Separation of Concerns in Process Compliance Checking: Divide-and-Conquer

Julieth Patricia Castellanos Ardila and Barbara Gallina

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

Abstract. Compliance with multiple standard's reference models has the potential to improve process quality but is a challenging task faced by manufacturers in the safety-critical context. To facilitate this task, we propose a method for automated process compliance checking that can be used as a basis for decision making. Our method requires users to create a knowledge base of formalized requirements and processes checkable for compliance. In this paper, we exploit the natural separation of concerns in the state of practice to offer adequate means to facilitate the creation of the required concepts by using a divide and conquer strategy. For this, we discuss the impact of process factors in compliance assessment and provide separation of concerns based on SPEM 2.0 (Systems and Software Process Engineering Metamodel). Then, we illustrate the defined concerns and discuss our findings.

Keywords: Software Process Compliance Checking · Safety Standards · SPI · SPEM 2.0 · Separation of Concerns · Divide and Conquer.

1 Introduction

In the safety-critical context, standards 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. With the growing software development complexity, there is a need to adequately allocated such resources during the software development lifecycle [25]. However, this task becomes difficult due to software process diversity, i.e., the simultaneous use of multiple reference models within a single project [18]. To tackle this situation, organizations produce generic software process baselines. In the analysis of these baselines, gaps to best practices could be discovered [5], and potential improvements, based on standard's information, can be performed [4]. Thus, part of the software process improvement effort required in the safety-critical context is expended in process-based compliance.

A high level of investment in process-based compliance could result in an improvement in productivity and quality, especially when there is process diversity [18]. Process-based compliance could be supported by checking that the process used to engineer safety-critical systems fulfill the properties set down by standards at given points. The resulting compliance checking reports can be

used not only to demonstrate to auditors that process plans fulfill the prescribed requirements [10], but also to discover essential improvement aspects. Thus, in previous work [1, 2], we presented a method for automated compliance checking of processes plans. Our method requires users to: 1) model a formal specification of the standard requirements by using FCL (Formal Contract Logic) [11] and 2) model a specification of the process plans that are checkable for compliance, i.e., processes augmented with compliance information, by using SPEM 2.0 (Systems and Software Process Engineering Metamodel) [16]. Thus, an essential step of our method is dedicated to creating well-formed specifications.

In this paper, we aim at facilitating the creation of the specifications required for automated compliance checking. Given the natural separation of concerns in the state of practice, we try to offer adequate means to support the separated concepts based on process structure and different standards, by using a divide-and-conquer strategy. For this, we discuss the impact of process factors in compliance assessment and justify the separation of concerns based on SPEM 2.0 concepts. SPEM 2.0 is a well-defined metamodel that not only permits the modeling of software processes but also the customization of elements to provide standards-related information. Then, we illustrate the use of the defined concerns with the requirements prescribed in the railway sector. Finally, we discuss the potential benefits and implications of our work.

The paper is organized as follows. We present essential background in Section 2. We discuss the separation of concerns within the regulatory space in Section 3. We illustrate the defined concerns in Section 4. We discuss our findings in Section 5. We present related work in Section 6. Finally, we conclude the work and outline future work in Section 7.

2 Background

This section presents essential background required in this paper.

2.1 Standards in the Safety-critical Context

Compliance with safety standards typically involves the provision of evidence regarding process plans since standards reference frameworks contain requirements that prescribe artifacts related to the planning of activities [21]. In particular, process reference models describe a set of tasks to be performed during the development of safety-critical systems. For example, ISO 26262 [14], which is the standard that applies in automotive, proposes a V-model, in which the activities related to the development of the software are contrasted with the ones related to verification and validation. The standard DO-178C [19] describes a set of objectives that implicitly define a lifecycle model. ECSS-E-ST-40C [7], which applies in space software projects, focuses on the definition of software development phases and their characteristics. In all the standards, the detailed breakdown of the work can be inferred from the requirements. Moreover, process-related standards commonly have sections in which they describe the necessary inputs

and the mandatory outputs of the safety lifecycle phases. The qualification of personnel may vary from one standard to the other. ISO 26262 mentions the importance of qualified personnel, but it leaves the decision to the company, which should have a minimum set of internal requirements in that matter. In ECSS E-ST-40C, the degree of independence between developers and reviewers is highlighted. In DO-178C, specific roles are defined for specific phases in the lifecycle. Similarly, tool qualification is required in the safety-critical context. In ECSS-E-ST-40C, supporting tools are a customer/supplier agreement that shall be documented in the plan. A specific standard annex, called DO-330 [20], defines that for Avionics, the tool qualification is in itself a process necessary to obtain qualification credit. For ISO 26262, evidence regarding the tool suitability for specific uses should be shown. All the standards prescribe methods and techniques that should be used to perform specific tasks in the form of guidance.

2.2 CENELEC EN 50128

CENELEC EN 50128 [3], which is the standard that focuses on software aspects regarding control and protection applications in railways, prescribes requirements that target the different elements described in Section 2.1. In Table 1, we recall a set of requirements that apply to the Architecture and Design Phase.

Table 1: CENELEC EN 50128-Architecture and Design Phase [3]

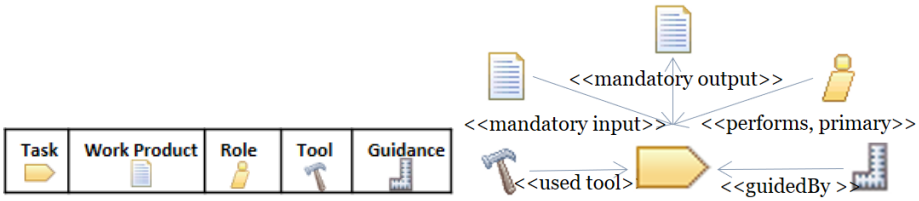
Element	**Description**
Inputs	Software Requirements Specification (SRS).
Outputs	Software Architecture Specification (SAS), Software Design Specification (SDS), Software Interface Specifications (SIS), Software Integration Test Specification (SITS), Software/Hardware Integration Test Specification (SHITS), Software Architecture and Design Verification Report (SADVR).
Tasks	1) Software Architecture Specification, 2) Software Interface Specification, 3) Software Design Specification, 4) Selection/Creation Coding Standards, 5) Software Integration Test Specification, 6) Software/Hardware Integration Test Specification, 7) Software Architecture and Design Verification Report
Roles	Designer for task 1), 2), 3) and 4). Integrator for tasks 5) and 6). Verifier for task 7). The designer shall be competent in safety design principles and EN 50128.
Tools	Suitable tools with a certificate of validation (e.g., Matlab and MS Word)
Guidelines	Guidance for the Software Architecture Specification task (req-7-3-4-5), guidance for SAS (req-7-3-4-10), guidance for the SIS (req-7-3-4-19), guidance for SDS (req-7-3-4-23), guidance for the selection/creating coding standards (req-7-3-4-25), guidance for the design method selection (req-7-3-4-28), guidance for the software integration test specification task (req-7-3-4-31), guidance for the software/hardware integration test specification (req-7-3-4-36), guidance for SHITS (req-7-3-4-37), guidance for the Software Architecture and Design verification report (req-7-3-4-42)

CENELEC EN 50128 also refers to quality management and continuous improvement of the systems within the organizations. Companies may have quality assurance mechanisms that conform to different frameworks such as Software Process Improvement and Capability Determination (SPICE), also known as ISO/IEC 15504. In particular, part 5 [13] provides processes that serve primary parties during the lifecycle of software. We select the process outcome *database*

design, as an example. Process outcomes are essential for determining the result of the execution of the process.

2.3 Software Processes and SPEM 2.0

A software process [8] provides a broad and comprehensive concept to frame and organize the different tasks required during the development of software. To ensure understanding, documentation, and exchange of process specifications, technological support is required [9]. In particular, SPEM 2.0 (Systems and Software Process Engineering Metamodel) [16] is a software process modeling language that provides the capability of modeling method content independently from their use in processes. Method content describes different process elements as presented in Fig. 1a. Such elements are related to each other, as presented in Fig. 1b. EPF (Eclipse Process Framework) Composer [6], which was recently migrated from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3. [15], provides the environment for modeling SPEM 2.0-like process models.



(a) SPEM 2.0 Elements (b) Elements Relationships

Fig. 1: Content Elements Definitions in SPEM 2.0 [16].

2.4 Formal Contract Logic

Formal Contract Logic (FCL) [11] is a logic that supports the modeling of norms representing obligations and permissions in a normative context that can be defeated by evolving knowledge. Thus, FCL is classified as a defeasible deontic logic. In FCL, a rule has the form: r: $a_1, ..., a_n \Rightarrow c$, where r is the rule identifier, $a_1, ..., a_n$ are the propositions that represent the conditions of the applicability of the norm, and c is the concluding proposition that contains normative effects.

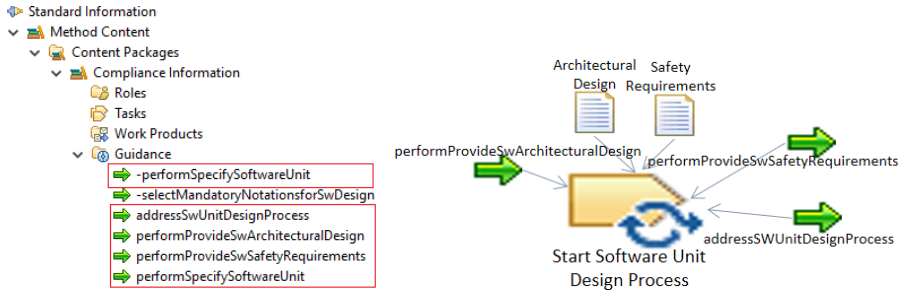
2.5 Automatic Compliance Checking Method

Our method for automated compliance checking of processes plans [1], requires users to model processes with SPEM 2.0 (recalled in Section 2.3) and formalize standards requirements with FCL (recalled in Section 2.4). Rules in FCL are composed of propositions that correspond to the properties described in the requirements of the standard. Such properties can be annotated to the process tasks that fulfill them. Annotations reflect not only the state of the task but also the effects such task produces on subsequent tasks. For this reason, FCL propositions describe compliance effects, which annotated on process models permit

the derivation of process models checkable for compliance (compliance state representation of such processes that permits automatic reasoning). We explain the modeling part of our method with an example from ISO 26262 presented in [1]. The modeled requirement is obtained from part 6 clause 8, number 8.1, which states: *"Specify software units in accordance with the architectural design and the associated safety requirements"*. The FCL representation of this requirement is presented in Equation 1.

$$\begin{aligned} r2.1 : addressSwUnitDesignProcess \Rightarrow [O] - performSpecifySwUnit \\ r2.2 : performProvideSwArchitecturalDesign, performProvideSwSafetyRequirements \\ \Rightarrow [P] performSpecifySwUnit \\ r2.2 > r2.1 \end{aligned} \tag{1}$$

To create the process models checkable for compliance, we fist need to model the compliance effects described in the propositions of the rules. For example, the rules on Equation 1 contains five propositions, namely addressSWUnitDesignProcess, -performSpecifySwUnit, performProvideSwArchitecturalDesign, performProvideSwSafetyRequirements and performSpecifySwUnit, which are presented in Fig. 2a. Then, we need to assign such compliance effects to the tasks that fulfill them. For example, the task Start software Unit Design Process indicates that the process is performed and has two inputs. Therefore the annotated compliance effects are addressSwUnitDesignProcess, performProvideSwArchitecturalDesign and performProvideSwSafetyRequirements (see Fig. 2b). The reader can discover more details about the previous modeling in [1].



(a) Compliance Effects. (b) Annotated Task.

Fig. 2: Method for Automatic Compliance Checking: The Modeling Part.

2.6 Separation of Concerns: Divide-and-conquer Strategy

The Romans had a strategy called divide-and-conquer, which considers that one power breaks another power into more manageable pieces to easier take control. In software engineering, this strategy is adopted as a principle to manage complexity [23]. Particularly, divide-and-conquer is seen in the principle of separation of concerns [24], which refers to the ability to separate and organize only those parts (or properties) of a system that are relevant to a particular concept or to a particular goal. A concern may be defined as a functional notion or more broadly as any piece of interest or focus.

3 Separation of Concerns within the Regulatory Space

The relationship between the requirements imposed by safety standards (recalled in Section 2.1) and the targeted software processes (recalled in Section 2.3) is complex. The reason is that a single requirement may be impacting one element in the process, causing effects to several elements. Moreover, each element in a process may be impacted by several requirements. In addition, software process diversity, as recalled in the introductory part, may lead to problems in the understanding of what is needed for managing the compliance. Thus, we have a compact set of requirements, which we need to manage appropriately. By applying the divide-and-conquer strategy, we could break down such complexity and provide a better view of the requirements.

Separation of concerns (recalled in Section 2.6) applied to the regulatory space could be oriented to the process-specific factors. In particular, the aspects that requirements regulate are the tasks, their specific order, the mandatory input and outputs of the tasks, the personnel performing the tasks, the tools as well as the recommended techniques that should be used to do the tasks. Thus, the concept of a task is central, to which properties such as the definition of roles, inputs, outputs, tools, and techniques must apply.

However, requirements not only define the properties of the tasks. For example, roles and tools should have a qualification. This kind of requirements does not directly affect the tasks. They directly affect other elements, which in turn have effects on tasks. Thus, a process can be deemed compliant if we can demonstrate that the process contains the permitted tasks, such tasks have associated the prescribed roles, inputs, outputs, tools, and techniques, and if the associated elements have associated their related properties. With such consideration, dividing requirements in terms of the elements they target, as well as the specific properties defined for each element, seems to be the natural way in which concerns should be separated.

According to SPEM 2.0 (recalled in Section 2.3), a task is performed by a role, requires inputs and provides outputs, is guided by guidance, and a tool is used (see Fig. 1b). Thus, the tasks are the central elements, to which the other elements are allocated. Our method for compliance checking (recalled in Section 2.5), requires that all the properties defined by the requirements of the standard are also allocated (or annotated) to the tasks included in the process plan since such annotations describe the permitted compliance states of the tasks. An abstraction of such a concept can be seen in Fig. 2b. However, not only tasks provide compliance effects to the overall process. As we previously concluded, elements different from tasks too.

Thus, we propose a new abstraction of model annotation, in which tasks will no longer be the placeholder of the compliance effects caused by the process elements ascribed to them. Instead, every element will carry out its own responsibility in terms of compliance information (see Fig. 3). The novelty of the approach is threefold. First, we free

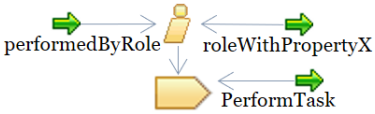


Fig. 3: Annotated Role.

the tasks from unnecessary annotations. Second, annotations on shared process elements should be done only once in a process model. Third, annotated elements have the potential to be reused in other processes and easily re-checked.

To facilitate the creation of compliance effects, which later can be used to form the propositions of the rules in FCL (recalled in section 2.4), we propose two aspects. The first aspect is the definition of icons, which includes the description of the targeted elements, as presented in Table 2. The second aspect is the definition of templates that facilitate the creation of compliance effects, as presented in Table 3. Both, icons and templates are based on the concepts described in SPEM 2.0 in Fig. 1.

Table 2: Icons Describing Specific Compliance Effects.

Role		Work Product		Guidance		Tool		Task
Definition	Property	Definition	Property	Definition	Property	Definition	Property	
								

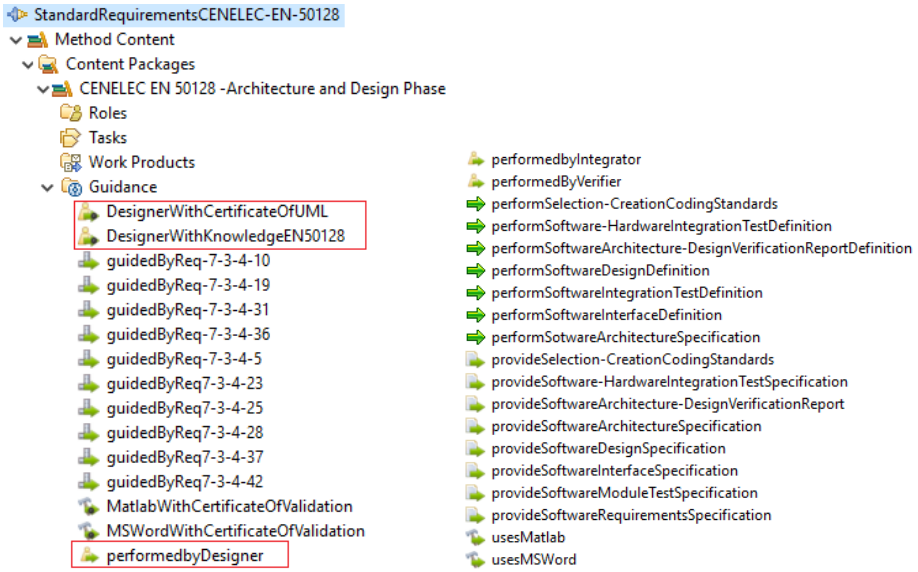
Table 3: Compliance Effects Targeting Differentiated Process Elements

Element target	**Definitional propositions**	**Property-based Propositions**
In/Output elements	provide{*Element*}	{*Element*}with{*Property*}
Roles	performedBy{*Role*}	{*Role*}with{*Property*}
Tools	used{*Tool*}	{*Tool*}with{*Property*}
Guidelines	guidedBy{*Guidance*}	{*Guidance*}with{*Property*}
Tasks	perform{*Task*}	

4 Illustrative Example

We illustrate the separation of concerns in the regulatory space by taking into account the requirements for the architecture and design phase proposed by CENELEC EN 50128 (see Section 2.2). Initially, we need to classify the requirements in terms of the process elements they target. This operation is already presented in Table 1. From this division, we can describe the definitional and property-based propositions derived from these requirements by using the propositions templates shown in Table 3, and the icons described in Table 2. Then, we model them as SPEM 2.0-like elements in the guidance part of EPF Composer. Fig. 4 presents the instantiation of the templates with the predefined icons. For example, the designer should be defined (definitional proposition), and the designer should have competence in safety design and EN 50128 (two property-oriented propositions). The previous propositions are highlighted in red in Fig. 4. A similar analysis is done with all the requirements.

The next step is to annotate the compliance effects defined in Fig. 4 into a process plan. For simplicity, we described a process plan taking into account the process elements described in the standard, recalled in Table 1 (see Fig. 5). As



(a) (b)

Fig. 4: CENELEC EN 50128 - Architecture and Design Phase.

we can see in Fig. 5, the process plan contains a series of tasks and elements ascribed to such tasks. To annotate the effect, we need to compare each element in Fig. 5 with the list of compliance effects in Fig. 4. In this case, the names of the process elements can be found in the names of the compliance effects since both models are taken from the standard. Thus, the annotation process is straightforward.

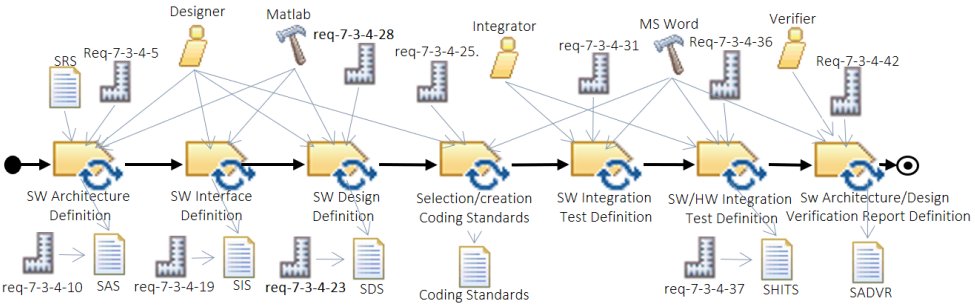


Fig. 5: Process Plan Targeting the Architecture and Design Phase

Fig. 6 shows the annotation of the task SW Design Definition. As the figure depicts, this task has one direct CENELEC EN 50128-related compliance effect, i.e., performSoftwareDesignDefinition. The remaining eight compliance effects are allocated to the elements that directly fulfill them, e.g., the task is performed by a designer, who should have a certificate of UML, and that has knowledge of EN 50128. As we can see in Fig. 5, some tasks are done by the same role. e.g., the

designer should perform the first four tasks, and the same tools should be used. Thus, our approach simplifies the annotations process since all those indirect compliance effects are not required to be annotated in each task. To make the process also compliant with ISO/IEC 15504, the outcome prescribed by the effect provideDatabase (we assume it was modeled as in Fig. 4), should be included in the modeling of the process (see the work product Database highlighted with a dotted line in Fig. 6).

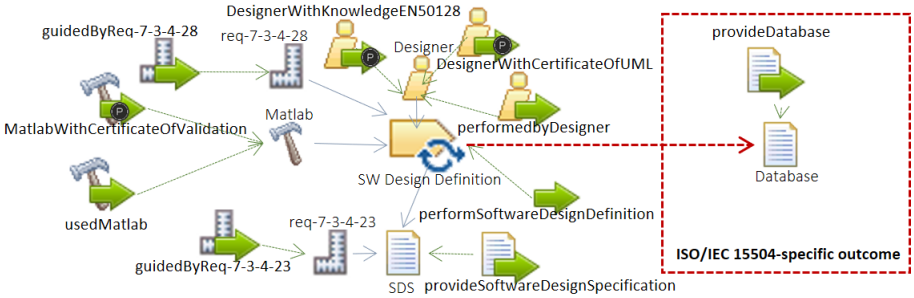


Fig. 6: Task and their Ascribed Elements Annotated with Compliance Effects

5 Discussion

In this section, we present a discussion regarding our method.

5.1 Compliance-related Process Information

Compliance management can benefit from our proposed modeling strategy. First, the icons describing definitions (see Table 2) will correspond to the software process elements required in a fully compliant process plan. Thus, such visual descriptions make the process engineer compliance-aware during software process modeling. Second, the templates presented in Table 3 aim at relating process elements with their properties. Thus, discovering the compliance effects, which the software process element produces in the context of the whole process, is facilitated. Third, as every process element carries out its compliance information, the annotation process is more efficient since it is expected that tasks share their associated elements, i.e., roles, guidance, tools, and work products (See Fig. 6). Moreover, compliance effects from different standards can be added to software process elements without limitations, helping to define multi-compliant process-checkable for compliance. Finally, we propose a standardized template-based mechanism for creating definitional and property-based compliance effects (See Table 3). Such mechanism can also be exploited for automating the generation of standard(s)-compliant process components that can be reused when assembling the processes required in different projects.

5.2 Software Process Diversity

Software process diversity is common in safety-related context, as recalled in the introductory part. Our approach implicitly takes into consideration process

diversity by providing a method that facilitates the selection of compliance artifacts as needed for specific compliance frameworks. In particular, the definition of compliance effects, as presented in Fig. 4, could help in the creation of compliance artifacts from one standard, that can also be enriched with the compliance effects of related standards, as depicted in Fig. 6, for configuring process models that are multi-compliant. This aspect results in the utilization of cohesive process components that have distinctive value attributes. Besides, process components that do not receive significant levels of resource commitments in terms of compliance could be identified as potentially less useful and could be eliminated without significantly impacting project outcomes.

5.3 Relation with the SPI Manifesto

The application of standards best practices is a way to learn from the experience of the functional experts. Such experience is valuable to define and improve specific, project-oriented software processes. Our approach provides a method for deploying compliant-related pieces required for controlling knowledge across standards and projects (as presented in Fig. 4). A process engineer can play with such pieces and learn how to use them to satisfy the demands, not only of the applicable standard(s) but also the company and customer needs. In this way, our approach addresses principle 4 of SPI Manifesto[17], which states that SPI *creates a learning organization.* Moreover, having a model of the required pieces could help the definition and improvement of baseline process models (see Fig. 5). The resulting artifacts aim not only at enabling certification according to the standard but also to change existing habits in the organization, incrementing their awareness regarding best practices and therefore making the business more successful. Thus, our approach also addresses principle 6 of SPI Manifesto, which states the *use of dynamic and adaptable models as needed.*

6 Related Work

In this section, we discuss other approaches that have proposed the separation of concerns for facilitating compliance checking with FCL. In [22], four types of control tags are defined for compliance checking of business processes. These control tags consist of the state and operations of propositions about the conditions that are to be checked on a task and are typed-linked, namely control tags represent compliance effects. Such tags are: the flow tag, which represents a control that would impact on the flow of the process activities; data tag, which identifies the data retention and lineage requirements; the resource tag, which represent access, role management and authorization; finally, time tag, which represents controls for meeting time constraints such as deadlines and maximum durations. Our work, as in [22], describes the compliance effects based on the type of elements that are present in a process. However, contrary to [22], we further separate the definition of the elements from the properties allocated to these elements, i.e., we propose definitional and property-oriented compliance

effects. Moreover, we provide a template for creating the compliance effect and icons that facilitate their description and its subsequent annotation in process elements. In [12], the concept of data tag described in [22] is revisited to create a methodology that permits their extraction from business process logs. Contrary to the work presented in [12], the extraction of our compliance effects is performed directly from the standards and not from process logs since we aim at having a faithful representation of the requirements prescribed by the standard at design time.

7 Conclusions and Future Work

(Process-oriented) Safety standards define process elements and their properties. Similarly, process modeling languages, such as SPEM 2.0, provide definitions that match precisely with the ones described in the standards. In this paper, we took advantage of these characteristics to offer a natural separation of concerns that could be applicable in compliance checking. From the definition of concerns, we proposed a template to describe the compliance effects that are expected from the process elements. Moreover, we proposed icons for their representation that permit their identification and annotation on the corresponding process elements. Our approach offers adequate means to support the separated concepts based on process competence and different standards, and thus it may facilitate the modeling part of our method for automated compliance checking.

Future work includes the evaluation of our approach in terms of usability. In addition, to put the approach into practice, extensions to the current algorithm used for compliance checking must be designed and implemented to permit the inclusion of co-occurrent compliance effects, which are annotated in process elements ascribed to tasks, into the compliance analysis. Moreover, to facilitate further the annotation process, algorithms that permit automatic mapping between compliance effects and company-specific processes, as well as algorithms that automatically permit the creation of process elements from the definitional compliance effects, should be created.

References

1. Castellanos Ardila, J.P., Gallina, B., Ul Muram, F.: Enabling Compliance Checking against Safety Standards from SPEM 2.0 Process Models. In: Euromicro Conference on Software Engineering and Advanced Applications. pp. 45 – 49 (2018)
2. Castellanos Ardila, J.P., Gallina, B., UL Muram, F.: Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance. In: 18th International SPICE Conference (2018)
3. CENELEC: EN 50128. Railway Applications-Communication, Signaling and Processing Systems Software for Railway Control and Protection Systems. British Standards Institution (2011)
4. Crabtree, C., Seaman, C., Norcio, A.: Exploring Language in Software Process Elicitation: A Grounded Theory Approach. In: 3rd International Symposium on Empirical Software Engineering and Measurement. pp. 324–335 (2009)

5. Eckey, M., Greiner, C., Peisl, T.: Why Do Organizations Focus on Assessments Instead of Their Process-Improvement Objectives? In: European Conference on Software Process Improvement. pp. 392–401 (2019)
6. Eclipse: Eclipse Process Framework (EPF) Composer. (2018), https://www.eclipse.org/epf/
7. ECSS: ECSS-E-ST-40C, Space Engineering Software (2009)
8. Fuggetta, A.: Software Process Patterns: A Roadmap. In: International Conference on Software Engineering,. pp. 25–34 (2000)
9. Gallina, B., Pitchai, K., Lundqvist, K.: S-TunExSPEM: Towards an Extension of SPEM 2.0 to Model and Exchange Tunable Safety-oriented Processes. Software Engineering Research, Management and Applications **496**, 215–230 (2014)
10. Gallina, B., Ul Muram, F., Castellanos Ardila, J.: Compliance of Agilized (Software) Development Processes with Safety Standards: a Vision. In: 4th international workshop on Agile Development of Safety-Critical Software. pp. 1–6 (2018)
11. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems. pp. 181–216 (2005)
12. Hashmi, M., Governatori, G., Wynn, M.: Business Process Data Compliance. In: International Workshop on Rules and Rule Markup Languages for the Semantic Web. vol. 7438 LNCS, pp. 32–46 (2012)
13. ISO/IEC 15504-5: Information Technology - Process assessment - An Exemplar Software Life Cycle Process Assessment model (2012)
14. ISO/TC 22/SC 32: ISO 26262: Road Vehicles Functional Safety (2018), https://www.iso.org/standard/68383.html
15. Javed, M., Gallina, B.: Get EPF Composer back to the future: a trip from Galileo to Photon after 11 years. In: EclipseCon (2018)
16. OMG: Software & Systems Process Engineering Meta-Model Specification. Version 2.0. (2008)
17. Pries-Heje, J., Johansen, J.: The SPI Manifesto (2009), https://2020.eurospi.net/images/eurospi/DownloadCenter/spi_manifesto.pdf
18. Ramasubbu, N., Bharadwaj, A., Tayi, G.K.: Software Process Diversity: Conceptualization, Measurement, and Analysis of impact on Project Performance. Management Information Systems **39**(4), 787–807 (2015)
19. RTCA/DO-178C: Software Considerations in Airborne Systems and Equipment Certification. (2011)
20. RTCA/DO-330: Software Tool Qualification Considerations. (2012)
21. Ruiz, A., Juez, G., Espinoza, H., de la Vara, J.L., Larrucea, X.: Reuse of safety certification artefacts across standards and domains: A systematic approach. Reliability Engineering & System Safety **158**, 153–171 (2017)
22. Sadiq, S., Governatori, G., Namiri, K.: Modeling Control Objectives for Business Process Compliance. In: International conference on business process management. pp. 149–164 (2007)
23. Smith, D.: The Design of Divide and Conquer Algorithms. Science of Computer Programming **5**, 37–58 (1985)
24. Sommerville, I.: Software Engineering. Pearson, ninth edn. (2011)
25. Yilmaz, M., O'Connor, R.: A Market Based Approach for Resolving Resource Constrained Task Allocation Problems in a Software Development Process. European Conference on Software Process Improvement pp. 25–36 (2012)