of process assessment model (PAM). As a running process capability level example, the authors use the capability level two (managed process) featuring PA2.1 performance management attribute and PA2.2 work product management process attribute from ISO/IEC 15504-5. The Measurement Framework is extracted from ISO/IEC 33020. The compliance checking is done by using OWL reasoners.

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. Elements defined are activity, role, and tools. Each activity defines contracts. The notion of contract is used to bind the components (activities) using Design by Contract. A notion of conditions is also associated with the contracts at both levels. This serves for specifying the pre/post conditions associated with an activity. For compliance checking, a mapping between the abstract and concrete process is performed. A process standard is translated into the abstract level of a process model only once for each standard. The metamodel is proposed, but there is not a mention of a specific tool, The example application is presented with the standard ECSS-E-ST-40C.

In S21, and S30, the authors present the evolution of a framework for enabling inference of maturity and capability levels of software processes. The concepts related to processes and work products are modeled in OWL. SWRL is used to create the compliance requirements. An SWRL rule is an implication between the antecedent and the consequent, which is a combination of zero or more atoms that are not allowing disjunctions or negation. The standards analysed are ISO/IEC 15504 and SEI CMMI v1.3. Test cases from different organizations and appraisals results published by the CMMI Institute[16], were used for testing the approach, which was modeled in OWL and analyzed with OWL reasoners, such as HermIT[17]. In S31, the authors a similar approach for the representation of GDPR, but the modeling is performed in OWL with constraint in DL.

**Compliance Checking from Process Modeling Languages** Some approaches take as a base consolidated process modeling languages and add a layer

[14] https://protege.stanford.edu/
[15] https://moodle.org/
[16] https://sas.cmmiinstitute.com/pars/
[17] http://www.hermit-reasoner.com/

of analysis by using formal languages. In S6, the authors propose a framework in which an OWL ontology is used to formalize domain standards and further domain knowledge required to understand processes. The information in the ontology is constrained with Description Logic (DL) rules and transformed into the SPEM 2.0 process models. Explicitly, the authors mention the provision of tasks in the process model, which are traceable to product models. Tooling is consolidated by using Protege for the ontology, XSLT transformation, and Eclipse Process Framework as the reference for SPEM 2.0 elements. Illustrations of the concepts presented in ISO 26262. In terms of tailoring, there are OWL structures defined for transferring only elements according to a determined ASIL. The limitation is that the formalized library only applied to ISO 26262.

In S7, the authors present an approach in which software process are implements in SPEM 2.0, and then translated in OWL ontologies to permit the application of constraints that can be derived from software engineering standards such as ISO 12207, process improvement frameworks such as CMMI or ISO/IEC 15504 and agile processes. Both, S6 and S7 present a basic approach as a proof of concept using OWL in Protegé. S7 in addition combines protegé with SWRL (Semantic Web Rule Language) rules to represent constraints as rules.

In S16, the authors propose a framework for compliance checking automation at planning time, which includes the formalization of process in BPMN and then transformed into timed petri nets. Compliance constraints, which are extracted from the regulations, are represented in SHACL, which is a constraint language able to retrieve information from RDF (Resource Description Framework)[18]. Process tasks are represented in Camunda BPM engine[19], which is a toolset that offers support for BPMN 2.0 (Business Process Management Notation). The authors have implemented a project-specific reasoner. The framework has been defined from an industry scenario from the railway automation domain in compliance with EN 50126.

In S19, S23, S25, S28, 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 (Software & Systems Process Engineering Metamodel), called EPF (Eclipse Process Framework) Composer, which permits the representation of process elements (i.e., tasks, roles, work products, guidance, and tools, and process workflows). In S19 and S23, the requirements from the standards are modeled in defeasible logic, and the approach permits to manage safety-oriented process lines, i.e., process that are highly related. The reasoner used for compliance checking is called SPINdle[20]. Initially, the authors focus on the automotive domain by using the standards ISO 26262, ASPICE, andthe cybersecurity handbook SAE J3061. In S25 and S28, the authors consolidate a tool supported framework by including Formal Contract Logic (FCL), which is an evolution of

[18] https://www.w3.org/RDF/
[19] https://camunda.com/
[20] http://spindle.data61.csiro.au/spindle/

defeasible logic augmented with the concepts of deontic logic. FCL, which can be analysed by using a compliance checker called Regorous[21], combines concepts and temporal knowledge representation characteristics to support the formalization of requirements representing obligations and permissions in a normative context that can be defeated by evolving knowledge. The standards used to illustrate the approach are ISO 26262 and CENELEC EN 50128 (which applies to railways). In S33, the authors augment the framework with process patterns extracted from ISO 26262. In S38, the authors include process compliance hints. Such hints are based on dividing requirements in terms of the elements they target as well as the specific properties defined for each element. As a result, customized icons an templates are provided for facilitating compliance effects creation, which are used to form the propositions of the rules in FCL. Compliance hints are illustrated with the formalization of CENELEC EN 50128. In S39, the framework adds the tool support for variability management offered by BVR-T (Base Variability Resolution Tool[22]), included in the tool-chain EPF-C ∘ BVR-T [86] to show process plan adherence with new versions of standards (in this case the family of the standard ISO 14971).

In S40, the authors compile the compete framework and present a case study taking into account the standards ECSS-E-ST-40. In S29, the frameworks is analysed focusing on support for agilized environments, specially R-Scrum [55], an agile process for avionics [112] and Safe Scrum [137].

In S37, the authors propose a method for managing compliance of processes that have similar characteristics. In this approach, the process elements are manually selected according to one specific standard and modeled in SPEM 2.0. Then, a standard of the same family, i.e, standard with similar characteristics is selected an manually compared with the initial one. Such comparison should highlight the common and variable process aspects mandated by the standards. Such aspects are modeled. The compliance checking is done by using BVR tool, which permits the creation of simple rules in Basic Constraint Language (BCL) to make possible the creation of compliant plans according to the selected standard.

**Compliance Checking from Role-based Access Controls** In S13, the authors present an approach for the cybersecurity domain, which permits to address the verification of security policies in role-based access control of enterprise software. The automated security policy verification approach describes a representation model and rules derived from the company's role-based access control (RBAC) policy in Answer Set Logic (ASL). ASL semantics is based on autoepistemic logic and default logic. For this reason, it makes a distinction between a strong (or traditional) negation and negation as failure (negation derived from incomplete information). It is a modeling concept illustrated with a web application software to assist the hiring process in a company that can be implemented with ASP solvers, e.g., LPARSE, DLV, GRINGO.

[21] https://research.csiro.au/data61/regorous/
[22] https://github.com/SINTEF-9012/bvr

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. The message chart, which represent 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. Compliance policies are formalized in FOL and resolved with SMT (Satisfiability Modulo Theories [11]) solvers. The control policy is checked on the access rights of the roles that are involved in the process. However, there is not check on tasks to be performed or other process elements. A Python-based tool is created to perform the compliance checking. Such tool uses the PySMT library[23] API to invoke the SMT solver MathSA[24]. The approach is illustrated with the a Personal Health Record (PHR) system and the processing of personal data to produce salary slips of employees.

In S36, control policies are represented as user histories, e.g., 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 by using XACML (eXtensible Access Control Markup Language). As a result a list of XACML policies encoding the GDPR's provisions are defined. The list of XACML policies are instantiated with actual attributes. An access control tool uses the derived attribute classification for mapping them into the user histories and enforce policies. Consequently the policies are applicable to the subject. This approach does not utilizes compliance checking as such, but helps for deriving test cases that could enforce the policies at testing time.

In S35, the authors propose a conceptual representation of the entities involved in GDPR (General Data Protection Regulation) in UML. The UML representation permits the creation of different types of data artifacts. However, for process-based compliance checking, the artifacts available are the roles (called actors). A set of OCL (Object Constraint Language)[25] constraints embedded in the UML classes are created to reflect the GDPR's obligations. It only tackles obligations, and the rules are embedded in the generic model.

**Other Methods** In S9, the authors present a workflow management system called NOVA, which is not specifically defined for compliance checking of software process but can be used for that purpose. The approach uses the time Compensable Workflow Modeling Language (CWMLT) extended with the time constraints of delay and duration in Linear Temporal Logic (LTL). In the workflow, it is possible to create units of work (or tasks). There is a small ontology in OWL 2.0 representing the facts and rules found in healthcare policies. The NOVA Engine is a workflow engine based on Service Oriented Architecture (SOA). The

[23] https://github.com/pysmt/pysmt
[24] http://mathsat.fbk.eu.
[25] https://www.omg.org/spec/OCL/2.4/PDF

approach is illustrated with a monitor system following the guidelines for managing cancer-related pain in adults. NOVA Editor uses a graphical environment, which permits the creation of correct by construction workflows (the incorrect composition of workflow activities is prevented). Manual changes have to be done in the workflow model if guidelines are tailored to specific cases.

In S15, S27, and S32, the authors propose an incremental life cycle model for medical software development based on model refinement, includes the main software engineering activities (specification, validation, verification, conformance checking), and is tool-supported. The approach is based on the Abstract Sate Machine (ASM) [19], which is a transition system that extend finite states machines with domain of objects with functions and predicates. ASM is 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 particular, it is possible to model the units of work and their sequence. ASM has rule constructors that represent common vulnerabilities and defects. Such rules are created in Computation Tree Logic (CTL) and can be used to check the ASM modeling for avoiding violations of suitable properties. The reasoner is part of a framework called AsmetaV. ASM is used to define the main phases and activities of the development process. Requirements modeling is based on model refinement; it starts by developing a high-level ground model that captures stakeholders requirements.

S15 and S27 show a case study related to the hemodialysis machine case study. In S32, the authors present an approach for checking the activities that are needed for creating a Smart Pill Box. The checks are implemented in a language called Avalla and tested with the validator AsmetaV. The compliance verification with IEC 62304 and the FDA general principles of software validation are manually mapped to the steps taken in the process verification of the approach presented. Thus, the approach is actually doing model checking to the device. In S17, the authors present a method for discovering actual software process models based on event logs and check conformance with the CMMI-DEV model. For this, an event log is used to automatically construct a petri net that explain the behavior discovered in the log. The conformance checking process aims to verify the discovered process with the "assessable" elements of CMMI-DEV model (development lifecycle proposed by CMMI-DEV), which are modeled by using Linear Temporal Logic (LTL).The result is a report presenting if certain properties (CMMI-DEV model rules) hold in a log. The method is tool supported via the ProM tool[26].

In S41, the authors present a tool-supported framework for tracking processes in the background of the actual software development, automatically standards constraints, e.g., DO-178C/ED-12C and informing quality violations. The approach is evaluated with an open source system for unmanned aerial vehicles and an industrial air traffic control system (ATC).

[26] http://www.promtools.org/doku.php

Table 9: 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 | ✓ | ✓ | | | | | FOL (Prolog) | PC | ✓ | Software Development | | ISO 9001 | | Quality |
| S2 | UML | | ✓ | | | | | FOL | PC | | Software Development | | ISO 12207 | | Quality |
| S3 | UML | ✓ | ✓ | | | | ✓ | UML | CM | | Recommendation Handling | ✓ | IEC 61508 | | Safety-critical |
| S4 | UML | ✓ | ✓ | | | | ✓ | UML | PC | | Programmable Electronic Systems | ✓ | IEC 61508 | | Safety-critical |
| S5 | UML | ✓ | ✓ | | | | | PPST | 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 | | Process Guidelines | ✓ | Process Verification |
| S8 | UML | ✓ | ✓ | ✓ | ✓ | | | OCL | PC | | Sub-Sea 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 | | Sub-Sea control | ✓ | IEC 61508 | | Safety-critical |
| S12 | fUML | ✓ | | | | | ✓ | LTL | IT | | Software Development | | Process Guidelines | ✓ | Process Verification |
| S13 | ASP | ✓ | ✓ | | | | ✓ | ASL | CM | | Human Resources | ✓ | RBAC policy | | Cybersecurity |
| S14 | CT | ✓ | ✓ | | | | | DL | PC | | Human Resources | ✓ | ISO/IEC TS 33053 | | Quality |
| 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 | | SPI |
| S18 | UML | ✓ | ✓ | | | ✓ | | Mapping | CM | | Space | | ECSS-ST-40C | | Safety-critical |

**Table 9 Continued:** Summary of the Reviewed Studies.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 principles | | Safety-critical |
| S28 | SPEM 2.0 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | FCL | PC | ✓ | Railway | | CENELEC EN 50128 | | Safety-critical |
| S29 | SPEM 2.0 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | FCL | PC | | Agilized Environments | | ISO 26262 DO-178C | ✓ | 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 principles | | Safety-critical |
| 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 |

**Table 9 Continued:** Summary of the Reviewed Studies.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S40 | SPEM 2.0 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | FCL | PC | ✓ | Space | | ECSS-E-ST-40C | | Safety-critical |
| S41 | UML | ✓ | ✓ | | | | ✓ | Declarative | PC | | Avionics | ✓ | DO-178C | ✓ | Safety-critical |

**Conventions**

(✓) Supported, (CM) Conceptual Model, (PC) Proof of Concept Prototype, (IT) Implemented Tool.

**Languages**

(UML) Unified Modeling Language[27], (OWL) Web Ontology Language[28], (SPEM 2.0) Software & Systems Process Engineering Metamodel[29]
(CWMLT) Compensable Workflow Modeling Language [125], (fUML) Foundational Subset for Executable UML Models[30],
(ASM) Abstract State Machines [19], (ASP) Answer Set Programming & (ASL) Answer Set Logic [107],
(BPMN) Business Process Management Notation[31], (SMT) Satisfiability Modulo Theories [11],
(XACML) eXtensible Access Control Markup Language, (FOL) First Order Logic, (PPST) Process Pattern Structure Tree [147],
(XSLT) Extensible Stylesheet Language Transformations[32], (SWRL) Semantic Web Rule Language[33], (OCL) Object Constraint Language[34],
(LTL) Linear Temporal Logic [122], (GAL) Governance Analysis Language [67], (DL) Description Logic[35],
(Def-L) Defeasible Logic[36], (FCL) Formal Contract Logic [62], (CTL) Computational Tree Logic, (BCL) Basic Constraint Language,
(SHACL) Shapes Constraint Language[37], (CT) Composition Trees

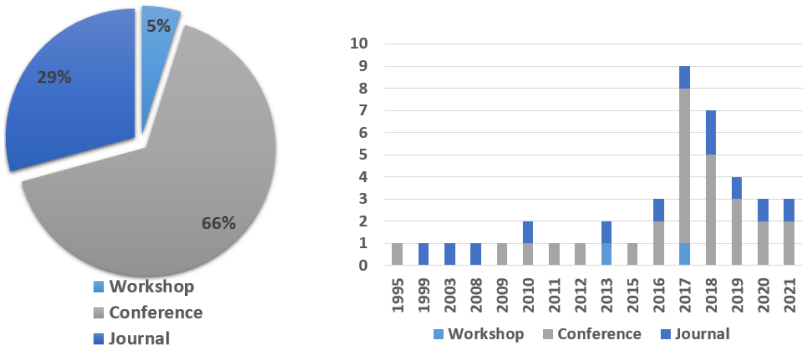
[27] https://www.omg.org/spec/UML/
[28] https://www.w3.org/OWL/
[29] https://www.omg.org/spec/SPEM
[30] https://www.omg.org/spec/FUML
[31] https://www.bpmn.org/
[32] https://www.w3.org/TR/2017/REC-xslt-30-20170608/
[33] https://www.w3.org/Submission/SWRL/
[34] https://www.omg.org/spec/OCL/
[35] http://dl.kr.org/
[36] http://www.defeasible.org/
[37] https://www.w3.org/TR/shacl/

### 4.2 Analysis

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

**RQ 1. Publications Distribution** This section presents the publication distribution of the 41 primary studies resulting for the SLR (i.e., time and venue) (see Figure 2) and active research groups in the context automatic compliance checking of software processes (see Table 10). In particular, Figure 2a, presents the distribution of the studies according to the types of publication venues. As the figure depicts, most primary studies were published in conferences (66%), while journals (29%) and workshops (5%) were the sources of fewer studies.

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 9 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 three months since we finished our search in March. Thus, the trend could increase during this year.




(a) Venues Types. (b) Years.

Fig. 2: Publications Distribution.

Concerning the active research groups within automated compliance checking for software processes, 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 10 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, 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.

Table 10: Active Research Groups.

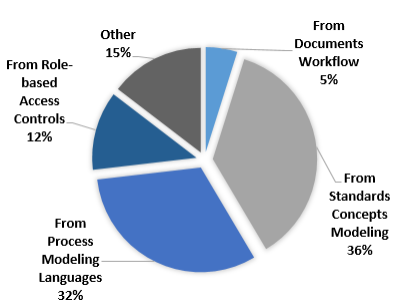
| **Affiliations** | **Primary Studies** | **Total** |
|---|---|---|
| Mälardalen University | S19, S23, S25, S28, S29, S33, S37, S38, S39, S40 | 10 |
| Griffith University | S14, S24, S26, S34 | 4 |
| Charles University | S15, S27 | 2 |
| Loughborough University | S3, S4 | 2 |
| Universidade de Lisboa | 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 |

***Analysis of the results for RQ 1.*** We did not set a lower boundary for the year of publication in our search process since, 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 2b 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, so we could consider the 1990's the initiation of this topic's work. This result corroborates with the publication of Osterweil's seminal paper [117] back in 1987, where the author discusses the nature of software processes and categorized them as a kind of software, which can also be programmed.

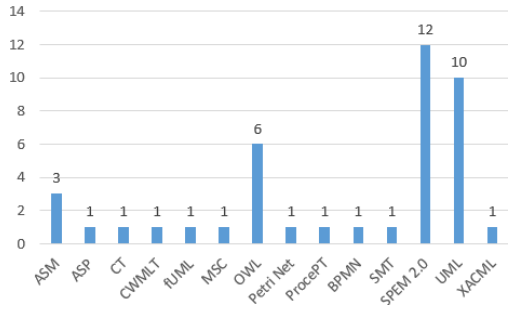
Osterweil's assumptions could be the source of interest for work related to the formalization of processes and their normative constraints. However, after analyzing the general temporal view of the studies, we can conclude that the number of studies about automated compliance checking of software processes is 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.

**RQ 2. Characteristics of the Methods** In this section, we present the characteristics of the methods described in the primary studies selected (summarized in Figure 3) by answering questions RQ 2.1, RQ 2.2, RQ 2.3 and RQ 2.4.

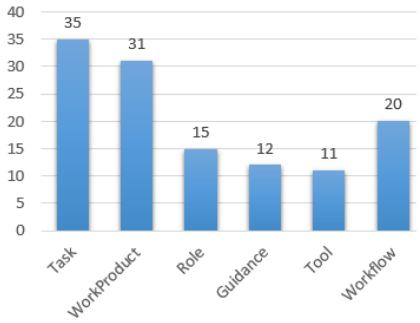
***RQ 2.1. Languages used to represent software processes entities and structures*** Five types of approaches (see Table 8) have been used to represent the information contained in software processes. Such approaches are distributed as presented in Figure 3a. 36% of the primary studies, namely S3, S4, S5, S8, S10, S11, S12, S14, S18, S21, S24, S26, S30, S31, S34, consider the modeling of process-related elements from specific standards concepts. 32% of the primary studies, namely S6, S7, S16, S19, S23, S25, S28, S29, S33, S37, S38, S39, S40, take as a base consolidated process modeling languages to which a layer of analysis using formal languages is added. The minority of the studies found are distributed as follows: 12% of the methods, namely S13, S20, S22, S35, S36, take into account



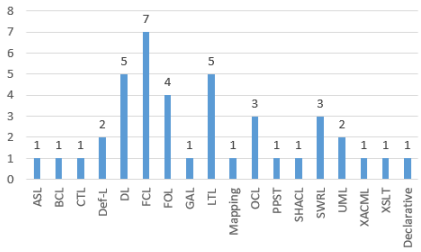

(a) General Approaches.



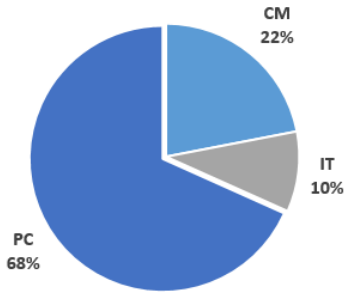

(b) Languages for Modeling Process.



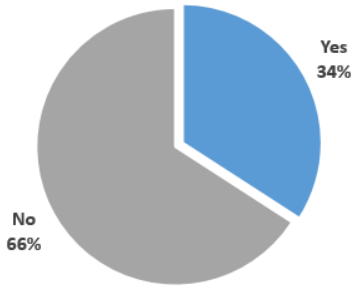

(c) Process Elements Represented.




(d) Languages for Compliance Requirements Modeling.




(e) State of the tool support.




(f) Evolution Handling Provision.

Fig. 3: Methods Characteristics.

access rights given to the roles in a process, 5% of the methods, namely, S1 and S2, take into account the documents workflow, and the final 15% of the methods, namely S9, S15, S17, S27, S32, S41, have other types of proposals. e.g., process mining, declarative programming, and workflow modeling.

The five types of approaches make use of 14 different languages to represent the process elements, and structures as Figure 3b depicts. n particular, S15, S27, S32 use ASM, S13 uses ASP, S14 uses CT, S9 uses CWMLT, S12 uses fUML,

S22 uses MSC, S21, S24, S26, S30, S31 and S34 use OWL, S17 uses Petri Net, S16 uses BPMN, S20 uses SMT, S1 uses ProcePT, S6, S7, S19, S23, S25, S28, S29, S33, S37, S28, S39 use SPEM 2.0, S2, S3, S4, S5, S8, S10, S11, S18 and S35 use UML, and finally S36 uses XACML (see Table 9).

It is important to note that models created in OWL and UML could also be considered as new languages. Thus, in the end, we have more than 14 languages used for modeling software processes. We also can see in Figure 3c that some process elements have more importance than others in the modeling languages created/reused, as each normative framework considers different kinds of process elements. In particular, significant attention in the modeling part of the processes is given to the tasks, work products, and workflows.

***RQ 2.2. Languages used to represent compliance requirements*** In the analysis of the selected primary studies, we found that a wide range of studies uses FCL (S25, S33, S28, S29, S33, S38 and S39) to formalize the requirements prescribed by the standard (see Figure 3d). In the second place, the preferred languages are LTL (S9, S12, S15, S17 and S27) and DL (S14, S24, S26, S31 and S34). In the third place, the selected language is FOL (S1, S2, S20, S22). However, there are languages in many other flavors that the researchers prefer to represent the requirements prescribed by the standards, i.e., ASL, BCL, CTL, Def-L, GAL, OCL, PPST, SHACL, SWRL, UML, XACML, XSLT, and Declarative languages (database approach). Thus, the modeling of requirements follows a similar trend as modeling processes: several languages, each selected according to specific needs.

***RQ 2.3. Level of automation*** The automation part claimed in the studies (see Section 4) is related to the compliance reasoning, namely the automatic comparison between the process and the normative documents. Frameworks composed of chained tools also automatically transform the information between the interrelated tools. Those that perform process mining also provide an automatic mining procedure. However, the formalization of requirements is performed mostly manually, in some cases, by using formalization patterns. The state of the tool support is also variable. We classify it in three groups: (CM) Conceptual Model, (PC) Proof of Concept Prototype, (IT) Implemented Tool, as presented in Figure 3e. 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, and S39, are prototypes that are used as a proof of concept, 22% of the methods, namely, S3, S13, S18, S19, S21, S23, S30, S31 and S36, are conceptual models, and only 10% of the methods, namely S10, S12, S17, and S20, are fully implemented tools.

***RQ 2.4. Evolution handling*** As presented in Figure 3f, only 34% of the primary studies 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 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 in the creation of processes models process, which is in charge of transferring only those requirements, methods and activities, which are relevant according to the system's ASIL. In S9, in addition, there is a monitor system that follows the guidelines for managing constraints and permits the creation of correct by construction workflows, preventing the incorrect composition of tasks. In S24, it is used a mechanism called powertype, which is pattern for modeling that combines instantiation and generalisation semantics in process metamodeling. In S19, S23, S37, and S39, the use of methodologies such as process lines, permit not only evolution handling but also to manage process diversity and reuse. In S25, S28, 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.

*Analysis of the results for RQ 2* We can see in the results that researchers use different kinds of approaches and methodologies to represent the software process to be used for automatic compliance checking. The purpose of the primary studies was 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 process workflows 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 (such as SPEM 2.0 and BPMN) could be reused and extended according to specific needs.

Compliance checking of software processes built on the capabilities provided by logic-based languages, especially for representing the requirements prescribed by the normative frameworks. In particular, the selection of languages in the primary studies was very diverse showing a similar trend than in languages used to represent software processes. In general, every formal method has its strengths and limitations as its own formal approaches and semantics. Some are easier to understand and use than others. The coverage, readability characteristics and tool support are also aspects that vary from one formal language to the other. Thus, it is important to find the correct balance between all those aspects to achieve the best fit for the problem at hand.

Essentially, the surveyed methods require human intervention, especially to implement the inputs of the reasoning process. The manual mapping or formalization of requirements as constraints requires considerable knowledge of the underlying formalisms and formal techniques. Therefore, formal approaches are often not easy to use for many process engineers. Given this aspect, there is a need for automate the transformation of normative requirements into formal representation, or at least, the provision of editors that could lessen the demands of its use. In addition, it is challenging to promote the use of methods for auto-

matic compliance checking in the industry when the tool support is lacking or nonexistent.

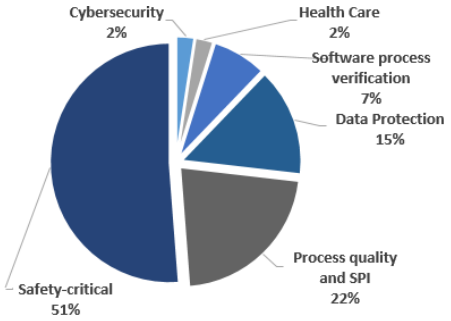
Evolution handling is a crucial aspect of process-related compliance management. However, the results of the SLR show that this aspect has been somewhat neglected. Another downside of the methods could be that the hard-coded rules could lessen the extensibility and generality and, therefore, the scope of application of these approaches. 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.

**RQ 3. Potential Impact** In this section, we present the potential impact of the studies in terms of application domain, normative documents targeted, illustrative Scenarios and agile support (summarized in Figure 4) by answering questions RQ 3.1, RQ 3.2, RQ 3.3 and RQ 3.4.

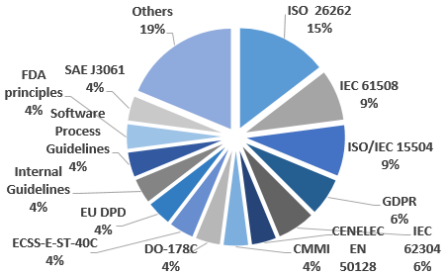
***RQ 3.1. Application domains*** Several application domains are addressed in the primary studies, as presented in Figure 4a. The most representative application domain is the safety-critical, with 51% of the studies tackling this sector, i.e., S3, S4, S6, S8, S11, S15, S16, S18, S19, S23, S25, S27, S28, S29, S32, S33, S37, S38, and S39. Then, we find that the researchers are interested in software process improvement SPI and quality (22%), i.e., S1, S2, S5, S14, S17, S21, S26, S30, and S34, and data protection (15%), i.e., S10, S20, S22, S31, S35, and S36. Other application domains are also represented in less quantity, i.e., software process verification (7%), i.e., S7, S12, and S24, Cybersecurity (2%), i.e., S13 and health care (2%), i.e., S9.

***RQ 3.2. 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 4b, the standards more used are ISO 26262 (15%), i.e., S6, S19, S23, S25, S29, S33, and S37, IEC 61508 (9%), i.e., S3, S4, S8, and S11, and ISO/IEC 15504 (9%), i.e., S21, S26, S30 and S34. To a lesser extent, the primary studies used GDPR (6%), IEC 62304 (6%), SAE J3062 (4%), FDA principles for software development (4%), software process guidelines (4%), internal guidelines (4%), ECSS-E-ST-40C (4%), DO-178C (4%) and CMMI (4%). Other standards were also used, representing 19% of the studies (i.e., ISO 12207, ISO 14971, ISO 9001, ISO/IEC 29110, ISO/IEC 90003, ISO/IEC TS 33053, PIPEDA, ASPICE, EN 50126.

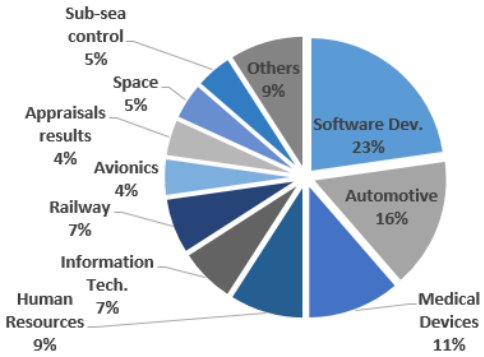
***RQ 3.3. Illustrative scenarios*** Illustrative scenarios are presented in Figures 4c and 4d. In particular, the studies focused primarily in general aspects of software development (23%), i.e., the GV-Model in S1, Case PSS-05 in S2, testing procedures and scrum processes), Automotive examples (16%), i.e., S6, S19, S23, S25, S33 and S37, and Medical devices development (11%), i.e., 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.




(a) Application Domain.

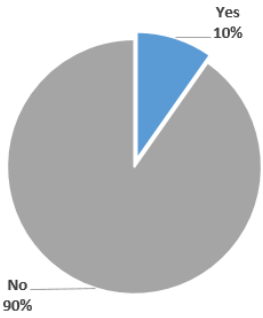



(b) Normative Frameworks Addressed.




(c) Illustrative Scenarios.


(d) Evaluation Data from Industry.




(e) Agile Support.

Fig. 4: Potential Impact.

Representative examples were also found in human resources systems (9%), i.e., i.e., S13, S14, S20, and S22, general applications in information technology (7%), i.e., S10. S17 and S35 railway (7%), i.e., S16, S28 and S28, avionics (4%), i.e., S29 and S41, Space (5%), i.e., S18 and S49, Sub-sea control (5%), i.e., S22 and S36 and appraisals results (4%), i.e., 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, 19 of

the 41 studies (approximately, 46%) used data extracted from industrial settings to evaluate their methods, i.e., S3, S4, S8, S9, S10, S11, S13, S14, S15, S16, S10, S21, S22, S27, S30, S31, S32, S36 and S41.

***RQ 3.4. Agile support*** Support for agile is not highly represented in the studies selected (the only 10% of the studies showed some agile-related information). 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, i.e., 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.

*Analysis of the results for RQ 3* The primary studies' methods provide 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, different standards targeted, and illustrative scenarios performed. 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 (46%), should be increased in order to provide real setting insights. 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 non concretized tool support (which is the common aspect) may be a factor that also hinders their application in practice.

**RQ 4. Challenges** Our investigation found that the existing literature related explicitly to compliance checking of software processes is scarce and scattered (see answer to RQ 1). The publication's irregularity in the initial years and the reduced amount of journal papers published may indicate that the topic has taken a long time to establish itself as a research subject. It seems also that research has been done in silos. Such independence may result in wasted research efforts since languages for process modeling are very often created from scratch.

In today's methods for automated compliance checking of software processes (see answer to RQ 2), diverse abilities are required from their potential users. In

particular, there is the need for knowledge regarding process modeling and the ability to formalize natural language in a specific formal language. As potential users, we have the process engineers who may already have some expertise in process modeling. However, different tools may approach modeling in different ways. Besides, the formalization of natural language in which the requirements are commonly specified is always perceived as demanding. Such perception may hinder the interest of the potential users and, thus, the methods' use.

The automation level claimed by the methods studied is related to two aspects. On the one hand, there are means to automate the compliance reasoning required to compare processes and the normative documents regulating them. On the other hand, there is conceptual integration of the tool-chain required to provide the reasoning aspects. However, in most primary studies, the concrete technological interaction between different tool-chain components is still a weak link in tool support provision. As a result, it is frequent to find that the tool support is still at the stage of conceptual modeling or proof of concept prototypes.

Finally, we can see impact problems (see the answer to RQ 3). Particularly, there is no consistent use of data from industry, limiting the evaluation of the studies. Moreover, in almost all the studies, the standards are addressed in isolation, reducing the results' generalizability. Finally, there is a lack of support for agile. These three aspects should be addressed in future research efforts to boost the implementation of the methods that are already available in the state-of-the-art.

## 5 Discussion

The previous parts of the paper have been objective accounts of the literature. Instead, in the remaining parts of this section, we discuss outstanding aspects regarding automatic compliance checking of software processes based on interpretations of the authors.

### 5.1 The use of software process modeling languages

Notably, a software product with desirable guaranteed attributes (e.g., safety, quality, reliability) is the result of several artifacts supplementing each other as well as actors performing on it with specialized techniques and tools in well-defined engineering processes. Consequently, it is essential to be able to describe all such concepts and structures included in a software process, as well as their properties, plus additional descriptive information. This could be the reason for the change of the trend in the last years, where researchers tend to use consolidated process modeling languages, such as SPEM 2.0 and BPMN. Such modeling languages have already defined characteristics, e.g., extensibility and reuse capapbilities. Consequently, new features can be modeled if needed by customizing or extending existing ones, permitting the modeling of more complete software processes that help the process users and auditors to understand what is needed

to be done, who will perform tasks, what resources will be used, and what results will be obtained. A software process model with such characteristics is especially needed for creating software products in the safety-critical context, which is often subject to a certification process. Most of the consolidated process modeling languages already offer tool support, which makes their use even 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

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). 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. However, in the tailoring process, the provision of a justification (called rationale) is a mandatory element aimed at legitimizing 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 be able to provide means that facilitate the description of the mentioned concepts since they are not only necessary but also sufficient to tackle the compliance checking problem of software processes.

In our analysis of the languages used in the primary studies, we found exploitable characteristics. For example, FCL explicitly provides the concepts of obligation, permission, and the rule priority, allowing reasoning with exceptions. Def-L permits to model facts, defeasible rules, and defeaters, providing the opportunity to model and reasoning with contradictory 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 to be deemed compliant. 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 is not provided either in an implicit or explicit form. Such reduced semantics lead to reduced reasoning capabilities, which also decreases the scope of the methods used for compliance checking. An ideal language for formalizing the requirements of normative frameworks could actually be a combination of several mechanisms and well-defined semantics that could work harmoniously to achieve idealized goals: compliance checking of single processes, variability management, agile processes as well as plan-driven, and finally, process planning and execution.

### 5.3 Towards a generic and domain-agnostic method

Most of the approaches aim at seeking compliance at design time. As such, compliance checking is able to demonstrate intentional compliance, i.e., distribution of responsibilities, such that if every actor fulfills its goals, then the compliance is ensured. [135]. However, intentional compliance can only permit, not guarantee, any quality attribute of the process. Non-conformance between process planning and execution can put the software development at risk in realizing the compliance required. Therefore, combinations between compliance of software process plans and follow-ups during process execution should be made sure. 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 is able to tackle, e.g., quality, SPI, safety, cybersecurity. A resulting method could be more attractive to organizations, and industrial applications could be made on a larger scale.

### 5.4 The need for diverse abilities

Converting normative requirements into formal specifications has many benefits. In particular, formal descriptions obligate the person who analyses the norms to see them from a causal perspective that would facilitate their interpretation. Moreover, a formal specification of normative requirements is a description that is precise and (if properly done) complete. These two characteristics may convince practitioners to use formal languages to do compliance checking tasks. However, there is nothing to do if the language used to perform such formalization is too difficult to understand. A key point for introducing any formal language in the industry is the usability aspects. We need to avoid the case of a new person feeling confused and frustrated with such formalisms. In particular, it could be interesting to develop short, straight-forward expressions, which are clear, and at the same time, readable when the complexity (and size) grows.

### 5.5 Increase the level of automation and tool support

It is difficult to guarantee industrial adoption when there is nonexistent or loosely coupled tool support. 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 for easing the creation of rules, i.e., rule editors and process models, since formalizing requirements still needs 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 then compliance may be challenging to

fix. In our opinion, process mining techniques can be included in a framework for facilitate the compliance checking lifecycle, but not as a standalone technique to guarantee compliance.

### 5.6 Going beyond technological dilemmas

Article 22 of the GDPR [141] 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 organizations and individuals' rights. 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. 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.

Assuming that the method for compliance checking and the tool support are correctly designed, good 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 no guarantee of correctness since 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 that there is no requirement poisoning, i.e., 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 papers addressing automatic compliance approaches of software processes. Therefore, we followed strictly the guidelines recommended in [97, 99]. However, there are threats that could undermine the validity of the results obtained in this systematic review. In this section, we address potential threats regarding publication bias (refers to the problem that positive results are more likely to be published than negative results), identification of primary studies (refers to the strategy to collect all possible studies), and data extraction consistency (refers to the strategy to extract all data required to address the review questions).

**Publication bias:** 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 [23, 115]) ensure appropriateness by performing an exhaustive review and assessment. We also pay careful attention to external reviewers' critical comments on an earlier version of this paper. 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:** 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. 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:** 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 was distributed in two tables. The first table contains selection criteria and articles identification (see Table 7). The second table contains 16 columns aimed at recording the information corresponding to the research questions (see Table 9). 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. [13] 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. Ly et al.'s [110] work provides a systematic comparison of existing approaches for monitoring compliance rules over business processes during run-time (compliance is checked during process execution). In the work done by Hashmi et al. [65], the authors evaluate selected frameworks regarding the modeling of different compliance requirements and their link with the business process. In the work done by Hashmi et al [66] and El Kharbili et al [95], the authors present an evaluation of compliance management strategies at different times of the compliance lifecycle, i.e., design-time, run-time, and auditing-time (compliance is checked after the process has been executed). In Hashmi et al.'s work, the author also review how control flow structures and norms are modeled. Like the previous SLRs, our work also found that different kinds of 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, i.e., the definition of roles, work products, guidance, and tools. Only the review presented by Hashmi et al. [66] 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. [16] and, more recently Akhigbe et al. [3], whose focus is surveying the representation of knowledge for legal and regulatory requirements engineering. On the one hand, Boella et al.'s work focuses on norms representation. On the other hand, Akhigbe et al.'s 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. [149] 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 Garcia et al [58] focuses on the identification of software process modeling languages. We do a similar thing, but we also include the models for normative frameworks required for compliance analysis. Finally, in the context of safety-related compliance management, we find Nair et al.'s work [116], whose work 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, which are often mandated by normative frameworks, i.e., 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. In the research arena, we can find several studies, which 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 systematic literature review 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, e.g., 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, especially to implement the inputs of the reasoning process. Tool support is still an issue since 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, we consider it 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 works harmoniously 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 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. Finally, we also mentioned incorporating a trust layer to guarantee that rules are correctly derived from the normative frameworks. This aspect can be reached in the future by using technological means. However, a shorter-term solution could be to 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.

## Acknowledgment

This work is supported by the EU and VINNOVA via the ECSEL JU project AMASS (No. 692474) [5].

## References

1. European Organisation for Civil Aviation Equipment & European Organisation for Civil Aviation Equipment: *RTCA/DO-178C – Software Considerations in Airborne Systems and Equipment Certification* (2011)
2. ISO/IEC JTC 1/SC 7: *ISO/IEC TS 33053 – Information technology — Process assessment — Process Reference Model (PRM) for quality management* (2019), https://www.iso.org/standard/55144.html
3. Akhigbe, O., Amyot, D., Richards, G.: A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory compliance. Requirements Engineering **24**(4), 459–481 (2019), https://doi.org/10.1007/s00766-018-0294-1
4. Alexander, I.F.: A taxonomy of stakeholders: Human roles in system development. International Journal of Technology and Human Interaction (IJTHI) **1**(1), 23–59 (2005)
5. AMASS: Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems. http://www.amass-ecsel.eu/
6. Arcaini, P., Bonfanti, S., Gargantini, A., Mashkoor, A.: Integrating formal methods into medical software development: The ASM approach. Science of Computer Programming **158**, 148–167 (2018)
7. 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. pp. 344–359 (2016)
8. Armour, P.G.: The Laws of Software Process: A New Model for the Production and Management of Software. CRC Press (2003)

9. Automotive SIG: *Automotive SPICE V. 3.0 – Process Assessment/Reference Model* (2015)
10. Bala, S., Cabanillas, C., Haselböck, A., Havur, G., Mendling, J., Polleres, A., Sperl, S., Steyskal, S.: A Framework for Safety-Critical Process Management in Engineering Projects. International Symposium on Data-Driven Process Discovery and Analysis, vol. 1, pp. 1–27 (2017)
11. Barrett, C., Tinelli, C.: Satisfiability Modulo Theories. Handbook of Model Checking pp. 305–335 (2018)
12. Bauer, K., Hinz, O., van der Aalst, W., Weinhardt, C.: Expl(ai)n it to me–explainable ai and information systems research. Business & Information Systems Engineering (2021)
13. 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 Research **5**(2), 221–247 (2012)
14. Biro, M.: Open services for software process compliance engineering. In: International Conference on Current Trends in Theory and Practice of Informatics. pp. 1–6. Springer (2014)
15. Biro, M.: Open services for software process compliance engineering. pp. 1–6. SOFSEM 2014: Theory and Practice of Computer Science, Springer International Publishing, Cham (2014)
16. 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. pp. 14–21. 7th International Workshop on Requirements Engineering and Law (2014)
17. 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. pp. 89–103. Springer (2019)
18. Bonatti, P.: Fast Compliance Checking in an OWL2 Fragment. In: 27th International Joint Conferences on Artificial Intelligence Organization. pp. 1746–1752 (2018)
19. Börger, E., Stark, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag, New York, Inc. (2003). https://doi.org/10.1007/978-3-642-18216-7
20. Bramberger, R., Martin, H., Gallina, B., Schmittner, C.: Co-engineering of safety and security life cycles for engineering of automotive systems. ACM SIGAda Ada Letters **39**(2), 41–48 (2020)
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 (2010)
22. Bundesamt für Wehrtechnik und Beschaffung (BWB): *General Directive 250: Software Development Standard for the German Federal Armed Forces, V-model, Software Lifecycle Process Model* (1992)
23. Carlan, C., Gallina, B., Soima, L.: Safety case maintenance: A systematic literature review. In: 40th International Conference on Computer Safety, Reliability and Security (2021)
24. Carroll, N., Richardson, I.: Software-as-a-medical device: demystifying connected health regulations. Journal of Systems and Information Technology (2016)

25. Castellanos Ardila, J., Gallina, B., Ul Muram, F.: Facilitating Automated Compliance Checking of Processes in the Safety-critical Context. Electronic Communications of the EASST **078**, 1–20 (2019)
26. Castellanos Ardila, J.P., Gallina, B.: Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards. 7th IEEE International Workshop on Software Certification (2017)
27. Castellanos Ardila, J.P., Gallina, B., Ul Muram, F.: Enabling Compliance Checking against Safety Standards from SPEM 2.0 Process Models. pp. 45 – 49. Euromicro Conference on Software Engineering and Advanced Applications (2018)
28. Castellanos Ardila, J.P., Gallina, B., UL Muram, F.: Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance. 18th International SPICE Conference (2018)
29. Castellanos Ardila, J., Gallina, B.: Towards increased efficiency and confidence in process compliance. The 24th EuroAsiaSPI Conference, vol. 748 (2017)
30. Castellanos Ardila, J.P., Gallina, B.: Separation of concerns in process compliance checking: Divide-and-conquer. pp. 135–147. European Conference on Software Process Improvement, Springer (2020)
31. Castellanos Ardila, J.P., Gallina, B.: Reusing (safety-oriented) compliance artifacts while recertifying. pp. 53–64. 9th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,, INSTICC, SciTePress (2021). https://doi.org/10.5220/0010224900530064
32. Castellanos Ardila, J.P., Gallina, B., Governatori, G.: Compliance-aware engineering process plans: The case of space software engineering processes. Artificial intelligence and law (2021)
33. Cha, S., Taylor, R.N., Kang, K.: Handbook of software engineering. Springer (2019)
34. Cheung, L., Chung, P., Dawson, R.: Managing process compliance. Information management: Support systems and multimedia technology pp. 48–62 (2003)
35. Chung, P., Cheung, L., Machin, C.: Compliance Flow-Managing the Compliance of Dynamic and Complex Processes. Knowledge-Based Systems **21**(4), 332–354 (2008)
36. Clarke, L.A., Osterweil, L.J., Avrunin, G.S.: Supporting human-intensive systems. pp. 87–92. FSE/SDP workshop on Future of software engineering research (2010)
37. Clarke, P., LepmetsF, M., McCaffery, F., Finnegan, A., Dorling, A., Flood, D.: Mdevspice - a comprehensive solution for manufacturers and assessors of safety-critical medical device software. pp. 274–278. Software Process Improvement and Capability Determination, Springer International Publishing, Cham (2014)
38. Cohen, D.: AP5 Reference Manual. https://ap5.com/doc/ap5-man.html (2019)
39. Colmerauer, A.: An Introduction to Prolog III. Computational Logic pp. 37–79 (1990)
40. Cooper, H.M.: Organizing knowledge syntheses: A taxonomy of literature reviews. Knowledge in society **1**(1), 104–126 (1988)
41. Cugola, G., Ghezzi, C.: Software processes: a retrospective and a path to the future. Software Process: Improvement and Practice **4**(3), 101–123 (1998)
42. Cusumano, M.A.: Who is liable for bugs and security flaws in software? Communications of the ACM **47**(3), 25–27 (2004)
43. Daoudagh, S., Marchetti, E.: A life cycle for authorization systems development in the gdpr perspective. pp. 128–140. ITASEC (2020)
44. De La Vara, J.L., Marín, B., Ayora, C., Giachetti, G.: An empirical evaluation of the use of models to improve the understanding of safety compliance needs. Information and Software Technology **126**, 106351 (2020)

45. De La Vara, J.L., Ruiz, A., Attwood, K., Espinoza, H., Panesar-Walawege, R.K., López, Á., Del Río, I., Kelly, T.: Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel. Information and Software Technology **72**(C), 16–30 (2016)
46. Denyer, D., Tranfield, D., Van Aken, J.E.: Developing design propositions through research synthesis. Organization studies **29**(3), 393–413 (2008)
47. Diebold, P., Scherr, S.: Software process models vs descriptions: What do practitioners use and need? Journal of Software: Evolution and Process **29**(11), 1–13 (2017)
48. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Checking Traces for Regulatory Conformance. pp. 86–103. Proceedings of the International Workshop on Runtime Verification (2008)
49. Emmerich, W., Finkelstein, A., Montangero, C., Antonelli, S., Armitage, S., Stevens, R.: Managing Standards Compliance. IEEE Transactions on Software Engineering **25**(6), 836–851 (1999)
50. European Committee for Electrotechnical Standardization: *CENELEC - EN 50128. Railway Applications-Communication, Signaling and Processing Systems Software for Railway Control and Protection Systems* (2011)
51. European Committee for Electrotechnical Standardization: *CENELEC - EN 50126. Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)* (2017)
52. European Parliament and Council of the European Union: *General Data Protection Regulation (GDPR)* (2016)
53. European Space Agency: *ECSS-E-ST-40C – Space Engineering Software* (2009), https://ecss.nl/standard/ecss-e-st-40c-software-general-requirements/
54. Fernandes, J.M., Duarte, F.J.: A reference framework for process-oriented software development organizations. Software & Systems Modeling **4**(1), 94–105 (2005)
55. Fitzgerald, B., Stol, K.J., O'Sullivan, R., O'Brien, D.: Scaling Agile Methods to Regulated Environments: An Industry Case Study. pp. 863–872. 35th International Conference on Software Engineering (ICSE), IEEE Computer Society (2013). https://doi.org/10.1109/ICSE.2013.6606635
56. Fuggetta, A.: Software process: a roadmap. pp. 25–34. Conference on the Future of Software Engineering, Orlando, Florida (2000)
57. Gallina, B., Ul Muram, F., Castellanos Ardila, J.: Compliance of Agilized (Software) Development Processes with Safety Standards: a Vision. pp. 1–6. 4th International Workshop on Agile Development of Safety-Critical Software (2018)
58. García-Borgoñon, L., Barcelona, M.A., García-García, J.A., Alba, M., Escalona, M.J.: Software process modeling languages: A systematic literature review. Information and Software Technology **56**(2), 103–116 (2014)
59. Generowicz, M.: The Easy Path to Functional Safety Compliance. pp. 1–3. I&E Systems Pty Ltda (2013), https://www.iesystems.com.au/wp-content/uploads/2015/04/Duty-of-Care-Article.pdf, Accessed March 30, 2021
60. Golra, F.R., Dagnat, F., Bendraou, R., Beugnard, A.: Continuous process compliance using model driven engineering. In: International Conference on Model and Data Engineering. pp. 42–56. Springer (2017)
61. Goverment of Canada: *PIPEDA – Personal Information Protection and Electronic Documents Act* (2000)
62. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems **14**(02n03), 181–216 (2005)

63. 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. pp. 247–254 (2017)
64. Harju, H., Lahtinen, J., Ranta, J., Nevalainen, R., Johansson, M.: Software safety standards for the basis of certification in the nuclear domain. pp. 54–62. 7th International Conference on the Quality of Information and Communications Technology (2010)
65. Hashmi, M., Governatori, G.: A methodological evaluation of business process compliance management frameworks. Proceedings of the Asia-Pacific Conference on Business Process Management pp. 106–115 (2013)
66. Hashmi, M., Governatori, G., Lam, H., Wynn, M.: Are we done with business process compliance: state of the art and challenges ahead. Knowledge and Information Systems **57**(1), 79–133 (2018)
67. Hassan, W., Logrippo, L.: Towards a process for legally compliant software. In: 2013 6th International Workshop on Requirements Engineering and Law (RELAW). pp. 44–52. IEEE (2013)
68. He, X., Guo, J., Wang, Y., Guo, Y.: An automatic compliance checking approach for software processes. pp. 467–474. Asia-Pacific Software Engineering Conference (2009)
69. Henderson, P.: Software processes are business processes too. pp. 181–182. 3rd International Conference on the Software Process. Applying the Software Process, IEEE (1994)
70. Hewett, R., Kijsanayothin, P., Bak, S., Galbrei, M.: Cybersecurity policy verification with declarative programming. Applied Intelligence **45**(1), 83–95 (2016)
71. Icheku, V.: Understanding ethics and ethical decision-making. Xlibris Corporation (2011)
72. Ingolfo, S., Siena, A., Mylopoulos, J.: Establishing regulatory compliance for software requirements. pp. 47–61. International Conference on Conceptual Modeling (2011)
73. Internation Organization for Standardization: *ISO/IEC 90003:2004-Software engineering – Guidelines for the application of ISO 9001:2000 to computer software* (2004)
74. Internation Organization for Standardization: *ISO 14971:2019 – Application of risk management to medical devices* (Dec 2019)
75. Internation Organization for Standardization - Technical Committee 210: *IEC 62304- Medical device software — Software life cycle processes* (2006)
76. International Electrotechnical Commission: *IEC 61508– Functional safety of electric/electronic/programmable electronic safety-related systems* (1998)
77. International Organization for Standardization: *ISO 9001-3– Quality Management and Quality Assurance Standards - Part 3* (1991)
78. International Organization for Standardization: *ISO 9000– Quality Management Systems-Fundamentals and Vocabulary* (2005)
79. International Organization for Standardization: *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)
80. International Organization for Standardization: *ISO/IEC 15504 – Information technology - Process assessment* (Jun 2013)
81. International Organization for Standardization - Technical Committe: ISO/IEC joint technical committee JTC 1: *ISO/IEC 27000– Information Technology* (2018)

82. International Organization for Standardization - Technical Committe: ISO/IEC JTC 1/SC 7: *ISO/IEC 330XX – Information technology - Process assessment – Concepts and Terminology.* (2015)
83. International Organization for Standardization - Technical Committe: ISO/TC 22/SC 32: *ISO 26262: Road Vehicles Functional Safety* (2018)
84. International Organization for Standardization/International Electrotechnical Commission: *ISO/IEC/IEEE 12207– Systems and software engineering — Software life cycle processes* (2017)
85. Jääskinen, N.: Better regulation programs: Some critical remarks. In: Changing Forms of Legal and Non-Legal Institutions and New Challenges for the Legislator. pp. 29–33. International Conference on Legislative Studies in Helsinki (2008)
86. Javed, M., Gallina, B.: Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool. pp. 23–28. 22nd International Systems and Software Product Line Conference (2018)
87. Jost, H., Hahn, A., Häusler, S., Köhler, S., Gačnik, J., Köster, F., Lemmer, K.: Supporting qualification: Safety standard compliant process planning and monitoring. pp. 1–6. Symposium on Product Compliance Engineering (2010)
88. Kabaale, E., Wen, L., Wang, Z., Rout, T.: Representing Software Process in Description Logics: An Ontology Approach for Software Process Reasoning and Verification. pp. 362–376. Software Process Improvement and Capability Determination Conference, Springer (2016)
89. Kabaale, E., Wen, L., Wang, Z., Rout, T.: An Axiom Based Metamodel for Software Process Formalisation : An Ontology Approach. International Conference on Software Process Improvement and Capability Determination, vol. 2, pp. 226–240 (2017)
90. Kabaale, E., Wen, L., Wang, Z., Rout, T.: Ensuring Conformance to Process Standards Through Formal Verification. International Conference on Software Process Improvement and Capability Determination, vol. 2, pp. 248–262. Springer International Publishing (2018)
91. Kabaale, E., Wen, L., Wang, Z., Rout, T.: Formalising Process Assessment and Capability Determination : An Ontology Approach. European Conference on Software Process Improvement, vol. 2, pp. 594–605. Springer International Publishing (2019)
92. Kemp, R.: Regulating the safety of autonomous vehicles using artificial intelligence. Communications Law **24**(1), 24–33 (2019)
93. Kerrigan, S., Law, K.H.: Logic-based regulation compliance-assistance. pp. 126–135. Proceedings of the 9th international conference on Artificial intelligence and law (2003)
94. Khan, A.A., 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. Information and Software Technology **87**, 180–205 (2017)
95. Kharbili, M.e., Medeiros, A.K.A.d., Stein, S., van der Aalst, W.M.: Business process compliance checking: Current state and future challenges. Modellierung betrieblicher Informationssysteme (MobIS 2008) (2008)
96. Khelladi, D.E., Bendraou, R., Baarir, S., Laurent, Y., Gervais, M.P.: A framework to formally verify conformance of a software process to a software method. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing. pp. 1518–1525 (2015)
97. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature reviews in Software Engineering. Tech. Rep. 4ve (2007)

98. Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering–a systematic literature review. Information and software technology **51**(1), 7–15 (2009)
99. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering. Information and software technology **55**(12), 2049–2075 (2013)
100. Kneuper, R.: Software Processes and Life Cycle Models. An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes. Springer, Cham (2018). https://doi.org/https://doi-org.ep.bib.mdh.se/10.1007/978-3-319-98845-0, iSBN 978-3-319-98845-0
101. Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Garousi, V., Felderer, M., Trektere, K., McCaffery, F., Linssen, O., Hanser, E., et al.: Hybrid software and system development in practice: waterfall, scrum, and beyond. pp. 30–39. International Conference on Software and System Process (2017)
102. Kuhrmann, M., Diebold, P., Munch, J., Tell, P., Trektere, K., McCaffery, F., Garousi, V., Felderer, M., Linssen, O., Hanser, E., et al.: Hybrid software development approaches in practice: a european perspective. IEEE Software **36**(4), 20–31 (2018)
103. Kuhrmann, M., Konopka, C., Nellemann, P., Diebold, P., Münch, J.: Software process improvement: where is the evidence?: initial findings from a systematic mapping study. pp. 107–116. International Conference on Software and System Process (2015)
104. Ladkin, P.B.: Duty of care and engineering functional-safety standards. Digital Evidence & Elec. Signature L. Rev. **16**, 51 (2019)
105. Leveson, N.: Safety : Why, What, and How. ACM Computing Surveys (CSUR) **18**(2), 125–163 (1986)
106. Leveson, N.G.: The use of safety cases in certification and regulation. Tech. rep., Massachusetts Institute of Technology. Engineering Systems Division (2011)
107. Lifschitz, V.: What is answer set programming. AAAI, vol. 8, pp. 1594–7 (2008)
108. Lonchamp, J.: A structured conceptual and terminological framework for software process engineering. pp. 41–53. 2nd International Conference on the Software Process-Continuous Software Process Improvement, IEEE (1993)
109. Lúcio, L., Rahman, S., Cheng, C.H., Mavin, A.: Just formal enough? automated analysis of ears requirements. pp. 427–434. NASA Formal Methods Symposium, Springer (2017)
110. Ly, L., Maggi, F., Montali, M., Rinderle-Ma, S., Van Der Aalst, W.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Information Systems **54**, 209–234 (2015)
111. 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. pp. 75–92 (2012)
112. Marsden, J., Windisch, A., Mayo, R., Grossi, J., Villermin, J., Fabre, L., Aventini, C.: Ed-12c/do-178c vs. agile manifesto: A solution to agile development of certifiable avionics. In: 9th European Congress Embedded Real Time Software and Systems (ERTS) (2018)
113. Mayr-Dorn, C., Vierhauser, M., Bichler, S., Keplinger, F., Cleland-Huang, J., Egyed, A., Mehofer, T.: Supporting quality assurance with automated process-centric quality constraints checking. IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (2021)

114. Munoz-Gama, J.: Conformance Checking and its Challenges. Conformance Checking and Diagnosis in Process Mining Comparing Observed and Modeled Processes pp. 11–18 (2016)
115. Muram, F.u., Tran, H., Zdun, U.: Systematic review of software behavioral model consistency checking. ACM Computing Surveys (CSUR) **50**(2), 1–39 (2017)
116. Nair, S., De La Vara, J., Sabetzadeh, M., Briand, L.: An extended systematic literature review on provision of evidence for safety certification. Information and Software Technology **56**(7), 689–717 (2014)
117. Osterweil, L.: Software processes are software too. In: 9th international conference on Software Engineering (ICSE 1987). IEE (1987)
118. Osterweil, L.J.: Formalisms to support the definition of processes. Journal of Computer Science and Technology **24**(2), 198–211 (2009)
119. 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. pp. 30–39 (2011)
120. 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. Information and Software Technology **55**(5), 836–864 (2013), http://dx.doi.org/10.1016/j.infsof.2012.11.009
121. Parnas, D.L., Clements, P.C.: A rational design process: How and why to fake it. IEEE transactions on software engineering (2), 251–257 (1986)
122. Pnueli, A.: The temporal logic of programs. pp. 46–57. 18th Annual Symposium on Foundations of Computer Science, IEEE (1977)
123. 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. pp. 197–210 (2017)
124. Proença, D., Borbinha, J.: Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3 – Enabling automatic inference of maturity and capability levels. Computer Standards and Interfaces (2018)
125. Rabbi, F., Wang, H., MacCaull, W.: Compensable workflow nets. pp. 122–137. International Conference on Formal Engineering Methods, Springer (2010)
126. Rahim, M.M., Idowu, S.O.: Social Audit Regulation: Development, Challenges and Opportunities. Springer (2015)
127. Ramasubbu, N., Bharadwaj, A., Tayi, G.K.: Software process diversity: Conceptualization, measurement, and analysis of impact on project performance. MIS Quarterly **39**(4), 787–808 (2015), https://www.jstor.org/stable/26628652
128. Ranise, S., Siswantoro, H.: Automated Legal Compliance Checking by Security Policy Analysis. In: International Conference on Computer Safety, Reliability, and Security. pp. 361–372 (2017)
129. Regan, G., Biro, M., Mc Caffery, F., Mc Daid, K., Flood, D.: A traceability process assessment model for the medical device domain. pp. 206–216. European Conference on Software Process Improvement, Springer (2014)
130. Rodriguez, D., Garcia, E., Sanchez, S., Nuzzi, C.R.S.: Defining software process model constraints with rules using owl and swrl. International Journal of Software Engineering and Knowledge Engineering **20**(04), 533–548 (2010)
131. SAE International: *SAE J3061 – Cybersecurity Guidebook for Cyber-Physical Vehicle Systems* (2016)
132. Scacchi, W.: Business processes can be software too: some initial lessons learned. pp. 183–184. 3rd International Conference on the Software Process. Applying the Software Process, IEEE Computer Society (1994)

133. Schieferdecker, I.: Responsible software engineering. In: The Future of Software Quality Assurance, pp. 137–146. Springer, Cham (2020)
134. Schwartz, A.: Statutory interpretation, capture, and tort law: The regulatory compliance defense. American Law and Economics Review **2**(1), 1–57 (2000)
135. Siena, A., Mylopoulos, J., Perini, A., Susi, A.: From Laws to Requirements. In: Requirements Engineering and Law. pp. 6–10 (2008)
136. Software Engineering Institute - Carnegie Mellon University: *CMMI for Development Version 1.3– Capability Maturity Model Integration* (2011)
137. Stålhane, T., Myklebust, T., Hanssen, G.: The application of safe scrum to IEC 61508 certifiable software. In: 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference. vol. 8, pp. 6052–6061 (2012)
138. Stallinger, F., Henderson-Sellers, B., Torgersson, J.: The oospice assessment component: Customizing. Business Component-Based Software Engineering **705**, 119 (2012)
139. Tarr, P.L., Wolf, A.L.: Introduction to "engineering of software: The continuing contributions of leon j. osterweil". Engineering of Software, Springer, Berlin, Heidelberg
140. The European Parlament and the Council of the European Union: *EU DPD – European Data Protection Directive* (1995)
141. The European Parliament and the Council of the European Union: *GDPR – General Data Protection Regulation* (2016)
142. Tonini, A.C., Mesquita Spinola, M.D., Barbin Laurindo, F.J.: Six sigma and software development process: Dmaic improvements. Technology Management for the Global Future - PICMET 2006 Conference, vol. 6, pp. 2815–2823 (2006)
143. Torre, D., Soltana, G., Sabetzadeh, M., Briand, L., Auffinger, Y., Goes, P.: Using Models to Enable Compliance Checking against the GDPR : An Experience Report. 22nd International Conference on Model Driven Engineering Languages and Systems pp. 1–11 (2019)
144. 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. pp. 385–401. Springer (2020)
145. Vakkuri, V., Jantunen, M., Halme, E., Kemell, K.K., Nguyen-Duc, A., Mikkonen, T., Abrahamsson, P.: Time for ai (ethics) maturity model is now. arXiv preprint arXiv:2101.12701 (2021)
146. Valle, A.M., Santos, E.A., Loures, E.R.: Applying process mining techniques in software process appraisals. Information and software technology **87**, 19–31 (2017)
147. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data & Knowledge Engineering **68**(9), 793–818 (2009), http://dx.doi.org/10.1016/j.datak.2009.02.015
148. Vilkomir, S., Bowen, J., Ghose, A.: Formalization and assessment of regulatory requirements for safety-critical software. Innovations in Systems and Software Engineering **2**(3-4), 165–178 (2006)
149. von Wangenheim, C.G., Hauck, J.C.R., Salviano, C.F., von Wangenheim, A.: Systematic literature review of software process capability/maturity models. International Conference on Software Process Improvement and Capabity Determination (SPICE), Pisa, Italy (2010)
150. Welzel, D., Walter, H., Schmidt, W.: Tailoring and conformance testing of software processes: the ProcePT approach. pp. 41–49. Software Engineering Standards Symposium (1995)

151. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. pp. 1–10. 18th International Conference on Evaluation and Assessment in Software Engineering (2014)
152. Yan, M., Xia, X., Zhang, X., Xu, L., Yang, D., Li, S.: Software quality assessment model: A systematic mapping study. Science China Information Sciences **62**(9), 1–18 (2019)
153. Zhang, H., Ali-Babar, M., Tell, P.: Identifying relevant studies in software engineering. Information and Software Technology **53**(6), 625–637 (2011)