Mälardalen University Doctoral Thesis
No.337

# A Safety-centered Planning-time Framework for Automated Process Compliance Checking

Julieth Patricia Castellanos Ardila

September 2021

MÄLARDALEN UNIVERSITY

Department of Computer Science and Engineering
Mälardalen University
Västerås, Sweden

ISBN 978-91-7485-511-1
ISSN 1651-4238
Printed by E-print AB, Stockholm, Sweden

# Populärvetenskaplig sammanfattning

Säkerhetskritiska system, som vid eventuella funktionsfel skulle kunna få katastrofala konsekvenser för oss, finns överallt. Det är inte bara miljöer med högriskfunktioner, som t ex kärnkraftverk, som är säkerhetskritiska system. Våra fordon, medicinsk utrustning som utför olika typer av behandlingar, flygplan och industrirobotar, är också säkerhetskritiska system. Ju mer skada systemet kan orsaka, desto noggrannare måste systemet designas, implementeras och underhållas. Genom att med rimlig noggrannhet följa den praxis som vanligtvis finns i branschstandarder, kan tillverkare visa att de använder metoder som förhindrar att de säkerhetskritiska systemen har brister eller ger upphov till olika typer av skador. Av denna anledning är det ett måste, för tillverkare av säkerhetskritiska system, att följa dessa standarder, särskilt säkerhetsstandarder.

Branschstandarder har ofta ett normativt upplägg som fokuserar på processrelaterade krav. För att följa sådana standarder måste tillverkare noggrant förbereda processplaner som korrekt uppfyller gällande krav. En lämplig processplan måste inkludera ordningsföljden av de arbetsuppgifter som ingår i gällande standarder samt de resurser som krävs för dessa uppgifter, t.ex. personal, arbetsmaterial, nödvändiga verktyg och metoder, samt viktiga egenskaper hos dem. En sådan arbetsuppgift kan stödjas genom att kontrollera att processplanerna uppfyller de krav som ställs på aktuella punkter i standarden.

Det räcker ofta inte att en sådan kontroll görs för bara en standard. Inom fordonsindustrin till exempel, rekommenderas tillverkare att åtminstone följa standarder för funktionssäkerhet, cybersäkerhet och förbättringar av mjukvaruprocesser. Tillverkare måste även utföra anpassningar av standarder, dvs. välja och ändra krav beroende på det enskilda projektet. I säkerhetsstandarder utförs ofta anpassningar i enlighet med befintliga säkerhetskritiska nivåer. Dessutom krävs omcertifiering när nya versioner av standarderna släpps, vilket sker ofta. Det är dessutom inte bara en enskild projektplan som kontrolleras. Företag behöver ofta planera flera processer parallellt. Det är därför inte lätt att manuellt kontrollera att processplanerna följer kraven i dessa standarder

Automatiserad kontroll av den typen av överensstämmelse skulle kunna hjälpa processingenjörer i säkerhetskritiska sammanhang att upptäcka överträdelser samt säkra att överensstämmelsen följs under planeringstiden. Huvudmålet för denna avhandling är därför att underlätta automatisk kontroll av överensstämmelsen av processplaner som används för att konstruera säkerhetskritiska system i enlighet med de standarder som är obligatoriska (eller rekommenderade) i säkerhetskritiska sammanhang. För att nå vårt mål utgår vi från moderna metoder och verktyg, anpassar dem genom att huvudsakligen fokusera på mjukvara- och riskanalysprocessplaner samt bidrar till den senaste tekniken enligt följande:

1. Vi identifierar aspekter som försvårar kontrollen av processplanens överensstämmelse med standarders fordrande och formulerar detaljerade krav på en teknisk lösning på dessa problem.

2. Vi introducerar ACCEPT (Automated Compliance Checking of Engineering Process plans against sTandards), ett iterativt och begripligt ram- verk som assisterar processingenjörer med att kontrollera och säkerställa att processreglerna efterföljs.

3. Vi föreslår mekanismer för att underlätta skapandet och återanvändningen av de specifikationer som krävs för att kontrollera processplanens överensstämmelse med standarderna.

4. Vi undersöker betydelsen av våra föreslagna lösningar genom att använda olika valideringsmekanismer. I och med detta visar vi att våra lösningar kan vara användbara för att stödja processingenjörer i de kontrolluppgifter som krävs för planering av processer i säkerhetskritiska sammanhang.

Avhandlingens bidrag syftar till att plantera frön för framtida utveckling av verktyg som stöder processingenjörers användning sig av automatiserade metoder för kontroll av processplaners överensstämmelse med standarder.

# Abstract

Safety-critical systems, whose failure could lead to catastrophic consequences, are everywhere. Not only environments with high-risk functions, e.g., nuclear power plants, are safety-critical systems. Our vehicles, medical devices that perform different kinds of treatments, airplanes, and industrial robots, are also safety-critical systems. The more harm the system can cause, the more careful the system has to be designed, implemented, and maintained. By following practices of reasonable care, typically collected within industry standards, manufacturers demonstrate that they aim at preventing safety-critical systems from failing or causing various types of damage. Thus, compliance with standards, especially safety standards, is a must-do for manufacturers of safety-critical systems.

Industry standards often adopt a prescriptive approach, which focuses on process-related requirements. To comply with such standards, manufacturers have to carefully prepare process plans that properly address the applicable requirements. A compliant process plan should include the sequence of tasks mandated by applicable standards as well as the resources allocated to such tasks, e.g., personnel, work products, required tools, and methods, which are also framed with key properties. The planning task could be supported by checking that planned processes fulfill the properties set down by standards at given points.

Compliance checking of process plans is rarely done for just one standard. In automotive, for instance, it is recommended that manufacturers follow at least standards for functional safety, cybersecurity, and software process improvements. Manufacturers also need to perform tailoring, i.e., select and modify requirements depending on the individual project. In safety standards, tailoring is often performed by taking into account existing safety criticality levels. Moreover, new versions of the standards, which are frequently released, demand recertification. In addition, compliance checking is not only done to one process plan. Companies commonly need to plan several processes simultaneously. Consequently, it is not easy to manually check that process plans comply with the requirements of standards.

Automated compliance checking could help process engineers in such organizations to detect compliance violations and enforce compliance at planning time. Thus, the main goal of this dissertation is to facilitate automated compliance checking of the process plans used to engineer safety-critical systems against the standards mandated (or recommended) in the safety-critical context. To reach our goal, we adopt modern methods and tools, adapt them by mainly focusing on software and risk analysis process plans, and contribute to the state-of-the-art as follows:

1. We identify aspects that make compliance checking of process plans demanding and formulate requirements for a technical solution to these problems.
2. We introduce ACCEPT (Automated Compliance Checking of Engineering Process plans against sTandards), an iterative and comprehensible framework for supporting process engineers to check and enforce process plan compliance.
3. We propose mechanisms for facilitating the creation and reuse of the specifications required to check process plan compliance.
4. We investigate the significance of our proposed solutions by applying different validation mechanisms. As a result, our solutions show to be useful to support process engineers in the compliance checking tasks required during process planning.

This dissertation's contributions aim at planting the seeds for the future development of tools that support process engineers moving towards automated compliance checking practices.

DEDICATED TO MY FAMILY

# Acknowledgments

First, I would like to express my appreciation to my principal supervisor, Barbara Gallina, who has supported me during the whole duration of my studies. Thanks to her guidance, I have been able to fulfill all the research goals. I will also thank my assistant supervisor, Faiz Ul Muram, for her contributions to my research. Special thanks to Guido Governatori, leader of the Software Systems Research Group at CSIRO's Data61, for sharing his knowledge and expertise.

I also want to thank the head of our division, Radu Dobrin, and the student representatives Viktorija Badasjane and Rachael Berglund for their support. Special thanks to Thomas Nolte, Federico Ciccozzi, Jenny Hägglund and Carola Ryttersson for facilitating the MDH routines. My gratitude is also for the people who are or have been colleagues at MDH. In particular, I thank Jan Carlson, Antonio Cicchetti, Luciana Provenzano, Soheila Sheikh Bahaei, Robbert Jongeling, Filip Markovic, Asha Kiran, Mirgita Frasheri, Irfan Sljivo, Simin Cai, LanAnh Trinh, Gabriel Campeanu, Omar Jaradat, Inmaculada Ayala, and Zulqarnain Haider, for taking their time to answer my countless questions, and for sharing their interests. Special thanks to Cristina Seceleanu for reviewing my thesis and giving me valuable comments.

I also want to give special thanks to my mother, Mercedes, who from Colombia is encouraging me to finish everything I start. Finally, and most importantly, I would like to express my gratitude and love to my husband Ola and my son Gabriel. Their company, patience, hugs, unconditional support, and love have strengthened me through this challenging experience.

The work in this Ph.D. thesis has been partially supported by EU and VINNOVA via the ECSEL JU project AMASS (No. 692474) [1].

Julieth Patricia Castellanos Ardila
Västerås, September, 2021

# List of Publications

## Papers Included in the Licentiate Thesis[1]

**Paper A:** *Facilitating Automated Compliance Checking of Processes in the Safety-critical Context*, Julieth Patricia Castellanos Ardila, Barbara Gallina and Faiz Ul Muram. Journal of Electronic Communications of the EASST. vol 78. 2019.

**Paper B:** *Separation of Concerns in Process Compliance Checking: Divide-and-Conquer*, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the European Systems, Software & Service Process Improvement & Innovation. EuroAsiaSPI 2020. Communications in Computer and Information Science, vol 1251. Springer, Cham. 2020.

**Paper C:** *A Personal Opinion Survey on Process Compliance Checking in the Safety Context*, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the 13th International Conference on the Quality of Information and Communications Technology. QUATIC 2020. Communications in Computer and Information Science, vol 1266. Springer, Cham. 2020.

**Paper D:** *Compliance-aware Engineering Process Plans: The case of Space Software Engineering Processes*, Julieth Patricia Castellanos Ardila, Barbara Gallina, and Guido Governatori. In Journal of Artificial Intelligence and Law. 2021.

**Paper E:** *Reusing (Safety-oriented) Compliance Artifacts while Recertifying*, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the

[1]The included papers have been reformatted to comply with the thesis layout

9th International Conference on Model-Driven Engineering and Software Development. Scitepress Digital Library - Volume 1: Modelsward. 2021.

**Paper F:** *Systematic Literature Review of Compliance Checking Approaches for Software Processes*, Julieth Patricia Castellanos Ardila, Barbara Gallina and Faiz Ul Muram. Technical Report, ISRN MDH-MRTC-336/2021-1-SE. Mälardalen Real-Time Research Center, Mälardalen University, June 2021. A version is submitted to a journal.

## Additional Peer-reviewed Publications Related to the Thesis[2]

**Paper 1:** *Towards Increased Efficiency and Confidence in Process Compliance*, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the 24th European Conference on Software Process Improvement (EuroAsiaSPI), Ostrava, Czech Republic, September 2017.

**Paper 2:** *Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards* Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the 7th IEEE International Workshop on Software Certification (WoSoCer), Toulouse, France, October 2017.

**Paper 3:** *Formal Contract Logic Based Patterns for Facilitating Compliance Checking against ISO 26262*, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the 1st Workshop on Technologies for Regulatory Compliance (TeReCom), Luxembourg, Luxemburg, December 2017.

**Paper 4:** *Compliance of Agilized (Software) Development Processes with Safety Standards: a Vision*, Barbara Gallina, Faiz Ul Muram and Julieth Patricia Castellanos Ardila. In Proceedings of the 4th international workshop on Agile Development of Safety-Critical Software (ASCS), Porto, Portugal, May 2018.

**Paper 5:** *Enabling Compliance Checking against Safety Standards from SPEM 2.0 Process Models*, Julieth Patricia Castellanos Ardila, Barbara Gallina and Faiz Ul Muram. In Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, Czech Repub-

[2]These papers are not included in this thesis

lic, August 2018.

**Paper 6:** *Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance*, Julieth Patricia Castellanos Ardila, Barbara Gallina and Faiz Ul Muram. In Proceedings of the 18th International Software Process Improvement and Capability Determination Conference (SPICE), Thessaloniki, Greece, October 2018.

**Paper 7:** *Lessons Learned while Formalizing Functional Safety Standards for Compliance Checking*, Julieth Patricia Castellanos Ardila, Barbara Gallina and Guido Governatori. In Proceedings of the 2nd Workshop on Technologies for Regulatory Compliance (TeReCom), Groningen, The Netherlands, December 2018.

**Paper 8:** *Process Compliance Re-Certification Efficiency Enabled by EPF-C ∘ BVR-T*, Barbara Gallina, Aleksandër Pulla, Antonela Bregu and Julieth Patricia Castellanos Ardila. In Proceedings of the 13th International Conference on the Quality of Information and Communications Technology. QUATIC 2020. Communications in Computer and Information Science, vol 1266. Springer, Cham. 2020.

## Licentiate Thesis

**Facilitating Automated Compliance Checking of Processes against Safety Standards**, Julieth Patricia Castellanos Ardila. Mälardalen University Press Licentiate Theses, ISSN 1651-9256; 277. p. 170. March 28th 2019.

# Contents

# I

# Thesis

# Chapter 1

# Introduction

Manufacturers of safety-critical systems[1] have the duty of care[2] [4]. Duty of care does not imply that the production of safety-critical systems is zero risks[3] but responsible. As such, manufacturers should follow accepted practices of reasonable care, usually found in industry standards [6]. Industry standards offer frameworks that encompass adequate practices refined by experts from historically successful experiences [7]. Standards also include knowledge and awareness of public policy, societal norms, and preferences [8]. Consequently, compliance with industry standards, particularly safety standards, is an essential requirement when performing safety-related engineering activities [9].

For industries developing safety-critical systems, safety considerations are sometimes less important than the economic ones [10]. Consequently, a major strand of social control has involved the imposition of standards [11]. For instance, in the UK, the HSE[4] has used compliance with IEC 61508 [12] as a guideline for bringing legal actions if harm is caused by safety-critical systems [4]. In the US, the NHTSA[5] has also the authority to subject manufacturers who fail to comply with the Federal motor vehicle safety standard to

[1] Safety-critical systems are those whose failure could lead to unacceptable consequences, e.g., death, injury, loss of property, or environmental harm [2]

[2] The obligation imposed on an individual or organization requiring adherence to a standard of reasonable care while performing any acts that could foreseeably harm others [3].

[3] There is not such a thing as zero risks. The reason is that no physical item has zero failure rate, no human being makes zero errors, and no piece of software design can foresee every operational possibility. Thus, the concept of tolerable risk prevails. [5]

[4] Health and Safety Executive https://www.hse.gov.uk/

[5] National Highway Traffic Safety Administration https://www.nhtsa.gov/

civil penalties [13]. We can also see that more and more knowledgeable customers individually demand the use of standards to influence responsible behavior among industry practices [14]. As a consequence, failure or inadequate compliance with industry standards could not only lead to unexpected accidents (see, for example, the inadequate compliance of the reconfigured MCAS[6] included in the Boeing 737 MAX 8, that allows the airplanes into market, causing 346 people died in two crashes[7]) but also legal risks, i.e., penalties [18] and prosecutions [19] (see, again, the case of Boeing[8]).

Given the previous considerations, it is clear that compliance with industry standards is not a voluntary choice (even though in some contexts it is not mandatory) but is a must-do for manufacturers of safety-critical systems. However, there are many normative frameworks, which aim at preventing harm and under-performance of safety-critical systems. In addition, for manufacturers acting in multiple jurisdictions (different regions with specific regulations and laws), the compliance risk increases due to the diversity and complexity of the normative frameworks that should be applied. Consequently, manufacturers may fail in ensuring the required compliance [20].

Demonstrating compliance with industry standards requires evidence from the object of conformity [21]. In particular, prescriptive standards, which assure the system technical properties by heavily commanding a rigorous development process [22], demand documented evidence of responsibilities, agreements, and established processes [23]. For this reason, an essential part of process assessments is the compliance of requirements related to the planning part of such processes [24]. The rationale is that every work product that a manufacturer provides is the outcome of a process [25]. Thus, normative imperatives should be designed into the system and not added later on.

The planning of the processes also servers safety-critical manufacturers to proceed in a systematic way [26] increasing predictability and transparency [27]. A compliant process plan can demonstrate intentional compliance [28], i.e., “the design-time distribution of responsibilities, such that if every actor ful-

[6]MCAS (Maneuvering Characteristics Augmentation System) is a software application used to adjust the angle-of-attack of the nose of the Boeing 737 MAX 8. MCAS was certified by the FAA (Federal Aviation Administration) based on an earlier ”system safety analysis” provided by Boeing to the regulators. Thus, FAA technical staff had not been fully aware of the latest details of MCAS functions [15]

[7]Lion Air Flight 610 in October 29, 2018 [16], and Ethiopian Airlines Flight 302 in March 10, 2019 [17]

[8]Boeing estimated US$20 billion in direct cost from the grounding, US$100 million in victim compensation fund, and legal liability to the families of the first 11 victims, at least US$1.2 million each https://edition.cnn.com/2020/11/17/business/boeing-737-max-grounding-cost/index.html

fills its goals, then the compliance is ensured." Thus, manufacturers of safety-critical systems can use process plans to set agreements at the initial stages of the development process with certification bodies and partners (for example, the certification liaison process, explicitly defined in DO-178C [29]).

Adjusting process plans in compliance with applicable standards is a matter of decision-making, in which managers should consider the selection of a strategy that fits within the boundaries allowed by the norms [28]. In particular, process plans should provide information regarding the types of compliance evidence that will be used and how and when it will be produced [30]. Such information is related to the sequencing of tasks, the resources required and produced (e.g., personnel, work products, and tools), and recommended techniques. Moreover, all these elements are framed with different properties. Standards also set down the points at which different sorts of compliance artifacts should be established [31]. For this reason, a common practice during the planning of the processes is to perform compliance checks.

Compliance checking reports can provide the right level of abstraction so that the conditions for compliance can be easily evaluated [28]. Such reports help to identify compliance errors, assist in creating process specifications, and prevent non-compliance tasks from being performed [32]. However, manually checking the compliance of process plans is a challenging task. In particular, compliance checking is rarely done for just one standard. For example, in automotive, manufacturers need to manage the introduction of guidelines aimed at providing a base to address cybersecurity (e.g., SAE J3061 [33]) besides already established normative documents such as standards for safety (i.e., ISO 26262 [34]) and process improvement (i.e., Automotive SPICE[9] [35]). Consequently, manufacturers need to face process diversity [36], which is the management of the multiple reference models within single projects.

Manufacturers also need to perform tailoring. Tailoring requires selecting applicable requirements, performing their eventual modifications, and explaining their implementation according to the project's particular circumstances. There are several drivers for tailoring [37], e.g., aspects regulated by standards (e.g., safety criticality levels), development constraints, product quality, and business objectives. Moreover, new versions of the standards are frequently released. For instance, the different versions of the standard ISO 14971-process for risk analysis and evaluation[10] make it challenging for the medical domain to gain approval for products within and outside Europe [41]. Standards evolution, process diversity, and tailoring involve dynamic complexity [42], which

[9]Software Process Improvement and Capability Determination

[10]ISO 14971:2007 [38], EN ISO 14971:2012 [39] and ISO 14971:2019 [40]

implies considering adequate means for handling change management.

Process-based assessments supported by tools, which can provide automated checks [43], could be a solution for permitting organizations to control their procedures and avoid compliance risk. In particular, it is crucial to support the process engineer in such organizations, who is the role responsible for selecting, composing, and documenting adequate process elements [44]. Automated compliance checking could help such engineers to detect compliance violations and enforce compliance at planning time. Thus, the main goal of this dissertation is to **facilitate automated compliance checking of the process plans used to engineer safety-critical systems against the standards mandated (or recommended) in the safety-critical context**. To reach our goal, we define four subgoals and fulfil them by adopting and adapting modern methods and tools. The subgoals and contributions are presented below.

1. **Elicit the requirements to be met to support the automated compliance checking of the process plans used in the safety-critical context.** This subgoal focuses on establishing the requirements for a technical solution that alleviates the compliance challenges that manufacturers of safety-critical systems have in terms of prescriptive standards, specifically at planning time. The desired solution shall facilitate: 1) the management of the artifacts required for compliance checking with prescriptive standards, i.e., the standards themselves, their requirements, the processes plans, and the compliance means; 2) keeping track of the applicable requirements; 3) the recognition of contradictions and ambiguities between applicable requirements; 4) managing the changing nature associated with requirements diversity.

2. **Identify mechanisms for supporting automated compliance checking of the process plans used in the safety-critical context.** This subgoal focuses on discovering how existing tools and methodologies can be appropriately combined to provide the support required for automated compliance checking of process plans. In particular, we adopt compliance by design [45], an approach aimed at integrating compliance requirements at design time, permitting to resolve compliance violations in engineering process plans before they are executed. As a result, we introduce ACCEPT (Automated Compliance Checking of Engineering Process plans against sTandards). ACCEPT is an iterative and comprehensible framework that allows the creation of engineering process plans checkable for compliance, i.e., process elements enriched with compliance information through annotations representing formalized standards requirements in FCL (Formal Contract Logic) [46]. FCL

is a language created to represent and reason upon normative knowledge. In particular, FCL permits to represent obligations (which are mandatory situations), prohibitions (which are situations to avoid), and permissions (which are allowed situations). Engineering processes checkable for compliance can be formally verified. ACCEPT is supported by the tool-chain constituted of Eclipse Process Framework Composer (EPF-C) [47], which has good coverage of SPEM 2.0 (Systems & Software Process Engineering Metamodel) [48] concepts, and the compliance checker Regorous [49], which provides automatic reasoning with FCL rules. We also define a set of methodological steps that facilitate the use of ACCEPT.

3. **Facilitate the creation of reusable specifications required for automated compliance checking of the process plans.** This subgoal focuses on discovering adequate means to model the concepts included in the specifications, as well as the necessary support for enabling systematic reuse. For this, we propose three mechanisms.

   - *Process compliance hints*, which are resources designed to support the creation of separated concepts included in the FCL specification based on process structure and the concepts supported by SPEM 2.0. (i.e., tasks, roles, work product, guidance, tools, and their relationships).
   - *Process compliance patterns*, which indicate common structures that an FCL designer is likely to encounter when formalizing standards (clauses in a standard commonly have title, prerequisites, mandated tools/techniques, work products, tailoring rules, and guidance).
   - *Systematic reuse and automatic generation of process compliance checking artifacts* through a solution that combines the reuse capabilities provided by SoPLE (Safety-oriented Process Line Engineering) [50] and the support provided by the tool-chain composed by EPF-C and BVR-T (Base Variability Resolution Tool) [51] (proposed in the context of the European AMASS Project [52, 53]). This solution offers the opportunity to specify constraints that reduce the selection of reusable process-related compliance checking artifacts for its subsequent composition.

4. **Investigate the significance of a solution for automated compliance checking of process plans in the safety-critical context.** This subgoal focuses on objectively analyze our solutions' pros and cons and define elements that permit their future improvement and usage. In particular, we investigate the significance of our proposed solutions by applying different validation mechanisms, with the results presented below.

- First, we investigate the *degree of acceptance* of methods for automated compliance checking via a personal opinion survey. In total, we obtained 15 valid responses from practitioners who participate in process compliance checking, mainly from the European consultatory branch. The practitioners show a favorable position regarding our solutions and indicated usability aspects to be improved.
- Second, we *evaluate ACCEPT*, via a standard-based case study, considering specific qualitative usefulness criteria described in [54]. In particular, we study the standard ECSS-E-ST-40C [55], which provides baseline criteria for software process planning in space projects. Initial observations show that the effort determined by task demand required to model compliance checking artifacts is significant. However, the effort is reduced in the long term since models are, to some extend, reusable and flexible. Reusing artifacts may simplify the work that process engineers need to perform in every process planning. Such a gain could be interpreted as a benefit in terms of resource savings since professionals' time is costly. We also analyzed the coverage level of the models based on design decisions. In our opinion, such a coverage level is adequate since it responds to the information needs required by the standard studied.
- Third, we *measure the reuse* permitted by our solutions when showing process plan adherence with new versions of standards. In particular, we consider three versions of the standard ISO 14971 [56]-process for risk management to medical devices, model the artifacts required for compliance checking of process plans by using our solutions and measure the enabled reuse by taking into account a metric for reuse measurement proposed by [57] (which is expressed in terms of percentage by considering the proportion of the number of new objects built and the total number of objects used). The reuse extent in the context of medical devices complying with the evolution of the standard ISO 14971 is significant. We conclude that processes and standards that evince low levels of variation could benefit from using our methodological framework during the modeling task required for compliance checking.
- Fourth, we *position our work with respect to the state-of-the-art and the state-of-the-practice*. First, we characterize the existent approaches for compliance checking of software processes by performing a systematic literature review. Second, we test our assumptions regarding compliance checking current practices and challenges with a set of questions that were answered in the personal opinion survey that was previously mentioned.

As a result, we find that our solutions provide a broader set of characteristics that seamlessly integrate with current practices. We also identified aspects that permit our solutions to improve in the future.

The resulting research efforts aim at planting the seeds for the future development of tools that support process engineers moving towards automated compliance checking practices.

## 1.1 Thesis Outline

We organize this thesis in two parts. In the first part, we summarize the research as follows: In Chapter 2, we recall essential background and prior work. In Chapter 3, we present the research summary. In Chapter 4, we describe the specific research contributions of this thesis. In Chapter 5, we discuss related work. Finally, in Chapter 6, we present conclusions and future work. The second part is a collection of the papers included in this thesis. We present a brief overview of the included papers.

**Paper A:** ***Facilitating Automated Compliance Checking of Processes in the Safety-critical Context***, Julieth Patricia Castellanos Ardila, Barbara Gallina, and Faiz Ul Muram. Journal of Electronic Communications of the EASST. vol 78. 2019.

**Abstract:** In some domains, the applicable safety standards prescribe process-related requirements. Essential pieces of evidence for compliance assessment with such standards are the compliance justifications of the process plans used to engineer systems. These justifications should show that the process plans are produced in accordance with the prescribed requirements. However, providing the required evidence may be time-consuming and error-prone since safety standards are large, natural language-based documents with hundreds of requirements. Besides, a company may have many safety-critical-related processes to be examined. In this paper, we propose a novel approach that combines process modeling and compliance checking capabilities. Our approach aims at facilitating the analysis required to conclude whether the model of a process plan corresponds to a model with compliant states. Hitherto, our proposed methodology has been evaluated with academic examples that show the potential benefits of its use.

**My contribution:** This paper summarizes the work done during the first two years of the Doctoral tenure. I was the primary driver of the paper. The coauthors, who are my Ph.D. supervisors, were highly involved in this research by providing all kinds of advice, ideas, reviews, comments, and more. This paper was initially presented as a research abstract in the Doctoral Symposium at ISOLA 2018[11].

**Paper B:** ***Separation of Concerns in Process Compliance Checking: Divide-and-Conquer***, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the European Systems, Software & Service Process Improvement & Innovation. EuroAsiaSPI 2020. Communications in Computer and Information Science, vol 1251. Springer, Cham. 2020.

**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.

**My contribution:** I was the primary driver of the paper under the supervision of the coauthor. My specific contribution included the description of the method. I also illustrated the method by creating a model checkable for compliance from the rail sector, and I wrote the paper. The coauthor contributed to the design of the paper, idea for the title, reviews, and comments for improving the paper.

**Paper C:** ***A Personal Opinion Survey on Process Compliance Checking in the Safety Context***, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the 13th International Conference on the Quality of Information and Communications Technology. QUATIC 2020. Communications in Computer and Information Science, vol 1266. Springer, Cham. 2020.

[11]https://www.isola-conference.org/isola2018/docsymp.html

**Abstract:** Manually checking the compliance of process plans against the requirements of applicable standards is a common practice in the safety-critical context. We hypothesize that automating this task could be of interest. To test our hypothesis, we conducted a personal opinion survey among practitioners who participate in process compliance checking. In this paper, we present the results of this survey. Practitioners indicated the methods used and their challenges, as well as their interest in a novel method that could permit them to move from manual to automated practices via compliance checking.

**My contribution:** I was the primary driver of the paper under the supervision of the coauthor. My specific contribution included the creation of a set of initial questions. The second author helped to structure and design the survey by providing comments for cleaning ambiguity and a more in-depth analysis that led to the formulation of further questions. The final evaluation was performed by both authors, improving textual explanations and questions. I also apply the survey, analyse the data and wrote the paper.

**Paper D:** ***Compliance-aware Engineering Process Plans: The case of Space Software Engineering Processes***, Julieth Patricia Castellanos Ardila, Barbara Gallina, and Guido Governatori. In Journal of Artificial Intelligence and Law. 2021.

**Abstract:** Safety-critical systems manufacturers have the duty of care, i.e., they should take correct steps while performing acts that could foreseeably harm others. Commonly, industry standards prescribe reasonable steps in their process requirements, which regulatory bodies trust. Manufacturers perform careful documentation of compliance with each requirement to show that they act under acceptable criteria. To facilitate this task, a safety-centered planning-time framework, called ACCEPT, has been proposed. Based on compliance-by-design, ACCEPT capabilities (i.e., processes and standards modeling, and automatic compliance checking) permit to design Compliance-aware Engineering Process Plans (CaEPP), which are able to show the planning-time allocation of standard demands, i.e., if the elements set down by the standard requirements are present at given points in the engineering process plan.

In this paper, we perform a case study to understand if the ACCEPT produced models could support the planning of space software engineering processes. Space software is safety and mission-critical, and it is often the result of industrial cooperation. Such cooperation is coordinated through compliance with relevant standards. In the European context, ECSS-E-ST-40C is the de-facto standard for space software production. The planning of processes in compliance with project-specific ECSS-E-ST-40C applicable requirements is mandatory during contractual agreements. Our analysis is based on qualitative criteria targeting the effort dictated by task demands required to create a CaEPP for software development with ACCEPT. Initial observations show that the effort required to model compliance and processes artifacts is significant. However, such an effort pays off in the long term since models are, to some extend, reusable and flexible. The coverage level of the models is also analyzed based on design decisions. In our opinion, such a level is adequate since it responds to the information needs required by the ECSS-E-ST-40C framework.

**My contribution:** I was the primary writer of the paper, and the coauthors contributed with reviews and comments to improve the paper.

**Paper E:** ***Reusing (Safety-oriented) Compliance Artifacts while Recertifying***, Julieth Patricia Castellanos Ardila and Barbara Gallina. In Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development. Scitepress Digital Library - Volume 1: Modelsward. 2021.

**Abstract:** Revisions of safety-related standards lead to the release of new versions. Consequently, products and processes need to be recertified. To support that need, product line-oriented best practices have been adopted to systematize reuse at various levels, including the engineering process itself. As a result, Safety-oriented Process Line Engineering (SoPLE) is introduced to systematize reuse of safety-oriented process-related artifacts. To systematize reuse of artifacts during automated process compliance checking, SoPLE was conceptually combined with a logic-based framework. However, no integrated and tool-supported solution was provided. In this paper, we focus on process recertification (interpreted as the need to show process plan adherence with the new version of the standard) and propose a concrete technical and tool-supported methodological framework for reusing (safety-oriented) compliance artifacts while recertifying. We illustrate the benefits of our methodological framework by considering ISO 14971 versions, and measuring the enabled reuse.

**My contribution:** I was the primary driver of the paper under the supervision of the coauthor. My specific contribution included the specification of the approach presented in this paper and its illustration. I also wrote the paper. The coauthor proposed the idea and title of the paper as well as the example that was used in the illustration of the method. The coauthor also helped to shape the contribution and commented on it to improve the paper.

**Paper F:** ***Systematic Literature Review of Compliance Checking Approaches for Software Processes***, Julieth Patricia Castellanos Ardila, Barbara Gallina and Faiz Ul Muram. Technical Report, ISRN MDH-MRTC-336/2021-1-SE. Mälardalen Real-Time Research Center, Mälardalen University, June 2021. A version is submitted to a journal.

**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.

**My contribution:** I was the primary driver of the paper under the supervision of the coauthors, who contributed with reviews and comments.

# Chapter 2

# Background and Prior Work

Safety-critical systems are primarily identified in environments that have to perform high-risk functions [58], for example, nuclear power systems, petrochemical plants, aircraft and airways, space systems, weapons, and DNA technology. Failure in these systems can quickly lead to severe consequences for humans (death or injury), the damage of the environment (e.g., oil spill), property or physical equipment (e.g., an spacecraft), and even situations where the mission is critical (e.g., electric power systems) [59]. A closer look reveals that there are many new types of systems that have the potential for very high consequences of failure [60]. The main reason is the increasing use of software for controlling almost everything (i.e., from home applications to warning systems in safety-critical applications) [61]. Such increment is closely related to the growing occurrence of systematic failures, which can lead to accidents [62].

As a counteract, proved strategies are at hand to support the production of safety-critical systems and facilitate their safer deployment into society. One of these strategies is compliance with industry standards, which consist of a set of requirements created to protect customers from poor quality products and bad design practices [63]. In some industries (e.g., civil aviation), manufacturers are subject to mandatory compliance. In those cases, a delegated agency (e.g., FAA in the USA) performs compliance assessments and issue some kind of approval (certification/licensing) that allows the system into market [59]. Not all industries are forced to comply with standards. However, there are societal factors that impose such compliance. We list some of those factors.

1. Consumers are demanding more and more information about the impacts of their acquired goods at all stages of the product lifecycle [14].

2. In some countries, showing compliance with standards is relevant evidence for a jury to consider in a product liability action [64], since compliance is a sign of reasonable care [6].

3. Safety-critical systems production is frequently the result of industrial cooperation. Meeting the highest levels of industry standards helps to coordinate lead firms and their suppliers on a global scale [65].

Demonstrating compliance with standards requires evidence regarding the object of conformity (e.g., products, and processes) [21]. To ground our investigation, we focus on compliance assessment of processes plans, a common and accepted procedure carried out in the safety-critical context. In this chapter, we introduce essential background and prior work related to the object of our study. In particular, we present aspects regarding compliance assessment of process plans (see Section 2.1), process models (see Section 2.2), process variability management (see Section 2.3), the process-based compliance by design approach (see Section 2.4), design hints and patterns (see Section 2.5), and empirical research aspects (see Section 2.6).

## 2.1 Compliance Assessment of Processes Plans

Prescriptive standards assure the technical properties of a system by commanding a rigorous development process [22]. For instance, standards for safety, such as RTCA DO-178C [29], RTCA/DO-330 [66], IEC 61508 [12], ISO 26262 [34], EN 50128 [67], ECSS series [68], ISO 14971 [56], cybersecurity guidelines SAE J3061 [33], software processes (e.g., ISO/IEC/IEEE 12207 [69]), and quality, process improvement and capability maturity frameworks such as ISO 9000 [70], ISO/IEC 15504 [71] and derivatives such as Automotive SPICE [35], and CMMI [72], are conceived to focus on processes. Safety-critical systems manufacturers use such standards to demonstrate that their products and services are created using acceptable criteria.

In particular, the planning of the processes servers safety-critical manufacturers to proceed in a systematic way [26] increasing predictability and transparency [27]. Consequently, one of the routine activities during compliance assessment is the evaluation of process plans. For instance, the standard DO-178C [29] defines the so-called certification liaison process, where the first interaction between the applicant and the certification body is expected to occur after the planning phase to ensure plan's approval [73].

A compliant process plan should be able to demonstrate intentional compliance [28], i.e., “the design-time distribution of responsibilities, such that if every actor fulfills its goals, then the compliance is ensured.” Thus, complete process plans should show not only the planning of tasks but also the resources required and produced, e.g., personnel, work products, and tools, which are additionally framed with essential properties. Furthermore, as recalled in [74], traceability between the requirements and the evidence provided in the process plan is also demanded by many normative frameworks. Thus, it is necessary to include a model connected with the original legal texts that can justify the outcomes of compliance checking activities [75].

In addition, compliance justifications, which are expected to be scrutinized by an auditor, should also be provided. A compliance justification is a document given in terms of either a checklist, an argument, or proof (e.g., a verification report) that can show/argue/prove that the process plan complies with the applicable requirements [73]. Compliant process plans should be agreed upon at the beginning of the project between regulatory bodies and applicants [30]. In the remaining parts of this section, we recall the process related standards ISO 26262 (see Section 2.1.1), SAE J3061 (see Section 2.1.2), ECSS-E-ST-40C (see Section 2.1.3) and ISO 14971 (see Section 2.1.4).

### 2.1.1 ISO 26262

ISO 26262 [34] is a standard that addresses functional safety in automotive. ISO 26262 introduces the notion of Automotive Safety Integrity Level (ASIL), which represent a criterion to specify the item’s necessary safety requirements, needed to ensure a certain level of confidence. In particular, ASIL correspond with one of four levels to specify the item’s necessary requirements of ISO 26262 and safety measures to apply for avoiding an unreasonable residual risk, which range from A (the milder) to D (the most stringent). Functional safety is influenced by the development lifecycle process. Therefore, ISO 26262 specifies a safety lifecycle that comprises the entirety of phases from concept through decommissioning of the system. Planning, coordinating and documenting the safety activities of all phases of the safety lifecycle are key management tasks during the implementation of ISO 26262. For ISO 26262, each requirement shall be fulfilled unless:

1. tailoring of the safety activities has been planned, or
2. an assessed rationale is available that the non-compliance is acceptable.

A clause in ISO 26262 states the objectives, general information of the clause, inputs for the clause, requirements and recommendations to be fulfilled, and finally the work products that are to be generated (see Table 2.1). Notes are also included, but they have informative character, i.e., they are expected to help the applicant in understanding and interpreting the requirements. The requirements and recommendations section describes not only the activities and the tasks required during the engineering process but also specific conditions required for compliance.

Table 2.1: ISO 26262:2011-Part 3.

**5. Item definition**  
**Objectives.** The first objective is to define and describe the...  
**General.** This clause lists the requirements and recommendations for...  
**5.3. Inputs of this clause.**  
*5.3.1. Prerequisites.* None.  
*5.3.2. Further supporting information.* Any information that already exists concerning the item, ...  
**5.4. Requirements and recommendations**  
5.4.1 Functional and non-functional requirements shall be made available, including:  
a) functional concept  
b)

...  
5.4.2. ...  
...  
**5.5 Work products**: Item definition resulting from the requirements of 5.4.

### 2.1.2 SAE J3061

SAE J3061 [33] is a guidebook that provides a process reference model, high-level guiding principles, and information on existing tools and methods to help organizations identify and assess cybersecurity threats and design cybersecurity into cyber-physical vehicle systems. The current version of SAE J3061 was released in January 2016, but the definition of Automotive Cybersecurity Integrity Level (ACsIL) is still a work in progress. A cyber-physical vehicle system contains a tight coupling between the computational elements, the physical elements of the system, and the environment around the system. Cybersecurity is an attribute of cyber-physical systems that should be built into the design. Thus, an appropriate lifecycle, which addresses threats from concept to decommissioning, is required. SAE J3061 proposes a lifecycle for handling cybersecurity which is based on ISO 26262's safety lifecycle. The lifecycle can be integrated into a safety process tailored from ISO 26262 by including the cybersecurity activities for each product lifecycle phase, with the related activities for each product lifecycle phase described in the safety process.

### 2.1.3 ECSS-E-ST-40C

The European Cooperation for Space Standardization (ECSS) developed a set of standards for use in all European space activities. The ECSS standard system includes three branches, i.e., Management (M), Engineering (E), and Product Assurance (Q). Handbooks (HB) guide the application of the requirements. The software engineering handbook, ECSS-E-HB-40A [37], states that in a space software project, a customer-supplier business agreement should be established. The customer shall produce the project requirements documentation, which could be produced by using the *ECSS Applicability Requirements Matrix (EARM)*. The EARM should have the list of applicable ECSS requirements with identifiers, applicability condition, i.e., applicable without change (A), applicable with modification (M), not applicable (D), and new generated requirement (N). The supplier responds with the *ECSS Compliance Matrix (ECM)*, indicating the compliance for each requirement provided in the EARM. Partial compliance needs to be detailed, such that the customer can assess the extent to which the objective of the ECSS is covered. Non-compliance also needs to be investigated in terms of feasibility and acceptability in the scope of the project. When a space project starts, the supplier has to identify a suitable software lifecycle process. Thus, discussions about the technical specifications based on the requirements baseline must start early in the lifecycle process [76].

In space software development, the requirements prescribed by the standard ECSS-E-ST-40C [55], which determines mission (non-safety) requirements on how the goals can be achieved, should be applied. Such requirements could be tailored, i.e., adapted for the characteristics of the project. For example, ECSS-E-ST-40C-Annex R, provides a pretailoring based on safety criticality categories, which rank from catastrophic to negligible (prescribed in ECSS-Q-ST-40C [77]). Thus, mission requirements have an inherent relationship with safety issues. Further tailoring should be analyzed in the scope of the project and its consequences assessed and documented. If requirements are tailored out, the associated expected outputs are also tailored out.

### 2.1.4 ISO 14971 and Its Evolution

ISO 14971 [56] specifies the process required to identify hazards, estimate, evaluate, control, and monitor the risk of medical devices during its lifecycle. The content of ISO 14971 has been evolving over the years [78]. When published, ISO 14971:2007 [38] was internationally endorsed. In contrast, EN ISO 14971:2012 [39] is harmonized with EU directives (90/385/EEC [79], 93/42/EEC [80], and 98/79/EC [81]) for the European market. As a consequence of the new release, recertification was mandatory. The latest version, ISO 14971:2019 [40], is internationally endorsed again. This situation is particularly challenging for manufacturers of medical devices that need approval from the different regulatory bodies within and outside the EU.

The risk analysis phase in ISO 14971:2007 and EN ISO 14971:2012 corresponds to clause 4 and requires the planning of three tasks, i.e., 1) *Define use/safety characteristics*, 2) *Estimate risks* and 3) *Identify hazards*. In contrast, the same phase corresponds to clause 5 in ISO 14971:2019 and the task *Define use/safety characteristics* should be divided into two. For ISO 14971:2007, *the manufacturer shall discard the negligible risk*. Annexes of EN ISO 14971:2012 and ISO 14971:2019 provide a deviation, i.e., *the manufacturer shall consider all risks*. In all versions, *the manufacturer* is the role in charge, *the risk management plan* is the prerequisite of the clause, and the work products are *the risk analysis document* and the *risk management file*. The risk analysis document requires information regarding the *medical device description and identification, the identification of the person and organization, the scope, date, the intended use, and reasonably foreseeable misuse, the qualitative/quantitative safety characteristics of the medical device, known and foreseeable hazards associated with the medical device, fault conditions, reasonably foreseeable sequences of events,* and *the resulting hazardous situation*. Additional information is prescribed by ISO 14971:2019, i.e., *intended medical indication, patient population, part of the body/tissue, user profile,* and *operating principle*.

## 2.2 Process Models

Process models are the main artifacts used for supporting the management of the processes, as they provide an explicit representation of the process knowledge [25]. A process is a sequence of units of work (phases, activities, tasks, and steps) that consume resources (employee energy and time, infrastructure) to transform inputs (data, material) into value-added outputs (products, services, or information) [82]. For this reason, an appropriate representation of a process serves several purposes [27]. For example, a process model can provide support for analyzing and improving complex organizational processes. Moreover, as presented in Section 2.1, the model of a process plan can be often used as a mechanism to convince third parties (regulatory bodies, customers) about certain qualities of a product. A process model should [83]:

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. Process modeling languages (PMLs) are available to give process engineers the means to create process models and management tools to control them [84]. PMLs have been created from different perspectives, such as programming-based languages, Petri net-based languages, and rule-based languages [85]. In the remaining parts of this section, we present the process engineers, an actor in charge of the creation of processes (see Section 2.2.1). We also recall essential features of SPEM 2.0 (see Section 2.2.2), and EPF-C (see Section 2.2.3)

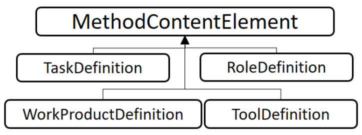
### 2.2.1 The Process Engineer

The process engineer is the role responsible for eliciting process knowledge, capturing this knowledge in a model, analyzing its execution, performing and disseminating process changes, and implementing systems to provide automated support to the process users [86]. Process engineers may adopt two different attitudes [87]: *descriptive* or *prescriptive*. A descriptive attitude requires the study of existing processes to characterize how they have been performed. A prescriptive attitude is instead related to the definition of a process to see how it should be developed. Commonly, a process engineer should adopt both approaches to deal with existing process representations and analyze process behavior, guide process users, enforce the rules (such as the ones prescribed by standards), or automate process steps [86].

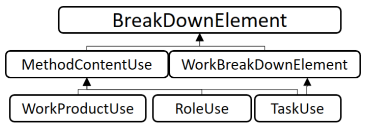
### 2.2.2 SPEM 2.0

SPEM 2.0 (Software and Systems Process Engineering Metamodel) [48] is a PML that defines the elements required for modeling software and systems processes. It is a good candidate to model processes mandated by standards, as demonstrated in [50] and, to some extent, it also supports the creation of compliance tables, i.e., the mapping between standard's requirements and process elements, as presented in [88, 89]. There is also the availability of tool support for the creation of SPEM 2.0-like elements, which permits the concretization of the models, i.e., EPF-C (recalled in Section 2.2.3).

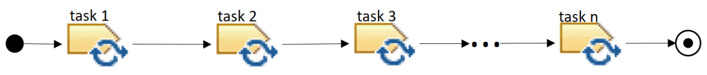
SPEM 2.0 concepts are defined in separated UML packages that are interrelated. For example, the meta-class *Task Definition*, which belongs to the package *MethodContentElement* (partially depicted in Figure 2.1a) is used to describe assignable units of work. Instances of *Task Definition* can be applied in a process breakdown structure by defining a proxy with a *Task Use*, a meta-class that belongs to the package *ProcessWithMethods* (partially depicted in Figure 2.1b). A similar approach is made for roles and work products. A tool definition is used to specify the tool's participation in a Task Definition.



(a) Method Content Elements Taxonomy.



(b) Work BreakDown Taxonomy.



(c) Activity Diagram Representing a Process Workflow.

Figure 2.1: SPEM 2.0 Taxonomies.

*Guidance*, which belongs to the package *Managed Content*, is a describable element that provides additional information to other elements. There are different guidance kinds, e.g., concept and reusable asset. A *Delivery Process*, which belongs to the package *Process Structure*, describes an approach for performing a specific project as a breakdown structure populated by different kind of units of work (e.g., a task). A *Category* is used to group elements in a recursive way. Some of the concepts previously mentioned are described with icons (see Table 2.2).

Table 2.2: Subset of Icons Used in SPEM 2.0.

| Task Definition/Use | Work Product | Delivery Process | Category | Role | Guidance |
|---|---|---|---|---|---|
| ▷ / ⟳ | 📄 | ⟳ | ▭ | 👤 | 📏 |

SPEM 2.0 supports variability management on breakdown structures and content elements, i.e., an element can be extended from a base that has a similar nature. We recall the variability mechanisms *extends* and *contributes*. With extends, the base method element inherits the attributes of the extended base element. Contributes is used to extend a base in an additive fashion.

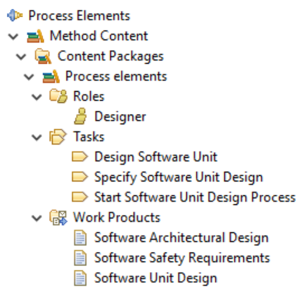
### 2.2.3 EPF-C

Eclipse Process Framework Composer (EPF-C) [47] is a stand-alone java application that uses the Eclipse Rich Client Platform (RCP) to implement UMA (Unified Method Architecture) Metamodel [90]. UMA is a metamodel that has been developed to provide the concepts and capabilities of different methods and process engineering languages. In particular, UMA contains a subset of SPEM 2.0 concepts (recalled in Section 2.2.2), providing capabilities for modeling systems and software processes. EPF-C is based on open standards and specifications to allow the exchange of process models specifications between different tools. In particular, XMI (XML Meta Interchange)[1] is used to store and exchange metadata in XML format. In addition, UML 2.0 Diagram Interchange Specification[2] is used to provide EPF-C with a proprietary activity diagram, which can be used to generate the execution semantics of a process from a process definition. The functionality of EPF-C mainly offers two capabilities, the method and the process authoring.

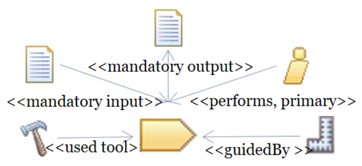
**Method Authoring:** Functionality used to capture a set of reusable building blocks, i.e., roles, tasks, work products, and guidance. Figure 2.2a shows an example of a plugin, called *Process Elements*. As the figure depicts, the plugin contains the definition of a role, i.e., *Designer*, three tasks, i.e., *Design Software Unit*, *Specify Software Unit Design* and *Start Software Unit Design Process* and three work products, i.e., *Software Architectural Design*, *Software Safety Requirements* and *Software Unit Design*. These elements are defined by using the subset of elements defined in SPEM 2.0 (see Table 2.2), which are also

[1] https://www.omg.org/spec/XMI/About-XMI/
[2] https://www.omg.org/spec/UMLDI/1.0/PDF

described by UMA metamodel. Conceptually, a task can be represented as a synergy between different process elements (see Figure 2.2b). Therefore, a task may have an used tool, mandatory input/output, a role that performs the task, and it is guided by guidance elements.


Process Elements
- Method Content
  - Content Packages
    - Process elements
      - Roles
        - Designer
      - Tasks
        - Design Software Unit
        - Specify Software Unit Design
        - Start Software Unit Design Process
      - Work Products
        - Software Architectural Design
        - Software Safety Requirements
        - Software Unit Design

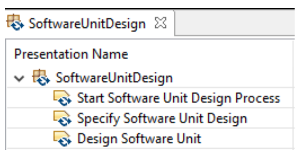
(a) EPF-C Plugin Structure.



(b) Elements relationships.

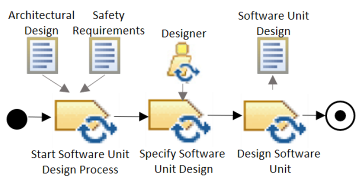
Figure 2.2: EPF-C Method Authoring.

**Process Authoring:** Functionality used to organize reusable process building blocks into processes by defining *Work Breakdown Structures*. As depicted in Figure 2.3a, a work breakdown structure describes the order of the units of work. With this functionality, we can also create process workflows. The workflow is represented as an *Activity Diagram* as depicted in Figure 2.3b.


SoftwareUnitDesign

Presentation Name
- SoftwareUnitDesign
  - Start Software Unit Design Process
  - Specify Software Unit Design
  - Design Software Unit

(a) Work Breakdown Structure.



(b) Activity Diagram.

Figure 2.3: EPF-C Process Authoring.

EPF-C implements the method plugin package, which defines the capabilities of modularization and extensibility. With these capabilities, processes and process elements defined in previous projects can be reused. EPF-C was recently ported from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 [91]).

## 2.3 Process Variability Management

In the context of safety-critical systems, a process exhibit variation due to several factors. For example, in automotive (see Figure 2.4) the growing connectivity of the systems that are included in cars is pushing towards the introduction of standards aimed at providing a baseline to address cybersecurity (e.g., SAE J3061) besides safety (i.e., ISO 26262) and software process capability maturity (i.e., Automotive SPICE). Consequently, manufacturers need to face process diversity [36], which is the management of the multiple reference models within single projects.

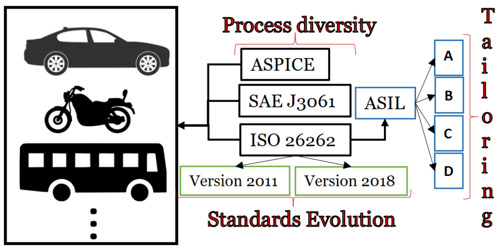


Figure 2.4: Process Variation in Automotive.

Manufacturers also need to perform tailoring. Tailoring requires selecting applicable requirements, performing their eventual modifications, and explaining their implementation according to the project's particular circumstances. In automotive (as recalled in Section 2.1.1) tailoring is commonly performed by taking into account the automotive-specific risk-based approach called ASIL. Moreover, new versions of the standards are frequently released. For example, there are already two versions of the standard ISO 26262, one released in 2011 and the other in 2018. Thus, processes assessed with the 2011 version may need recertification with the 2018 version.

Process diversity, tailoring, and standards evolution imply variation in the process plans used to engineer safety-critical systems. Such variation may imply the reuse of core assets that are common in the processes. In the remaining parts of this section, we recall the tool-supported methodological frameworks used in this thesis to manage the variability. In particular, we recalled SoPLE (see Section 2.3.1), which aims at systematizing reuse in process models. We also recall the support of EPF-C ∘ BVR-T (see Section 2.3.2), which is a composition of tools that can be used for supporting SoPLE and the variability management of processes.

### 2.3.1 SoPLE

SoPLE (Safety-oriented Process Line Engineering) [50] is a methodological approach that permits process engineers to systematize the reuse of process-related information. SoPLE builds on top of principles and concepts defined in PLE (Product Line Engineering) [92]. PLE is a reuse-oriented engineering method that consists of the systematization of commonalities and variabilities characterizing a set of products belonging to the same family/product line. Similarly, a process line is a set of processes that capture common reusable process elements and controlled variabilities. Thus, each of the family's processes is then developed from the common set of core assets (features) in a prescribed way [93]. SoPLE is constituted of two phases.

**1. Engineering reusable safety process-related commonalities and variabilities.** This phase requires that the process engineer selects those characteristics that are common in the process family. Then, the variabilities need to be also determined. For example, Table 2.3 (recalled from [94]) presents the comparison of activities between the description of the software unit design and implementation phase presented in the functional safety standard ISO 26262 (recalled in Section 2.1.1) and its counterpart presented in the cybersecurity guidebook SAE J3061 (recalled in Section 2.1.2). We called the activities that are common *Commonality Points (CP)*, and the ones that vary *Variability Points (VP)*. In the table, we can find four commonality points marked with the unique identifier CP1, CP2, CP3, and CP4. These commonality points can be extended with the variants that belong to the corresponding standard. For example, there is a variability point called VP1a, which contains the information regarding the ISO 26262 requirement, that corresponds to the first activity (IA1), *Design concerning safety*. Similarly, there is a variability point called VP1b, which contains the information regarding the SAE J3061 requirements, that corresponds to the first activity (JA1) *Design concerning cybersecurity*.

**2. Engineering single safety processes.** Through the selection and composition of previously engineered reusable process elements, we create specific processes for specific projects. First, we need to describe the skeleton that defines the software process line. As depicted in Figure 2.5, the skeleton is conformed by the Commonality Points, CP1, CP2, CP2, and CP4, which describes its essential structure. The variants are added to the commonality points by using the variability type called *contributes*, which is provided by SPEM 2.0 (recalled in Section 2.2.2). Currently, SoPLE is supported by the integration of

Table 2.3: Activities Comparison ISO 26262/SAE J3061.

| ID | IR | JR | Common Name |
|---|---|---|---|
| CP1 | IA1 | JA1 | Unit design |
| VP1a | IA1 | | Design concerning safety |
| VP1b | | JA1 | Design concerning cybersecurity |
| CP2 | IA2 | JA3 | Unit design review |
| VP2a | IA2 | | Design review concerning safety |
| VP2b | | JA3 | Design review concerning cybersecurity |
| CP3 | IA3 | JA2 | Unit implementation |
| VP3a | IA3 | | Unit implementation concerning safety |
| VP1b | | JA2 | Unit implementation concerning cybersecurity |
| CP4 | IA4 | JA4 | Unit implementation review |
| VP4a | IA4 | | Implementation review concerning safety |
| VP4b | | JA4 | Implementation review concerning cybersecurity |

EPF-C and Base Variability Resolution Tool (BVR-T), recalled in Section 2.3.2

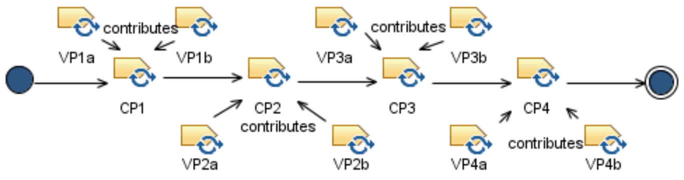


Figure 2.5: A Process Line Skeleton.

## 2.3.2 The Integration of EPF-C and BVR-T

EPF-C ∘ BVR-T [51] is a tool-chain obtained by integrating EPF-C (recalled in Section 2.2.3) and Base Variability Management (BVR-T)[3]. We focus on BVR-T. As summarized in [41], BVR-T is used to model (VSpec), resolve (Resolution) and realise (Realization) the variability.

**VSpec.** Permits users to model the variability in a feature diagram-like fashion, which defines distinctive user visible aspects, or characteristic of a pro-

[3]https://github.com/SINTEF-9012/bvr

cess. As Table 2.4 recalls, a choice represents a yes/no decision, a constraint, given in BCL (Basic Constraint Language), specifies restrictions on permissible resolution models, and a group dictates the number of choice resolutions. Relationships between a parent and its child features are of various types, i.e., 0.. *, refers to none or many selections; 1..1, which refers to alternative/xor in which one of the child features must be selected; 1..* which refers to or means that at least one of the child features must be selected. Relationships between the features can be mandatory, which means that the child feature is required, or optional, which means that the child feature can be or not selected.

Table 2.4: BVR Modelling Elements.

| **Choice** | **Constraint** | **Group** |
|---|---|---|
| ▭ | ▱ | △ |

Figure 2.6 presents an example of a BVR VSpec related to a car modeling (recalled from [78]). As Figure 2.6 depicts, the **Car** (represented by *choice* element) is compounded by **Engine** (which is either **HP140** or **HP110**, defined by the *grouping* element), **Parking_Assist** (which is present only in the cars with the type of engine HP140, as the *Constraint* element denotes) and Seats (denoted with a *Multiplicity* element that we do not use in our research).

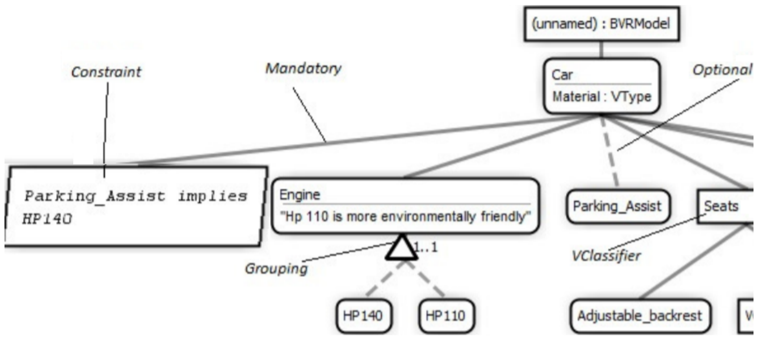


Figure 2.6: An Example of VSpec Model.

**Resolution.** Permits users to make choices at variation points, where desired variants can be selected. Resolution also includes the possibility to validate the choices. Erroneous choices violating the cross-variation points requirements can be detected. Figure *2.7* illustrates the same example provided in Figure *2.6*, but generated from the Resolution editor. Because of the constraint that was put in the VSpec model (as Figure *2.6* illustrates), the choice of HP110 is set to 'false', while the Parking_Assist choice is 'true'.

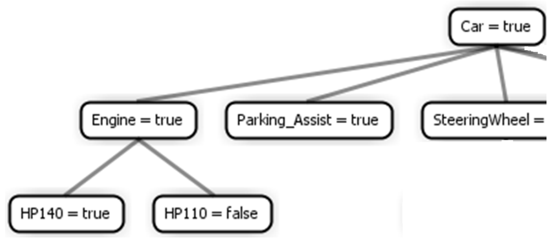


Figure 2.7: An Example of Resolution Model.

**Realization.** Permits users to bind conceptual resolutions with the concrete elements in the base model. In particular, it requires that users establish the relationship between placements and replacements within the fragment substitutions and the triggering of these substitution in the within the VSpec features.

## 2.4 Process-based Compliance by Design

Compliance assessment of process plans can be understood as the consistency between a specified process and the reference model embedded in a prescriptive standard [31]. There are two main approaches towards achieving process compliance [95]. The first one is called Compliance by Detection (CbDt). CbDt is an approach that entails a conformity check during or after the runtime stage (in the execution environment). In CbDt, the detection of non-compliant situations is problematic since it complicates the reconstruction of a process in a compliant form after the process has been executed. The second one is called Compliance by Design (CbD). CbD means that the set of applicable requirements are considered in the design stage of the process. In CbD, the conformity check takes place in advance when the process is planned. This approach has several advantages, as presented in [96].

1. The approach is flexible as the generation can be repeated when rules are added, removed, or changed.
2. Compliance is not only detected but actually enforced. Thus, CbD is considered a preventive approach.

In CbD, the requirements prescribed in a normative system are propagated into process plans [45]. For such propagation, the normative requirements have to be represented in a machine-readable form. In the remaining parts of this section, we present theoretical assumptions regarding normative representation (see Section 2.4.1). Then, we present Formal Contract Logic (FCL) (see Section 2.4.2), a language explicitly created for the representation or norms in legal and business compliance. Finally, we present Regorous (see Section 2.4.3), a compliance checker able to do reasoning with FCL rules.

### 2.4.1 Normative Representation

Normative documents contain principles of behavior used to regulate target subjects by defining what is legal and what is not [97]. Such documents arise from different sources, e.g., regulations, laws, standards, branch-specific guidelines, internal code of conduct, social and moral rules [98]. In particular, normative documents contain the conditions under which they are applicable (i.e., the meaning of the terms or concepts where the norms are valid), and the normative provisions they cause when applied [99]. Normative provisions correspond to the legally binding notions that are anchored to the structure of legislative text, laws, and regulations [100].

Normative documents are usually represented in natural language, which can freely describe a wide range of concepts of the real, abstract, and imaginary worlds [101]. Natural language is very expressive, but it has problems of ambiguity, subjectivity, inconsistency, and inaccuracy [102]. For this reason, formal models, which are a set of domain theorems that are amenable to formal proving through reasoning, are of growing interest for compliance checking [103]. Moreover, using formal methods provides rigorous methodologies that facilitate compliance tasks [94]. However, the analysis of compliance with normative frameworks is as good as the models used for such analysis [104].

In particular, the normative provisions of importance for process compliance are those notions regarding the *obligations*, the *permissions*, and the *prohibitions* [105] (see Figure 2.8). Obligations and prohibitions are constraints that limit the behavior of processes. The difference between obligations and prohibitions and other types of normative provisions is that they can finish in a violation. A permission is the lack of obligation to the contrary and cannot be violated. A violation could be compensated with a set of new obligations arising after the violation. Compensations are also obligations that can be violated and compensable as well. Thus, recursive definitions for the notion of compensated obligation is vital in the compliance analysis.

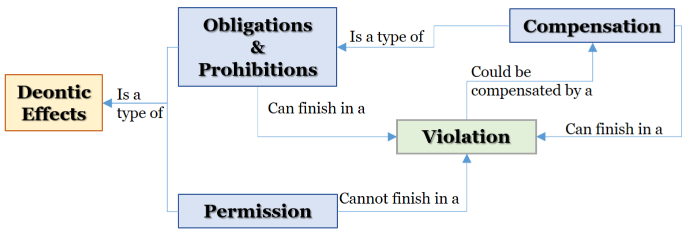


Figure 2.8: Normative Provisions Classification.

The formalization of normative documents is complex [106], and requires precise notions that can adequately describe them [101]. A formal model of the norms must have an unambiguous, mathematically defined syntax and semantics of such norms [107]. Several approaches aimed at addressing the representation of normative texts for system development have been created, (see [108]). We focus on Formal Contract Logic (FCL) (see Section 2.4.2), a framework that unambiguously represents normative knowledge as presented in Figure 2.8.

## 2.4.2 Formal Contract Logic

Formal Contract Logic (FCL) [46] is a language created to represent norms. An FCL rule is represented as follows:

$$r : a_1, ..., a_n \Rightarrow c,$$

where r is the name of the rule (unique for each rule), $a_1, ..., a_n$ are the premises (the conjunction of the propositions $a_1 \wedge a_2 \wedge ... \wedge a_n$ ), which represent the conditions of the applicability of the norm, and $c$ is the conclusion of the rule (also a proposition of the logic), which represents normative effects. The propositions of the logic are built from a finite set of atomic propositions, and the following operators: $-$ (for negation), $O$ (for obligation), $P$ (for permission), and $\otimes$ (for violation/reparation). The formation rules are as follows [45].

- Every atomic proposition is a proposition. A simple proposition corresponds to a factual statement.
- If p is an atomic proposition, then $-p$, is a proposition.
- If p is a proposition then Op is an obligation proposition and Pp is a permission proposition. Obligation propositions and permission propositions are deontic propositions;
- if $p_1, ..., p_n$ are obligation propositions and q is a deontic proposition, then $p_1 \otimes ... \otimes p_n \otimes q_n$ is a reparation chain. The conclusion $c$ could also represent a reparation chain.

FCL is a skeptical non-monotonic logic, meaning that it does not support contradictory conclusions but seeks to resolve conflicts. In case there is sustainable support to conclude both c and $-c$, FCL does not conclude any of them. However, if the support for c has priority over the support of $-c$, then c is concluded. This means that a designer of FCL rules has to identify pairs of incompatible propositions, such as c and $-c$. Once defined, a *superiority relation* ($>$) among rules is used to determine priorities (see Formula 2.1).

$$\begin{aligned} r : a_1, ..., a_n &\Rightarrow c, \\ r' : b_1, ..., b_n &\Rightarrow -c, \\ r' &> r \end{aligned} \tag{2.1}$$

An FCL rule can specify that an obligation is in force at a particular time point, i.e., a norm indicates when an obligation is active (see Figure 2.9). Thus, the obligation $O$ is in force at the point n in time t. An obligation is considered to remain in force until it is terminated or removed.

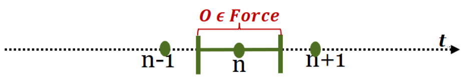


Figure 2.9: Obligation in Force.

FCL provides a classification model on temporal validity of the obligations and the effects of the violations. If an obligation needs to be obeyed for the whole duration within the interval t in which it is in force, it is categorized as a *maintenance obligation (OM)* (see Figure 2.10).

If achieving the content of the obligation at least once is enough to fulfill it, it is called *achievement obligation (OA)*. An OA is *Preemptive* if it could be

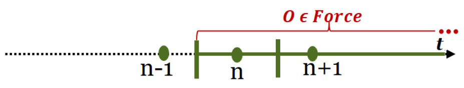


Figure 2.10: Maintenance Obligation.

fulfilled even before the obligation is in force. An OA is *non-preemptive* if it only can be fulfilled after it is in force (see Figure 2.11).

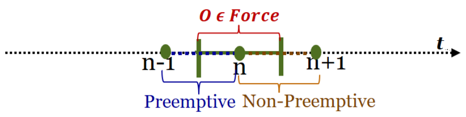


Figure 2.11: Achievement Obligation (Preemptive and Non-Preemptive).

An OA is *Perdurant* if after being violated, the obligation is still required to be fulfilled (see Figure 2.12). An OA is *Non-Perdurant* if after being violated, the obligation does not require to be fulfilled.

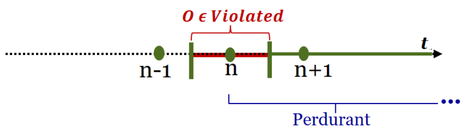


Figure 2.12: Achievement-Perdurant Obligation.

A *permission (P)* is an allowed situation. If something is permitted the obligation to the contrary does not hold. Prohibitions are forbidden situations that are represented as the negation of the content of a maintenance obligation $-OMp$. The different types of normative effects are summarized in Table 2.5.

Requirements in international standards (e.g., ISO 26262, ECSS standards, CENELEC EN 50128, MIL STD 882D [109]) commonly use the modal "shall" to define obligations, "shall not" to define prohibitions and "can" or "may" and "need not" (this is explicit in ECSS standards) to define permission. Thus, in the following, we present some everyday examples of the statements that could be modeled in FCL by using the types of obligations presented in Table 2.5.

1. **For the party in John's house, you can dress informally.** This statement is a permission. Thus, a representation of the statement in FCL could have as an antecedent the proposition that refers to the fact that there is a *party in*

Table 2.5: FCL Rule Notations.

| Notation | Description |
|---|---|
| [P]P | A proposition P is permitted |
| [OM]P | There is a maintenance obligation for the proposition P |
| [OM]-P | There is a prohibition for proposition P |
| [OAPP]P | There is an achievement, preemptive, and perdurant obligation for the proposition P |
| [OANPP]P | There is an achievement, non-preemptive and perdurant obligation for the proposition P |
| [OAPNP]P | There is an achievement, preemptive and non-perdurant obligation for the proposition P |
| [OANPNP]P | There is an achievement, non-preemptive and non-perdurant obligation for proposition P |

*John's house*, which conclusion is that there is a permit for *dressing informally* (see Formula 2.2).

$$r : PartyInJohn'sHouse \Rightarrow [P]dressInformally \tag{2.2}$$

2. **You <u>shall not</u> walk the dog in the avenue.** This statement is a prohibition. In FCL a prohibition is represented as the negation of the content of an OM. Thus, a representation of the statement in FCL could have as an antecedent the proposition that refers to the action *walk the dog*, whereas the conclusion is negation of the content of the obligation *walk the dog in the avenue* (see Formula 2.3).

$$r : walkTheDog \Rightarrow [OM] - walkTheDogInAvenue \tag{2.3}$$

3. **You <u>shall</u> pay the loan fee every month**. This statement is an OA because doing it once in the month is enough. It is also a *preemptive* because the fee could even be paid before the deadline. It is also perdurant because the no payment on time does not mean that should be not paid. Thus, a representation of the statement in FCL have as an antecedent the fact that the *month starts*, which conclusion is the *OAPP* to *pay the loan fee*(see Formula 2.4).

$$r : theMonthStarts \Rightarrow [OAPP]payLoanFee \tag{2.4}$$

4. **The package <u>shall</u> be registered once it arrives.** This statement is an OA since the package shall be registered only once. The OA is *non-preemptive* since you cannot register it before it arrives. In addition, if the package is not registered as soon as it arrives, there is a violating, but it does not

mean that the packages should not be registered anyway. Thus, the OA is also *perdurant*. A representation of the statement in FCL is presented in Formula 2.5.

$$r : PackageArrives \Rightarrow [OANPP]registerPackage \tag{2.5}$$

5. **You shall buy the machine with discount. Discounts are only today.** In this case, buying the machine only once is enough. However, you cannot buy before today, so, it is a OA that is *non-preemptive*. Moreover, if you do not buy today, you cannot buy tomorrow with discount. Thus the OA is also *non-perdurant*. A representation of the statement in FCL is presented in Formula 2.6.

$$r : BuyMachine \Rightarrow [OANPNP]buyingWtihDiscount \tag{2.6}$$

6. **You shall buy the machine with discount. The discounts finish next week.** In this case, buying the machine only once is enough to fulfil the obligation. Moreover, you can buy the machine before the discount time finish, so, it is a OA that is *preemptive*. If you do not buy the machine before the discounts finish, you cannot buy it with discount. Thus the OA is also *non-perdurant*. A representation of the statement in FCL is presented in Formula 2.7.

$$r : BuyMachine \Rightarrow [OAPNP]buyingWtihDiscount \tag{2.7}$$

### 2.4.3 Regorous

Regorous [49] is a checker that permits the automatic analysis of compliance of a process against a set of regulations formalized in FCL (recalled in Section 2.4.2). For this, the process should be enriched with semantic annotations. Unlike classic text annotation (used by humans to read associated information), machines can use semantic annotations to refer to and compute information. Annotations on process elements are literals that record data, resources, and other information representing compliance effects. The annotated process is taken by Regorous and converted in a logical state representation that is compared with the set of FCL rules containing the standards information (see Figure 2.13). The procedure executed by Regorous is the following:

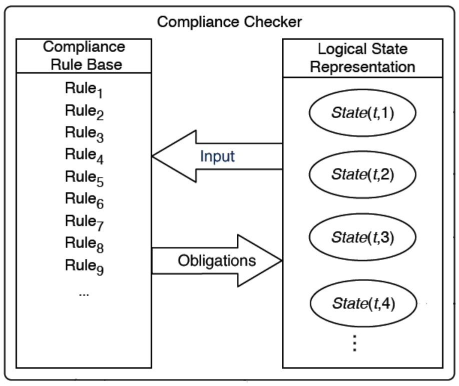


Figure 2.13: Regorous Architecture.

1. Generate an execution trace of the process.

2. Traverse the trace:

   - For each task in the trace, cumulate the effects of the task. Remark: if an effect in the current task conflicts with previous annotations, update using the effects of the current task.
   - Use the set of cumulated effects to determine which obligations enter into force at the current task. This is done by a call to of FCL reasoner.
   - Add the obligations obtained from the previous step to the set of obligations carried over the previous task.
   - Determine which obligations have been fulfilled, violated or pending, and if there are violated obligations, check whether they have been compensated.

3. Repeat for all traces.

A trace is a sequence of tasks in which a process can be executed. In particular, for the n-th element in a trace t, Regorous uses the function *State (t,n)* to semantically describe the set of facts in the computation to determine which rules are activated (or in other words, which rules fire). Consequently, obligations are in force after rules fire. Rules in force are described by Regorous with the function *Force(t, n+1)*. In addition, *Force (t,n)* contains the obligations that are in force but are not terminated in n. An obligation can be terminated for the following three reasons:

1. the obligation reaches its deadline

2. the obligation has been fulfilled

3. the obligation has been violated and it is not perdurant

To assess which obligations have been complied with or violated in n, Regorous compares the elements of Force(t, n) and State(t, n). A process is fully compliant if and only if all traces are compliant (all obligations have been fulfilled, or if violated, they have been compensated).

Figure 2.14, shows a fictional FCL rule set and a compliance annotated process. The ruleset in FCL contains four rules. The first rule, r1, implies the obligation of providing A. The second rule, r2, implies the obligation of providing B given the provision of A. The third rule, r3, implies the permission to not provide C given the provision of B. And r4 implies the obligation of D given the provision of B. From the FCL rule base, we have four compliance effects, i.e., A, B, C, and D. As seen, the compliance effects are extracted from the formulas composing the rules. The tasks in the process are annotated with the effects as follows. T1 is annotated with effect A, T2 is annotated with effect B, T3 is annotated with effect C, and T4 is annotated with effect D.

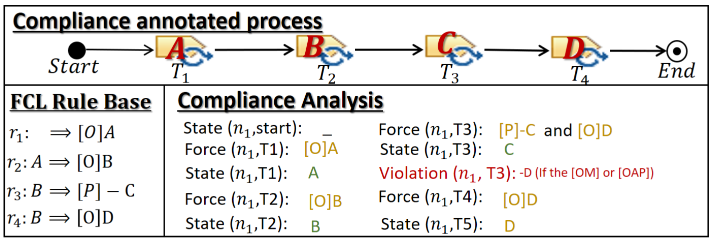


Figure 2.14: Analysis of Compliance.

To check compliance, we use the functions *State* and *Force*. The state of the start point is empty because we have not defined any effect. After the start point, the compliance checking process is activated. Thus, the first rule is in force. The first task is expected to provide the effect A since there is the obligation to provide A. Then, we check the state after the task T1. As we see in the figure, T1 produces the effect A. So, the rule is fulfilled. Then, providing A forces the provision of B in T2. In the figure, we can see that T2 provides effect B. So, the second rule is also fulfilled. After B is provided, it implies two normative effects. The first one is the permission to not providing C in T3. Second, it implies the obligation of providing D in T3. When checking T3, we can see that it provides the effect C. However, having C as the produced effect does not imply a violation of rule r3 because the force function has a permit, not an obligation. However, in T3, we should have D, and the tasks T3 is not providing E. If the obligation of providing D is a OM or OA-preemptive, we have a violation. A violation means that the process is not compliant. If the obligation is OA-non-preemptive, it can be fulfilled in T4. In this case, there is no violation, and the process is compliant.

## 2.5 Design Hints and Patterns

Design hints and patterns are two kinds of resources used in computational thinking [110]. Hints are rules of thumb used to solve a problem, while patterns indicate common situations that are likely to be encounter. Our research uses hints and patterns to facilitate creating the specifications required for automated compliance checking of process plans. In the remaining parts of this section, we present a hint widely used in computer science, namely the divide and conquer strategy (see Section 2.5.1). We also recall the notion of property specification patterns (see Section 2.5.2).

### 2.5.1 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 [111]. Particularly, divide-and-conquer is seen in the principle of separation of concerns [112], 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.

### 2.5.2 Property Specification Patterns

The property specification patterns, formulated by Dwyer et al.'s [113], are "generalized descriptions of commonly occurring requirements on the permissible state sequence of a finite state model of a system." A selected set of Dwyer et al.'s patterns[4] is presented in Table 2.6. Each pattern has a *scope*, which is the extent of the program execution over which the pattern must hold. The types of scope that we consider in this thesis are: *global*, which represent the entire program execution, *before*, which includes the execution up to a given state, and *after* which includes the execution after a given state.

Table 2.6: Dwyer's Specification Patterns.

| **Name** | **Description** |
|---|---|
| **Absence** | A given state P does not occur within a scope |
| **Existence** | A given state P must occur within a scope |
| **Universality** | A given state P must occur throughout a scope |
| **Precedence** | A state P must always be preceded by a state Q within a scope |
| **Response** | A state P must always be followed by a state Q within a scope |

## 2.6 Empirical Research Aspects

In this section, we recall the aspects related to empirical research in computer science that we use in this thesis. First, we present an overview of the design science methodology for information systems and software engineering (see Section 2.6.1). Then, we present the methods used in this thesis to validate our research outputs. In particular, we present the methodology for performing personal opinion surveys (see Section 2.6.2), systematic literature review (see Section 2.6.3), the technology acceptance model (see Section 2.6.4), the qualitative criteria to assess compliance approaches usefulness (see Section 2.6.5), and a metric called reuse measurement (See Section 2.6.6).

### 2.6.1 Design Science Methodology

The design science methodology for information systems and software engineering [114] aims at designing and investigating treatments (an artifact improving something in a problem context) used in software and information systems. Research problems in design science contain design problems (which call for a change in the real world) interacting with knowledge challenges (request understanding about the world as it is). Research problems also have a

[4]https://matthewbdwyer.github.io/psp/

problem context, including the social context (i.e., those who may affect or be affected by the treatment) and the knowledge context (i.e., existing theories).

### Design Cycle

The design cycle (see Figure 2.15) starts with a problem investigation, where the phenomena to be improved and the reasons for that improvement are studied. Then, there is a treatment design, where designs aimed at treat the problem are created. Finally, there is a treatment validation, where the suitability of the artifacts for the treatment is assessed. Researchers iterate over these activities many times in a design science research project.

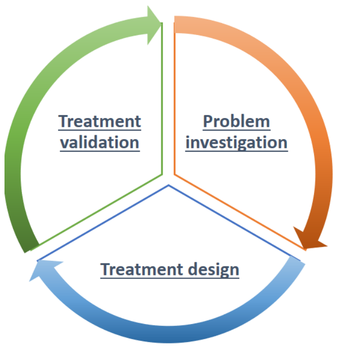


Figure 2.15: Design Cycle.

### Relevant Tasks

Peffers et al. [115] define a design science research methodology (DSRM) that is a slightly elaborated design cycle. The relevant tasks are presented in Table 2.7.

### Goals's Definition

According to Wieringa [114], design science also defines a range of goals. There is a social context goal at the higher level, which is a goal aimed at improving a problem context. To support this higher level goal, there may be several types of goals. For example, prediction goals (guess concerning what will happen in the future), knowledge goals (to answer knowledge questions),

Table 2.7: Peffers' DSRM Tasks.

| Task | Description |
|---|---|
| **Problem identification** | Define the research problem. |
| **Objectives definition** | Infer the objectives of a potential solution from the problem definition and knowledge of what is possible and feasible. |
| **Design and development** | Create an artifact (constructs, models, methods, instantiations or new properties of technical, social, and/or informational resources) by taking into account the knowledge of theory that can be brought to bear in a solution. |
| **Validation** | Demonstrate the use of the artifact to solve the problem. |
| **Evaluation** | Observe the degree of support provided by the artifact to solve the problem. |
| **Communication** | Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to relevant audiences. |

instrument design goals (to (re)design a research instrument), and artifact (or technical) research goals (to (re)design an artifact). Those types of goals have defined relationships. For example, to make predictions, we need knowledge. Thus, knowledge goals may also be required to describe phenomena and to explain them. For this, design science research projects require design instruments to answer knowledge challenges, e.g., personal opinion surveys, systematic literature reviews. Design science research projects also need to validate the suitability of the treatment, i.e., justify that it would contribute to stakeholder goals if implemented. The following sections recall the instruments used in this research to answer knowledge challenges and validate treatments.

### 2.6.2 Personal Opinion Surveys

A personal opinion survey [116] is a comprehensive research method for collecting information using a questionnaire completed by subjects. When creating a survey, the first step is to define the expected outcomes. Then, the survey should be designed, e.g., cross-sectional (participants are asked for information at one fixed point in time). It is also essential to define options related to how the survey would be administered. Once designed, the survey instrument should be developed, evaluated, and applied to a sample population, from which obtained data is analyzed.

In the creation of surveys, Likert Scales [117] are widely used. Likert Scales are psychometric response scales, e.g., a five-point scale ranging from "Strongly Disagree" to "Strongly Agree" used to ask respondents to indicate their level of agreement with a given statement. On a Likert scale, each specific question can have its response analyzed separately, or have it summed with other related items to create a score for a group of statements. Individual responses are generally treated as ordinal data because although the response levels do have a relative position, we cannot presume that participants perceive the difference between adjacent levels to be equal.

### 2.6.3 Systematic Literature Review

A Systematic Literature Review (SLR) [118] is a review methodology used for identifying, evaluating, and interpreting all available research relevant to a particular research question. The first activity in a SLR is *planning the review*. This activity includes the identification of the need for the SLR, the specification of the research questions, and the design of the review protocol. In the second activity, *conducting the review*, the researchers apply the review protocol and answer the questions. In the last activity, *reporting the review*, the researchers define the means to illustrate the findings. The guidelines, described by Wohlin [119], consider a technique called snowballing, which can be used to reach more relevant primary studies. Snowballing can be backward (search of relevant studies by taking into account the reference list of an initial set of primary studies) or forward (identifying more relevant studies based on those papers citing the paper being examined).

### 2.6.4 Technology Acceptance Model

The Technology Acceptance Model (TAM) [120] provides general determinants of computer acceptance. TAM is capable of explaining user behavior across a broad range of end-user computing technologies and user populations, while at the same time being theoretically justified. TAM focuses on three main facets of user acceptance. The first is the degree to which a person believes that using a particular method will be free of effort (Perceived Usability). The second is related to a person's subjective probability that using a particular system would enhance his/her job (Perceived Usefulness). The third is the extent to which a person intends to use a particular system (Intention to Use).

### 2.6.5 Qualitative Criteria to Assess Compliance Approaches

The usefulness of compliance management approaches can be qualitatively assessed taking into account specific criteria in terms of effort and level of coverage [54].

1. **Effort to model** needed to establish a model for managing compliance.

2. **Effort to comprehend** processes and regulations models.

3. **Effort to document compliance**, i.e., verify whether processes model comply with a legislation model.

4. **Effort to manage evolution** needed to find potential instances of non-compliance in the models when regulatory documents are amended or when their policies change.

5. **Level of coverage for the model**, how much of the policies and processes can be modeled.

6. **Level of coverage for compliance documentation**, i.e., approach's success in terms of documenting the compliance and ensuring compliance to the legislation.

7. **Level of coverage for the evolution management**, i.e., approach's success in handling the changes and assessing their overall impact.

The effort, according to to [121], is a variable that could be estimated during task performance in two ways: the actual effort (determined by task demands) and the perception of effort (relative to a subject's capacity to recognize the effort). In theory, the actual effort can be used to determine the intent to complete a task a priori, independently of any conscious actor. In this thesis, we use actual effort to gain a deeper understanding of our methods.

### 2.6.6 Reuse Measurement

A metric for reuse measurement is proposed by [57] (see below).

$$\%\,Reuse = (1 - \frac{Number\ of\ new\ objects\ built}{Total\ number\ of\ objects\ used}) * 100$$

The metric can be applied in hierarchical structures that permit the identification of the objects and the applications to which they were originally created. This metric is expressed in terms of percentage by considering the proportion of the number of new objects built (created from scratch) and the total number of objects used (in the absence of reuse). Besides, it focuses on the total benefit attributable to reuse. Thus, objects that are reused multiple times are considered to represent multiple instances of reuse.

# Chapter 3

# Research Summary

In this chapter, we present a summary of the research addressed in this thesis. In particular, we present the research process used (See Section 3.1), the problem formulation (see Section 3.2) and the research goals (see Section 3.3).

## 3.1 Research Process

The research process used in this thesis is based on the methodology for design science research for information systems and software engineering, described in Section 2.6.1. We adapt the design cycle presented in Figure 2.15 by including the tasks presented in Table 2.7. As a result, we have a research process that encompasses three stages. *A. Research Initiation* and *C. Research Finalization* are carried out once at a global scale, while *B. Research Development* is a cyclic stage of research aimed at fulfilling the general research goal

**A. Research initiation.** In this stage, we perform the problem investigation (first activity in the research cycle in Figure 2.15) at a global scale. We include two tasks described in DSRM (see Table 2.7).

1. **Problem identification.** In this task, we define the overall problem. For this, we require the *state-of-the-practice* (design problems which call for a change in the real world), and *state-of-the-art* (design problems must interact with knowledge challenges). The output is the *problem formulation*, which includes the problem context and the research motivation.

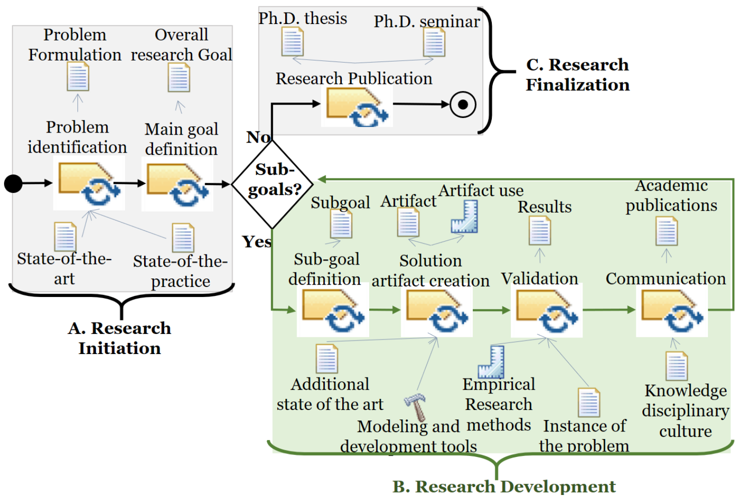


Figure 3.1: Research Process.

2. **Main goal definition.** In this task, we infer the overall research goal by considering the *problem formulation* and the knowledge of what is possible and feasible. The output of this task is the *overall research goal*.

**B. Research development:** This stage, which is composed by four tasks, supports the achievement of the main goal.

1. **Sub-goal definition.** A subgoal should describe a specific problem and justify the value of a specific solution. This task considers the *problem formulation* and the *overall research goal*, which are the work products of the research initiation. The output is the formulation of a *subgoal*, which defines a specific focus. As defined in Section 2.6.1 goals can be of different types. In this thesis, we define knowledge and technical research goals.

2. **Solution artifact creation.** In this task, we design an artifact, i.e., constructs, models, methods, or instantiations, new properties of technical, social, and/or informational resources, that solves the fulfill the specify subgoal. Within the artifact, its desired functionality, architecture, and actual development have to be described. Resources required for moving from the goal to creation of the artifact include knowledge of the theoretical approaches, i.e., *additional state of the art*, that can be brought to bear in a

solution and the *tools* required for modeling and development. The output of this task is the produced *artifact* and the guidance required for it use.

3. **Validation.** In this task, the solution artifact is used to solve a specific instance of the problem. We also observe the degree of support provided by the artifact when solving such a problem instance. The validation could involve the use of examples or other appropriate actions. Resources required for the validation include adequate knowledge of how to use the artifact, which is given by the *artifact use guidance*, and the actual *artifact*. Besides, the description of the *specific instance of the problem* and guidance related *empirical research methods* that would be used. In our research, we used different research methods for validation. In particular, we use methodologies for gathering data such as personal opinion survey (recalled in Section 2.6.2) and systematic literature reviews (recalled in Section 2.6.3). For the validation of the artifacts created, we used the technology acceptance model (recalled in Section 2.6.4), qualitative criteria to assess compliance approaches (recalled in Section 2.6.5) and the reuse measurement metric (recalled in Section 2.6.6). The output of this task is the *validation results*. These three steps are repeated for every research goal.

4. **Communication**. In this task, the problem and its importance, artifact, utility, novelty, its design, and capability to solve the problem (validation results) are communicated to the research community and practitioners. The *knowledge of the disciplinary culture* is a basic input for this task.

**C. Research finalization:** A Ph.D. position is a research tenure that is constrained by a time period. Results of this tenure should be visible at the end of such a period. For this reason, we define the stage research finalization, which is a communication activity as described in Table 2.7. In this stage, the activity **research publication** is performed. In this activity, we compile all the partial results obtained during the research development. The inputs of this stage are all the work products resulting from the research initiation and research development. The outputs are *Ph.D thesis* and the *Ph.D seminar*.

## 3.2 Problem Formulation

In this section, we formulate the problem to be tackled in this thesis. In particular, we define the problem context (see Section 3.2.1) and the research motivation (see Section 3.2.2).

### 3.2.1 Problem context

Manufacturers of safety-critical systems (see introduction of Chapter 2) have to comply with industry standards to show that they act under acceptable criteria. Compliance requires that manufacturers adapt their practices and provide evidence regarding the object of conformity (e.g., products, processes). In this thesis, we focus on the process used to engineer safety-critical systems as an object of conformity. In particular, adjusting process plans in compliance with applicable standards is a common, accepted, and mandatory procedure when producing safety-critical systems (see Section 2.1). However, selecting a strategy that fulfills business objectives in the process plan within the boundaries allowed by all the requirements included in the applicable normative documents is challenging (e.g., many standards, with different purposes and frequently changing). Therefore, there is a need to provide mechanisms for supporting the modeling of compliant processes at planning time. In the remaining parts of this section, we describe the object of study and the target stakeholder.

#### The Object of Study

Prescriptive standards, as recalled in Section 2.1, include requirements that demand the planning of tasks, the resources required and produced, and the use of specific techniques, which are framed with different properties. Such standards also set down the points at which different sorts of compliance artifacts should be established. For this reason, compliance checking is a common practice during the creation of process plans. Thus, the object of study in this research is the compliance checking of process plans against industry standards.

#### Target Stakeholder

One key consideration when conducting compliance assessment is that regulations are obligations on the licensee to fulfill in order to get authorizations. As such, there are two different sides: the industry side and the regulatory body side. Both sides involve different practitioners, i.e., process engineers, consultants, auditors, and evaluators. When process-related compliance is used to support contractual obligations between companies acting as customer and supplier, respectively, lawyers may also be involved. However, the main responsible for the processes in a company is the process engineer (as recalled in Section 2.2.1). We focus on such a role since a process engineer is responsible for selecting, composing, and correctly documenting adequate process elements and workflows to achieve the required process goals according to the applicable standards.

### 3.2.2 Research Motivation

Process-related compliance responds to a cycle of activities that start with selecting applicable standards to the demonstration of compliance with the selected standards. Such activities are very complex. In particular, compliance checking is typically considered time-consuming since standards contain hundreds of requirements, which usually reference each other. Compliance checking is also considered error-prone since there is much variation in process planning (see, for instance, Figure 2.4). Finally, compliance checking is a repetitive job since industries commonly manage multiple process plans. Given such complexity, errors in process-related compliance checking reports may arise. Such errors may endanger the compliance process, leading to delays in the production of safety-critical systems and, thus, economical losses.

The provision of solutions for automatizing compliance checking are imperative to permit organizations to have more control over their process plans and avoid compliance risk, by:

1. detecting compliance violations, and

2. enforcing compliance at planning time.

## 3.3 Research Goals

In this section, we identify the main research goal (see Section 3.3.1) and the derived research subgoals (see Section 3.3.2).

### 3.3.1 Main Goal

As presented in Section 3.2.1, we aim at addressing process compliance at planning time to help process engineers do their job. As presented in Section 3.2.2, we consider the provision of solutions that facilitate the automation of compliance checking of processes so that compliance violations are detected and compliance is enforced at planning time. With this, a process engineer can demonstrate intentional compliance, i.e., the planning-time allocation of responsibilities, such that if every actor fulfills its duties, then compliance is ensured. Based on the previous reasoning, we define a social context (main) goal that drives this thesis as follows:

> **Facilitate automated compliance checking of the process plans used to engineer safety-critical systems against the standards mandated (or recommended) in the safety-critical context**

### 3.3.2 Research Sub-Goals

In order to address the main goal (presented in Section 3.3.1), we define four concrete sub-goals. As recalled in Section 2.6.1, different kinds of goals can be defined in the design science methodology. In particular, we defined knowledge goals, which help us explain the phenomena under study and the impact of the proposed solution (subgoals 1 and 4). We also define technical goals since we aim at proposing a technological solution (subgoals 2 and 3).

**1. Elicit the requirements to be met to support the automated compliance checking of the process plans used in the safety-critical context.** This subgoal focuses on establishing the requirements for a technical solution that alleviate the compliance challenges that manufacturers of safety-critical systems have in terms of prescriptive standards, specifically at planning time.

**2. Identify mechanisms for supporting automated compliance checking of the process plans used in the safety-critical context.** This subgoal focuses on discovering how existing tools and methodologies can be appropriately combined to provide the support required for automated compliance checking of process plans.

**3. Facilitate the creation of reusable specifications required for automated compliance checking of the process plans.** This subgoal focuses on discovering adequate means to model the concepts included in the specifications, as well as to include the necessary support for enabling systematic reuse.

**4. Investigate the significance of a solution for automated compliance checking of process plans in the safety-critical context.** This subgoal focuses on objectively analyze our solutions' pros and cons and define elements that permit their future improvement and usage.

# Chapter 4

# Thesis Contributions

In this chapter, we present the technical contributions presented this thesis. The contributions are matched with the subgoals in Table 4.1.

Table 4.1: Contributions and Research Subgoals.

| No. | Contribution | Goal addressed |
|---|---|---|
| 1 | Requirements for Automated Process Compliance Checking (see Section 4.1) | Elicit the requirements to be met to support the automated compliance checking of the process plans used in the safety-critical context |
| 2 | ACCEPT (see Section 4.2) | Identify mechanisms for supporting automated compliance checking of the process plans used in the safety-critical context |
| 3 | Process Compliance Hints and Patterns (see Section 4.3) | Facilitate the creation of reusable specifications required for automated compliance checking of the process plans. |
|  | Systematic reuse of compliance artifacts (see Section 4.4) |  |
| 4 | Solutions validation (see Section 4.5) | Investigate the significance of a solution for automated compliance checking of process plans in the safety-critical context. |

## 4.1 Requirements for Automated Process Compliance Checking

In this section, we establish the requirements for a technical solution that alleviates the challenges that manufacturers of safety-critical systems have regarding prescriptive standards. In particular, in Section 4.1.1, we consider the compliance reasons of a manufacturer of safety-critical systems. In Section 4.1.2, we highlight the process plan. In Section 4.1.3, we emphasize the challenges regarding compliance checking of process plans. Finally, in Section 4.1.4, we present the requirements for a technical solution.

### 4.1.1 Compliance Reasons

When producing safety-critical systems, manufacturers seek to fulfill the compliance requirements provided by a set of applicable normative frameworks (e.g., standards, policies, regulations) to demonstrate that they act under publicly accepted criteria. For example, compliance with functional safety standards represents that the risks associated with the safety-critical system operation have been considered, analyzed, and mitigated to a sufficient level for their confident deployment into a specific context in the society.

For some industries, compliance is not a simple voluntary act. For example, software aspects of airborne systems and equipment need to comply with airworthiness requirements described in DO-178C. Conversely, in automotive, compliance certifications with a specific regulatory framework are not de rigueur. However, since its inception in 2011, ISO 26262 has been considered the state-of-the-art automotive safety engineering. Thus, self-assessment with ISO 26262 is essential in the production and commercialization of cars. In any case, compliance with industry standards grants several benefits to manufacturers involved in the production of safety-critical systems. For example:

1. Compliance requirements provide manufacturers a reference point for their production operations based on expertise and best practices. For instance, the functional safety standard ISO 26262 prescribes the specific actions to be taken into consideration by a car manufacturer to achieve acceptable safety levels in their fabricated or assembled vehicles.

2. Compliance requirements help to realize the ever more globalized safety-critical systems production more confidently. For example, in the European space context, the ECSS standards provide baseline requirements that space agencies use to formulate the project-specific needs and requirements that have to be met by their worldwide distributed suppliers.

3. A compliance stamp is a mark that customers trust. For instance, the CE marking [1], which is the declaration that a product meets European standards

[1] https://ec.europa.eu/growth/single-market/ce-marking_en

for health, safety, and environmental protection, is commonly preferred by European consumers.

4. A standard-compliant safety-critical system support manufacturer in legal actions if the product causes harm. For example, compliance with IEC 61508, which is the functional safety of electrical/electronic/programmable electronic safety-related systems, is relevant evidence for a jury to consider in a product liability action in England.

### 4.1.2 The Process Plan

Industry standards demand documented evidence of responsibilities and agreements. Commonly, standards adopt a highly prescriptive approach where the decision-making process is facilitated by integrating desired attributes, e.g., safety, security, and quality, into the development lifecycle of the safety-critical systems. For this reason, manufacturers should plan the fulfillment of standards requirements at the beginning of the engineering activities. By following such an approach, manufacturers can ensure that a rigorous process has been designed to build the system. For example, safety standards demand a safety plan, which contains the planning of activities related to the assurance of the system's safety. Compliant engineering process plans are used to coordinate and track engineering progress, and in some contexts, e.g., avionics, they should get initial approval from regulatory bodies.

Rigorous compliant process plans should include the sequence of tasks mandated by standards (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). Task, resources, and properties outside of the standards prescriptions are also possible to plan, but they require a rationale in which compliance should be justified. Consequently, a process plan is a detailed proposal of action that demonstrates the planning-time allocation of responsibilities. Given these characteristics, process-related compliance management could be supported by checking at design time that processes planned to engineer safety-critical systems fulfill the properties set down by standards at given points.

### 4.1.3 Compliance Checking Challenges

The process reference frameworks included in prescriptive standards focus on what needs to be done, who should be involved in the process, and the recommended techniques to be used to achieve desirable results. In some contexts, tool qualification is also required, e.g., in avionics, the standard annex DO-330 defines that tool qualification is in itself a process necessary to obtain qualification credit. Still, standards do not restrict organizations from using a particular development process, e.g., organizations can develop safety-critical software by using plan-driven or agile methodologies if they consider it appropriate.

The standards may outline some guidance that organizations can use in their application. However, a concrete plan of action or exact specification on how the process should be done is usually not provided. Thus, a process engineer is responsible for selecting, composing, and documenting process plans aimed at achieving the organization's goals while complying with the applicable standards' requirements.

The creation of compliance checklists facilitates this job. An accurately filled-in compliance checklist highlights missed requirements providing hints to improve the compliance of a process plan. However, manually checking compliance of process plans could be challenging. We identify five aspects that contribute to making this task difficult.

1. **Requirements Volume.** Standards contain a sheer volume of requirements. For instance, the ECSS compliance matrix V. 0.8 contains 38569 requirements[2].

2. **Interlaced Requirements.** Requirements in one standard usually refer to other requirements in the same standard or other standards. For example, the standard CENELEC EN 50128 also refers to quality management and continuous improvement of the systems, usually described in the Software Process Improvement and Capability Determination (SPICE).

3. **Contradictory/Ambiguos Requirements.** Normative frameworks are written in natural language by people who have a specific context in their minds. Requirements in one part of the standard may contradict requirements in another part of the same standard or other applicable standards. Sometimes, there may also be ambiguities derived from the difficulty that interpreting natural language involves. Contradictory/Ambiguous requirements are challenging to highlight due to the requirements volume (challenge 1) and interlaced requirements (challenge 2).

4. **Requirements diversity.** Requirements from the same or different standards may apply to one or several projects at different times and jurisdictions. See, the following cases.

[2]See https://ecss.nl/standards/downloads/earm/

- *Standards evolve*, i.e., there is a continuous release of new versions of the standards. For example, the content of ISO 14971 (recalled in Section 2.1.4), which defines the process for risk management of medical devices, has been evolving over the years resulting in different versions that apply to different jurisdictions.
- To apply standards, a *tailoring process* is usually required. Tailoring is the activity of selecting applicable requirements from relevant standards, performing their eventual modifications, and explaining their implementation according to the project's particular circumstances. For example, ISO 26262 (recalled in Section 2.1.1) required tailoring requirements according to integrity levels of safety, which could be different for different developed artifacts.
- *Simultaneous use of multiple reference processes* within a single project is commonplace. For instance, in automotive, it is recommended to manufacturers to comply with the functional safety standard ISO 26262, the process improvement framework ASPICE, and cybersecurity guidelines, such as SAE J3061 (see Figure 2.4).

5. **Repetitive work.** Industries commonly manage multiple process plans and, thus, perform compliance checking repeatedly. In each case, if there is no a mechanism to manage appropriately the information regarding standards requirements, (re)interpretations may be needed (e.g., when the expert leaves the company).

### 4.1.4 Requirements for a Technical Solution

A technical solution that alleviates the challenges described in Section 4.1.3 could support the process engineer when planning of compliant processes in the safety-critical context. In particular, the desired solution:

1. shall facilitate the management of the artifacts required for compliance checking with prescriptive standards, i.e., the standards themselves, their requirements, the processes plans and the compliance means (challenge 1 and 5),
2. shall facilitate keeping track of the applicable requirements (challenge 2),
3. shall facilitate the recognition of contradictions and ambiguities between applicable requirements (challenge 3), and
4. shall facilitate managing the changing nature associated with requirements diversity (challenge 4).

This contribution was initially defined in paper A (see Chapter 7)). Then, it was refined in papers B (see Chapter 8), C (see Chapter 9), D (see Chapter 10), and E (see Chapter 11).

## 4.2 ACCEPT

In this section, we present ACCEPT, which stands for Automated Compliance Checking of Engineering Processes against sTandards. ACCEPT is an iterative, comprehensible, and reusable framework for supporting process engineers in creating compliant process plans. In the remaining parts of this Section, we present the details regarding the definition of ACCEPT. In Section 4.2.1, we identify the conditions for automatically checking compliance. In Section 4.2.2, we identified the tool-supported methodological approaches that have the potential to back up the previously identified conditions. In Section 4.2.3, we present the design of a framework that defines the ACCEPT structures. Finally, in Section 4.2.4, we describe the methodological steps necessary to use ACCEPT.

### 4.2.1 Conditions for Automatically Checking Compliance

We have adopted compliance by design (recalled in Section 2.4), which is a state-of-the-art methodology considering the propagation of normative requirements into process elements for demonstrating compliance at design time, i.e., process compliance before the process is executed. As such, two specifications are required (see Figure 4.1), i.e., the specification that describes the requirements prescribed by the standards (on the right side) and the specification that describes the process plans (on the left side).

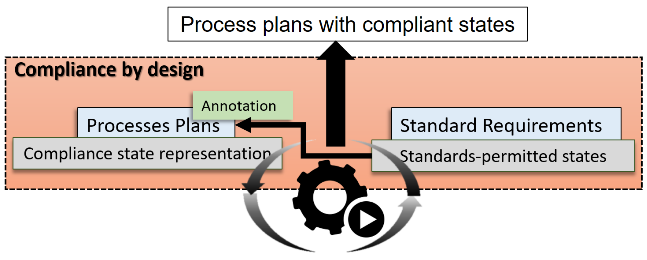

Process plans with compliant states
Compliance by design
Annotation
Processes Plans
Compliance state representation
Standard Requirements
Standards-permitted states

Figure 4.1: Conditions for Automating Process Compliance Checking.

The propagation of normative requirements is possible via the annotation of compliance effects. Compliance effects correspond to the standard-permitted states, which are extracted from the specification corresponding to the standard requirements. Unlike other effects caused by process elements, compliance effects should correlate with the standard-permitted states. For this reason, the annotation process is an alignment between the states exhibited by the process elements and the standard-permitted states. Properly annotated compliance effects allow the generation of a compliance state representation of the process, which can be used to perform the automatic analysis of compliance. A process engineer can iteratively apply automated compliance checking to reach process plans with compliant states.

### 4.2.2 Tool-supported Methodological Approaches

The identified tool-supported methodological approaches that have the potential to back up the conditions presented in Figure 4.1 are defined as follows.

**Process Modeling and Annotation Capabilities.**

A process engineer (as recalled in Section 2.2.1) is the person who deals with representing of the processes used to engineer safety-critical systems. However, a process representation can be composed of hundreds of entities. Process modeling languages (see Section 2.2) have been developed to provide process engineers more control over their processes. We chose SPEM 2.0 (recalled in Section 2.2.2), as opposed to other process modeling notations, because:

1. SPEM 2.0 is a standardized language, based on the UML;
2. SPEM 2.0 has the ability to capture several types of information;
3. SPEM 2.0-like artifacts can be captured via Eclipse Process Framework Composer (EPF-C) (recalled in Section 2.2.3). EPF-C models can be ported to other tools, via model-driven transformations.
4. SPEM 2.0 provides variability mechanisms that can be exploited for flexible process derivation. Such mechanisms are currently tool-supported via the composition of EPC-C with BVR (recalled in Section 2.3.2).
5. SPEM 2.0 elements can also be customized to permit the definition of artifacts beyond process elements. e.g., standards, requirements, rules and their compliance effects. In addition, EPF-C is an environments that provides hierarchal structures that facilitates keeping track of such artifacts.

6. SPEM 2.0 is widely accepted by the research community and industry.

### Normative Representation Capabilities

Norms typically describe the conditions under which they are applicable, i.e., the meaning of the terms or concepts where the norms are valid. Norms also describe the normative effects they produce when applied, i.e., legally-binding effects. From a compliance perspective, the normative effects of importance are deontic, which have intrinsic relations (as depicted in Figure 2.8).

There are three primary deontic effects. First, we have the obligations and prohibitions, which are situations to which the bearer is legally bounded, or that should avoid. In industry standards, obligations and prohibitions are commonly defined by the modal "shall", and "shall not," respectively. Second, we have permissions, which are allowed situations. Industry standards commonly use "can" or "may" or "need not" to define that something is permitted. It is also essential to provide reasoning about normative violations and exceptions. Normative violations correspond to the failure in fulfilling the requirements, while exceptions are conditions that emerge in situations not considered by the standard. In particular, exceptions defeat the established normative provisions. Thus, we need to be able to represent also this type of information.

The normative provisions included in industry standards can be encoded as implications. A normative implication has an antecedent, which is read as a property of a state of affairs, while the conclusion has a deontic nature. If one rule defeats another rule (i.e., rules contradict each other, as in the case of exceptions), there must be a mechanism that permits plausible conclusions. Giving these circumstances, we argue that deontic defeasible reasoning formalisms, such as Formal Contract Logic (FCL) (recalled in Section 2.4.2), can be used to generate automatic support to reason from standard's requirements and the description of the process they regulate.

### Reasoning Capabilities

As recalled in Section 2.4.3, Regorous is a tool-supported methodology for compliance checking designed and implemented in the legal and business context. Regorous is of particular interest since it implements compliance by design, which, as described in Section 4.2.1, is a methodology that is adequate for facilitating the planning-time allocation of mandatory pieces of evidence required by the majority of industry standards. In the process of compliance checking, Regorous provides constructive proofs (a proof from which it is possible to trace its derivation). Constructive proofs permit process engineers to trace back the sources of uncompliant situations (or violations), facilitating their detection and enforcing compliance. Regorous methodology is process modeling language agnostic, i.e., only requires a process description that contains compliance effects. Thus, it is theoretically adaptable with SPEM 2.0.

### 4.2.3 Compliance Checking Framework

In this section, we present the framework (see Figure 4.2) designed from the assumptions defined in Section 4.2.1 and the identified tool-supported methodological approaches presented in Section 4.2.2.

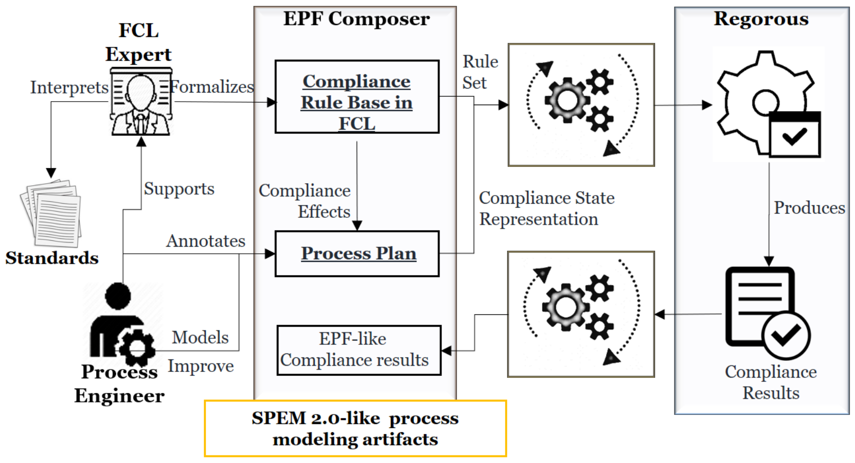


Figure 4.2: Compliance Checking Framework.

As Figure 4.2 depicts, EPF-C is combined with Regorous. In EPF-C, the process engineer can create process models, and an FCL-trained person can represent the requirements of the standard in FCL (a.k.a. FCL rule set).

The FCL rule set contains the rules and the requirements from which they are derived and the normative provisions, which are the source of compliance effects. The process engineer uses the normative provisions included in the FCL rule set to annotate the process models and create a compliance state representation of the process plan.

The FCL rule set and the compliance state representation of the process plans are then provided in the Regorous syntax by performing a set of automated transformations. With this information, Regorous automatically checks compliance and produces compliance results that include the violations occurring concerning the FCL rule set provided and the possible resolutions. Such results are aimed at being backpropagated into EPF-C to facilitate analysis and improvement of process plans.

### 4.2.4 Methodological Steps

Performing compliance checking with ACCEPT requires the application of the five methodological steps shown in Figure 4.3. In the remaining part of this section, we explain each step.

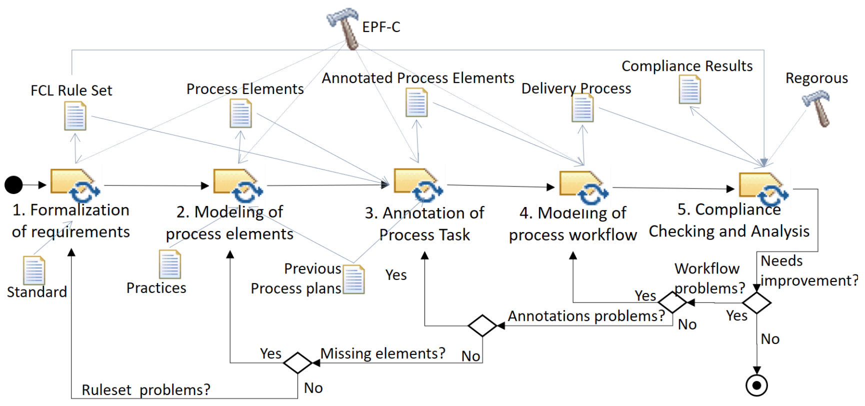


Figure 4.3: Methodological Steps.

**Step 1: Formalization of Requirements.** Standard requirements are formalized in FCL by the person(s) with FCL and domain knowledge (e.g., process engineers). This step has two inputs. The first input is the normative document (i.e., industry standard) from which the rules are derived. The second input is the list of customized compliance elements that we have designed to facilitate the representations of requirements in EPC-C (see Figure 4.4). In particular, those artifacts are SPEM 2.0-like elements that represent the elements required for modeling compliance information, i.e., compliance effects, rule sets, requirements, and rules. The output is the FCL-based ruleset captured in an EPF-C plugin.

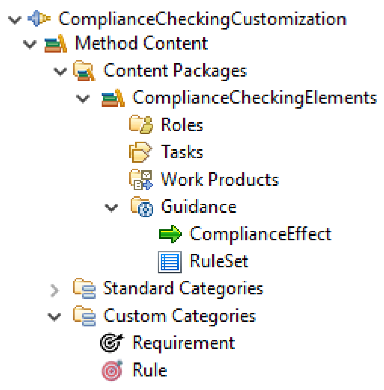

ComplianceCheckingCustomization
Method Content
Content Packages
ComplianceCheckingElements
Roles
Tasks
Work Products
Guidance
ComplianceEffect
RuleSet
Standard Categories
Custom Categories
Requirement
Rule

Figure 4.4: Elements Customization.

With our initial assumptions on formalization of requirements, we present an example taking into account two requirements (see Table 4.2) of the standard ISO 26262 (recalled in Section 2.1.1)

Table 4.2: Requirements for ISO 26262:6-Clause 8.

| **ID** | **Description** |
|---|---|
| R1 | Specify software units in accordance with the architectural design and the associated safety requirements. |
| R2 | The software unit design shall be described using specific notations, according to ASIL and recommendation levels. |

Initially, we create a rule that represents the obligation to address the software unit design process required by the standard (see Formula 4.1). For simplification, we use the requirement ID described in the first column of Table 4.2 to identify the requirements and their derived rules.

$$r_1 :\Rightarrow [OAPNP]addressSwUnitDesignProcess \tag{4.1}$$

Requirement R1 mentions that the specification of the software units has to be done in accordance with the architectural design and the associated safety requirements. The expression *in accordance with* recalls the concept of precondition, namely a task is prohibited until the previous tasks or elements are provided. In this sense, architectural design and safety requirements have to be provided before the software units are specified (see Formula 4.2).

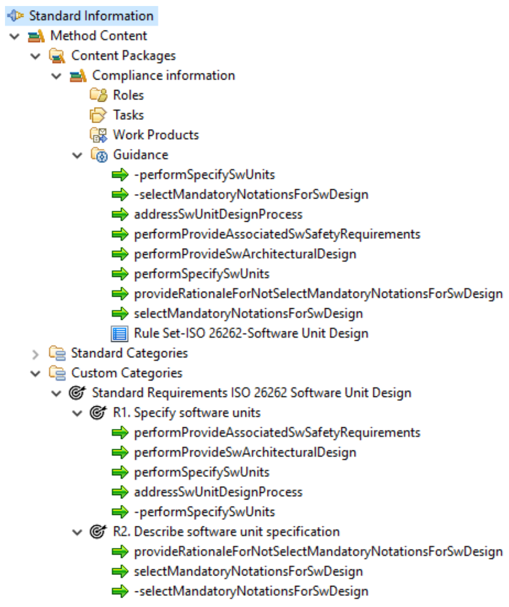
$$\begin{aligned} r_1a &: addressSwUnitDesignProcess \Rightarrow [OA] - performSpecifySwUnit \\ r_1a' &: performProvideSwArchitecturalDesign, performProvideSwSafetyRequirements \\ &\Rightarrow [P]performSpecifySwUnit \\ &r_1a' > r_1a \end{aligned} \tag{4.2}$$

Requirement R2 implies the use of mandatory methods in the description of software units which are conditioned by the ASIL and the recommendation levels. However, a rationale can be given that the selected methods (different to the ones prescribed) comply with the corresponding requirement. The formalization of the R2 is presented in Formula 4.3.

$$\begin{aligned} r_2 &: performSpecifySwUnit \Rightarrow [OAPNP]selectMandatoryNotationsforSwDesign \\ r_2' &: provideRationaleForNotSelectMandatoryNotationsforSwDesign \\ &\Rightarrow [P] - selectMandatoryNotationsforSwDesign \\ &r_2' > r_2 \end{aligned} \tag{4.3}$$

The previous requirements, and their derived FCL rules are captured in an EPF-C plugin as presented in Figure 4.5. In this plugin, we define a custom category root called *Standard Requirements ISO 26262 Software Unit Design*, to which we associate the two requirements (as nested custom categories) presented in Table 4.2 with a short but descriptive name, i.e., *R1. Specify software units* and *R2. Describe software unit specification*. Both requirements are described further in their respective *Brief description* field. Then, we extract the compliance effects from the Formulas 4.1, 4.2 and 4.3. Recalling, compliance effects correspond to the conclusions that compound the rules. For example, the compliance effects associated to the Formula 4.2 are *performProvideAssociatedSwSafetyRequirements*, *performProvideSwArchitecturalDesign*, *-performSpecifySwUnits* and *performSpecifySwUnits*. Compliance effects are defined as a customized *concept* and then assigned to their corresponding requirements in the custom categories (see Figure 4.5). Finally, we define the rule set in a customized *reusable asset* (called *Rule Set-ISO 26262-Software Unit Design*).

**Step 2: Modeling of Process Elements** Capturing a process plan elements is a task performed by the process engineer. The required input is information about process plans, which could steam from the organization's practices

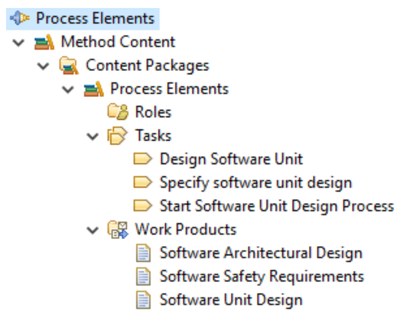


```
Standard Information
  Method Content
    Content Packages
      Compliance information
        Roles
        Tasks
        Work Products
        Guidance
          -performSpecifySwUnits
          -selectMandatoryNotationsForSwDesign
          addressSwUnitDesignProcess
          performProvideAssociatedSwSafetyRequirements
          performProvideSwArchitecturalDesign
          performSpecifySwUnits
          provideRationaleForNotSelectMandatoryNotationsForSwDesign
          selectMandatoryNotationsForSwDesign
          Rule Set-ISO 26262-Software Unit Design
    Standard Categories
    Custom Categories
      Standard Requirements ISO 26262 Software Unit Design
        R1. Specify software units
          performProvideAssociatedSwSafetyRequirements
          performProvideSwArchitecturalDesign
          performSpecifySwUnits
          addressSwUnitDesignProcess
          -performSpecifySwUnits
        R2. Describe software unit specification
          provideRationaleForNotSelectMandatoryNotationsForSwDesign
          selectMandatoryNotationsForSwDesign
          -selectMandatoryNotationsForSwDesign
```

Figure 4.5: Standard's Requirements Plugin.

and previous process plans. The output is the representation of the process elements in EPF Composer. In this example, the definition of the process elements is based on our interpretations of the standard's requirements provided in Table 4.2. From R1, we deduce that there is one task called *Specify Software Unit*. These tasks should be preceded by a task called *Start Software Unit Design Process*, in which the requirements for the process is collected, namely, the *Software Architectural Design* and its corresponding *Software Safety requirements*, which are work products resulting from previous phases. From R2, we deduce that we have a task called *Design Software Unit* and a work product called *Software Unit Design*. Work products are associated to their corresponding tasks. The modeling of the tasks and the work products is depicted in Figure 4.6.

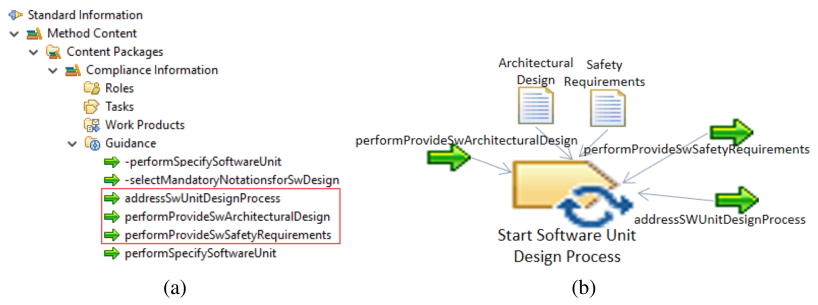
**Step 3: Annotation of Process Tasks** The annotation is a task performed by a process engineer. Previous process plans, as well as compliance effects contained in the FCL rule set, and process elements, are the inputs of this



- Process Elements
  - Method Content
    - Content Packages
      - Process Elements
        - Roles
        - Tasks
          - Design Software Unit
          - Specify software unit design
          - Start Software Unit Design Process
        - Work Products
          - Software Architectural Design
          - Software Safety Requirements
          - Software Unit Design

Figure 4.6: Process Elements Plugin.

step. The output is the annotated process tasks in EPF Composer. The annotation process consists of assigning the compliance effects to the tasks that fulfill them. For example, the task *Start Software Unit Design Process* (see Figure 4.7b) is performed and has two inputs, i.e., architectural design and safety requirements. Thus, the annotated compliance effects (see Figure 4.7a) are *addressSwUnitDesignProcess*, *performProvideSwArchitecturalDesign* and *performProvideSwSafetyRequirements*.



- Standard Information
  - Method Content
    - Content Packages
      - Compliance Information
        - Roles
        - Tasks
        - Work Products
        - Guidance
          - -performSpecifySoftwareUnit
          - -selectMandatoryNotationsforSwDesign
          - addressSwUnitDesignProcess
          - performProvideSwArchitecturalDesign
          - performProvideSwSafetyRequirements
          - performSpecifySoftwareUnit

Architectural Design
Safety Requirements
performProvideSwArchitecturalDesign
performProvideSwSafetyRequirements
addressSWUnitDesignProcess
Start Software Unit Design Process

(a) (b)

Figure 4.7: Compliance Annotated Process Plugin.

**Step 4: Modeling of Process Workflow** For modeling the process workflow, the process engineer uses the compliance annotated process elements defined in step 3. The output is the delivery process in EPF Composer, which contains the process plan checkable for compliance, i.e., the compliance state representation of the process plan (as depicted in Figure 4.8).

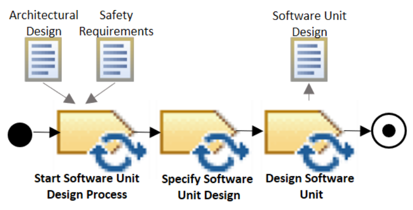
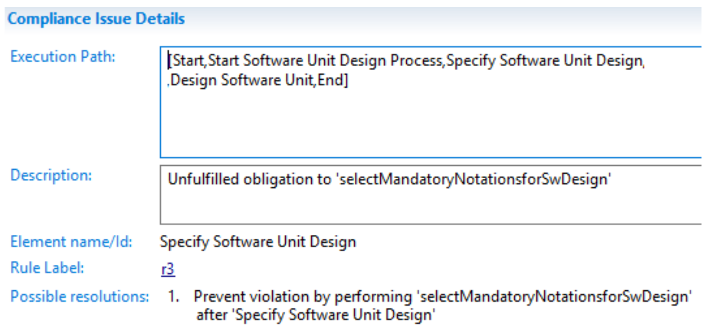


Figure 4.8: Activity Diagram of the Software Unit Design Process.

**Step 5: Checking and Analysis** Checking and analyzing compliance is a task performed by the process engineer. The required inputs are the FCL-based ruleset and the delivery process. The output is the compliance analysis (See Figure 4.9).


Compliance Issue Details

| | |
|---|---|
| Execution Path: | [Start,Start Software Unit Design Process,Specify Software Unit Design, ,Design Software Unit,End] |
| Description: | Unfulfilled obligation to 'selectMandatoryNotationsforSwDesign' |
| Element name/Id: | Specify Software Unit Design |
| Rule Label: | r3 |
| Possible resolutions: | 1. Prevent violation by performing 'selectMandatoryNotationsforSwDesign' after 'Specify Software Unit Design' |

Figure 4.9: Regorous Results.

As Figure 4.9 depicts, a compliance report contains information regarding the rules violated by the process and their possible reparation policies. Regorous also provides the set of rules that did not fire, i.e., rules that were not used during the compliance analysis. Such information can be used to make the adjustments required in the process plans to be checked iteratively. Reasons for improvements could be workflow problems, failure in the annotation process, failure to select process elements or a rule set with deprecated rules (i.e., rules that in the light of new version of the standards have change or disappeared). In the example depicted in Figure 4.9, it is possible to conclude that task *Specify Software Unit design* is missing the annotation *selectMandatoryNotationsS-*

*wDesign*. With this information, we can revise such a task and understand the reasons for the failure in the annotation, e.g., guidance regarding mandatory notations is probably missing in the task.

## 4.3 Process Compliance Hints and Patterns

Skillful FCL rule set design can be reached by applying computational thinking resources, in particular, process compliance hints and patterns (as recalled in Section 2.5). Hints are rules of thumb found in previous FCL formalization experiences, while patterns indicate common situations an FCL designer is likely to encounter. Both process compliance hints and patterns aim at facilitating the formalization of process-related requirements into FCL rules. In Section 4.3.1, we present the process compliance hints. In Section 4.3.2, we present the process compliance patterns.

### 4.3.1 Process Compliance Hints

The relationship between the requirements imposed by prescriptive standards and the targeted processes 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, process diversity (i.e., the application of several normative process reference frameworks in a single project) may lead to problems in the understanding of what is needed for managing process compliance. Thus, we have a compact set of requirements, which we need to manage appropriately.

The divide-and-conquer strategy (recalled in Section 2.5.1) is a design hint that can be applied in the formalization of process-related requirements. By applying the divide-and-conquer strategy, we could break down such complexity and provide a more comprehensible view of the requirements. In particular, the aspects that requirements in prescriptive standards regulate are the tasks, their specific order, the mandatory in/outputs of the tasks, roles performing the tasks, and the tools/recommended techniques 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 be qualified. 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 2.2.2), a task is performed by a role, requires inputs and provides outputs, is guided by guidance, and a tool is used (see Figure 2.2b). Thus, the tasks are the central elements, to which the other elements are allocated. The methodological steps for compliance checking (recalled in Section 4.2.4), 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 Figure 4.7b. 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 responsibility in terms of compliance information (see Figure 4.10).

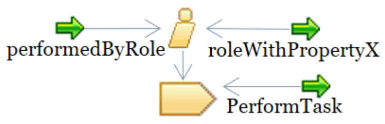


Figure 4.10: Annotated Role.

The novelty of the approach is threefold. First, we free 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, we propose two aspects.

1. Icons based on targeted elements (see Table 4.3).

2. Templates that facilitate the creation of compliance effects (see Table 4.4).

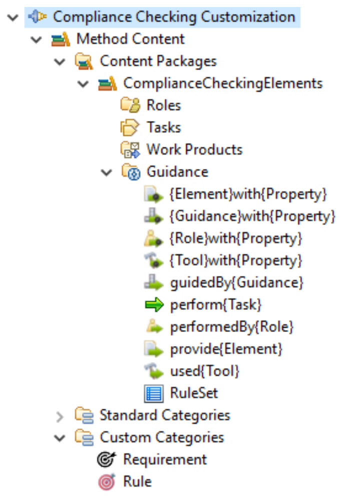
Table 4.3: Icons Describing Specific Compliance Effects.

| **Role** | | **Work Product** | | **Guidance** | | **Tool** | | **Task** |
|---|---|---|---|---|---|---|---|---|
| Def | Property | Def | Property | Def | Property | Def | Property | |
| ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ |

Table 4.4: 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*} | |
| **Note:** Fragments between { } should be replaced by the specific element or its property. | | |

The result of this customization is the addition of new elements to the compliance checking customizations plugin that was previously defined in Figure 4.4 (See Figure 4.11). We performed a complete example to illustrate our customizations and templates in paper B (see Chapter 8).



- Compliance Checking Customization
  - Method Content
    - Content Packages
      - ComplianceCheckingElements
        - Roles
        - Tasks
        - Work Products
        - Guidance
          - {Element}with{Property}
          - {Guidance}with{Property}
          - {Role}with{Property}
          - {Tool}with{Property}
          - guidedBy{Guidance}
          - perform{Task}
          - performedBy{Role}
          - provide{Element}
          - used{Tool}
          - RuleSet
    - Standard Categories
    - Custom Categories
      - Requirement
      - Rule

Figure 4.11: Enhanced Elements Customization.

### 4.3.2 Process Compliance Patterns

Formalizing the requirements prescribed by industry standards requires skills, which cannot be taken for granted. Patterns could represent a solution. In the remaining parts of this section, we define process compliance patterns (PCP's) and the PCP's templates.

#### PCP's Definition

Performing automatic compliance checking of a process requires the definition of a finite state model of the process, where normative requirements provide the permissible states of the process elements. This statement allows us to think of a process as a kind of system that can be verified. Thus, we can translate the property specification pattern definition (recalled in Section 2.5.2) into our context as follows:

> **Process compliance patterns (PCP) are patterns that describe commonly occurring normative requirements on the permissible state sequence of a finite state model of a process.**

With this definition, we developed the mapping between property specification patterns and PCP's. In this mapping, the state resulting from applying a requirement on a process element is considered analogous to the state resulting from applying a requirement on a system. The scope corresponds to the interval in a process when the obligations formulated by a requirement are in force. We have delineated five methodological steps to identify PCP's, as depicted in Figure 4.12.

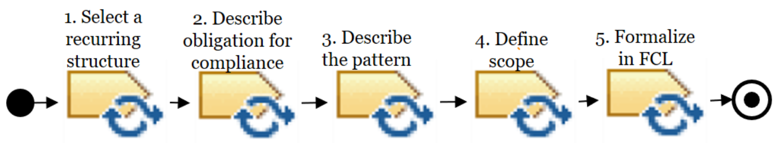

1. Select a recurring structure
2. Describe obligation for compliance
3. Describe the pattern
4. Define scope
5. Formalize in FCL

Figure 4.12: Methodological Steps for Identifying PCPs.

According to the steps depicted in Figure 4.12, the first step consists is to *select a recurring structure*. Standards, e.g., ISO 26262 (see Table 2.1) have implicit and explicit structures that we can use for that purpose, e.g., a clause has a title, prerequisites, work products. The second step is to *describe the* *obligation for compliance*, which is to find the deontic (the mandatory or permissible) nature of the structure to be defined. The third step is *describe the pattern*, based on similar (or a combination of) behaviors of the property specification patterns described in Section 2.5.2. This description is contextualized to process compliance based on the mapping previously done. In this step, we also assign a name for the PCP, reflecting the related obligation for compliance. The fourth step is to *define the scope* of the pattern, which we also based on the scopes defined to the property specification patterns. The fifth step is to *formalize in FCL* (we present this step below).

### FCL-based PCP's Templates

The PCP template description is based on similar (or a combination of) behaviors described for the property specification patterns, which are mapped to the notations provided in FCL (see Table 2.5), as follows:

- A global scope, which represents the entire process model execution, is defined as a [OM]P.
- A before scope, which includes the execution of the process model up to a given state, is mapped to a partial [OAP ].
- An after scope, which includes the execution of the process model after a given state, is mapped to a partial [OANP ].
- If an obligation admits an exception, e.g., tailoring, the part of the pattern corresponding to the exception is described as [P] since if something is permitted, the obligation to the contrary does not hold. The excepted obligation is modeled as non-perdurant, since the permission is not a violation of the obligation. Thus, the obligation does not persist after the permission is granted.

In principle, the requirements included in standards can be tailored. Thus, the achievement obligations are modeled as non-perdurant, i.e., [OAPNP] or [OANPNP]. In this case, obligation and permission have contradictory conclusions, but the permission is superior since it represents an exception.

In the remaining part of this section, we present the templates of the identified PCPs. In all templates, $\{\#\}$ should be replaced with the number that identifies the requirement in the standard. When it is described as $\{\#.i\}$, the *i* should be replaced by a, b, ..., n, where n is the number of sub-items, e.g., if there is a requirement with two parts that is identified with the number 5, the rules' identifiers are 5.a and 5.b.

**Tailoring requirements.** Tailoring means to adapt (omit or perform differently) the requirements in a standard to a specific project in a compliant form. Tailoring requires a rationale (or justification). For being valid, a rationale should always be verified by an expert. The rationale is an input element, and its verification is a property. An expert with specific qualifications should also be appointed. Thus, we use the templates for definitional and property-based propositions described in Table 4.4 for in/output elements and roles, i.e., provide$\{Rationale\}$, $\{Rationale\}$withVerificationByExpert, performedBy$\{Expert\}$ and $\{Expert\}$with$\{Qualification\}$. $\{Rationale\}$, $\{Expert\}$ and $\{Qualification\}$ should be replaced with the title of the required justification (if any), the role required and the necessary qualifications respectively. Providing those four conditions permit to omit the requirement (in other words, permit not to perform the requirement). Any of the definitional and property-based propositions described in Table 4.4 could be the target of such omission. For explanations purposes, we consider omitting a requirement that imposes the definition of a task, i.e., $\Rightarrow [P] - perform\{Task\}$ (see the pattern 4.4). In this case, $\{Task\}$ should be replaced with the name of the task that will be omitted,

$$\begin{aligned}\textbf{r\{\#\}.Omitted:} provide\{Rationale\}, \{Rationale\}withVerificationByExpert,\\ performedBy\{Expert\}, \{Expert\}with\{Qualification\}\\ \Rightarrow [P] - perform\{Task\}\end{aligned} \tag{4.4}$$

A second rule is included in case the task is done in a different way, where [OANPNP]perform$\{DifferentTask\}$ corresponds to the new task replacing the previous one (see Pattern 4.5).

$$\begin{aligned}\textbf{r\{\#\}.ChangedRule:} - perform\{Task\}\\ \Rightarrow [OANPNP]perform\{DifferentTask\}\end{aligned} \tag{4.5}$$

**Provide a prerequisite.** A prerequisite is an obligatory input element, which should be fulfilled before it is in force. $\{prerequisite\}$ should be replaced with the name of the prerequisite (see Pattern 4.6). If a previous rule triggers the prerequisite, its conclusion is included in the $\{optionalTrigeringObligation\}$, e.g., when the prerequisite is produced by a previous task. Prerequisite could have properties. In this case, the $\{optionalTrigeringObligation\}$ could be a list of such properties, using the template $\{Element\}$with$\{Property\}$. Otherwise, it is left empty.

$$\begin{aligned}&\textbf{r\{\#.i\}:}\{optionalTrigeringObligation\}\\&\Rightarrow [OAPNP]provide\{prerequisite.i\}\end{aligned} \tag{4.6}$$

**Perform a unit of work.** Pattern 4.7 represents the prescription of a unit of work in a process (i.e., phase/activity/task). It considers the prerequisites, if any, as the conditions of the applicability of the rule, which normative conclusion is performing a unit of work (e.g., a phase). It could be preemptive ([OAPNP]), if the prerequisites and the task are provided at the same time the task is performed.

$$\begin{aligned}&\textbf{r\{\#\}:}provide\{Prerequisite1\}..., provide\{Prerequisite.i\}\\&\Rightarrow [OAPNP]perform\{TitleClause\}\end{aligned} \tag{4.7}$$

It can be non-preemptive ([OANPNP]) as in Pattern 4.8, if the prerequisite is another task, that have to be done first. In the example in Pattern 4.8, $\{TitleClause\}$ should be replaced with the specific clause title.

$$\begin{aligned}&\textbf{r\{\#\}:}perform\{Task\}\\&\Rightarrow [OANPNP]perform\{FollowingTask\}\end{aligned} \tag{4.8}$$

**Provide guidance.** Guidance elements may not be required during standards compliance auditing. However, internal policies in a company may impose guidance elements. In that case, guidance elements should be provided at the moment the element guided is created. We create the propositions by using the template for guidance provided Table 4.4, i.e., *guidedBy*$\{Guidance\}$ and $\{Guidance\}$*with*$\{Property\}$. Guidance can be defined for any element in the process (tasks, work product, tool, or role). For explanation purposes, we consider *perform*$\{Task\}$ (see Pattern 4.9).

$$\begin{aligned}&\textbf{\{\#.i\}:}perform\{Task\}, \{Guidance\}with\{Property\}\\&\Rightarrow [OAPNP]guidedBy\{Guidance\}\end{aligned} \tag{4.9}$$

**Provide a work product.** Work products are the result of certain requirements. Thus, these requirements are presented as antecedents that oblige the provision of the related work product. Pattern 4.10 presents this aspect in FCL, where $\{providePreviousObligations\}$ should be replaced with the conditions that oblige the work product's production, usually the execution of a task (*perform*$\{Task\}$). Work product properties may be also required, i.e., $\{WorkProduct\}with\{Property\}$, where $\{WorkProduct\}$ and $\{Property\}$ corresponds to the work product's name and its corresponding property.

$$\begin{aligned}\textbf{r\{\#.i\}:}&\{providePreviousObligations\}, \{WorkProduct\}with\{Property\}\\ &\Rightarrow [OANPNP]provide\{WorkProduct\}\end{aligned} \tag{4.10}$$

PCP's where initially defined in Paper A (see Chapter 7). Then, the PCP's where improved and applied in Paper D (see Chapter 10).

## 4.4 Systematic Reuse of Compliance Artifacts

The engineering of safety-critical systems must comply with different normative frameworks, which often exhibit common requirements or at least a significant potential for synergy. If the semantic interplay of all applicable normative frameworks is not adequately understood, a twofold negative consequence needs to be faced:

1. the system's safety might be compromised due to the potentially contradictory influences of different regulatory frameworks;
2. compliance management might become even more time-consuming, risking stealing time and focus from other activities related to safety-critical systems production, e.g., verification of systems behavior.

In this section, we consider reuse systematization as a potential solution for coping with the demands of several standards. In particular, in Section 4.4.1, we present a logic-based framework for enabling the reuse of compliance proofs, called SoPLE&Logic-basedCM. In Section 4.4.2, we propose a technical and methodological solution for concretizing such reuse.

### 4.4.1 SoPLE&Logic-basedCM

Safety-oriented Process Line Engineering (SoPLE) (as recalled in Section 2.3.1) permits process engineers to systematize the reuse of process-related information. However, to argue about or prove compliance, SoPLE is not enough.

Therefore, we intend to provide an additional layer of confidence by combining SoPLE with a logic-based framework that enables formal proofs of compliance. To do that, we build on top of previous results. Specifically, we use defeasible logic (the predecessor of FCL), a rule-based approach for efficient reasoning with incomplete and inconsistent information, a typical scenario in normative systems. As a result, we obtain SoPLE&Logic-basedCM (as depicted in Figure 4.13), a framework that has the potential to increase efficiency in process compliance via systematic reuse of compliance proofs.

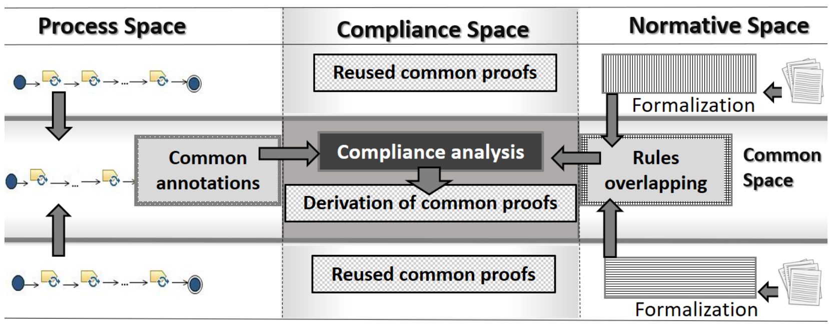


Figure 4.13: SoPLE&Logic-basedCM Framework.

As Figure 4.13, the framework is conformed by four spaces, as follows.

1. **The process space.** It is located in the left side of our framework. This space represents a zone where families of processes (i.e., processes that exhibit several commonalities) are managed by systematizing the reuse of their process elements (process structure) and the sequences of tasks (process behavior). For this, the process engineer uses SoPLE. As a result, a Safety-oriented Process Line (SoPL) is created. A SoPL should include selecting common and variable process elements (tasks, work products, roles, tools, and guidance) and the composition of those elements in a skeleton representing the process sequence.

2. **The normative space.** It is located in the right side of our framework. This space represents a zone where families of standards (i.e., standard that exhibit several commonalities, e.g., different versions of the same standard) are managed. In this space, a person trained in logics (especially, logics of the family of defeasible logics, such as FCL) formalise the standards requirements and select the set of rules that overlap.

3. **The common space.** It is located in the horizontal middle area of our framework. This space represents a zone where common aspects extracted from the processes meet the common rules extracted from the normative frameworks. In this space, the process engineer needs to analyze the SoPL and annotate it with the compliance effects found in the overlapping rules to derive a compliance state representation of the SoPL.

4. **The compliance space.** It is located in the vertical middle area of our framework. This space, which overlaps with the common space, represents a zone where a process engineer performs two tasks. First, the process engineer performs automatic compliance checking between the annotated SoPL and the set of overlapping rules to obtain a set of common proofs. Second, he or she analyzes the obtained set of proofs of compliance to define their potential reusability.

The potential reusability aspect of the proofs of compliance depends on the compliance effects of the tasks that vary in the process line. Changes in the compliance status of the derived standard-specific processes are linked to the normative effect of the variant. Depending on such effects, a process engineer can fully or partially reuse the common proofs of compliance. Proofs may be fully reused when the derived process does not replace any variability points defined in a SoPL. Instead, proofs could be partially reused when the general compliance is not affected by the compliance of the variability point once it is replaced. However, the compliance of the variability point needs to be assessed since the compliance analysis previously performed only included common aspects. In this case, compositional reasoning may be a solution.

Compositional reasoning replaces the analysis of the global state of the process with localized reasoning. In this sense, the process sequences affected by the variability are analyzed separately, based on assumptions about the behavior of process sequences already deemed compliant by the common proofs. For example, see the SoPL represented as the sequence C1-V1-C2 (see Figure 4.14). In such SoPL, the C1 and C2 represent the commonalities in the family, and V1 represents a variability. For compliance checking, C1 and C2 are annotated with the compliance effects a and b, respectively. The effects a and b are extracted from overlapping rules in the common normative space. When deriving processes from the family, the variability, V1, could be replaced either by R1 or R2, according to some aspect, e.g., the selection of a specific standard. Moreover, R1 needs to fulfill specific obligations derived from one of the standards in the family analyzed in the normative framework.

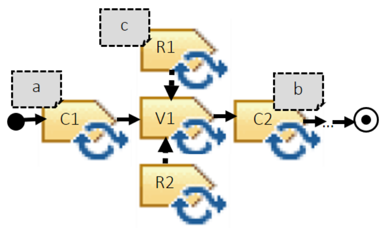


Figure 4.14: SoPL Skeleton of a Family of Processes.

As R2 does not have compliance effects to fulfill, its replacement in the process lines does not affect compliance. In this case, the proof of compliance obtained for the sequence C1-V1-C2 can be reused in the sequence C1-R2-C2. When R1 is used to derive the process-specific member, the compliance may be affected. Thus, the compliance status of task R1 needs to be checked and, based on assumptions, included in the general compliance status of the sequence C1-R2-C2. SoPLE&Logic-basedCM was defined in Paper A (see Chapter 7).

## 4.4.2 Tool-supported Methodological Framework

In Section 4.2, we proposed a logic-based method for facilitating compliance checking of single process plans, called ACCEPT. In Section 4.4.1, we also proposed a methodological framework called SoPLE&Logic-basedCM that extends the use of logic-based methods by incorporating SoPLE (Safety-oriented Process Line Engineering), to enable systematic reuse during compliance checking of process plans. In this contribution, we integrate these methods and include the support provided by the integration of EPF-C and BVR-T (recalled in Section 2.3.2). As a result, we offer a novel solution that encompasses process modeling, compliance checking, and variability/change management capabilities to enable systematic reuse and automatic generation of process compliance checking artifacts.

The model of reuse was previously described in SoPLE&Logic-basedCM as depicted in Figure 4.14. The management of such model in our new solution needs to be translated to VSpec models. As depicted in Figure 4.15, the skeleton of the family C1-V1-C2 is described in VSpec (as recalled in Section 2.3.2) as features connected to the parent feature (Checking_Management) via solid lines. The variability R1 and R2 are connected via dashed lines. Additional information can be modeled. In particular, the standard versions (e.g., S1 and S2) are modeled with a group element. Moreover, BCL constraints are created to restrict the selection of the variations according to the selected standard, e.g., if S1 is selected, then R1 and its effect c become mandatory features.

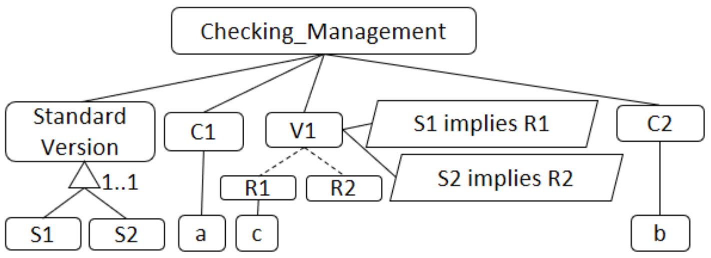


Figure 4.15: VSpec Model of the Checking Management.

The compliance status of task R1 needs to be analyzed to understand if the general compliance status of the sequence C1-R2-C2 is the same the one obtained for C1-V1-C2. If c = 0, the composition of the process elements would not affect compliance since there are no new compliance effects to be fulfilled. If c $\neq$ 0, there are two cases. First, the effect is local to the task, i.e., the effect is triggered and fulfilled in the variant R2. Second, the variant effect is not triggered by a previous task and/or make a new influence in the subsequent task effect. In both cases, the compliance status may be affected. For these cases, we consider the separations of concerns within the regulatory space and check the structural compliance (first case) separately from the compliance of the sequence of tasks (second case). The former permits the integration of the proof in the line without affecting the general compliance status. Such checking can be performed by BVR-T, which checks the presence/absence of process elements features. The latter makes the reuse of proof conditioned to additional compliance analysis of the tasks surrounding the variant (C1 and C2). This analysis can be performed by Regorous. The systematic reuse of compliance artifacts requires four steps (see Figure 4.16).

1. **Manage single process plan compliance.** We seek for single process plan compliance by using the automated compliance checking methodological steps presented in Section 4.2.4. Resulting artifacts are three EPF-C plugins and the compliance results issued by Regorous.

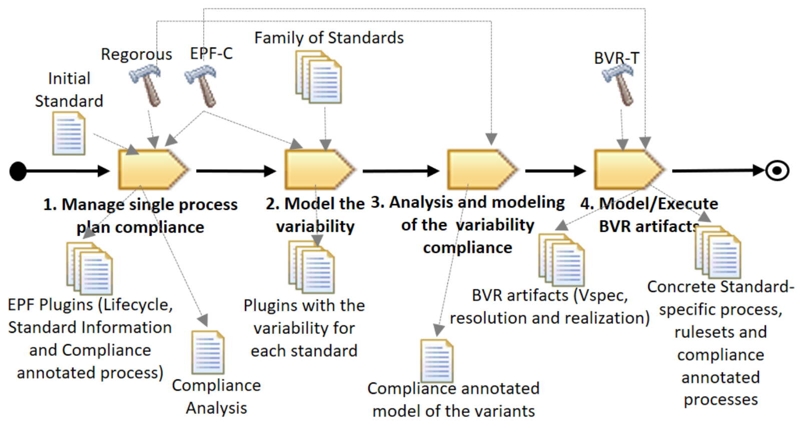

Family of Standards
Regorous
EPF-C
BVR-T
Initial Standard
1. Manage single process plan compliance
2. Model the variability
3. Analysis and modeling of the variability compliance
4. Model/Execute BVR artifacts
EPF Plugins (Lifecycle, Standard Information and Compliance annotated process)
Compliance Analysis
Plugins with the variability for each standard
BVR artifacts (Vspec, resolution and realization)
Compliance annotated model of the variants
Concrete Standard-specific process, rulesets and compliance annotated processes

Figure 4.16: Family-oriented Compliance Annotated Processes.

2. **Model the variability.** We evaluate the commonalities and variabilities regarding the models obtained in step 1) while adding standards of the same family, e.g., different versions of the same standard. For this, we use SoPLE&Logic-basedCM, which is described in Section 4.4.1. The artifacts that vary are modeled in EPF-C. Thus, the resulting models are a lifecycle plugin and standard information plugin for each standard evolution, containing only artifacts related to the variability.

3. **Analysis and modeling of the variability compliance.** An analysis of the changes in the compliance status of the standard-specific artifacts that are part of the variability is performed by taking into account the annotated compliance information. If needed, we use Regorous. However, if the variant is small, such analysis can be done manually. The result of this step is the compliance annotated process model of the variants.

4. **Model BVR artifacts.** BVR-T is used to create the abstract representation of the families involved in compliance checking, i.e., process elements, standard information and compliance annotated processes. Resolution models are automatically generated from the VSpec models, and use to validate the membership of the elements according to the selected standard. In a final step, which is not part of the scope of this thesis, realization models are created. Realization permits to define the replacements that should be part of the concrete standard-related artifacts that are exported back to EPF-C.

Thus, in this step we use the tool-chain integrated by EPF-C and BVR-T. This contribution is presented in Paper E (see Chapter 11).

## 4.5 Solutions Validation

In this section, we present the validation performed to our proposed solutions. In Section 4.5.1, we present the results of a personal opinion survey applied among practitioners who participate in safety-related process compliance checking. In Section 4.5.2, we present a case of compliance-aware space software engineering processes. In Section 4.5.3, we present the benefits of our methodological framework by measuring the enabled reuse. Finally, In Section 4.5.4, we position our research in the context of existing research efforts realized for compliance checking of software processes against different normative frameworks.

### 4.5.1 Personal Opinion Survey

We design and apply a cross-sectional web-based personal opinion survey (as recalled in Section 2.6.2). This survey goal is twofold. First, it aims at gathering information about current industrial practices and challenges in process compliance checking. For tackling this part of the survey, we used multiple option questions. Second, it focuses on evaluating the acceptance level of ACCEPT (recalled in Section 4.2). For addressing this part of the survey, we present an example of our solution applied to automotive, i.e., ISO 26262 (recalled in Section 2.1.1). Then, we present a series of claims regarding such a method from which we seek practitioners' degree of acceptance about the aspects described in the TAM model (recalled in Section 2.6.4), i.e., the method usefulness, usability, and user's intention to use it. To collect the answers, we use a five-point Likert Scale ranging from Strongly Agree to Strongly Disagree.

Our survey was answered by practitioners who participate in safety-related process compliance checking mostly in the European context. The valid answers were obtained from practitioners mostly working in the consultatory branch and have experience demonstrating process compliance checking in 13 different countries, predominantly Europe. The practitioners have experience in different safety-related domains and standards where automotive is the most represented. Our findings are described in three categories.

1. **Current practices in process compliance checking.** For the practitioners, process compliance checking is not only the way towards a safety certifi-

cate but also a mechanism for process improvement. Their current practices include the checking of a variety of process plans, reuse of previous process models and the use of software tools for supporting their practices. Thus, process compliance checking is an important factor in compliance management in the safety-critical context, which can benefit from the use of frameworks that permit automation and reuse.

2. **Process compliance checking challenges.** In general, practitioners consider that compliance checking is prone-to-error. For them, it is possible to miss requirements. Moreover, they consider that it is not easy to determine the kind of information that should be provided as evidence, and that there are different possible interpretations provided by the assessors. In addition, practitioners consider that compliance checking is time- and resource-consuming since it requires many hours of work, several iterations and many people involved. Thus, there is a need for solutions that provide more confidence and efficiency in process compliance checking.

3. **Acceptance level of the compliance checking method.** Practitioners consider that, in general, there are advantages regarding the method proposed for automated process compliance checking. In particular, there is a good degree of acceptance for the characteristics provided by FCL, for requirements representation. Moreover, the ability to represent processes graphically is seen as an advantage. In addition, practitioners consider that it is easy to trace uncompliant situations. For this reasons, practitioners show a willingness to use the method, which could be helpful for evolving from the current manual practices to automated practices via compliance checking. However, there is some hesitation regarding its usage, as expected with formal methods. Thus, it is necessary to explain further the formalization part of the method by providing more guidance and examples. We also need to provide mechanisms for improving the tool's usability in terms of compliance information representation, which appears to be not easy to use by practitioners. For facilitating this aspects, we consider to provide more specific graphical representations of the compliance artifacts.

This contribution is presented in Paper C (see Chapter 9).

### 4.5.2 The case of Space Software Engineering Processes

We perform a case study to understand if the ACCEPT (presented in Section 4.2) produced models could support the planning of space software engineering processes. Space software is safety and mission-critical, and it is often the result of industrial cooperation. Such cooperation is coordinated through compliance with relevant standards. In the European context, ECSS-E-ST-40C (recalled in Section 2.1.3) is the de-facto standard for space software production. The planning of processes in compliance with project-specific ECSS-E-ST-40C applicable requirements is mandatory during contractual agreements. Our analysis is based on the qualitative criteria to assess compliance approaches (recalled in Section 2.6.5), which consider two variables effort and converge level. We consider actual effort, which is determined by task demands since, in theory, it can be used to determine the intent to complete a task a priori, independently of any conscious actor. The coverage level is analyzed considering how the models respond to the information that needs to be required by the ECSS-E-ST-40C framework, i.e., information regarding standards, process plans, and compliance (i.e., EARM and ECM matrices).

**Case study insights**

When planning a software engineering process plan, the challenge is to understand how many process elements should be specified and their order. In the case study conducted, we define a model of a software process plan by the book, i.e., we extract the process elements suggested by the selected portion of ECSS-E-ST-40C standard without any tailoring. It is called Compliance-aware Engineering Process Plan (CaEPP) since such process elements are enriched with compliance information. In case of deviation (e.g., tailoring), we can also know if the requirements are tailored out or modified. ACCEPT states that creating a CaEPP requires several models, which design is a process that is not free of effort. However, initial observations have shown that the effort required to comprehend processes and standards models and document models is significantly less. The reason is that formal specifications are accompanied by informal explanations that clarify their meaning and place them in context. Moreover, the visual approach adopted allows for more focused reviews.

It is clear that organizations may depart from normative practices (i.e., to not create process plans by the book) for project-inherent reasons that can be justified. In such cases, logic-based requirements representations can be effortlessly superseded. In addition, the level of coverage of the models is higher. Thus, we can take good advantage of such an initial effort in the long term. Specifically, the models are created in an authoring environment that permits a well-defined organization of compliance-related artifacts in a hierarchical, visual, and enriched structure, which can be reused. This modeling strategy minimizes the distance between the specification of the requirements' normative intention and the process elements that should respond to such requirements.

We could also include the possible exceptions that are derived from the deviations. Once the models are created, the process plans' validity can be established by doing automatic reasoning about the standard conditions. In particular, compliance violations could be drafted better since failure to requirements is connected to textual sources. Therefore, the comprehension of processes, standards, and their relationships is more natural, and the documentation of compliance and the management of evolution get better support than in manual checklists. These features are a valuable gain since once industry standards and process plans are formalized, process engineers do not need to expend valuable hours on reading regulatory documentation to infer the actions that must be taken to maintain compliance.

#### Challenges and Potential Improvements

A key challenge in the use of ACCEPT is that standards are currently written in natural language, and formalizing them is an intimidating and fairly sophisticated task. The reason is that the number of requirements in a standard is significant and context-specific. Thus, their interpretation requires expertise. However, FCL has a limited set of constructs, which provide the expressivity required for formalizing requirements. Such constructs also provide a framework for thinking about the requirements in terms of deontic notions and exceptions, which could simplify their interpretation. Therefore, showing process engineers the FCL potential and its easy to use aspect may strive the interest for its exploitation.

The work to be done when creating a CaEPP for software projects in the space context has the tendency to be repetitive. Repetition could cause a drop in a subject's capacity to perform the modeling task (i.e., disinterest, boredom, fatigue), making relative task demands greater than necessary. Further automation of such tasks might reduce the absolute demands, and thus the actual effort. For example, the manual creation of the compliance effects is a repetitive task that has to be done for each effect. It was also prone to error since the effects' names have similarities (e.g., almost all the tasks' effects have the word design). Thus, we needed to review our design several times and manually track the information we were writing in EPF-C. However, this task is systematic and supported by templates. As such, it could be automatized by using a domain-specific language that permits an adequate characterization of the specific compliance effects and their production.

In general, different mechanisms can be defined to determine the meaning of context-dependent situations that could affect rules' formalization. In particular, patterns facilitate the recognition of relevant requirements, improving efficiency and consistency when producing rules. Our current selection of compliance patterns is limited to general situations, and they are also manually instantiated. Still, they can provide some assistance. Moreover, the process compliance hints could be used to establish conceptual relationships between the elements composing process plans and their compliance effects in the general compliance status. Thus, automated formalization of requirements could also be provided by performing an intermediate translation step into controlled English. For the compliance annotation of processes, programming scripts that examine the semantic similarity between process elements and compliance effects can be created. The modeling part could also be facilitated by providing general-purpose process model repositories to process engineers. Indeed, EPF-C offers such kind of repositories with libraries that can be downloaded and assemble in specific projects[3].

This contribution is presented in Paper D (see Chapter 10).

### 4.5.3 Reuse Measurement within the Evolution of ISO 14971

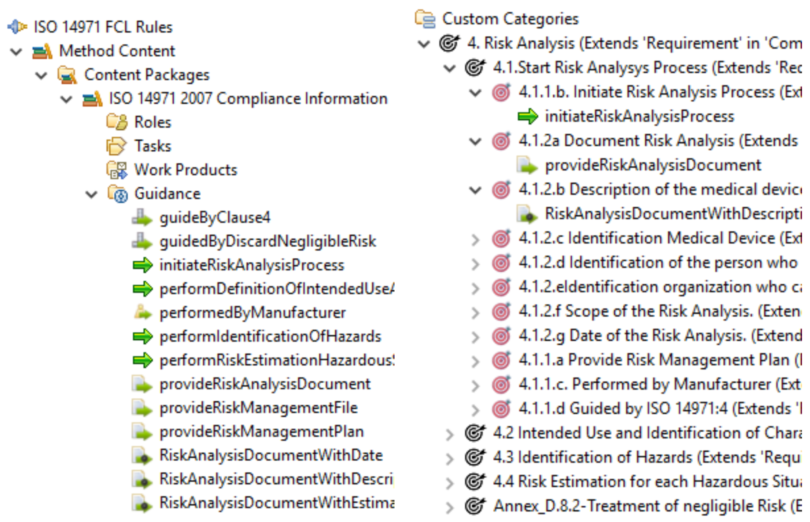
In this section, we illustrate the benefits in terms of our tool-supported methodological solution for the reuse of compliance checking artifacts presented in Section 4.4. We focus on recertification by showing process plan adherence to new versions of standards. For this, we considered the evolution (i.e., new versions of the standard resulting from revisions) of ISO 14971 (recalled in Section 2.1.4), which is related to the process for risk management to medical devices. By measuring the enabled reuse, we answer the question: To what extent process-related compliance artifacts can be reused?

#### ISO 14971 Evolved Artifacts Management

We follow the steps described in Figure 4.16. The results are three plugins as follows: the compliance effects and rules plugin (see Figure 4.17), the plugin that contains the process elements and their relationships (see Figure 4.18), and finally the compliance annotated plugin, which contains the annotations (see Figure 4.19) and the process workflow (see Figure 4.20). Regorous automatically generates a compliance state representation of the annotated process plan

[3]https://www.eclipse.org/epf/downloads/praclib/praclib_downloads.php

and analyses compliance against the FCL rule set. When no counterexamples exist, Regorous defines that the process is compliant (see Figure 4.21).


ISO 14971 FCL Rules
Method Content
Content Packages
ISO 14971 2007 Compliance Information
Roles
Tasks
Work Products
Guidance
guideByClause4
guidedByDiscardNegligibleRisk
initiateRiskAnalysisProcess
performDefinitionOfIntendedUse/
performedByManufacturer
performIdentificationOfHazards
performRiskEstimationHazardous!
provideRiskAnalysisDocument
provideRiskManagementFile
provideRiskManagementPlan
RiskAnalysisDocumentWithDate
RiskAnalysisDocumentWithDescri
RiskAnalysisDocumentWithEstima

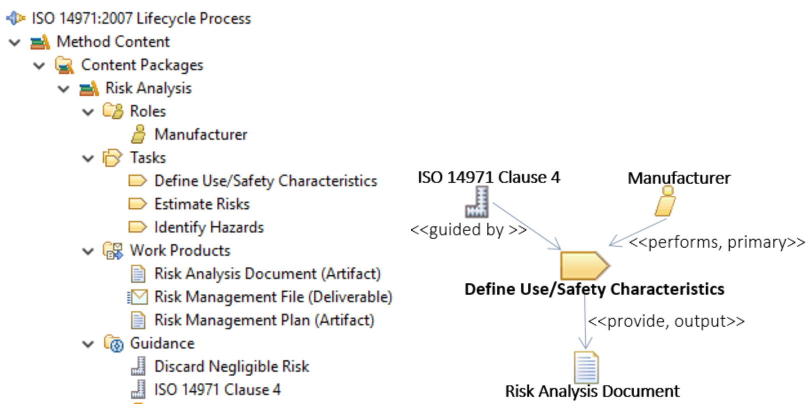
Custom Categories
4. Risk Analysis (Extends 'Requirement' in 'Com
4.1.Start Risk Analysys Process (Extends 'Rec
4.1.1.b. Initiate Risk Analysis Process (Ext
initiateRiskAnalysisProcess
4.1.2a Document Risk Analysis (Extends
provideRiskAnalysisDocument
4.1.2.b Description of the medical devic
RiskAnalysisDocumentWithDescripti
4.1.2.c Identification Medical Device (Ext
4.1.2.d Identification of the person who
4.1.2.eIdentification organization who c
4.1.2.f Scope of the Risk Analysis. (Exten
4.1.2.g Date of the Risk Analysis. (Extend
4.1.1.a Provide Risk Management Plan (I
4.1.1.c. Performed by Manufacturer (Ext
4.1.1.d Guided by ISO 14971:4 (Extends 'I
4.2 Intended Use and Identification of Chara
4.3 Identification of Hazards (Extends 'Requ
4.4 Risk Estimation for each Hazardous Situ
Annex_D.8.2-Treatment of negligible Risk (E

(a) (b)

Figure 4.17: ISO 14971:2007-Compliance Effects and Rules.

The second step consist of modeling the variability, i.e., we perform a gap analysis and model the additional artifacts imposed by the new standard versions. In particular, a new process element is additionally required for compliance with EN ISO 14971:2012, i.e., the guidance related to the inclusion of all risks for the treatment of negligible risk (see Figure 4.22a). In contrast, four new elements are required for compliance with ISO 14971:2019, i.e., two guidance elements (ISO 14871 clause 5 and the treatments of negligible risk), and two additional tasks, which are the result of splitting the task *Define/use safety characteristics* (see Figure 4.22b). In Step 3, the compliance analysis of the variability is performed. In our case, we found that the compliance with EN ISO 14971:2012 requires that one new element, specifically a guidance called *Treatment of Negligible Risk-Take all Risks* is annotated with a compliance effect called *guidedByTakeIntoAccountAllRisks*. A more complex analysis is performed in the case of ISO 14971:2019 (see Table 4.5).

We model the compliance effects. Figure 4.23 shows compliance effects



(a) (b)

Figure 4.18: ISO 14971:2007-Process Elements.

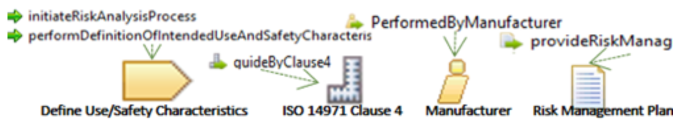

initiateRiskAnalysisProcess
performDefinitionOfIntendedUseAndSafetyCharacteris
PerformedByManufacturer
provideRiskManag
guideByClause4
Define Use/Safety Characteristics
ISO 14971 Clause 4
Manufacturer
Risk Management Plan

Figure 4.19: Compliance Effects Annotations.

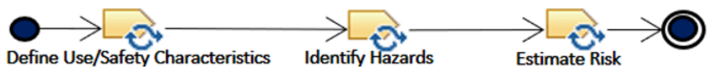
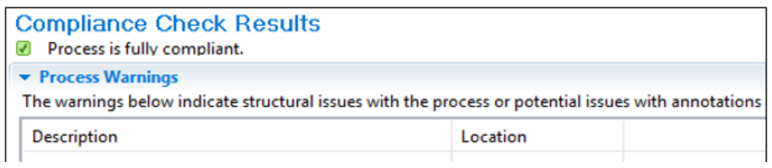


Figure 4.20: Compliance Annotated Process Plan.


Compliance Check Results

✓ Process is fully compliant.

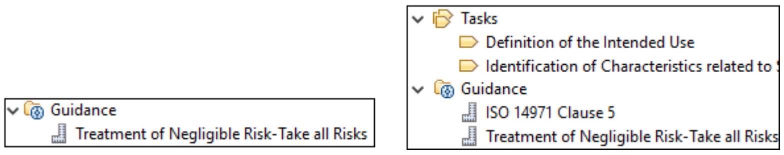
Process Warnings

The warnings below indicate structural issues with the process or potential issues with annotations

| Description | Location |
|---|---|

Figure 4.21: Regorous Results.

extracted from ISO 14971:2007 (on the left side) and the variations (highlight with different colors) respect the other two standards (on the right side).



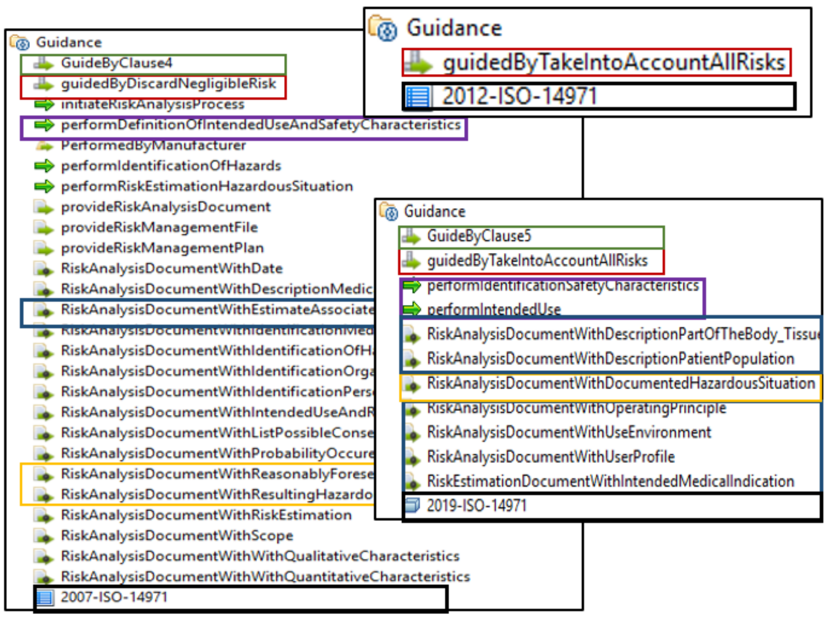
(a)

Guidance
Treatment of Negligible Risk-Take all Risks

(b)

Tasks
Definition of the Intended Use
Identification of Characteristics related to
Guidance
ISO 14971 Clause 5
Treatment of Negligible Risk-Take all Risks

Figure 4.22: ISO 14971:2007-Process Elements.



Guidance
guidedByTakeIntoAccountAllRisks
2012-ISO-14971

Guidance
GuideByClause4
guidedByDiscardNegligibleRisk
initiateRiskAnalysisProcess
performDefinitionOfIntendedUseAndSafetyCharacteristics
PerformedByManufacturer
performIdentificationOfHazards
performRiskEstimationHazardousSituation
provideRiskAnalysisDocument
provideRiskManagementFile
provideRiskManagementPlan
RiskAnalysisDocumentWithDate
RiskAnalysisDocumentWithDescriptionMedic
RiskAnalysisDocumentWithEstimateAssociate
RiskAnalysisDocumentWithIdentificationMed
RiskAnalysisDocumentWithIdentificationOfH
RiskAnalysisDocumentWithIdentificationOrga
RiskAnalysisDocumentWithIdentificationPers
RiskAnalysisDocumentWithIntendedUseAndR
RiskAnalysisDocumentWithListPossibleConse
RiskAnalysisDocumentWithProbabilityOccure
RiskAnalysisDocumentWithReasonablyForese
RiskAnalysisDocumentWithResultingHazardo
RiskAnalysisDocumentWithRiskEstimation
RiskAnalysisDocumentWithScope
RiskAnalysisDocumentWithWithQualitativeCharacteristics
RiskAnalysisDocumentWithWithQuantitativeCharacteristics
2007-ISO-14971

Guidance
GuideByClause5
guidedByTakeIntoAccountAllRisks
performIdentificationSafetyCharacteristics
performIntendedUse
RiskAnalysisDocumentWithDescriptionPartOfTheBody_Tissu
RiskAnalysisDocumentWithDescriptionPatientPopulation
RiskAnalysisDocumentWithDocumentedHazardousSituation
RiskAnalysisDocumentWithOperatingPrinciple
RiskAnalysisDocumentWithUseEnvironment
RiskAnalysisDocumentWithUserProfile
RiskEstimationDocumentWithIntendedMedicalIndication
2019-ISO-14971

Figure 4.23: Effects Variability between the Standards.

The fourth step is the modeling of BVR artifacts. In this step, we model the VSpecs of the families corresponding to the process, ruleset, and checking management. All the families are created under the same root, i.e., ISO_14971 (see Figure 4.24). A branch of the feature model tree contains the version of the standards used to make the changes at variation points.

Table 4.5: ISO 14971: 2019-related Annotations.

| Element | Compliance Effect |
|---|---|
| Task: Definition of the Intended Use | performIntendedUse, initiateRiskAnalysisProcess |
| Task: Identification of Characteristics related to Safety | performIdentificationSafetyCharacteristics |
| Work Product: Identification of Characteristics related to Safety | RiskAnalysisDocumentWithDescriptionPartOfTheTissue, RiskAnalysisDocumentWithDescriptionPatientPopulation, RiskAnalysisDocumentWithDocumentedHazardousSituation, RiskAnalysisDocumentWithIntendedMedicalIndication, RiskAnalysisDocumentWithIntendedUseAndReasonablyForeseeableMisuse, RiskAnalysisDocumentWithOperatingPrinciple, RiskAnalysisDocumentWithUseEnvironment, RiskAnalysisDocumentWithUserProfile |
| Guidance: ISO 14971 Clause 5 | GuideByClause5 |
| Guidance: Treatment of Negligible Risk-Take all Risks | guidedByTakeIntoAccountAllRisks |

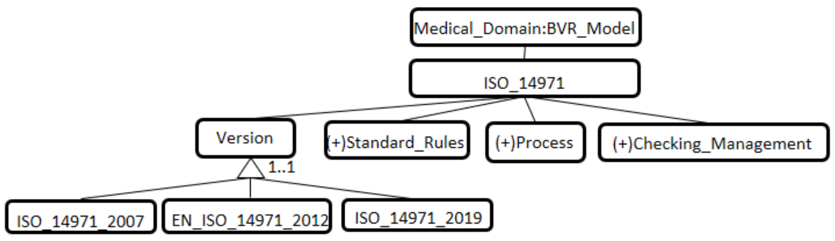


Figure 4.24: BVR VSpec.

### Reuse Measurement

For compliance with EN ISO 14971:2012, we need to create 1 new process artifacts, 2 new compliance effects and perform 1 compliance annotations. The number of total artifacts was 9 process elements, and 27 compliance effects and 26 compliance annotations. Thus, reuse is 88,9%, 92,3% and 96,2%% respectively (See Table 4.6). Similar measurement is performed in the case of compliance with ISO 14971:2019 (see Table 4.7).

Table 4.6: Reuse Measurement related to EN ISO 14971:2012.

| Type of artifacts | New | Total Used | Reuse Percentage |
|---|---|---|---|
| **Process** | 1 | 9 | 88,9% |
| **Compl. Effects** | 2 | 27 | 92,3% |
| **Compl. Annotations** | 1 | 26 | 96,2% |

Table 4.7: Reuse Measurement related to ISO 14971:2019.

| Type of artifacts | New | Total Used | Reuse Percentage |
|---|---|---|---|
| **Process** | 4 | 10 | 60% |
| **Compl. Effects** | 12 | 32 | 61,3% |
| **Compl. Annotations** | 13 | 32 | 59,4% |

### Discussion about the Enabled Reuse

Reusing compliance checking artifacts is a plausible solution for showing process plan adherence with new versions of standards. The reason is that recertification demands process reconfiguration. Considering the reuse measurement previously performed, we answer the question initially posed, *to what extent process-related compliance artifacts can be reused?*, as follows: the reuse extent measured is significant (the minimum gain was 59,4%). Thus, the context of medical devices complying with ISO 14971-clause 4 (the risk analysis phase) can positively benefit from the systematic reuse of compliance checking artifacts. In general, processes and standards that evince low levels of variation could be part of a family that exhibits high reuse levels in terms of compliance checking artifacts and could benefit from using our methodological framework during the required modeling task.

It is widely recognized that standards requirements are challenging to understand due to their wordiness and how they relate to each other. Their evolution is also challenging, due to the need to handle the normative changes and the recertification effort. When using our methodological framework, process engineers need to analyze new requirements systematically. This analysis is required to determine whether an existing compliance checking-related artifact can fulfill a specific requirement as-is or with modifications (new properties should be added/deleted), or if new artifacts have to be modeled from scratch. It also highlights problems between requirements, which may put compliance at risk, i.e., contradictory requirements, real/fake dependencies between requirements and new compliance information applicable to existing process plans.

The most important is that the process engineer's analysis is recorded in graphical models, which not only provide automated checks but also automated process plan's reconfiguration. Thus, our methodological framework supports a confident reduction of the work required to be done when new instances of compliant process plans have to be modeled.

This contribution is presented in Paper E (see Chapter 11).

### 4.5.4 SLR of Compliance Checking Approaches for Software Processes

Practitioners who participate in safety-related process compliance checking find it advantageous to have automatic means for performing compliance checking (as presented in Section 4.5.1). For this, it is required to use a unifying logic that permits automatic reasoning between the process models and the normative frameworks regulating them (as presented in Section 4.2.1). Several studies have approached this idea by formulating methods for compliance checking of software processes against different software process-related normative frameworks. 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 (Systematic Literature Review) (as recalled in Section 2.6.3). In this section, we present the results of the SLR.

#### Publication Distribution

We selected 41 primary studies after a thoughtful evaluation from a list of 2033 articles found in recognized online libraries. The selected primary studies where dating from 1995 to 2021, with a peak in 2017. The most preferred venue to publish the articles are conferences. However, we found also relevant papers published in workshops and journals.

#### Methods Characteristics

The primary studies were classified according to how compliance checking is performed. Initial methods for process-based compliance checking with quality standards are based on the verification of document evolution. The second group of methods considers the process-related concepts prescribed in the standards to defined models for checking compliance. The third group of methods take as a base consolidated process modeling languages and add a layer of analysis by using formal languages. The fourth group of methods is aimed at checking compliance from role-based access controls. The final group uses different methods, i.e., workflow engines, model refinements, model checking, and process mining. The five types of approaches use 14 different languages to represent the process elements and structures. Similarly, 14 different languages are used to represent the requirements prescribed by the standards.

The automation part claimed in the studies is related to 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, requirements formalization is mainly performed manually, in some cases, by using formalization patterns. The state of the tool support is also variable. In particular, the majority of the methods provide prototypes that are used as proof of concept.

Few primary studies present 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). Some studies show specific structures that permit the deletion and modification of products and activities according to the project's characteristics, such as the integrity levels prescribed by safety standards. In others, the reuse of process elements and the use of methodologies such as process lines are also implemented to manage process diversity and standards evolution

**Potential Impact**

Several application domains are addressed in the primary studies. The most represented application domain is characterized by the safety-critical systems context (defined by the application of safety standards). There are also methods addressing specific aspects of software process improvement (SPI) and quality, data protection, software process verification, Cybersecurity and Health care. Different standards have been modeled and used in the experimentations or illustration results provided in the primary studies. The most represented is ISO 26262, which is a functional safety standard used in automotive.

The illustrative scenarios are also variable. It is important to highlight that, in total, 19 of the 41 studies used data extracted from industrial settings to evaluate their methods There are few studies that also addressed the problem of compliance checking in agile environments. Such approaches are mainly based on mining techniques. Thus, they are not applicable in the case of seeking compliance checking of process plans since process minign requires process logs, which is information resulting after executing the process.

### Discussion about Existing Methods

Existing literature related explicitly to compliance checking of software processes is scarce and scattered. 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. We found most of the studies after 2017, which may mean that automated compliance checking of software processes has started to be a consolidated topic in the research agenda. In our opinion, these are relevant news since compliance checking is not optional nor either an easy task.

1. **The use of software process modeling languages.** New languages (metamodels, and ontologies) with limited scope have been created. The continuous creation of ad-hoc software process-related modeling solutions could be a disadvantage, especially when there are already well-defined process modeling languages that could be reused and extend according to specific needs. In addition, the creation of many new languages is a sign of research efforts done in silos. Such independence may result in wasted research efforts since languages are always created from scratch.

   We consider that new research efforts in automatic compliance checking, specifically for software processes, could take into account existing process modeling languages to accelerate results in the topic and standardize the techniques and tool support. For this, we could perform comparative studies between existing process modeling languages and case studies showing their capabilities.

2. **Language suitability for addressing normative requirements.** 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 rationales) is a must-do task 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 explicit 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 such exploitable characteristics in FCL.

3. **Towards a generic and domain-agnostic method.** The methods described in the primary studies provide a set of interesting, applicable, and useful aspects contributing to the automation of compliance checking of software processes. However, in most cases, normative documents have been considered in isolation resulting in ad-hoc solutions. In reality, process engineers have to deal with 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 may be a factor that also hinders their application in practice.

   In our opinion, it is crucial to spread out research results regarding compliance checking of software processes by consolidating a method that can handle the different concepts, structures, and scenarios. 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.

4. **The need for diverse abilities.** In today's methods for compliance checking, 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 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.

   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 (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

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.

6. **Going beyond technological dilemmas.** In this point, we have two parts to discuss.

   (a) Article 22 of the GDPR 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.

   (b) 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. 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.

This contribution is presented in Paper F (see Chapter 12).

# Chapter 5

# Related Work

In this chapter, we discuss the related work. In particular, we discuss approaches for automated compliance checking (see Section 5.1), approaches for facilitating requirements specification (see Section 5.2), and approaches for facilitating the reuse of proofs (see Section 5.3).

## 5.1 Automated Compliance Checking

Means for compliance checking of processes have been provided in previous research. In [122], the authors propose a semi-automatic compliance process to support the formal verification of software requirements. In [102], the authors present an approach to reason about the correctness of the process structure, which is based on the combination of CTN (Composition Tree Notations) [123] and Description Logic (DL). In [124], the authors extend the work done in [102] to include the capability levels proposed in ISO/IEC 15504 [71]. Similarly, in [125] and [126], authors present models to enable the definition process capability levels, according to ISO/IEC 15504 and CMMI (Capability Maturity Model Integration) v1.3 [72]. In [127], the author presents a formalization of data usage policies in a fragment of OWL (Web Ontology Language) [128], which is a DL-based modeling language. All the previous approaches consider the use of DL to reason about the compliance of the process structure against specific normative frameworks (i.e., ISO/IEC 15504 and CMMI). As presented in [129], DL has relative expressiveness, which makes it more difficult to model certain concepts. Besides, the previous approaches only consider the analysis of the process structure. Instead, our framework considers the information that represents the effects caused by the tasks (a.k.a. compliance effects annotation), which is used to check the compliance of a process structure and its behavior. Our framework is not tied to specific normative frameworks and uses SPEM 2.0 concepts, which offers a standardized structure for modeling processes and their related information.

SPEM 2.0 community is also interested in addressing compliance checking and monitoring. In [130] and [131], the authors propose a framework that uses LTL (Linear Temporal Logics) on top of SPEM 2.0 to monitor and control process instances according to its defined process model. The work presented in [132] and [133] aim at facilitating the checking of constraints that can be defined as part of a specific process model (e.g., standards requirements, metrics) by using SWRL (Semantic Web Rule Language) [134]. In [135], the authors present an approach for representing SPEM 2.0 process models in DL to provide consistency checks. In [136], the authors provide an approach in OWL for tailoring processes, in which outputs are transformed into SPEM 2.0 process models. Approaches to systematically exploit the modeling capabilities of SPEM 2.0 for collecting the compliance elements required by specific standards are also part of the state-of-the-art. Examples of those approaches are presented in [137], which collects compliance information for EN 50128 [67], and in [138], which collects evidence for supporting compliance with DO-178C [29]. Models for representing Deployment Packages and Implementation Guides for the Standard ISO/IEC 29110 [139] in EPF-C are presented in [140].

The use of SPEM 2.0 in the previous approaches is limited to modeling the elements required to create the process structure. In addition to the process structure, other approaches use concepts provided by SPEM 2.0 for modeling additional elements required for compliance. For instance, in [89] customization of the elements defined in SPEM 2.0 is performed to give the possibility to generate compliance tables. The modeling of standards requirements in SPEM 2.0 presented in [141] is used to detect whether the process model contains sufficient evidence for supporting the requirements, providing feedback to the safety engineers regarding detected fallacies and recommendations to solve them. As in [89] and in [141], our approach combines the modeling capabilities for modeling standard's requirements, plus customization of preexisting modeling concepts to generate a centralized compliance-related knowledge base. In addition, we add a layer of confidence by considering the use of methods to derive proof of compliance. However, we do not use semantic web methods of logics from the temporal logics family for deriving our proofs since they are not expressive enough for modeling compliance notions [75]. Semantic web methods, in particular, are computational methods that deal with ontologies and rules, whose combination could be undecidable [142].

Approaches for compliance checking have been widely studied in the business context. In [143], the authors propose a metamodel called REALM (Regulations Expressed As Logical Models), which uses Real-time Temporal Object Logic [144], for the representation of norms. In [145], the authors propose an SOA (Service Oriented Architecture)-based compliance governance, called COMPAS, to define compliant process fragments. In [146], authors propose a compliance checking method for business process models, in which norms are expected to be modeled in BPSL (Business Property Specification Language) and then formalized in LTL (Linear Temporal Logic). Once the two formal specifications are given, model checking via the NuSMV2-based Open Process Analyser is performed for checking compliance. In [147], the authors propose a solution for ensuring compliance by using a formal specification of business rules. There is also compliance checking frameworks that combine the modeling capabilities provided by BPMN (Business Process Model and Notations) [148] and Temporal Logics for the modeling of regulations. Examples of these kind of frameworks can be found in [149], [150], and [151]. Our framework uses a similar methodology, i.e., we define formal specifications of the normative requirements and the processes. The main difference regarding the previous approaches is the use of the modeling languages. In particular, we use FCL and not any language derived from the family of Temporal Logics for creating the formal specification of the requirements of the standards. The reason for not using TL is that this logic cannot provide conceptually sound representations of the regulatory requirements governing a process [152].

## 5.2 Requirements Specification

One of the significant challenges associated with the formalization of normative texts is requirements interpretation. Currently, research in this area aims at bridging the gap between the normative expertise required to interpret the regulatory text and the modeling skills required to build the knowledge base. In particular, in [153], the authors focus on the representation of constitutive rules in a machine-readable form. In [154], the authors propose a framework for identifying terms and refining the goals proposed by privacy legislation. In [155], the authors present a pattern-based approach to capture the knowledge of domain experts. In [156], the authors present a methodology to formalize norms as logic programs. In [157], the authors use SBVR (Semantics of Business Vocabulary and Business Rules), a controlled natural language, to develop business vocabulary and business rules. In [158], the authors use high-Level methodological approaches for the extraction and representation of compliance objectives, rules, and constraints by using semantic web technologies. In [159], the authors also use conceptual models to extract information from standards that apply to agricultural production. In [160], the authors describe a methodology for requirements formalization for flight-critical systems verification. Like the previous works, our goal is to facilitate the formal specification of requirements by considering specific normative concepts. We focus, mainly, on processes and provide a mechanism for facilitating the formalization of process-related requirements prescribed by the standards in FCL based on a standardized language, i.e., SPEM 2.0.

Separation of concerns within the normative space in FLC has also been proposed in the state-of-the-art. In [161], 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 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 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. As in [161], our work describes the compliance effects based on the type of elements present in a process. However, contrary to [161], 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 subsequent annotation in process elements. In [162], the concept of data tag described in [161] is revisited to create a methodology that permits their extraction from business process logs. The extraction of our compliance effects, as opposite the work presented in [162], is performed directly from the standards requirements since we aim at having a faithful representation of the requirements prescribed by the standard at design time.

Guidelines are also widely used to spread the used of novel methods in engineering tasks. In [163], the authors describe the Oracle Policy Modeling best practices guidelines for business rules modeling. Similarly, in [164], the authors describe a methodology to guide companies to establish Cyber-Physical Social System data subjects' consent and data usage policies. In [103], the authors present guidelines for supporting the formal representation of safety regulatory requirements. In [165], the authors present the use of tabular expressions to generate formal models of system requirements. The authors of FCL have also published explicative examples of the modeling of FCL rules within the business context, e.g., [166, 49], which can be used as a guideline for learning the language. In [167], the authors present a methodology for extracting models from ISO 26262, called “snowball”, in which high-level requirements (objectives section of the standard) are modeled first, and then the low-level requirements (requirements and recommendations) are added as rolling a snowball in the snow. The use of FCL for supporting compliance management tasks in safety-related processes is a novelty. We did not find specific examples or guidelines that apply to the domain yet. Thus, the guidelines provided by this thesis may be of interest for process engineers involved in the different industries manufacturing safety-critical systems. We also consider that our guidelines can be used to derive domain-specific guidance applicable to process-based standards beyond the ones tackled in this thesis.

## 5.3 Reuse of Proofs

“State explosion phenomenon” [168] refers to the exponential growth of the state space required to be verified by automatic checking methods. Essentially, research impacting the reduction of this phenomenon is studied in software verification. In [169], the authors present an approach for exploiting the evolutionary nature of programs for reusability of proofs, in which a tree of connected proved fragments of a program is built. In [170], the authors present a reuse method that uses incremental proof construction for reusing correctness proofs of Java programs. In [171], the authors give an overview of techniques for reusing information obtained in verifications results from past trials. In [172], the authors present a family-based strategy for minimizing the efforts in verifying product lines. In particular, this verification can be achieved by combining all code of all features in a single product simulator, which checks all products’ valid execution paths without generating and checking any individual product. These approaches are focused on products, while our focus is the processes to achieve such products.

The change triggered by updated standards for the software process is a topic tackled from different perspectives. In [173], the authors propose a method that permits the attachment of change information to process documents to facilitate change understandability. However, no systematic methods to reuse modeling artifacts facilitating the changes are proposed as we do in our work. Methods for modeling the change/variation and reuse of processes result from applying software process lines methodologies, as recently surveyed in [174]. In particular, SoPLE [50] has been exploited to provide a representation of family members with safety information, e.g., reusable process arguments used in safety cases [175] and tailoring of process models according to safety integrity levels of products [176]. In our work, we also use SoPLE to provide mechanisms to support variation knowledge reuse regarding compliance checking artifacts, which has not been yet addressed in other approaches.

In the area of process compliance, only a few approaches that consider the reuse of proofs are presented in the state-of-the-art. In [145], the authors consider that after doing many customization steps, which can include techniques, such as abstraction or parametrization, the functionality of a process chunk can be reused. However, for performing the compliance checking of process chunks, special formalizations have to be performed. Some researchers tackled reuse by defining building blocks that implement compliance requirements, e.g., compliance scopes [177], and compliance fragments [178, 179]. Reuse is also approached with the use of process patterns [180], and rule patterns [149]. In [150], business processes are augmented with reusable fragments, called process fragments for compliance, stored in a fragment repository that supports versioned storage and retrieval. The intention is to ensure process compliance to the corresponding compliance requirements. In contrast ot the previous works, we propose a holistic modeling framework for safety-related process compliance checking that permits modeling artifacts, which can be automatically interleaved with evolutionary/changing artifacts originated from new versions of standards. In that way, building blocks that implement compliance requirements are reusable and process models and rulesets denoting formalized requirements from standards.

# Chapter 6

# Epilogue

In this chapter, we present the final remarks of this thesis. In particular, we present conclusions (see Section 6.1), limitations of the study (see Section 6.2) and future work (see Section 6.3).

## 6.1 Conclusions

Industry standards, especially safety standards, establish requirements that serve as a baseline for developing safety-critical systems. For instance, compliance with ISO 26262, the functional safety standard targeting automotive, represents that the risks associated with the safety-critical system operation installed in a vehicle have been considered, analyzed, and mitigated to a sufficient level for their confident deployment into society. In some industries, compliance with specific normative frameworks is mandatory (see, for instance, the certification of airworthiness, which is based on standards applied by national aviation authorities), while in others, the same procedure is not de rigueur (for instance, the application of the standard ISO 26262 in automotive is voluntary).

Mandatory or not, compliance with industry standards serves as a benchmark to design, implement, maintain and evaluate technical products. It also grants several benefits to manufacturers. For example, compliance requirements provide a reference point for production optimization. Moreover, compliance requirements help manufacturers to formulate contractual obligations with their partners in a globalized sphere. In addition, a compliance stamp, which is a mark that consumers generally trust, could make manufacturers gain a lead against competitors. A standard-compliant safety-critical system is also, to some extent, ”backed up” in legal actions if it unexpectedly causes harm since compliance with standards is a sign of reasonable care. Given the previous considerations, is it clear that compliance with standards is a must-do for manufacturers of safety-critical systems.

Commonly, industry standards adopt a highly prescriptive approach, which focuses on process reference models and process-related requirements. Such an approach also demands that normative imperatives are planned at the beginning of the development process and not added later on. A process plan correctly designed is a detailed proposal of action that demonstrates the planning-time allocation of responsibilities. If every actor fulfills his/her duties, compliance at execution time can also be guaranteed. Consequently, planning processes is an essential requirement for manufacturers of safety-critical systems.

Compliance of process at planning time requires that process engineers include the sequence of tasks mandated by standards (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). Process elements and their features are described in the standards and mandated to be achieved at certain points of the development process. Thus, process-related compliance management can be supported by checking that the processes planned to engineer safety-critical systems have such elements and characteristics at the specific times they are required.

Manually checking process plan compliance is a complex task since standards contain a sheer volume of requirements. Requirements also have an intricate nature, i.e., requirements in one standard usually refer to other requirements in the same standard or other standards. Moreover, requirements in one part of the standard may contradict requirements in another part of the same standard or other applicable standards. In addition, requirements from one or different standards may apply to one or several processes at different times and jurisdictions. Last but not less, manufacturers commonly manage multiple process plans and, thus, perform compliance checking repeatedly. In each case, if there are no mechanisms to manage the information regarding standards requirements appropriately and the processes they regulate, (re)interpretations and rework may be needed (e.g., when the process expert leaves the company), and compliance risks may become commonplace.

Process management tasks supported by tools, which can provide automated checks, could be a plausible solution. Indeed, automated compliance checking could help process engineers to detect compliance violations and enforce compliance at planning time when allocating the resources for engineering safety-critical systems. Tools can also help the process engineers manage the changing conditions required to comply with standards (new versions of the standards, tailoring) and provide means for tracking such changes. Thus, the main goal of our thesis is to **facilitate automated compliance checking of the process plans used to engineer safety-critical systems against the standards mandated (or recommended) in the safety-critical context**. For reaching this goal, we defined four subgoals and provided contributions distributed in six papers (see Section 6.1.1).

### 6.1.1 Research Goals Revisited

In this section, we present the relationship between the research goals proposed in Section 3.3, the contributions discussed in Chapter 4, and the included papers presented in Chapters 7 to 11 (see Figure 6.1).

**Contribution 1.** We identified challenges that manufacturers of safety-critical systems have regarding compliance with prescriptive standards at planning time. Based on them, we specified requirements for a suitable technical solution that shall facilitate: 1) the management of the artifacts required for compliance checking with prescriptive standards, i.e., the standards themselves, their requirements, the processes plans, and the compliance means; 2) keeping track of the applicable requirements; 3) the recognition of contradictions and ambiguities between applicable requirements; and 4) managing the changing nature associated with requirements diversity. With this contribution, we fulfilled our first subgoal, namely, *elicit the requirements to be met to support the automated compliance checking of the process plans used in the safety-critical context.* This contribution, which was summarized in Section 4.1, is found throughout the articles A, B, C, D, and E.

**Contribution 2.** We adopted compliance by design and introduced ACCEPT (Automated Compliance Checking of Engineering Process plans against sTandards). ACCEPT is an iterative and comprehensible framework that allows the creation and management of the artifacts required for compliance checking, i.e., the standards themselves, their requirements, the processes plans, and the compliance means. It also permits the creation of engineering process plans checkable for compliance, i.e., process elements enriched with compliance information through annotations representing formalized standards requirements in FCL. FCL is a language based on defeasible logic, which permit to manage the contradictory and ambiguous nature present in the normative documents.

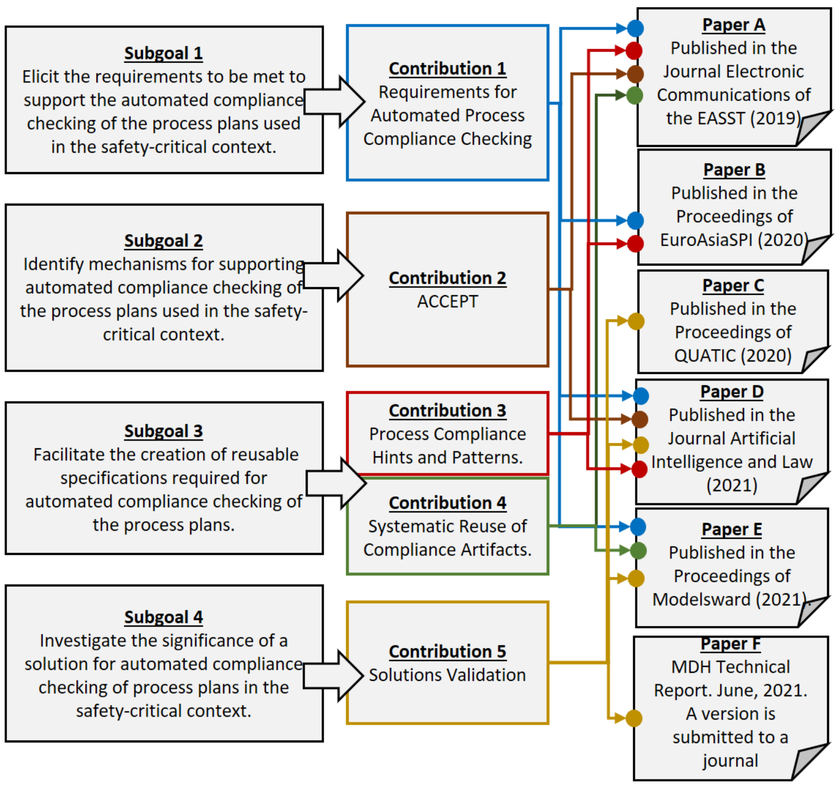

Subgoal 1
Elicit the requirements to be met to support the automated compliance checking of the process plans used in the safety-critical context.
Contribution 1
Requirements for Automated Process Compliance Checking
Subgoal 2
Identify mechanisms for supporting automated compliance checking of the process plans used in the safety-critical context.
Contribution 2
ACCEPT
Subgoal 3
Facilitate the creation of reusable specifications required for automated compliance checking of the process plans.
Contribution 3
Process Compliance Hints and Patterns.
Contribution 4
Systematic Reuse of Compliance Artifacts.
Subgoal 4
Investigate the significance of a solution for automated compliance checking of process plans in the safety-critical context.
Contribution 5
Solutions Validation
Paper A
Published in the Journal Electronic Communications of the EASST (2019)
Paper B
Published in the Proceedings of EuroAsiaSPI (2020)
Paper C
Published in the Proceedings of QUATIC (2020)
Paper D
Published in the Journal Artificial Intelligence and Law (2021)
Paper E
Published in the Proceedings of Modelsward (2021)
Paper F
MDH Technical Report. June, 2021. A version is submitted to a journal

Figure 6.1: Relationships between Research Goals, Contributions and Papers.

ACCEPT is supported by the tool-chain constituted of EPF-C, which has good coverage of SPEM 2.0 concepts, and the compliance checker Regorous, which can perform automatic analysis of FCL rules. EPF-C is an environments that provides hierarchal structures that facilitates keeping track of the applicable requirements. ACCEPT is equipped with a set of methodological steps that facilitate its use. ACCEPT provides a baseline for fulfilling the requirements 1, 2 and 3 elicited in contribution 1. With this contribution, we fulfilled our second subgoal, namely, *identify mechanisms for supporting automated compliance checking of the process plans used in the safety-critical context*. This contribution was summarized in Section 4.2 and found in papers A and D.

**Contribution 3.** We proposed three mechanisms for facilitating the creation and reuse of the specifications required to check compliance of process plans automatically, which is the four requirement elicited in contribution 1. The first two mechanisms are called process compliance hints and patterns. Process compliance hints (summarized in Section 4.3.1) are resources designed to support the creation of separated concepts included in the FCL specification based on process structure and the concepts supported by SPEM 2.0. (i.e., tasks, roles, work product, guidance, tools, and their relationships). Process compliance patterns (summarized in Section 4.3.2) indicate common situations an FCL designer is likely to encounter when formalizing standards. Both process compliance hints and patterns provide reusable means for facilitating the formalization of process-related requirements into FCL rules. The third mechanism (summarized in Section 4.4) is the systematic reuse of compliance checking artifacts. This solution is based on the reuse capabilities provided by SoPLE and the support provided by the integration of the tools EPF-C and BVR-T. With these contributions, we fulfilled our third subgoal, namely, *facilitate the creation of reusable specifications required for automated compliance checking of the process plans*. These contributions are found in the papers as follows. The process compliance hints are found in paper B. The process compliance patterns are found in papers A and D. Finally, the systematic reuse of compliance artifacts is found in papers A and E.

**Contribution 4.** Finally, we validated our proposed solutions. First, we investigated the degree of acceptance of methods for automated compliance checking via a personal opinion survey, which was applied to practitioners who participate in safety-related process compliance checking. Second, we assessed ACCEPT, via a standard-based case study applicable to space software engineering processes, considering a specific set of qualitative usefulness criteria. Third, we measured the reuse permitted by our solutions when showing process plan adherence with new versions of standards related to the medical domain. Fourth, we positioned our work with respect to the state-of-the-art and the state-of-the-practice by characterizing the existent approaches for compliance checking of software processes and testing our assumptions regarding compliance checking current practices and challenges. We found that our solutions provide a set of characteristics that seamlessly integrate with current practices. We also identified aspects that permit our solutions to improve in the future. With this contribution, we fulfilled our fourth subgoal, namely, *investigate the significance of a solution for automated compliance checking of process plans in the safety-critical context*. This last contribution, which was summarized in Section 4.5, is found in papers C, D, E, and F.

## 6.2 Limitations

Our solutions focus on compliance checking of the process plans used to engineer safety-critical systems. The findings of this study have to be seen in light of some limitations.

1. Manufacturers of safety-critical systems commonly make use of different strategies to meet their needs regarding process compliance management. In particular, this thesis address one of these strategies, i.e., compliance checking of processes at planning time. In general, process compliance can also be checked at runtime or post-execution time. Our framework can provide a static view of the compliance status of the process at a specific execution point that could support the decision-making process. However, it cannot help the process engineers enforce compliance in the same way it is done at planning time since activities already executed cannot be changed.

2. SPI frameworks, in addition to the process reference framework, also consider the process outcomes/attributes ratings to determine the capability level of the process (i.e., how well a manufacturer performs its processes). Our framework provides the status of the process compliance to the process reference framework provided by such standards, i.e., whether the tasks and work products are achieved. The compliance reports obtained from our framework can only be used as a baseline to manually determine the capability levels of the checked process.

3. The annotation of the compliance effects for the tasks, roles, work products, tools, and guidance, is done manually, i.e., the process engineers, based on their own experience, deduce the effects that the process elements can eventually cause. This procedure could pose some challenges when the process plans are extensive, i.e., the manual annotation could be time-consuming.

4. Process compliance hints, which are our rule of thumb to address separated concerns within the compliance information of a process, are only conceptually achieved in our contribution. For this reason, such separation of concerns is not still taken into account in practice by the checker.

5. The compliance checking methodology used in our approach is process modeling language agnostic. However, the tool support lacks agnosticism,

i.e., it depends on a specific modeling tool to provide compliance checking results. This characteristic is currently impeding the back-propagation of the compliance results in our selected process modeling language.

## 6.3 Future Work

We believe that the work needed to be done by a process engineer regarding compliance management will benefit from the contributions achieved in this dissertation since they provide a baseline for automation and systematic reuse. We also consider that there are research directions that could increase the scope of our current results, as presented below.

1. **Validation.** With the time we had at hand, we have performed a limited evaluation of our achieved contributions. Thus, future work will include more empirical research with the use of interviews and observations to see, for instance, how practitioners carry out their compliance checking practices with our solutions in natural settings, i.e., industrial environments. In particular, the experience of effort (or perception of effort) is a factor that is very important to analyze since it can provide feedback on task difficulty. With these evaluation results, we can revisit usability aspects and improve our models' representation if needed. In addition, we consider including more elaborated measurement frameworks (e.g., metrics) to provide evidence concerning our solution's efficiency in terms of time and cost reduction (manual vs. automatic work). Finally, we could generate fitness functions that facilitate calculations regarding the adequacy of the information coverage level provided by the models, the compliance documentation, and the evolution management (number of artifacts modeled vs. number of artifacts required by the normative framework).

2. **Automation.** In general, the methodologies defined in our solution are systematic and can be further automated. For example, extensions to the current algorithm used for compliance checking must be designed and implemented to permit the inclusion of co-occurrent compliance effects (effects that coincide in a task), which are annotated in process elements ascribed to tasks. In addition, we consider that algorithms that automatically enable the creation of process elements from the definitional compliance effects should be investigated. We also need to find strategies to represent the compliance checking results in an agnostic way to permit their automatic backpropagation in our selected process modeling tool. We also consider the creation

of rule editors that facilitate the use of templates for process compliance hints and patterns. Finally, algorithms that facilitate the automation of the formalization of requirements, based on natural language processing, could be proposed.

3. **Impact.** A trust mechanism to guarantee that rules are correctly derived from the normative frameworks, i.e., it validates that requirements are not lost or poisoned during formalization processes, is required. A solution for this aspect 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 standards or new versions of already positioned standards. With this strategy, we could reduce undesired room for interpretation of the normative texts. In addition, our solutions should be integrated into the platform created by the European project AMASS (Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems) [181]. With this strategy, we aim at concretizing our work in terms of tool support.

4. **Extension.** Additional mechanisms for supporting compliance at runtime or post-execution time (e.g., automated alerts, task post-execution analysis, and recommendations systems) can be provided to manage compliance issues once violations are detected in static compliance checks. These mechanisms could be beneficial in agile environments, which can make use of static checks after each process iteration to support subsequent iterations in a compliant-informed fashion. Algorithms that consider the ratings required for process capability determination could also be designed and implemented for fully automated support regarding process improvement standards.

5. **Generalization.** We consider it essential to provide case studies by considering a broader range of standards in contexts beyond the ones already evaluated.

# Bibliography

[1] AMASS., "Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems." http://www.amass-ecsel.eu/.

[2] N. Leveson, "Safety : Why, What, and How," *ACM Computing Surveys (CSUR)*, vol. 18, no. 2, pp. 125–163, 1986.

[3] V. Icheku, *Understanding Ethics and Ethical Decision-Making*. Xlibris Corporation, 2011.

[4] P. B. Ladkin, "Duty of Care and Engineering Functional Safety Standards," *Digital Evidence & Elec. Signature L. Rev.*, vol. 16, p. 51, 2019.

[5] D. J. Smith and K. G. Simpson, *The Safety Critical Systems Handbook: A Straightforward Guide to Functional Safety: IEC 61508 (2010 Edition), IEC 61511 (2015 Edition) and Related Guidance*. Butterworth-Heinemann, 2020.

[6] M. Generowicz, "The Easy Path to Functional Safety Compliance." https://www.iesystems.com.au/wp-content/uploads/2015/04/Duty-of-Care-Article.pdf, 2013.

[7] J. Knight and J. Rowanhill, "The Indispensable Role of Rationale in Safety Standards," in *35th International Conference SAFECOMP*, vol. 2788, pp. 39–50, Springer, 2016.

[8] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 2016.

[9] S. O. Hansson, "Safe Design," *Techné: Research in Philosophy and Technology*, vol. 10, no. 1, pp. 45–52, 2006.

[10] J. Bowen, “The Ethics of Safety-critical Systems,” *Communications of the ACM*, vol. 43, no. 4, pp. 91–97, 2000.

[11] G. T. Marx, “Technology and Social Control,” 2015.

[12] International Electrotechnical Commission, “*IEC 61508-Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*,” 2010.

[13] J. Finch, “Toyota Sudden Acceleration: A Case Study of the National Highway Traffic Safety Administration - Recalls for Change,” *Loy. Consumer L. Rev.*, vol. 22, p. 472, 2009.

[14] M. M. Rahim and S. O. Idowu, *Social Audit Regulation: Development, Challenges and Opportunities*. Springer, 2015.

[15] P. A. Defazio and R. Larsen, “Final Committe Report - The Design, Development & Certification of the Boeing 737 MAX ,” Tech. Rep. September, The House Committe on Transportation & Infrastructure, 2020.

[16] *Komite Nasional Keselamatan Transportasi (KNKT), West Java, Republic of Indonesia*, “Final KNKT.18.10.35.04 Aircraft Accident Investigation Report, PT. Lion Mentari Airlines, Boeing 737-8 (MAX),” Tech. Rep. October 29, 2019.

[17] *Federal Democratic Republic of Ethiopia, Ministry of Transport, Aircraft Accident Investigation Bureau*, “Interim Investigation Report on Accident to the B737-8(MAX) Registered ET-AVJ operated by Ethiopian Airlines on 10 March 2019,” Tech. Rep. March 9, 2020.

[18] M. A. Cusumano, “Who is Liable for Bugs and Security Flaws in Software?,” *Communications of the ACM*, vol. 47, no. 3, pp. 25–27, 2004.

[19] S. Ingolfo, A. Siena, and J. Mylopoulos, “Establishing Regulatory Compliance for Software Requirements,” in *International Conference on Conceptual Modeling*, pp. 47–61, Springer, 2011.

[20] A. Tiron-Tudor and C. Bota-Avram, “New Challenges for Internal Audit: Corporate Social Responsibility Aspects,” in *Social Audit Regulation*, pp. 15–31, Springer, 2015.

[21] International Organization for Standardization/International Electrotechnical Commission, "*ISO/IEC 17000:2020 - Conformity Assessment — Vocabulary and General Principles*," 2020.

[22] R. Kemp, "Regulating the Safety of Autonomous Vehicles using Artificial Intelligence," *Communications Law*, vol. 24, no. 1, pp. 24–33, 2019.

[23] F. Moyón, D. Méndez, K. Beckers, and S. Klepper, "How to Integrate Security Compliance Requirements with Agile Software Engineering at Scale?," in *International Conference on Product-Focused Software Process Improvement*, pp. 69–87, Springer, 2020.

[24] A. Ruiz, G. Juez, H. Espinoza, J. de la Vara, and X. Larrucea, "Reuse of Safety Certification Artefacts across Standards and Domains: A Systematic Approach," *Reliability Engineering & System Safety*, vol. 158, pp. 153–171, 2016.

[25] C. Di Ciccio, A. Marrella, and A. Russo, "Knowledge-intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches," *Journal on Data Semantics*, vol. 4, no. 1, pp. 29–57, 2015.

[26] G. Cugola and C. Ghezzi, "Software Processes: a Retrospective and a Path to the Future," *Software Process: Improvement and Practice*, vol. 4, no. 3, pp. 101–123, 1998.

[27] R. Kneuper, *Software Processes and Life Cycle Models. An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes*. Springer, Cham, 2018. ISBN 978-3-319-98845-0.

[28] A. Siena, J. Mylopoulos, A. Perini, and A. Susi, "From Laws to Requirements," in *2008 Requirements Engineering and Law*, pp. 6–10, IEEE, 2008.

[29] Radio Technical Commission for Aeronautics (RTCA) & European Organisation for Civil Aviation Equipment (EUROCAE), "*RTCA/DO-178C - Software Considerations in Airborne Systems and Equipment Certification.*," 2011.

[30] Swedish Ratiation Safety Authority, "Licensing of Safety Critical Software for Nuclear Reactors," tech. rep., 2018.

[31] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, "Managing Standards Compliance," *IEEE Transactions on Software Engineering*, vol. 25, no. 6, pp. 836–851, 1999.

[32] P. Chung, L. Cheung, and C. Machin, "Compliance Flow-Managing the Compliance of Dynamic and Complex Processes," *Knowledge-Based Systems*, vol. 21, no. 4, pp. 332–354, 2008.

[33] SAE, "Surface Vehicle Recommended Practice," tech. rep., 2016.

[34] International Organization for Standardization, "*ISO 26262:2018 - Road vehicles – Functional safety*," 2018.

[35] VDA QMC Working Group 13 / Automotive SIG, "*Automotive SPICE Process Assessment – Reference Model*," 2015.

[36] N. Ramasubbu, A. Bharadwaj, and G. K. Tayi, "Software Process Diversity: Conceptualization, Measurement, and Analysis of impact on Project Performance," *Management Information Systems*, vol. 39, no. 4, pp. 787–807, 2015.

[37] European Space Agency, "*ECSS-E-HB-40C – Space engineering - Software engineering handbook*," 2013. https://ecss.nl/hbstms/ecss-e-hb-40a-software-engineering-handbook-11-december-2013/.

[38] International Organization for Standardization, "*ISO 14971:2007 – Application of Risk Management to Medical Devices*," Mar. 2007.

[39] EN International Organization for Standardization, "*ISO 14971:2012 - Medical Devices – Application of Risk Management to Medical Devices (ISO 14971:2007, Corrected version 2007-10-01)*," July 2012.

[40] International Organization for Standardization , "*ISO 14971:2019 - Medical Devices – Application of Risk Management to Medical Devices*," Dec. 2019.

[41] B. Gallina, A. Pulla, A. Bregu, and J. P. Castellanos Ardila, "Process Compliance Re-Certification Efficiency Enabled by EPF-C $\circ$ BVR-T: A Case Study," in *International Conference on the Quality of Information and Communications Technology*, pp. 211–219, Springer, 2020.

[42] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 2011.

[43] R. O'Connor and C. Laporte, *An Innovative Approach to the Development of an International Software Process Lifecycle Standard for Very Small Entities*, vol. III. IGI Global, 2018.

[44] B. Gallina, E. Gómez-Martínez, and C. Benac-Earle, "Promoting MBA in the Rail Sector by Deriving Process-related Evidence via MDSafeCer," *Computer Standards & Interfaces*, vol. 54, pp. 119–128, 2017.

[45] R. Lu, S. Sadiq, and G. Governatori, "Compliance Aware Business Process Design," in *International Conference on Business Process Management*, pp. 120–131, 2007.

[46] G. Governatori, "Representing Business Contracts in RuleML," *International Journal of Cooperative Information Systems*, vol. 14, no. 02n03, pp. 181–216, 2005.

[47] "*Eclipse Process Framework (EPF) Composer*." http://www.eclipse.org/epf/.

[48] Object Management Group Inc., "Software & Systems Process Engineering Meta-Model Specification. Version 2.0.," *OMG Std., Rev*, p. 236, 2008.

[49] G. Governatori, "The Regorous Approach to Process Compliance," in *19th International Enterprise Distributed Object Computing Workshop*, pp. 33–40, IEEE, 2015.

[50] B. Gallina, I. Sljivo, and O. Jaradat, "Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification," in *35th Annual IEEE Software Engineering Workshop*, pp. 148–157, 2012.

[51] M. Javed and B. Gallina, "Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool," in *22nd International Systems and Software Product Line Conference*, pp. 23–28, ACM, 2018.

[52] A. Ruiz, B. Gallina, J. L. de la Vara, S. Mazzini, and H. Espinoza, "AMASS: Architecture-driven, Multi-concern, Seamless,

Reuse-Oriented Assurance and Certification of CPSs," in *5th International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems (SASSUR). SafeComp, International conference on computer safety, reliability and security, Trondheim*, 2016.

[53] J. Luis de la Vara, A. Ruiz Lopez, B. Gallina, G. Blondelle, E. Alaña, J. Herrero, F. Warg, M. Skoglund, and R. Bramberger, "The AMASS Approach for Assurance and Certification of Critical Systems," in *Embedded world 2019 ewC-2019, 26 Feb 2019, Nuremberg, Germany*, 2019.

[54] S. Ghanavati, D. Amyot, and L. Peyton, "Comparative Analysis between Document-based and Model-based Compliance Management Approaches," in *Requirements Engineering and Law*, pp. 35–39, 2008.

[55] European Space Agency, "*ECSS-E-ST-40C – Space Engineering Software*," 2009. https://ecss.nl/standard/ecss-e-st-40c-software-general-requirements/.

[56] International Organization for Standardization, "*ISO 14971:2000 – Application of Risk Management to Medical Devices*," Dec. 2000.

[57] R. Banker, R. Kauffman, and D. Zweig, "Repository Evaluation of Software Reuse," *IEEE Transactions on Software Engineering*, vol. 19, no. 4, pp. 379–389, 1993.

[58] C. Perrow, *Normal accidents: Living with High Risk Technologies - Updated edition*. Princeton university press, 2011.

[59] N. Leveson, "SAFETY III: A Systems Approach to Safety and Resilience," 2020.

[60] J. C. Knight, "Safety Critical Systems: Challenges and Directions," in *24rd International Conference on Software Engineering*, pp. 547 – 550, ACM, 2002.

[61] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones, "Certifiably Safe Software-dependent Systems: Challenges and Directions," *Proceedings of the on Future of Software Engineering*, pp. 182–200, 2014.

[62] N. G. Leveson, "System Safety in Computer-Controlled Automotive Systems," *SAE transactions*, vol. 109, pp. 287–294, 2000.

[63] N. Fenton and M. Neil, “A Strategy for Improving Safety Related Software Engineering Standards,” *IEEE Transactions on Software Engineering*, vol. 24, no. 11, pp. 1002–1013, 1998.

[64] A. Schwartz, “Statutory Interpretation, Capture, and Tort Law: The Regulatory Compliance Defense,” *American Law and Economics Review*, vol. 2, no. 1, pp. 1–57, 2000.

[65] N. Brunsson and B. Jacobsson, *A World of Wtandards*. Oxford University Press, 2010.

[66] Radio Technical Commission for Aeronautics, “*RTCA/DO-330- Software Tool Qualification Considerations*,” 2012.

[67] European Committee for Electrotechnical Standardization, “*CENELEC - EN 50128 - Railway Applications – Communication, Signaling and Processing Systems Software for Railway Control and Protection Systems*,” 2011.

[68] European Space Agency, “*European Cooperation for Space Standardization*,” 1993. https://ecss.nl/.

[69] International Organization for Standardization/International Electrotechnical Commission, “*ISO/IEC/IEEE 12207 - Systems and Software Engineering — Software Life Cycle Processes*,” 2017.

[70] International Organization for Standardization, “*ISO 9000 - Quality Management Systems-Fundamentals and Vocabulary*,” 2005.

[71] International Organization for Standardization (ISO) and International Electrotechnical Commission , “*IEC ISO/IEC 15504 - Information Technology – Process assessment - An Exemplar Software Life Cycle Process Assessment model*,” 2012.

[72] Software Engineering Institute, Carnegie Mellon, “CMMI® for Development, Version 1.3 CMMI-ACQ, V1.3,” Tech. Rep. November, Software Engineering Institute, Carnegie Mellon, 2010.

[73] B. Gallina, F. U. Muram, and J. P. Castellanos Ardila, “Compliance of Agilized (Software) Development Processes with Safety Standards: A Vision,” in *Proceedings of the 19th International Conference on Agile Software Development: Companion*, pp. 1–6, 2018.

[74] G. Regan, M. Biro, F. Mc Caffery, K. Mc Daid, and D. Flood, “A Traceability Process Assessment Model for the Medical Device Domain,” in *European Conference on Software Process Improvement*, pp. 206–216, Springer, 2014.

[75] M. Palmirani and G. Governatori, “Modelling Legal Knowledge for GDPR Compliance Checking,” *Frontiers in Artificial Intelligence and Applications*, vol. 313, pp. 101–110, 2018.

[76] E. Ahmad, B. Raza, R. Feldt, and T. Nordebäck, “ECSS Standard Compliant Agile Software Development: An Industrial Case Study,” in *National Software Engineering Conference*, pp. 1–6, 2010.

[77] European Space Agency, “*ECSS-Q-ST-40C – Space Product Assurance - Safety*,” 2017. https://ecss.nl/standard/ecss-q-st-40c-rev-1-safety-15-february-2017/.

[78] A. Pulla and A. Bregu, “Master Thesis: Evaluating the Compliance Re-Certification Efficiency Enabled by the AMASS Platform for Medical Devices, Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden,” 2020.

[79] Council of the European Union, “*Council Directive 90/385/EEC of 20 June 1990 on the approximation of the laws of the Member States relating to active implantable medical devices*,” June 1990.

[80] The Council of the European Communities, “*Council Directive 93/42/EEC of 14 June 1993 concerning medical devices*,” June 1993.

[81] European Parliament & Council of the European Union, “*Directive 98/79/EC of the European Parliament and of the Council of 27 October 1998 on in vitro diagnostic medical devices*,” Oct. 1998.

[82] T. Boutros and T. Purdie, *The Process Improvement Handbook: A Blueprint for Managing Change and Increasing Organizational Performance*. 2014.

[83] L. Osterweil, “Software Processes are Software Too,” Engineering of Software, pp. 323–344, Springer, 2011.

[84] A. Fuggetta and E. Di Nitto, “Software Process,” in *Future of Software Engineering*, pp. 1–12, 2014.

[85] L. García-Borgoñón, M. Barcelona, J. García-garcía, M. Alba, and M. Escalona, "Software Process Modeling Languages : A systematic Literature Review," *Information and Software Technology*, vol. 56, no. 2, pp. 103–116, 2014.

[86] U. Becker-Kornstaedt, D. Hamann, R. Kempkens, P. Rösch, M. Verlage, R. Webby, and J. Zettel, "Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance," *International Conference on Advanced Information Systems Engineering*, pp. 119–133, 1999.

[87] J. Lonchamp, "A Structured Conceptual and Terminological Framework for Software Process Engineering," *2nd International Conference on the Software Process*, pp. 41–53, 1993.

[88] B. McIsaac, "IBM Rational Method Composer: Standards Mapping.," tech. rep., IBM Developer Works, 2015.

[89] ECSEL Research and Innovation actions (RIA) - AMASS, "D6.5 Prototype for Cross/Intra-Domain Reuse (b)," 2017.

[90] IBM Corporation, "Key Capabilities of the Unified Method Architecture (UMA)." http://deg.egov.bg/LP/core.base_concepts/guidances/concepts/introduction_to_uma_931F5B93.html.

[91] M. Javed and B. Gallina, "Get EPF Composer Back to the Future: A Trip from Galileo to Photon After 11 Years," in *EclipseCon*, 2018.

[92] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Addison-Wesley Reading, 2002.

[93] T. Ternité, "Process Lines : A Product Line Approach designed for Process Model Development," in *35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).*, pp. 173–180, 2009.

[94] J. P. Castellanos Ardila and B. Gallina, "Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards," in *The 7th IEEE International Workshop on Software Certification*, 2017.

[95] P. Casanovas, J. González-Conejero, and L. De Koker, "Legal Compliance by Design (LCbD) and through Design (LCtD): Preliminary Survey," in *1st Workshop on Technologies for Regulatory Compliance Legal*, pp. 33–49, 2017.

[96] S. Sadiq and G. Governatori, “A Methodological Framework for Aligning Business Processes and Regulatory Compliance,” *Handbook of Business Process Management’, Springer*, 2009.

[97] G. Governatori, M. Hashmi, H. Lam, S. Villata, and M. Palmirani, “Semantic Business Process Regulatory Compliance Checking using LegalRuleML,” in *European Knowledge Acquisition Workshop*, pp. 746–761, Springer, 2016.

[98] F. Koetter, M. Kochanowski, T. Renner, C. Fehling, and F. Leymann, “Unifying Compliance Management in Adaptive Environments through Variability Descriptors,” *6th International Conference on Service-Oriented Computing and Applications*, pp. 214–219, 2013.

[99] M. Hashmi, G. Governatori, and M. Wynn, “Normative Requirements for Regulatory Compliance: An Abstract Formal Framework,” *Information Systems Frontiers.*, vol. 18, no. 3, pp. 429–455, 2016.

[100] E. Francesconi, “Semantic Model for Legal Resources: Annotation and Reasoning over Normative Provisions,” *Semantic Web*, vol. 7, no. 3, pp. 255–265, 2016.

[101] L. Lúcio, S. Rahman, C.-H. Cheng, and A. Mavin, “Just Formal Enough? Automated Analysis of EARS Requirements,” NASA Formal Methods Symposium, pp. 427–434, Springer, 2017.

[102] E. Kabaale, L. Wen, Z. Wang, and T. Rout, “Representing Software Process in Description Logics: An Ontology Approach for Software Process Reasoning and Verification,” in *Software Process Improvement and Capability Determination*, pp. 362–376, Springer, 2016.

[103] S. Vilkomir, J. Bowen, and A. Ghose, “Formalization and Assessment of Regulatory Requirements for Safety-critical Software,” *Innovations in Systems and Software Engineering*, vol. 2, no. 3-4, pp. 165–178, 2006.

[104] J. Munoz-Gama, “Conformance Checking and its Challenges,” *Conformance Checking and Diagnosis in Process Mining Comparing Observed and Modeled Processes*, pp. 11–18, 2016.

[105] M. Hashmi, *Evaluating Business Process Compliance Management Frameworks*. Doctoral dissertation, Queensland University of Technology (QUT), 2015.

[106] A. Toval, A. Olmos, and M. Piattini, “Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection,” in *IEEE Joint International Conference on Requirements Engineering*, pp. 95–103, IEEE, 2002.

[107] D. Brown, H. Delseny, K. Hayhurst, and V. Wiels, “Guidance for Using Formal Methods in a Certification Context,” ERTS2 2010, Embedded Real Time Software & Systems, 2010.

[108] P. N. Otto and A. I. Antón, “Addressing Legal Requirements in Requirements Engineering,” in *15th IEEE international requirements engineering conference (RE 2007)*, pp. 5–14, IEEE, 2007.

[109] Department of Defense Standard Practice, “*MIL-STD-882E-System Safety*,” 2012.

[110] P. J. Denning and M. Tedre, *Computational Thinking*. MIT Press, 2019.

[111] D. Smith, “The Design of Divide and Conquer Algorithms,” *Science of Computer Programming*, vol. 5, pp. 37–58, 1985.

[112] I. Sommerville, “Software Processes,” in *Software Engineering*, ch. 2, pp. 43–55, Addison-Wesley, 9th editio ed., 2011.

[113] M. Dwyer, G. Avrunin, and J. Corbett, “Property Specification for Finite-State Verification,” in *International Conference on Software Engineering.*, pp. 411–420, 1998.

[114] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, Berlin, Heidelberg, 2014.

[115] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[116] B. Kitchenham and S. Pfleeger, “Personal Opinion Surveys,” in *Guide to Advanced Empirical Software Engineering*, ch. 3, pp. 63–92, Springer Science & Business Media, 2008.

[117] D. Bertram, “Likert Scales Are the Meaning of Life. CPSC 681-Topic Report,” 2006. http://poincare.matf.bg.ac.rs/ kristina/topic-dane-likert.pdf.

[118] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering," Tech. Rep. 4ve, 2007.

[119] C. Wohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering," *18th International Conference on Evaluation and Assessment in Software Engineering*, pp. 1–10, 2014.

[120] F. Davis, "A Technology Acceptance Model for Empirically Testing New End-user Information Systems: Theory and Results.," *Massachusetts Institute of Technology*, 1985.

[121] J. Steele, "What is (Perception of) Effort? Objective and Subjective Effort during Task Performance," *PsyArXiv, doi: 10.31234/osf.io/kbyhm*, 2020.

[122] P. Engiel, J. Sampaio do Prado Leite, and J. Mylopoulos, "A Tool-Supported Compliance Process for Software Systems," in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pp. 66–76, IEEE, 2017.

[123] L. Wen, D. Tuffley, and T. Rout, "Using Composition Trees to Model and Compare," in *International Conference on Software Process Improvement and Capability Determination*, no. March 2014, pp. 1–15, Springer, 2011.

[124] E. Kabaale, L. Wen, Z. Wang, and T. Rout, "Ensuring Conformance to Process Standards Through Formal Verification," in *International Conference on Software Process Improvement and Capability Determination*, vol. 2, pp. 248–262, Springer International Publishing, 2018.

[125] D. Proença and J. Borbinha, "Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3 – Enabling Automatic Inference of Maturity and Capability Levels," *Computer Standards and Interfaces*, 2018.

[126] G. Soydan and M. Kokar, "A Partial Formalization of the CMMI-DEV — A Capability Maturity Model for Development," *Journal of Software Engineering and Applications*, vol. 5, no. 10, pp. 777–788, 2012.

[127] P. Bonatti, "Fast Compliance Checking in an OWL2 Fragment," in *27th International Joint Conferences on Artificial Intelligence Organization*, pp. 1746–1752, 2018.

[128] W3C, "Web Ontology Language (OWL)," 2012. https://www.w3.org/2001/sw/wiki/OWL.

[129] A. Borgida, "On the Relative Expressiveness of Description Logics and Predicate Logics," *Artificial intelligence*, vol. 82, no. 1-2, pp. 353–367, 1996.

[130] R. Bendraou, B. Combemale, X. Crégut, and M. Gervais, "Definition of an Executable SPEM 2.0," in *14th Asia-Pacific Software Engineering Conference.*, pp. 390–397, 2007.

[131] F. Golra, F. Dagnat, R. Bendraou, and A. Beugnard, "Continuous Process Compliance Using Model Driven Engineering," in *International Conference on Model and Data Engineering*, pp. 42–56, Springer, 2017.

[132] D. Rodríguez, E. Garcia, S. Sanchez, and C. Rodríguez-Solano, "Defining Software Process Model Constraints with Rules using OWL and SWRL," *International Journal of Software Engineering and Knowledge Engineering*, vol. 20, no. 4, pp. 533–548, 2010.

[133] M. Valiente, E. García-Barriocanal, and M. Sicilia, "Applying Ontology-Based Models for Supporting Integrated Software Development and IT Service," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 42, no. 1, pp. 61–74, 2012.

[134] I. Horrocks, P. Patel-schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.

[135] S. Wang, L. Jin, and C. Jin, "Represent Software Process Engineering Metamodel in Description Logic," in *World Academy of Science, Engineering and Technology*, vol. 11, pp. 109–113, 2006.

[136] H. Jost, S. Köhler, and F. Köster, "Towards a Safer Development of Driver Assistance Systems by Applying Requirements-Based Methods," in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1144–1149, IEEE, 2011.

[137] B. Gallina, E. Gómez-Martínez, and C. Earle, "Deriving Safety Case Fragments for Assessing MBASafe ' s Compliance with EN 50128," in *International Conference on Software Process Improvement and Capability Determination*, vol. 1, pp. 3–16, 2016.

[138] A. Gannous, A. Andrews, and B. Gallina, "Toward a Systematic and Safety Evidence Productive Verification Approach for Safety-Critical Systems," *The 29th IEEE International Symposium on Software Reliability Engineering (ISSRE 2018)*, pp. 329–336, 2018.

[139] International Organization for Standardization (ISO) and International Electrotechnical Commission , "*(IEC) ISO/IEC 29110:2016 - Systems and software engineering – Lifecycle profiles for Very Small Entities (VSEs)*," 2016.

[140] École de technologie supérieure of Canada., "Deployment Packages for the Generic Profile Group for VSEs Developing Systems and/or Software," 2010.

[141] F. Ul Muram, B. Gallina, and L. Gomez Rodriguez, "Preventing Omission of Key Evidence Fallacy in Process-based Argumentations," in *11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, 2018.

[142] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC,, 2009.

[143] C. Giblin, S. Müller, and B. Pfitzmann, "From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation," tech. rep., IBM Research Laboratory, Zurich, 2006.

[144] C. Giblin, A. Liu, S. Müller, B. Pfitzmann, and X. Zhou, "Regulations Expressed As Logical Models (REALM)," tech. rep., IBM China Research Lab, 2005.

[145] F. Daniel, F. Casati, E. Mulo, U. Zdun, S. Strauch, D. Schumm, F. Leymann, S. Sebahi, F. De Marchi, and M. S. Hacid, "Business Compliance Governance in Service-oriented Architectures," in *International Conference on Advanced Information Networking and Applications (AINA)*, pp. 113–120, 2009.

[146] A. Awad, G. Decker, and M. Weske, "Efficient Compliance Checking Using BPMN-Q and Temporal Logic," *International Conference on Business Process Management*, pp. 326–341, 2008.

[147] L. Ly, K. Göser, S. Rinderle-ma, and P. Dadam, "Compliance of Semantic Constraints – A Requirements Analysis for Process Management

Systems," in *1st International Workshop on Governance, Risk and Compliance - Applications in Information Systems*, 2008.

[148] Object Management Group, "Business Process Model and Notation Version 2.0," 2011.

[149] A. Elgammal, O. Turetken, W. van den Heuvel, and M. Papazoglou, "Formalizing and Applying Compliance Patterns for Business Process Compliance," *Software and Systems Modeling.*, pp. 119–146, 2016.

[150] D. Schumm, O. Turetken, N. Kokash, A. Elgammal, F. Leymann, and W. van den Heuvel, "Business Process Compliance through Reusable Units of Compliant Processes," in *International Conference on Web Engineering*, pp. 325–337, 2010.

[151] M. El Kharbili, "Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation," in *8th Asia-Pacific Conference on Conceptual Modelling.*, pp. 23–32, 2012.

[152] G. Governatori and M. Hashmi, "No Time for Compliance," no. Section II, 2015.

[153] M. Ceci, T. Butler, L. Brien, and F. Al Khalil, "Legal Patterns for Different Constitutive Rules," *AI Approaches to the Complexity of Legal Systems*, pp. 105–123, 2015.

[154] S. Islam, H. Mouratidis, and S. Wagner, "Towards a Framework to Elicit and Manage Security and Privacy Requirements from Laws and Regulations," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 255–261, 2010.

[155] S. Faßbender and M. Heisel, "A Computer-Aided Process from Problems to Laws in Requirements Engineering," in *Software Technologies*, pp. 215–234, Springer Berlin Heidelberg, 2014.

[156] B. Akinkunmi and M. Babalola, "Knowledge Representation for High-Level Norms and Violation Inference in Logic Programming," *arXiv preprint arXiv:1801.06740*, 2018.

[157] K. Bouzidi, C. Faron-Zucker, B. Fies, and N. Le Thanh, "An Ontological Approach for Modeling Technical Standards for Compliance Checking," in *International Conference on Web Reasoning and Rule Systems*, pp. 244–249, Springer, 2011.

[158] S. Bala, C. Cabanillas, A. Haselböck, G. Havur, J. Mendling, S. Sperl, and S. Steyskal, "A Framework for Safety-Critical Process Management in Engineering Projects," in *International Symposium on Data-Driven Process Discovery and Analysis*, vol. 1, pp. 1–27, 2015.

[159] E. Nash, J. Wiebensohn, R. Nikkilä, A. Vatsanidou, S. Fountas, and R. Bill, "Towards Automated Compliance Checking based on a Formal Representation of Agricultural Production Standards," *Computers and Electronics in Agriculture*, vol. 78, no. 1, pp. 28–37, 2011.

[160] G. Brat, D. Bushnell, M. Davies, D. Giannakopoulou, F. Howar, and K. Temesghen, "Verifying the Safety of a Flight-Critical System," in *International Symposium on Formal Methods*, pp. 308–324, Springer, 2015.

[161] S. Sadiq, G. Governatori, and K. Namiri, "Modeling Control Objectives for Business Process Compliance," *Business Process Management*, pp. 149–164, 2007.

[162] M. Hashmi, G. Governatori, and M. Wynn, "Business Process Data Compliance," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, vol. 7438 LNCS, pp. 32–46, 2012.

[163] J. Lee, "Oracle Policy Automation (OPA). Best Practice Guide for policy Modelers.," 2018. https://www.oracle.com/technetwork/apps-tech/policy-automation/learnmore/opabestpracticeguidev12-3697709.pdf.

[164] J. Fernandez, "Deliverable 6.1: Privacy Policy Formalization (v. 1)," 2018. http://cityspin.net/wp-content/uploads/2017/10/D6.1-Privacy-policy-formalization.pdf.

[165] N. Singh, M. Lawford, T. Maibaum, and A. Wassyng, "Use of Tabular Expressions for Refinement Automation," in *International Conference on Model and Data Engineering*, pp. 167–182, 2017.

[166] G. Governatori, "Practical Normative Reasoning with Defeasible Deontic Logic Practical Normative Reasoning with Defeasible Deontic Logic," no. September, 2018.

[167] Y. Luo, M. Van Den Brand, L. Engelen, J. Favaro, M. Klabbers, and G. Sartori, "Extracting Models from ISO 26262 for Reusable Safety Assurance," *Lecture Notes in Computer Science*, vol. 7925 LNCS, pp. 192–207, 2013.

[168] D. Giannakopoulou, K. Namjoshi, and K. Pasareanu, "Compositional Reasoning," in *Handbook of logic and language*, pp. 345–383, Springer, 2018.

[169] W. Reif and K. Stenzel, "Reuse of Proofs in Software Verification," in *International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pp. 284–293, Lecture Notes in Computer Science, 1993.

[170] B. Beckert and V. Klebanov, "Proof Reuse for Deductive Program Verification," in *Proceedings of the Second International Conference on Software Engineering and Formal Methods*, pp. 77–86, IEEE, 2004.

[171] D. Beyer and P. Wendler, "Reuse of verification results conditional model checking, precision reuse, and verification witnesses," *Model Checking Software*, vol. 7976, pp. 1–17, 2013.

[172] S. Apel, A. Von Rhein, P. Wendler, A. Groslinger, and D. Beyer, "Strategies for Product-line Verification: Case Studies and Experiments," *International Conference on Software Engineering*, pp. 482–491, 2013.

[173] A. Ocampo and J. Münch, "Rationale Modeling for Software Process Evolution Alexis," *Software Process: Improvement and Practice*, vol. 14, no. 2, pp. 85–105, 2009.

[174] E. N. Teixeira, F. A. Aleixo, F. D. de Sousa Amâncio, E. OliveiraJr, U. Kulesza, and C. Werner, "Software Process Line as an Approach to Support Software Process Reuse: A Systematic Literature Review," *Information and Software Technology*, vol. 116, p. 106175, 2019.

[175] H. Martin, M. Krammer, R. Bramberger, and E. Armengaud, "Process- and Product-based Lines of Argument for Automotive Safety Cases," in *7th International Conference on Cyber-Physical Systems*, 2016.

[176] L. Bressan, A. L. de Oliveira, F. Campos, Y. Papadopoulos, and D. Parker, "An Integrated Approach to Support the Process-Based Certification of Variant-Intensive Systems," in *International Symposium on Model-Based Safety and Assessment*, pp. 179–193, 2020.

[177] D. Schleicher, S. Grohe, F. Leymann, P. Schneider, D. Schumm, and T. Wolf, "An Approach to Combine Data-related and Control-flow-related Compliance Rules," in *International Conference on Service-Oriented Computing and Applications*, pp. 1–8, 2011.

[178] K. Görlach, O. Kopp, F. Leymann, and D. Schumm, "WS-BPEL Extension for Compliance Fragments (BPEL4CFrags)," tech. rep., Institute of Architecture of Application Systems, University of Stuttgart., 2011.

[179] Z. Ma, *Process Fragments: Enhancing Reuse of Process Logic in BPEL Process Models*. Ph.d. dissertation, University of Stuttgart, 2012. http://elib.uni-stuttgart.de/opus/volltexte/2013/8075/.

[180] M. Kabir, Z. Xing, P. Chandrasekaran, and S. Lin, "Process Patterns: Reusable Design Artifacts for Business Process Models," *International Computer Software and Applications Conference*, vol. 1, pp. 714–721, 2017.

[181] J. L. de la Vara, E. P. Corredor, A. R. Lopez, and B. Gallina, "The AMASS Tool Platform: An Innovative Solution for Assurance and Certification of Cyber-Physical Systems," in *26th International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2020.

# II

# Included Papers

# Chapter 7

# Paper A:
Facilitating Automated Compliance Checking of Processes in the Safety-critical Context

Julieth Patricia Castellanos Ardila, Barbara Gallina, Faiz Ul Muram.
Journal of Electronic Communications of the EASST. vol 78. 2019

## Abstract

In some domains, the applicable safety standards prescribe process-related requirements. Essential pieces of evidence for compliance assessment with such standard are the compliance justifications of the process plans used to engineer systems. These justifications should show that the process plans are produced in accordance with the prescribed requirements. However, providing the required evidence may be time-consuming and error-prone since safety standards are large, natural language-based documents with hundreds of requirements. Besides, a company may have many safety-critical-related processes to be examined. In this paper, we propose a novel approach that combines process modeling and compliance checking capabilities. Our approach aims at facilitating the analysis required to conclude whether the model of a process plan corresponds to a model with compliant states. Hitherto, our proposed methodology has been evaluated with academic examples that show the potential benefits of its use.

## 7.1 Introduction

The production of safety-critical systems is regulated by safety standards, which in some domains prescribe processes-related requirements. Those requirements suggests proved procedures and methods as well as specific characteristics of the process that aim at increasing the safety of the engineered systems. For compliance assessment with such standards, complete and convincing justifications, which show that the process-oriented requirements are fulfilled within the planning of the development process, are required [1]. To support the production of compliance justifications, compliance checking reports can be used, since they facilitate the auditor's job in detecting the defects of the inspected processes [2]. However, their manual production may be time-consuming and prone-to-error since they require that process engineers check hundreds of requirements based on the information provided by process specifications, which may be large and complicated.

Modeling languages are available to give process engineers the means to generate process models and management tools to control them [3]. In particular, SPEM 2.0 (Systems & Software Process Engineering Metamodel) [4] is a well-defined standard that is used to model engineering processes. In addition, SPEM 2.0 provide support for Safety-oriented Processes Lines Engineering (SoPLE) [5], which is an approach that facilitate the reuse of process-related elements. However, the model of a process is not enough to prove compliance. The reason is that process compliance is not only related to the structure of a process, but also what the tasks in a process do and their effects in the general process behavior [6]. Therefore, we intend to provide an additional layer of confidence by offering a logic-based framework that facilitates the reasoning from standards requirements and the processes they regulate. For this, we have selected Formal Contract Logic (FCL) [7]. FCL permits to encode rules as conditionals in which the antecedent is read as a property of a state of affairs, and the conclusion has a deontic nature, i.e., notions regarding the obligatory, the permitted and the forbidden. FCL is based on defeasible logic [8], which contrary evidence defeats earlier reasoning, allowing the management of inconsistencies. A set of process-related requirements encoded in FCL can be used to automatize the compliance checking of a given process model. Reasoning with FCL rules is possible with Regorous [9], a compliance checker available on the shelf, which provides traceable conclusions [10].

In this paper, we propose a novel approach for facilitating automatic checking of processes against safety standards that combines: 1) process modeling capabilities for representing systems and software process specifications,

2) normative representation capabilities for adequately encoding the requirements prescribed by the safety standards, 3) compliance checking capabilities to provide the analysis required to conclude whether a process model corresponds to the model with compliant states, and 4) process-line modeling capabilities to systematize the reuse of process-related information. Hitherto, our proposed methodology has been evaluated with academic examples that show the potential benefits of its use.

The rest of the paper is structured as follows. Section 7.2 recalls essential background information. Section 7.3 presents related work. Section 7.4 presents the research summary. Section 7.5 presents preliminary results. Finally, Section 7.6 presents the concluding remarks and next steps.

## 7.2 Background

This section introduces essential background required by the current research.

### 7.2.1 Process-based Compliance

Process-based standards provide detailed guidance, in the form of best practices, that is used by regulatory bodies to tell suppliers what to achieve, and how [11]. Such standards prescribe a safety lifecycle, which describes specific activities related to assuring the safety of the system [12]. One of the key pieces of evidence for process-based compliance management is the safety plan, which represents that a plan has been conceived and documented. However, the provision of the safety plan is not sufficient during the compliance assessment process. A compliance justification in terms of, e.g., a compliance checking report, should also be provided to show that the safety plan complies with the requirements [1]. Both, safety plan and compliance justification, should be agreed upon at the beginning of the project between the regulatory body and the applicant [13] and used to manage the execution of safety activities during the engineering of safety-critical systems.

### 7.2.2 Safety Standard ISO 26262

ISO 26262 [14] addresses functional safety in automotive. ISO 26262 introduces the notion of Automotive Safety Integrity Level (ASIL), which represents a criterion to specify the item's necessary safety requirements needed to ensure a certain level of confidence. ISO 26262 specifies a safety lifecycle that comprises the entirety of phases from concept through decommissioning of the system. ISO 26262 is structured in several parts that contain clauses. The first three clauses, which are similar in all the parts of the standard, only have an informative nature. Clause 4 is of particular importance since it describes two compliance conditions required along with all the standard: the general requirements for compliance, and the interpretation of tables. The rest of the clauses states the objectives, general information of the clause, inputs for the clause, requirements, and recommendations to be fulfilled, and the work products that are to be generated. Notes are also included, but they have informative character. The requirements and recommendations section describes not only the activities and the tasks required during the engineering process but also the specific conditions required for compliance.

### 7.2.3 SPEM 2.0

SPEM (Software & Systems Process Engineering Metamodel) 2.0 [4] is a standard that describes *Method Content* (knowledge base of reusable elements) and *Processes*. Some elements of SPEM 2.0 are depicted in Table 7.1. A *task definition* is an assignable unit of work which has expected input/output *work products*. When a task is assigned to a process, it is called *Task Use*. *Guidance* provides additional descriptions to method content elements. *Custom Category* is a way to organize elements. A *Delivery Process* is an integrated approach for performing a project. SPEM 2.0 supports variability management, e.g., *Contributes*, which allows extending a base in an additive fashion without altering its existing properties. The open-source tool *EPF (Eclipse Process Framework) Composer* [15], implements UMA (Unified Method Architecture), a metamodel that exhibits a good coverage of SPEM 2.0 concepts. Also, EPF Composer has a proprietary activity diagram which partially generates the execution semantics of a defined process, and permits the importing and exporting libraries with projects (a.k.a. plugins) allowing reusability.

Table 7.1: Subset of Icons Used in SPEM 2.0.

| **Task Definition/Use** | **Work Product** | **Guidance** | **Custom Category** | **Delivery Process** |
|---|---|---|---|---|
| ▷ / ▷ | 📄 | 📁 | 📁 | ⊞ |

### 7.2.4 Safety-oriented Process Line Engineering

Safety-oriented Process Line Engineering (SoPLE) [5] is a methodological approach that permits process engineers to systematize the reuse of process-related information. Two phases conform SoPLE. The first phase is aimed at engineering reusable safety process-related commonalities and variabilities. The second phase is aimed at engineering single safety processes via the selection and composition of the reusable process elements. Currently, SoPLE is supported by the integration of EPF Composer [15], which is used to model the base processes, and Base Variability Resolution (BVR) Tool [16], which allows users to bind the conceptual representation of the variable elements. The integration of EPF Composer and BVR Tool is described in more details in [17].

### 7.2.5 Defeasible Logic

Defeasible logic [18] is a rule-based logic that provides reasoning with incomplete and inconsistent information. A defeasible theory is a knowledge base in defeasible logic, which contains:

1. *Facts:* indisputable statements;
2. *Strict rules:* rules in the classical sense, whenever the premises are indisputable, so is the conclusion;
3. *Defeasible rules:* rules that can be defeated by contrary evidence;
4. *Defeaters:* rules used only to prevent conclusions;
5. *Superiority relation:* a relation among rules used to define priorities.

Formally,

$$r : A(r) \hookrightarrow C(r),$$

where a rule r consists of an antecedent A, the consequence of the rule C, and the rule $\hookrightarrow = \{\rightarrow (strict), \Rightarrow (defeasible), or \rightsquigarrow (defeater)\}$. A defeasible proof requires that we:

1. Put forward a supported rule for the conclusion we want to prove;
2. consider all possible reasons against the desired conclusion
3. rebut all counterarguments, by either showing that some premises of the counterargument do not hold, or another argument defeats the argument.

### 7.2.6 Formal Contract Logic

Formal Contract Logic (FCL) [7] is a language based on defeasible logic (described in Section 7.2.5) and deontic logic of violations [19]. An FCL rule is represented as follows:

$$r : a_1, ..., a_n \Rightarrow c,$$

where $a_1, ..., a_n$ are the conditions of the applicability of the norm, and $c$ is the normative effect. Normative effects can be of two types. One type describes the environment in which the process will be executed (constitutive rules). The second type triggers deontic effects, such as *Obligations*, which are mandatory situations, *Prohibitions*, which are forbidden situations and *Permissions*, which are allowed situations. In addition, if something is permitted the obligation to the contrary does not hold. There are different types of normative effects, as presented in Table 7.2. An obligation that has to be obeyed during all instants of the process is called *Maintenance*, while obligations that only require to be fulfilled once are called *Achievement*. An achievement obligation is *Preemptive* if it could be fulfilled even before the obligation is in force. Otherwise, it is *Non-Preemptive*. If the obligation persists after being violated, it is considered *Perdurant*. Otherwise, it is a *Non-Perdurant*.

Table 7.2: FCL Rule Notations.

| **Notation** | **Description** |
|---|---|
| [P]P | P is permitted |
| [OM]P | There is a maintenance obligation for P |
| [OAPP]P | There is an achievement, preemptive, and non-perdurant obligation for P |
| [OANPP]P | There is an achievement, non-preemptive and perdurant obligation for P |
| [OAPNP]P | There is an achievement, preemptive and non-perdurant obligation for P |
| [OANPNP]P | There is an achievement, non-preemptive and non-perdurant obligation for P |

### 7.2.7 Compliance by Design Approach

Compliance by design [20] is an approach in which compliance of a process with a set of rules is verified during process design. For applying this approach, *process traces*, which are sequence of tasks in which a process can be executed, should be defined. Moreover, semantic annotations, which are functions that describe the environment in which a process operates, are required. In particular, two types of functions are necessary. The function *Ann(n,t,i)*, which returns the state of a *trace (n)* obtained after a *task (t)*, in the *step (i)*. The function *Force(n,t,i)* = $\{o\}$ associates to each *task (t)* in a *trace (n)*, in the *step (i)* a set of *obligations (o)*.

### 7.2.8 Regorous

Regorous [9] is a compliance checker, which assists process engineers during the design of the processes with mapping regulations to specific process and process steps, so that processes can be designed or re-designed in a compliant way. Regorous is the result of the implementation of *the compliance by design approach*, recalled in Section 7.2.7. To check compliance of an annotated process model against a relevant normative system, the procedure executed is the following:

1. Generate an execution trace of the process.

2. Traverse the trace. For each task in the trace, cumulate the effects of the task. Use the set of cumulated effects to determine which obligations enter into force at the current task. Add the obligations obtained from the previous step to the set of obligations carried over the previous task. Finally, determine which obligations have been fulfilled, violated or a pending, and if there are violated obligations, check whether they have been compensated.

3. Repeat for all traces.

An obligation can be terminated if the deadline is reached, the obligation has been fulfilled, or if the obligation has been violated and it is not perdurant. A process is fully compliant if all its traces are compliant (all obligations have been fulfilled, or if violated, they have been compensated). A process is partially compliant if there is at least one trace that is compliant.

### 7.2.9 Specification Patterns

The specification patterns, formulated by Dwyer et al.'s [21], are *"generalized descriptions of commonly occurring requirements on the permissible state sequence of a finite state model of a system."* A selected set of Dwyer et al.'s patterns is presented in Table 7.3. The reader may refer to [22] to see the complete set of patterns with their entire descriptions. Each pattern has a *scope*, which is the extent of the program execution over which the pattern must hold.

The types of scope that we consider in this paper are: *global*, which represent the entire program execution, *before*, which includes the execution up to a given state, and *after* which includes the execution after a given state.

Table 7.3: Dwyer's Specification Patterns.

| **Name** | **Description** |
|---|---|
| **Absence** | A given state P does not occur within a scope |
| **Existence** | A given state P must occur within a scope |
| **Universality** | A given state P must occur throughout a scope |
| **Precedence** | A state P must always be preceded by a state Q within a scope |
| **Response** | A state P must always be followed by a state Q within a scope |

## 7.3 Related work

Compliance to standards is a matter of decision-making. Supporting that decision-making process requires the provision of the right level of abstraction of the boundaries prescribed by the standard in a way that the conditions for compliance can be evaluated. In [23], the authors propose a semi-automatic compliance process to support the definition of a formal specification of software requirements. In [24], the authors present an approach to reason about the correctness of the process structure, which is based on the combination of CTN (Composition Tree Notations) [25] and Description Logic (DL). Similarly, in [26] and [27] approaches for enabling the definition process capability levels, according to ISO/IEC 15504 and CMMI (Capability Maturity Model Integration) v1.3 [28] are presented. In [29], the author presents a formalization of data usage policies in a fragment of OWL (Web Ontology Language) [30]. All the previous approaches, consider the use of DL to reason about the compliance of the process structure. One of the problems of DL, as presented by [31], is its relative expressiveness, which makes more difficult the modeling of certain concepts. Besides, the previous approaches only consider the analysis of the process structure. Instead, our approach considers the use of a mechanism that permits the recording of the information that represents the effects caused by the tasks, which is called compliance effects annotation. This mechanism is not only useful for checking the compliance of a process structure, but also its behavior. Other difference, we have included in our approach, is the use of a SPEM 2.0-compatible software process modeling language.

SPEM 2.0 community is interested in addressing checking and monitoring capabilities. In [32], the authors propose a framework that uses LTL (Linear Temporal Logics) on top of SPEM 2.0 for adding the ability to monitor and control a real process according to its defined process model. The methodology provided in [32] is also used in [33], to ensure process compliance during execution time. The work presented in [34] aims at facilitating the checking of constraints that can be defined as part of a specific process model by using SWRL (Semantic Web Rule Language) [35]. The approach in [34] is also used in [36], to permit that the description of IT (Information Technology) process models are checked with the constraints provided by the business perspective. An approach for representing SPEM 2.0 process models in DL, to provide process analysis such as reasoning and consistency checks, is presented in [37]. The generation of the tailored process, in the automotive domain, is done by using ontologies created in OWL, which outputs are transformed into SPEM 2.0 process models [38]. Our approach combines the capabilities for modeling standard's requirements, plus customization of preexisting modeling concepts to generate a centralized compliance-related knowledge base. Besides, we add a layer of confidence by considering the use of methods that allow us to derive proofs of compliance. However, we do not use semantic web methods for deriving our proofs since they are computational methods that deal with ontologies and rules, whose combination could be undecidable [39].

Approaches for compliance checking have been widely studied in the business context. For instance, in [40], the authors propose to capture high-level policies with a compliance metamodel called REALM (Regulations Expressed As Logical Models), to support the formalization of compliance requirements in Real-time Temporal Object Logic [41]. In [42], an object life cycle approach is used to generate a set of actions for the generation of process models, in which the order of the model of the process actions is determined and then combined into process fragments that are connected to decision and merge nodes. In [43], the authors propose a Service Oriented Architecture-based compliance governance, called COMPAS, to define compliant process fragments. In [44], authors propose a compliance checking method for business process models, in which norms are expected to be modeled in BPSL (Business Property Specification Language) and then formalized in LTL (Linear Temporal Logic). In [45], the authors propose a solution for ensuring compliance by using a formal language for specifying a subset of business rules and the necessary mechanisms for parsing the constraints and ensuring compliance of process management systems. There are also compliance checking frameworks that combine the modeling capabilities provided by BPMN (Business Process

Model and Notations) [46] and Temporal Logics for the modeling of regulations, e.g., [47], and [48]. In our approach, we are using a similar methodology that those previously presented. However, we do not use Temporal Logics for creating the formal specification of the standards requirements since such logic is not able to provide conceptually sound representations of the regulatory requirements governing a process [49].

## 7.4 Proposed Research

In this section, we present the research methodology used. Then, we present the motivation and the goals of our intended research.

### 7.4.1 Research Methodology

Our research methodology, which was inspired in the research methodology for information systems research proposed in [50], consists of three main stages.

1. **Research Initiation:** Defines the overall research. In this stage, we *identify and motivate the problem* and *define the main goal*. The resources required in this stage include the knowledge of state of the art and the state of the practice. A problem formulation, which describes the main problem and formulates a motivation about the need to solve it, and an overall research goal, which is designed to address the main problem, are produced.

2. **Research Development:** Supports the achievement of the main goal. Initially, we *identify a sub-problem* and *define a subgoal*, which should describe a specific problem and justify the value of a solution. Later, we *design and develop* a solution artifact, i.e., constructs, models, methods, or instantiations, new properties of technical, social, and/or informational resources, that solves the specific problem. Within the artifact, its desired functionality, architecture and actual development have to be described. Then, the *demonstration*, which could involve the use of the artifact in experimentation, case study, proof or other appropriate activity, is carried out. These four steps are repeated for every research goal. Every iteration may finish in a global activity called *communication*, in which the problem and its importance, the artifact, its utility and novelty, the rigour of its design, and its effectiveness is communicated to the research community and practitioners.

3. **Research Finalization:** Compile the project. We *integrate* the solutions of the subgoals and *validate* the overall research contribution, namely, we observe how well the artifact produced solves the overall problem.

### 7.4.2 Motivation

Companies aiming at complying with process-based safety standards should adapt their practices, and provide evidence that demonstrates the fulfillment of the requirements. In particular, compliance checking of process plans against safety standards is a mechanism that can be used to demonstrate the adherence of the safety plan to the standard requirements regarding processes. The result of this demonstration, which can take the form of a compliance checking report, can support the provision of the compliance justification, which is required during the interaction with the certification bodies in the planning phases. Compliance checking may involve several steps. Initially, a process engineer should know and understand the range of the criteria provided in the standard's requirements. Then, a careful examination of the process description and the interactions between process elements should be done to identify whether the elements involved in the planning of the process conforms to the standards prescriptions. Fulfilled requirements can be considered checkable for compliance. However, the checking mark is not enough. It is expected that a compliance checking report informs not only the fulfillment of the requirements but also what is the evidence collected that demonstrates that the process satisfies the requirements. Thus, information regarding the identified elements is also considered evidence that demonstrates compliance and should be documented within the checking mark, to produce a proper compliance checking report. The process engineer can use the compliance checking report to identify areas in the process that are uncompliant and, if needed, improve the process. The improvement can be made by modifying or deleting existing process elements, or by adding new process elements, according to the compliance checking report recommendations. However, improving some process elements may affect the behavior of others, resulting in new uncompliant situations. Therefore, complete re-checking may be required. Once fully compliance is reached, the compliance checking report itself can be used as the evidence required for the certification bodies to justify process compliance. However, manually performing all the steps described before can be time-consuming and prone-to-error since standards are large documents with hundreds of process-related requirements. Besides, a company can have many safety-critical-related processes to be checked. Thus, support for automated compliance checking may be of interest to facilitate the production of compliance checking reports required during planning phases.

### 7.4.3 Research Goals

Given the research motivation presented in Section 7.4.2, we formulate our overall research goal as follows:

> *Provide an approach that facilitates compliance checking of the processes used to engineer safety-critical systems against the standards mandated (or recommended) in the safety-critical context.*

In order to address the overall research goal, we define concrete subgoals that address specific challenges. The subgoals are described as follows:

1. Elicit the requirements to be met to support the automation of process-based compliance checking in the safety-critical context.
2. Identify methodologies that contribute to automate the compliance checking of planned process against process-based safety standards.
3. Facilitate the creation of formal specifications of the process-based requirements prescribed by safety standards.
4. Analyse existing methodological approaches that could be used for increasing efficiency in process compliance.

## 7.5 Preliminary Results

Hitherto we have achieved five technical contributions, which we describe in this section.

### 7.5.1 Conditions for Checking Compliance

As presented in Section 7.4.2, automatizing the compliance checking is considered useful to facilitate the procurement of the compliance justification report required during the planning phases. For facilitating this task, we have selected the *compliance by design approach* (recalled in Section 7.2.7). As the definition recalls, for performing compliance by design we need to model two components: the model that describes the norms, which will be propagated into the model that describes the process. This propagation is possible by a mechanism called compliance effects annotation. This mechanism consists of recording the information that represents the effects caused by the tasks that are aligned with the requirements influences. Giving this appreciation, we could assume that the compliance effects unlike other effects caused by the process tasks, corresponds to the permissible states allowed by the standard's requirements. The permissible states trigger other (possible) permissible states that describe a model with compliant states. When permissible states are possible to be annotated into a process model, the requirements that represent are considered to be checkable for compliance, because they can occur in the process model. Thus, we can assign a boolean function to the requirements that is true when it occurs and false otherwise. Based on the previous reasoning, we have defined the conditions for automatically checking compliance in the safety-critical context as follows:

> *Automatic compliance checking of a safety plan involves the annotation of the process elements defined to manage and guide the execution of safety activities with compliance effects, which correspond to the permissible states provided by the standard's requirements, to describe a model with standard-compliant states.*

These conditions require the association of three components as depicted in Figure 7.1.

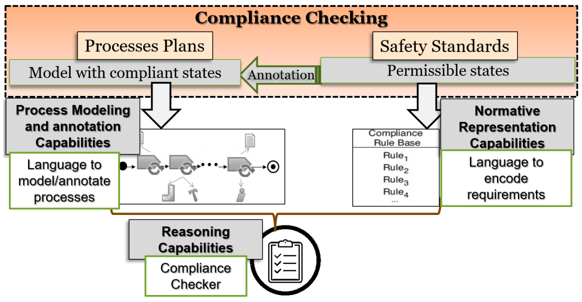

Compliance Checking
Processes Plans
Safety Standards
Model with compliant states
Annotation
Permissible states
Process Modeling and annotation Capabilities
Language to model/annotate processes
Compliance Rule Base
Rule$_1$
Rule$_2$
Rule$_3$
Rule$_4$
...
Normative Representation Capabilities
Language to encode requirements
Reasoning Capabilities
Compliance Checker

Figure 7.1: Required Components for Automating Compliance Checking.

The first component is a language to model processes that provides not only the process modeling capabilities but also the annotation capabilities that allows the enrichment of process tasks with compliance effects. The second component is a language to encode requirements that provides normative representation capabilities, to permit the interpretation of the standard's requirements in an adequate machine-readable form, and the generation of the permissible states that will be used as the compliance effects required for the annotation process. Finally, the third component is a compliance checker that provides the reasoning capabilities necessary to conclude whether the annotated process model corresponds to a model with compliant states. This contribution is presented in [51].

## 7.5.2 Automated Compliance Checking Vision

Our compliance checking vision (see Figure 7.2), which has the potential to automatize the compliance checking in the safety-critical context, considers the combination of the tool-supported methodological approaches that provide the required capabilities described in Figure 7.1. In particular, the vision includes the provision of a compliance rule base in FCL (recalled in Section 7.2.6), which provides the normative representation capabilities required for annotating the process models and check compliance. Moreover, we include EPF Composer, which provides the SPEM 2.0-like process modeling and annotation capabilities (as recalled in Section 7.2.3), as well as a basic platform for FCL rule edition. Finally, we include Regorous (recalled in Section 7.2.8), which provides reasoning capabilities with FCL rules required for compliance checking. The vision also includes two main roles, i.e., a process engineer, who should support the interpretation of the standard's requirements, model, annotate the process, and analyze the compliance report, and the FCL expert, who should interpret standard's requirements and formalize them in FCL.

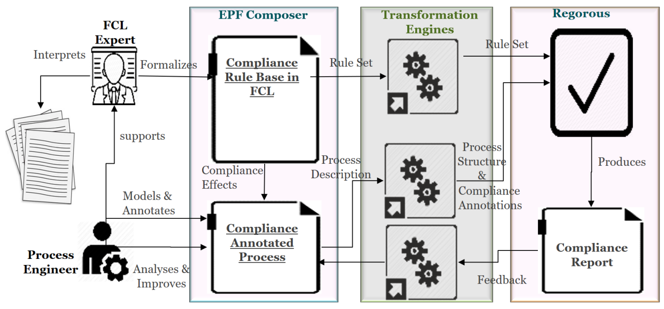


Figure 7.2: Automated Compliance Checking Vision.

The tool-support previously described is conceived in three steps. First, we consider the definition of the mechanisms to annotate process models, to support the process engineers. Then, we consider the definition of the facilities required for editing FCL rules to produce the rule set supporting FCL experts. Finally, we created the mechanisms to ensure EPF Composer and Regorous compatibility. These mechanisms consist of a series of transformations that take the models produced by EPF Composer and convert them into the models that Regorous can process. During the production of the transformations, we realize that the tool-support provided by Regorous is not process modeling language agnostic, as the Regorous methodology. In particular, Regorous depends on a specific process modeling language, i.e., BPMN (Business Process Model and Notation) to produce the compliance report. We need to detach the compliance report from the modeling language to be able to backpropagate the compliance results into EPF composer. The result of this discovering is that Regorous has entered a refactoring period, from which we expect to concretize our automated compliance checking vision in the future. This contribution is presented in [52, 53].

### 7.5.3 ISO 26262-related Compliance Patterns Definition

Formalizing safety requirements in FCL is not an easy task, since it requires skills, which cannot be taken for granted. For this issue, patterns could represent a solution. In particular, property specification patterns (as recalled in Section 7.2.9) were created to ease the formalization of systems requirements for finite state system model verification. We follow property specification patterns style, to draw a general definition of safety compliance patterns as follows:

> *Safety compliance patterns describe commonly occurring normative requirements on the permissible state sequence of a finite state model of a process plan.*

With this definition, we developed the mapping between specification patterns and safety compliances patterns. In this mapping, the state of the obligation imposed to an element in the process is considered in a similar way as the presence of a state in a system, and that the scope corresponds to the interval in a process when the obligations formulated by the pattern are in force. For the identification of ISO 26262-related compliance patterns, we have delineated five methodological steps, which are depicted in Figure 7.3, by using SPEM 2.0 elements (recalled in Table 7.1).

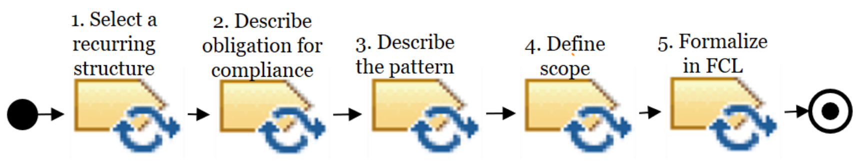

1. Select a recurring structure
2. Describe obligation for compliance
3. Describe the pattern
4. Define scope
5. Formalize in FCL

Figure 7.3: Methodological Steps for Identifying Safety Compliance Patterns.

The first step consists of the selection of a recurring structure in the standard since, as recalled in Section 7.2.2, safety requirements in ISO 26262 have implicit and explicit structures. The second step is the description of the obligation for compliance described in the requirement. The third step is the pattern description, based on similar (or a combination of) behaviors of the property specification patterns described in Section 7.2.9. This description is contextualized to safety compliance, based on the mapping previously done. In this step, we also assign a name for the safety compliance pattern, which reflects the related obligation for compliance. The fourth step is the definition of the scope of the pattern, which we also based on the scopes defined to the property specification patterns. The fifth step is the formalization in FCL. To formalize the pattern, the scope defined for the pattern requires being mapped into the rule notations provided by FCL. Therefore, a *global scope*, which represents the entire process model execution, can be mapped to *maintenance obligation*, which represents that an obligation has to be obeyed during all instants of the process interval. A *before scope*, which includes the execution of the process model up to a given state, can be mapped to the concept of *preemptive obligation*, which represents that an obligation could be fulfilled even before it is in force. An *after scope*, which includes the execution of the process model until a given state, can be mapped to the concept of *non-preemptive obligation*, which represents that an obligation cannot be fulfilled until it is in force. It should be noted that, in safety compliance, it is possible to define exceptions for the rules. Therefore, if the obligation admits an exception, the part of the pattern that corresponds to the exception is described as a permission. The obligation, to which the exception applies, is modeled as *non-perdurant*, since the permission is not a violation of the obligation, and therefore the obligation does not persist after the permission is granted. In this case, the obligation and the permission have contradictory conclusions, but the permission is superior since it represent an exception. This contribution is presented in [54].

### 7.5.4 Methodological Guidelines for Formalizing ISO 26262

To be able to formalize effectively, we consider that doing a pre-processing of ISO 26262 (recalled in Section 7.2.2) was a necessary task. The pre-processing, which is depicted in Figure 7.4, includes three tasks. Initially, we identify the essential normative structures, namely those structures that define the safety process to be adopted for developing the car's safety-critical systems. Then, we identified the repetitive structures of the standard that can be considered Safety Compliance Patterns. With the identified Safety Compliance Patterns, we create templates to consolidate a reusable knowledge base for future formalization jobs. Finally, the knowledge gathered in the pre-processing is used to define a methodological guideline for facilitating the formalization of normative clauses in ISO 26262. This contribution is presented in [55].

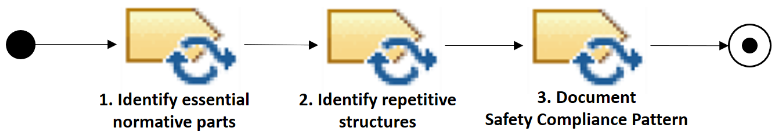

1. Identify essential normative parts
2. Identify repetitive structures
3. Document Safety Compliance Pattern

Figure 7.4: Pre-processing.

From the pre-processing tasks described above, we got an understanding of what to formalize and how we could proceed in the formalization process. The parts to be formalized are those that determine the safety lifecycle, namely, those clauses that start from Clause 5 in every part of the standard ISO 26262. As Figure 7.5 depicts, initially, the given context of the phase, which is described in the safety standard, should be understood. For this, the reading and the analysis of the objectives and the main general information of the clause to be formalized are required. Then, the formalization process initiates with the prerequisites and followed by the title. After, one requirement is selected from the list of Requirements and Recommendations. We suggest that the requirements are selected in the order they are presented and that the rules are named following the requirement numeration to ensure consistency and traceability. During the formalization of the requirements, Safety Compliance Patterns templates could be used to facilitate this task. However, if there are no templates, brainstorming sessions are required. The brainstorming session can be carried out in different ways, but the most relevant is that the group takes one requirement at the time, discuss its importance in the compliance process (e.g., related requirements or permits for tailoring), divide the requirement into smaller sentences that have only one idea, and discuss every sentence. Finally, when all requirements available in Requirements and Recommendations are covered, the work products can be formalized.

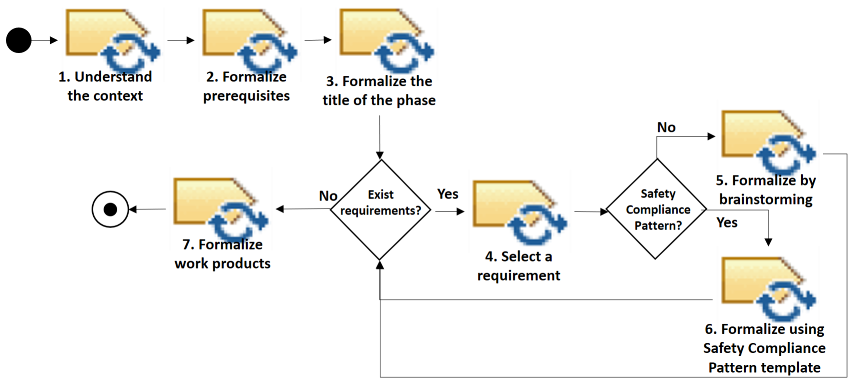

1. Understand the context
2. Formalize prerequisites
3. Formalize the title of the phase
Exist requirements?
No
Yes
4. Select a requirement
Safety Compliance Pattern?
No
Yes
5. Formalize by brainstorming
6. Formalize using Safety Compliance Pattern template
7. Formalize work products

Figure 7.5: Methodological Guidelines.

### 7.5.5 Logic-based Framework for Enabling Reuse of Compliance Proofs

Safety-oriented Process Line Engineering (SoPLE) (recalled in Section 7.2.4), permits process engineers to systematize the reuse of process-related information. However, to argue about or prove compliance, SoPLE is not enough. Therefore, we intend to provide a layer of confidence by offering a logic-based framework that enables formal proofs of compliance. To do that, we build on top of results stemming from the legal compliance and business process-related community. Specifically, we use defeasible logic formalisms (recalled in Section 7.2.5), which permit efficient reasoning with incomplete and inconsistent information, a typical scenario in normative systems. Our approach, which is called *SoPLE&Logic-basedCM*, is depicted in Figure 7.6. As Figure depicts, a process engineer is expected to: 1) Model a SoPL, which includes manually modeling the skeleton of the process sequence; 2) Formalize the standards rules, select the set of rules that overlap, and analyze the compliance of the SoPL commonalities with the overlapping rules; and 3) Analyze the effects of the tasks that contribute to the variabilities in the in the standard-specific process. This contribution is presented in [56, 57].

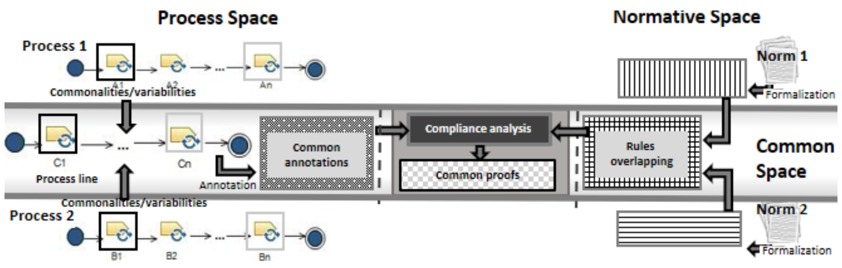


Figure 7.6: SoPLE&Logic-basedCM Framework.

## 7.6 Conclusions and Next Steps

This section present concluding remarks and next steps in the research

### 7.6.1 Conclusions

In this paper, we present a proposal for providing an approach that facilitates automated compliance checking of the process plans against the standards mandated (or recommended) in the safety-critical context. For reaching this goal, we have defined the conditions for automatically checking compliance based on the application of the compliance by design methodology. The definition of these conditions, allow us to proposed an automated compliance checking vision that suits the needs in the safety-critical context. The compliance checking vision combines: 1) process modeling and process annotation capabilities that are required for defining process models checkable for compliance, 2) normative representation capabilities that permit the interpretation of the standard's requirements in an adequate machine-readable form, and the generation of the compliance effects, which are the permissible states required for the annotation process, 3) reasoning capabilities necessary to conclude whether an annotated process model corresponds to the model with the compliant states described in the standard's requirements, and 4) process-line modeling capabilities to systematize the reuse of process-related information. These capabilities are tool-supported. SPEM 2.0 (Software and Systems Process Engineering Metamodel)-like implementation, called EPF (Eclipse Process Framework) provides the modeling and annotation capabilities. FCL (Formal Contract Logic) provides the normative representation capabilities. Regorous provides compliance checking capabilities. In addition, the combination of

EPF Composer and BVR (Base Variability Resolution) provides the process-line modeling capabilities. To support the compliance checking vision, we identified the essential elements required to generate process models checkable for compliance in SPEM 2.0-like process models, and the transformations necessary to automatically generate the models that can be processed by Regorous. Hitherto, our proposed methodology has been evaluated with academic examples that show the potential benefits of its use. Our work represents a novelty in process-based compliance in the safety-critical context, which may contribute to increasing efficiency, via automation, and confidence, via formal checking. It also contributes to cross-fertilize previously isolated communities, i.e., the safety-critical and the legal contexts.

## 7.6.2 Next steps

The results of our thesis can be improved in several directions. Here, we present the suggested areas of research in the future.

- The mapping of regulations to the process tasks, i.e. the annotation of the compliance effects, is done manually, by deducing the effects that can eventually be caused by the tasks in the general compliance status. When the processes are small, this mapping is straightforward. However, when processes are extensive, the mapping may be difficult to achieve. Therefore, methodologies and tactics should be investigated so that the process annotation does not become a burden for the application of the compliance checking approach.

- The automated compliance checking vision, described in this paper, only permits that the analysis of compliance is performed in the sequence of tasks assigned to a process plan. However, a process plan is not only comprised by tasks but also it contains other process elements, such as roles and work products. We aim at extending our approach for permitting that compliance effects annotated to process elements beyond tasks are also included in the analysis of compliance.

- The compliance checking methodology used in our approach is, undoubtedly, process modeling language agnostic. However, the current tool-support lacks agnosticism, i.e., it depends on a specific modeling tool to provide compliance checking results. This characteristic impedes the back-propagation of the compliance results in our selected process modeling language. Therefore, we need to investigate methods and strategies that allow us to represent the compliance checking results in an agnostic way so that we can concretize our compliance checking vision.

- We have limited our analysis of patterns and methodological guidelines to the functional safety standard ISO 26262. This restriction may also limit the applicability of our approach. To expand our horizon, we need to generalize the use of patterns and methodological guidelines so that we can incorporate a wide range of standards. Therefore, comparative studies between standards and definition of generalized patterns, as well as standard-specific patterns could be investigated.

- The reuse of proofs of compliance may increase efficiency and confidence in compliance checking. Thus, we aim at studying in deep the conditions that are required for compositionality of proofs of compliance. We also need to provide metrics for measuring increased confidence and increased efficiency.

- Our work has only be evaluated with academic examples. Therefore, we require to further validating the approach with more complex cases, i.e., industrial cases.

- We need to better situate our work in the context of the state of the art. Therefore, an extended and systematic literature review will be performed.

- To augment the impact for our results, we plan to integrate our automated compliance checking approach to the platform created by the European project AMASS (Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems) [58].

# Bibliography

[1] B. Gallina, F. Ul Muram, and J. Castellanos Ardila, "Compliance of Agilized (Software) Development Processes with Safety Standards: a Vision," in *4th international workshop on Agile Development of Safety-Critical Software (ASCS)*, 2018.

[2] J. Jiménez, J. Amelio, M. Merodio, and L. Sanz, "Computer Standards & Interfaces Checklists for compliance to DO-178C and DO-278A standards," *Computer Standards & Interfaces*, vol. 52, pp. 41–50, 2017.

[3] A. Fuggetta and E. Di Nitto, "Software process," in *Future of Software Engineering*, pp. 1–12, 2014.

[4] O. M. Group, "*Software & Systems Process Engineering Meta-Model Specification*. Version 2.0.," 2008.

[5] B. Gallina, I. Sljivo, and O. Jaradat, "Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification," in *35th Annual IEEE Software Engineering Workshop (SEW)*, pp. 148–157, 2012.

[6] M. Hashmi, G. Governatori, and M. Wynn, "Business process data compliance," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pp. 32–46, 2012.

[7] G. Governatori, "Representing business contracts in RuleML," *International Journal of Cooperative Information Systems*, vol. 14, no. 02n03, pp. 181–216, 2005.

[8] D. Nute, "Defeasible Logic," in *International Conference on Applications of Prolog*, pp. 151–169, Springer, 2001.

[9] G. Governatori, “The Regorous Approach to Process Compliance,” in *IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pp. 33–40, IEEE, 2015.

[10] S. Sadiq, G. Governatori, and K. Namiri, “Modeling Control Objectives for Business Process Compliance,” in *International Conference on Business Process Management*, pp. 149–164, 2007.

[11] S. Vilkomir, J. Bowen, and A. Ghose, “Formalization and assessment of regulatory requirements for safety-critical software,” *Innovations in Systems and Software Engineering*, vol. 2, no. 3-4, pp. 165–178, 2006.

[12] IEC, “Functional safety. Essential to overall safety,” 2015.

[13] Bel V, BfS, CSN, ISTec, ONR, SSM, and STUK, “Licensing of safety critical software for nuclear reactors. Common position of seven European nuclear regulators and authorised technical support organisations,” tech. rep., 2010.

[14] International Organization for Standardization (ISO) 26262:2018, “*Road vehicles – Functional safety*,” 2018.

[15] Eclipse Foundation, “Eclipse Composer Framework.” https://www.eclipse.org/epf/.

[16] Ø. Haugen and O. Øgård, “BVR – Better Variability Results,” in *International Conference on System Analysis and Modeling*, pp. 1–15, 2014.

[17] M. Javed and B. Gallina, “Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool,” in *22nd International Systems and Software Product Line Conference*, pp. 23–28, ACM, 2018.

[18] G. Antoniou, D. Billington, G. Governatori, and M. J. Maher, “Representation Results for Defeasible Logic,” *ACM Transactions on Computational Logic*, no. 2, pp. 255–287, 2000.

[19] G. Governatori and A. Rotolo, “Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligations,” *Australasian Journal of Logic*, vol. 4, no. 4, pp. 193–215, 2006.

[20] R. Lu, S. Sadiq, and G. Governatori, “Compliance Aware Business Process Design,” in *International Conference on Business Process Management*, pp. 120–131, 2007.

[21] M. Dwyer, G. Avrunin, and J. Corbett, “Property Specification for Finite-State Verification,” in *2nd Workshop on Formal Methods in Software Practice*, pp. 7–15, 1998.

[22] Santos Laboratory, “Specification Patterns .” http://patterns.projects.cs.ksu.edu/.

[23] P. Engiel, J. Sampaio do Prado Leite, and J. Mylopoulos, “A Tool-Supported Compliance Process for Software Systems,” in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pp. 66–76, IEEE, 2017.

[24] E. Kabaale, L. Wen, Z. Wang, and T. Rout, “Representing Software Process in Description Logics: An Ontology Approach for Software Process Reasoning and Verification,” in *Software Process Improvement and Capability Determination*, pp. 362–376, Springer, 2016.

[25] L. Wen, D. Tuffley, and T. Rout, “Using Composition Trees to Model and Compare,” in *International Conference on Software Process Improvement and Capability Determination*, no. March 2014, pp. 1–15, Springer, 2011.

[26] D. Proença and J. Borbinha, “Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3 – Enabling automatic inference of maturity and capability levels,” *Computer Standards and Interfaces*, 2018.

[27] G. Soydan and M. Kokar, “A Partial Formalization of the CMMI-DEV — A Capability Maturity Model for Development,” *Journal of Software Engineering and Applications*, vol. 5, no. 10, pp. 777–788, 2012.

[28] SEI Carnegie Mellon, “CMMI® for Development, Version 1.3 CMMI-ACQ, V1.3,” Tech. Rep. November, Software Engineering Institute, Carnegie Mellon, 2010.

[29] P. Bonatti, “Fast Compliance Checking in an OWL2 Fragment,” in *27th International Joint Conferences on Artificial Intelligence Organization (IJCAI)*, pp. 1746–1752, 2018.

[30] OWL Working Group, “Web Ontology Language (OWL),” https://www.w3.org/OWL/.

[31] A. Borgida, “On the relative expressiveness of description logics and predicate logics,” *Artificial intelligence*, vol. 82, no. 1-2, pp. 353–367, 1996.

[32] R. Bendraou, B. Combemale, X. Crégut, and M. Gervais, "Definition of an executable SPEM 2.0," in *14th Asia-Pacific Software Engineering Conference (ASPEC)*, pp. 390–397, 2007.

[33] F. Golra, F. Dagnat, R. Bendraou, and A. Beugnard, "Continuous Process Compliance Using Model Driven Engineering," in *International Conference on Model and Data Engineering*, pp. 42–56, Springer, 2017.

[34] D. Rodríguez, E. Garcia, S. Sanchez, and C. Rodríguez-Solano Nuzzi, "Defining software process model constraints with rules using OWL and SWRL," *International Journal of Software Engineering and Knowledge Engineering*, vol. 20, no. 04, pp. 533–548, 2010.

[35] I. Horrocks, P. Patel-schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.

[36] M. Valiente, E. García-Barriocanal, and M. Sicilia, "Applying Ontology-Based Models for Supporting Integrated Software Development and IT Service," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 1, pp. 61–74, 2012.

[37] S. Wang, L. Jin, and C. Jin, "Represent S oft w are Process Engineering Metamode l in Description Logic," *World Academy of Science, Engineering and Technology*, vol. 11, pp. 109–113, 2006.

[38] H. Jost, S. Köhler, and F. Köster, "Towards a Safer Development of Driver Assistance Systems by Applying Requirements-Based Methods," in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1144–1149, IEEE, 2011.

[39] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC,, 2009.

[40] C. Giblin, S. Müller, and B. Pfitzmann, "From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation," tech. rep., IBM Research Laboratory, Zurich, 2006.

[41] C. Giblin, A. Liu, S. Müller, B. Pfitzmann, and X. Zhou, "Regulations Expressed As Logical Models (REALM)," tech. rep., IBM China Research Lab, 2005.

[42] J. Küster, K. Ryndina, and H. Gall, “Generation of Business Process Models for Object Life Cycle Compliance,” in *International Conference on Business Process Management*, pp. 165–181, Springer, 2007.

[43] F. Daniel, F. Casati, E. Mulo, U. Zdun, S. Strauch, D. Schumm, F. Leymann, S. Sebahi, F. De Marchi, and M. S. Hacid, “Business compliance governance in service-oriented architectures,” in *International Conference on Advanced Information Networking and Applications (AINA)*, pp. 113–120, 2009.

[44] A. Awad, G. Decker, and M. Weske, “Efficient Compliance Checking Using BPMN-Q and Temporal Logic,” *International Conference on Business Process Management*, pp. 326–341, 2008.

[45] L. Ly, K. Göser, S. Rinderle-ma, and P. Dadam, “Compliance of Semantic Constraints – A Requirements Analysis for Process Management Systems,” in *1st Int’ernational Workshop on Governance, Risk and Compliance - Applications in Information Systems (GRCIS)*, 2008.

[46] Object Management Group, “Business Process Model and Notation Version 2.0,” 2011.

[47] D. Schumm, O. Turetken, N. Kokash, A. Elgammal, F. Leymann, and W. van den Heuvel, “Business Process Compliance through Reusable Units of Compliant Processes,” in *International Conference on Web Engineering*, pp. 325–337, 2010.

[48] M. El Kharbili, “Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation,” in *8th Asia-Pacific Conference on Conceptual Modelling.*, pp. 23–32, 2012.

[49] G. Governatori and M. Hashmi, “No Time for Compliance,” *IEEE 19th International Enterprise Distributed Object Computing Workshop, (EDOCW)*, pp. 9–18, 2015.

[50] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[51] J. Castellanos Ardila, *Facilitating Compliance Checking of Processes against Safety Standards*. Licentiate thesis, Mälardalen University, 2019.

[52] J. P. Castellanos Ardila, B. Gallina, and F. UL Muram, “Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance,” in *18th International SPICE Conference*, 2018.

[53] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, “Enabling Compliance Checking against Safety Standards from SPEM 2.0 Process Models,” in *Euromicro Conference on Software Engineering and Advanced Applications*, 2018.

[54] J. P. Castellanos Ardila and B. Gallina, “Formal Contract Logic Based Patterns for Facilitating Compliance Checking against ISO 26262,” in *1st Workshop on Technologies for Regulatory Compliance (TeReCom)*, pp. 65–72, 2017.

[55] J. Castellanos Ardila, B. Gallina, and G. Governatori, “Lessons Learned while formalizing ISO 26262 for Compliance Checking,” in *2nd Workshop on Technologies for Regulatory Compliance (TeReCom)*, pp. 1–12, CEUR-Workshop Proceedings, 2018.

[56] J. Castellanos Ardila and B. Gallina, “Towards increased efficiency and confidence in process compliance,” in *The 24th EuroAsiaSPI Conference*, vol. 748, 2017.

[57] J. P. Castellanos Ardila and B. Gallina, “Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards,” in *The 7th IEEE International Workshop on Software Certification (WoSoCer)*, 2017.

[58] AMASS., “Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems,” 2016. http://www.amass-ecsel.eu/.

# Chapter 8

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

Julieth Patricia Castellanos Ardila, Barbara Gallina.
Proceedings of the European Systems, Software & Service Process Improvement & Innovation. EuroAsiaSPI 2020. Communications in Computer and Information Science, vol 1251. Springer, Cham. 2020.

## 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.

## 8.1 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 [1]. However, this task becomes difficult due to software process diversity, i.e., the simultaneous use of multiple reference models within a single project [2]. To tackle this situation, organizations produce generic software process baselines. In the analysis of these baselines, gaps to best practices could be discovered [3], 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 [2]. 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 [5], but also to discover essential improvement aspects. Thus, in previous work [6, 7], 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) [8] 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) [9]. 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 8.2. We discuss the separation of concerns within the regulatory space in Section 8.3. We illustrate the defined concerns in Section 8.4. We discuss our findings in Section 8.5. We present related work in Section 8.6. Finally, we conclude the work and outline future work in Section 8.7.

## 8.2 Background

This section presents essential background required in this paper.

### 8.2.1 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 [10]. In particular, process reference models describe a set of tasks to be performed during the development of safety-critical systems. For example, ISO 26262 [11], 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 [12] describes a set of objectives that implicitly define a lifecycle model. ECSS-E-ST-40C [13], 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 [14], 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.

### 8.2.2 CENELEC EN 50128

CENELEC EN 50128 [15], 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 8.2.1. In Table 8.1, we recall a set of requirements that apply to the Architecture and Design Phase.

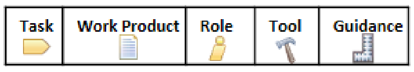
Table 8.1: CENELEC EN 50128-Architecture and Design Phase

| **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) |

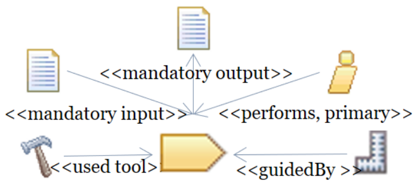
CENELEC EN 50128 also refers to quality management and continuous improvement of the systems within the organizations. Companies may have quality assurance mechanisms that conform to different frameworks such as Software Process Improvement and Capability Determination (SPICE), also known as ISO/IEC 15504. In particular, part 5 [16] 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.

### 8.2.3 Software Processes and SPEM 2.0

A software process [17] 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 [18]. In particular, SPEM 2.0 (Systems and Software Process Engineering Metamodel) [9] 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 Figure 8.1a. Such elements are related to each other, as presented in Figure 8.1b. EPF (Eclipse Process Framework) Composer [19], which was recently migrated from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3. [20], provides the environment for modeling SPEM 2.0-like process models.



(a) SPEM 2.0 Elements



(b) Elements Relationships

Figure 8.1: Content Elements Definitions in SPEM 2.0.

### 8.2.4 Formal Contract Logic

Formal Contract Logic (FCL) [8] 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.

### 8.2.5 Automatic Compliance Checking Method

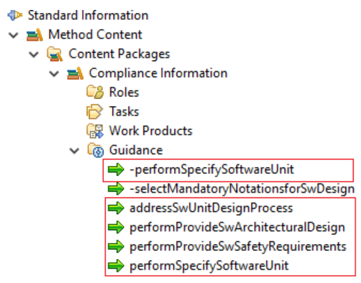
Our method for automated compliance checking of processes plans [6], requires users to model processes with SPEM 2.0 (recalled in Section 8.2.3) and formalize standards requirements with FCL (recalled in Section 8.2.4). Rules in FCL are composed of propositions that correspond to the properties described 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 [6]. 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 8.1.

$$\begin{aligned}
r2.1: addressSwUnitDesignProcess \Rightarrow [O] - performSpecifySwUnit \\
r2.2: performProvideSwArchitecturalDesign, \\
performProvideSwSafetyRequirements \Rightarrow [P] performSpecifySwUnit \\
r2.2 > r2.1
\end{aligned} \tag{8.1}$$

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 8.1 contains five propositions, namely addressSWUnitDesignProcess, -performSpecifySwUnit, performProvideSwArchitecturalDesign, performProvideSwSafetyRequirements and performSpecifySwUnit, which are presented in Figure 8.2a. 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 Figure 8.2b). The reader can discover more details about the previous modeling in [6].

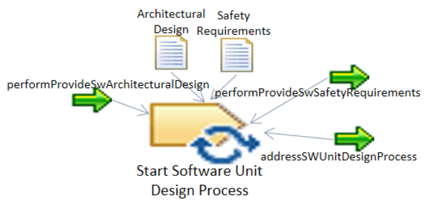
### 8.2.6 Separation of Concerns: Divide-and-conquer Strategy

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 [21]. Particularly, divide-and-conquer is seen in the principle of separation of concerns [22], which refers to the ability to separate and organize only those parts (or properties) of a system that are relevant to a particular



```
Standard Information
Method Content
  Content Packages
    Compliance Information
      Roles
      Tasks
      Work Products
      Guidance
        -performSpecifySoftwareUnit
        -selectMandatoryNotationsforSwDesign
        addressSwUnitDesignProcess
        performProvideSwArchitecturalDesign
        performProvideSwSafetyRequirements
        performSpecifySoftwareUnit
```

(a) Compliance Effects.


Architectural Design
Safety Requirements
performProvideSwArchitecturalDesign
performProvideSwSafetyRequirements
addressSWUnitDesignProcess
Start Software Unit Design Process

(b) Annotated Task.

Figure 8.2: Method for Automatic Compliance Checking: The Modeling Part.

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.

## 8.3 Separation of Concerns within the Regulatory Space

The relationship between the requirements imposed by safety standards (recalled in Section 8.2.1) and the targeted software processes (recalled in Section 8.2.3) 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 8.2.6) 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 8.2.3), a task is performed by a role, requires inputs and provides outputs, is guided by guidance, and a tool is used (see Figure 8.1b). Thus, the tasks are the central elements, to which the other elements are allocated. Our method for compliance checking (recalled in Section 8.2.5), 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 Figure 8.2b. 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 Figure 8.3). The novelty of the approach is threefold. First, we free 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.

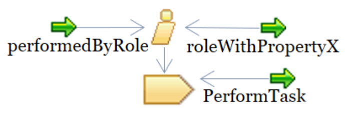


Figure 8.3: Annotated Role.

To facilitate the creation of compliance effects, which later can be used to form the propositions of the rules in FCL (recalled in section 8.2.4), we propose two aspects. The first aspect is the definition of icons, which includes the description of the targeted elements, as presented in Table 8.2. The second aspect is the definition of templates that facilitate the creation of compliance effects, as presented in Table 8.3. Both, icons and templates are based on the concepts described in SPEM 2.0 in Figure 8.1.

Table 8.2: Icons Describing Specific Compliance Effects.

| **Role** | | **Work Product** | | **Guidance** | | **Tool** | | **Task** |
|---|---|---|---|---|---|---|---|---|
| Definition | Property | Definition | Property | Definition | Property | Definition | Property | |
| ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ |

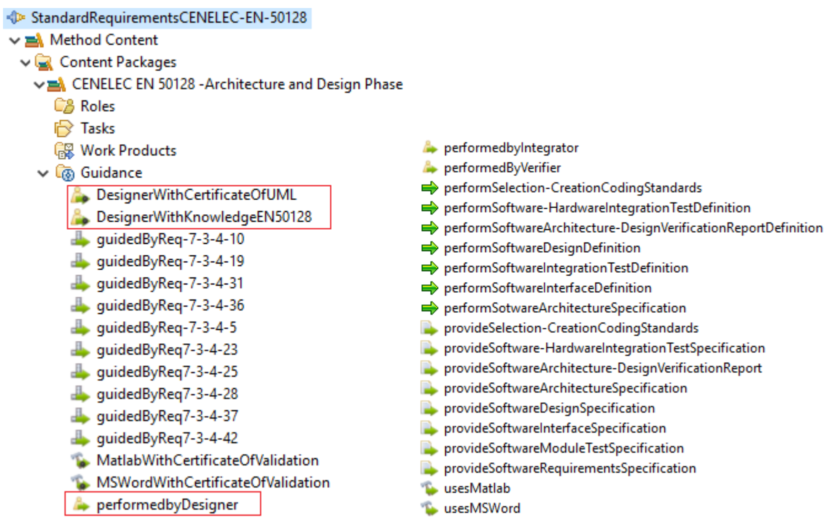
Table 8.3: Compliance Effects Targeting Differentiated Process Elements

| **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*} | |

## 8.4 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 8.2.2). Initially, we need to classify the requirements in terms of the process elements they target. This operation is already presented in Table 8.1. From this division, we can describe the definitional and property-based propositions derived from these requirements by using the propositions templates shown in Table 8.3, and the icons described in Table 8.2. Then, we model them as SPEM 2.0-like elements in the guidance part of EPF Composer. Figure 8.4 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 Figure 8.4. A similar analysis is done with all the requirements.

The next step is to annotate the compliance effects defined in Figure 8.4 into a process plan. For simplicity, we described a process plan taking into account the process elements described in the standard, recalled in Table 8.1 (see Figure 8.5). As we can see in Figure 8.5, 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 Figure 8.5 with the list of compliance effects in Figure 8.4. In this case, the names of the process elements can be found in the



StandardRequirementsCENELEC-EN-50128
- Method Content
  - Content Packages
    - CENELEC EN 50128 -Architecture and Design Phase
      - Roles
      - Tasks
      - Work Products
      - Guidance
        - DesignerWithCertificateOfUML
        - DesignerWithKnowledgeEN50128
        - guidedByReq-7-3-4-10
        - guidedByReq-7-3-4-19
        - guidedByReq-7-3-4-31
        - guidedByReq-7-3-4-36
        - guidedByReq-7-3-4-5
        - guidedByReq7-3-4-23
        - guidedByReq7-3-4-25
        - guidedByReq7-3-4-28
        - guidedByReq7-3-4-37
        - guidedByReq7-3-4-42
        - MatlabWithCertificateOfValidation
        - MSWordWithCertificateOfValidation
        - performedbyDesigner

(a)

- performedbyIntegrator
- performedByVerifier
- performSelection-CreationCodingStandards
- performSoftware-HardwareIntegrationTestDefinition
- performSoftwareArchitecture-DesignVerificationReportDefinition
- performSoftwareDesignDefinition
- performSoftwareIntegrationTestDefinition
- performSoftwareInterfaceDefinition
- performSotwareArchitectureSpecification
- provideSelection-CreationCodingStandards
- provideSoftware-HardwareIntegrationTestSpecification
- provideSoftwareArchitecture-DesignVerificationReport
- provideSoftwareArchitectureSpecification
- provideSoftwareDesignSpecification
- provideSoftwareInterfaceSpecification
- provideSoftwareModuleTestSpecification
- provideSoftwareRequirementsSpecification
- usesMatlab
- usesMSWord

(b)

Figure 8.4: CENELEC EN 50128 - Architecture and Design Phase.

names of the compliance effects since both models are taken from the standard. Thus, the annotation process is straightforward.

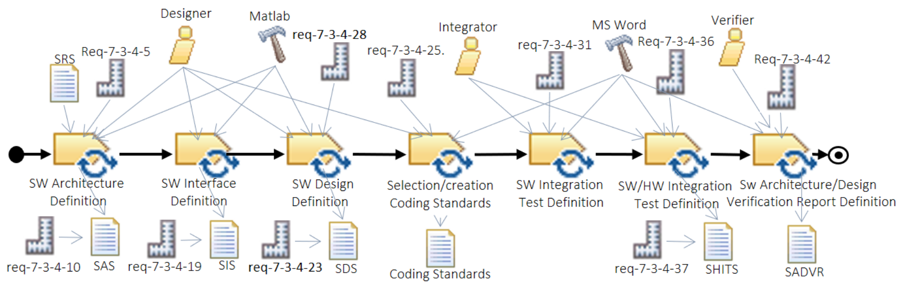
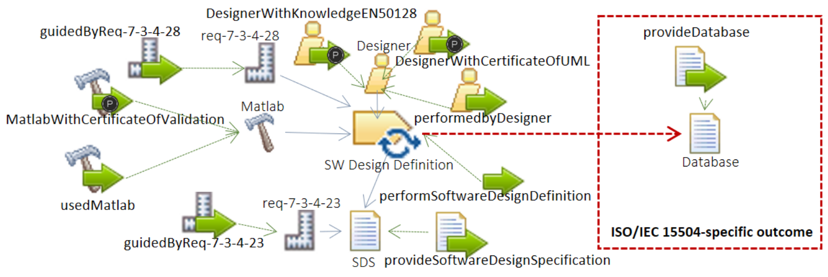


Figure 8.5: Process Plan Targeting the Architecture and Design Phase

Figure 8.6 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 Figure 8.5, 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 Figure 8.4), should be included in the modeling of the process (see the work product Database highlighted with a dotted line in Figure 8.6).



DesignerWithKnowledgeEN50128
guidedByReq-7-3-4-28
req-7-3-4-28
Designer
DesignerWithCertificateOfUML
provideDatabase
MatlabWithCertificateOfValidation
Matlab
performedbyDesigner
SW Design Definition
Database
usedMatlab
req-7-3-4-23
performSoftwareDesignDefinition
guidedByReq-7-3-4-23
SDS
provideSoftwareDesignSpecification
ISO/IEC 15504-specific outcome

Figure 8.6: Task and their Ascribed Elements Annotated with Effects

## 8.5 Discussion

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

### 8.5.1 Compliance-related Process Information

Compliance management can benefit from our proposed modeling strategy. First, the icons describing definitions (see Table 8.2) 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 8.3 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 Figure 8.6). 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 8.3). 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.

### 8.5.2 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 Figure 8.4, 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 Figure 8.6, 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.

### 8.5.3 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 Figure 8.4). 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[23], 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 Figure 8.5). 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.*

## 8.6 Related Work

In this section, we discuss other approaches that have proposed the separation of concerns for facilitating compliance checking with FCL. In [24], 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 [24], describes the compliance effects based on the type of elements that are present in a process. However, contrary to [24], 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 [25], the concept of data tag described in [24] is revisited to create a methodology that permits their extraction from business process logs. Contrary to the work presented in [25], 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.

## 8.7 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.

# Bibliography

[1] M. Yilmaz and R. O'Connor, "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.

[2] N. Ramasubbu, A. Bharadwaj, and G. K. Tayi, "Software Process Diversity: Conceptualization, Measurement, and Analysis of impact on Project Performance," *Management Information Systems*, vol. 39, no. 4, pp. 787–807, 2015.

[3] M. Eckey, C. Greiner, and T. Peisl, "Why Do Organizations Focus on Assessments Instead of Their Process-Improvement Objectives?," in *European Conference on Software Process Improvement*, pp. 392–401, 2019.

[4] C. Crabtree, C. Seaman, and A. Norcio, "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] B. Gallina, F. Ul Muram, and J. Castellanos Ardila, "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.

[6] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, "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.

[7] J. P. Castellanos Ardila, B. Gallina, and F. UL Muram, "Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance," in *18th International SPICE Conference*, 2018.

[8] G. Governatori, "Representing Business Contracts in RuleML," *International Journal of Cooperative Information Systems.*, pp. 181–216, 2005.

[9] OMG, "Software & Systems Process Engineering Meta-Model Specification. Version 2.0.," 2008.

[10] A. Ruiz, G. Juez, H. Espinoza, J. L. de la Vara, and X. Larrucea, "Reuse of safety certification artefacts across standards and domains: A systematic approach," *Reliability Engineering & System Safety*, vol. 158, pp. 153–171, 2017.

[11] ISO/TC 22/SC 32, "*ISO 26262 - Road vehicles – Functional safety*," 2018.

[12] Radio Technical Commission for Aeronautics (RTCA) & European Organisation for Civil Aviation Equipment (EUROCAE) RTCA, "*RTCA/DO-178C - Software Considerations in Airborne Systems and Equipment Certification.*," 2011.

[13] ESA, "*ECSS-E-ST-40C – Space Engineering Software*," 2009. https://ecss.nl/standard/ecss-e-st-40c-software-general-requirements/.

[14] Radio Technical Commission for Aeronautics, "*RTCA/DO-330-Software Tool Qualification Considerations*," 2012.

[15] European Committee for Electrotechnical Standardization – CENELEC - EN 50128, "*Railway applications – Communication, signaling and processing systems Software for railway control and protection systems*," 2011.

[16] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "*Information Technology – Process assessment - An Exemplar Software Life Cycle Process Assessment Model*," 2012.

[17] A. Fuggetta, "Software Process Patterns: A Roadmap," in *International Conference on Software Engineering,*, pp. 25–34, 2000.

[18] B. Gallina, K. Pitchai, and K. Lundqvist, "S-TunExSPEM: Towards an Extension of SPEM 2.0 to Model and Exchange Tunable Safety-oriented Processes," *Software Engineering Research, Management and Applications*, vol. 496, pp. 215–230, 2014.

[19] Eclipse, "Eclipse Process Framework (EPF) Composer.," 2018. https://www.eclipse.org/epf/.

[20] M. Javed and B. Gallina, "Get EPF Composer back to the future: a trip from Galileo to Photon after 11 years," in *EclipseCon*, 2018.

[21] D. Smith, "The Design of Divide and Conquer Algorithms," *Science of Computer Programming*, vol. 5, pp. 37–58, 1985.

[22] I. Sommerville, *Software Engineering*. Pearson, ninth ed., 2011.

[23] J. Pries-Heje and J. Johansen, "The SPI Manifesto," 2009. https://2020.eurospi.net/images/eurospi/DownloadCenter/spi_manifesto.pdf.

[24] S. Sadiq, G. Governatori, and K. Namiri, "Modeling Control Objectives for Business Process Compliance," in *International conference on business process management*, pp. 149–164, 2007.

[25] M. Hashmi, G. Governatori, and M. Wynn, "Business Process Data Compliance," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, vol. 7438 LNCS, pp. 32–46, 2012.

# Chapter 9

# Paper C: A Personal Opinion Survey on Process Compliance Checking in the Safety Context

Julieth Patricia Castellanos Ardila, Barbara Gallina.
Proceedings of the 13th International Conference on the Quality of Information and Communications Technology. QUATIC 2020. Communications in Computer and Information Science, vol 1266. Springer, Cham. 2020.

## Abstract

Manually checking the compliance of process plans against the requirements of applicable standards is a common practice in the safety-critical context. We hypothesize that automating this task could be of interest. To test our hypothesis, we conducted a personal opinion survey among practitioners who participate in safety-related process compliance checking. In this paper, we present the results of this survey. Practitioners indicated the methods used and their challenges, as well as their interest in a novel method that could permit them to move from manual to automated practices via compliance checking.

## 9.1 Introduction

Safety standards usually include requirements that prescribe the planning of tasks, and the resources required and produced, e.g., personnel, work products, and tools. Nair et al. [1], reports 9 essential process plans required in safety assessment, i.e., Safety Management, Communication, Risk Management, Configuration Management, Development, Verification and Validation, Modification Procedures, Operation Procedures, and Staff Competence. Manually checking the compliance of such plans against the requirements of applicable standards is a common practice. The checklists used can be obtained by listing the requirements of the standard, or listing personal or organizational practices [2]. A process compliance checklist, which has been accurately filled-in, requires a proper evaluation of the satisfaction of the requirements. Thus, missed requirements are highlighted, providing hints to improve the process.

Process compliance checking could be overwhelming due to the sheer volume and complexity of the knowledge included in the standards. Thus, we hypothesize that automating this task could be of interest. To test our hypothesis, we conducted a personal opinion survey [3] among practitioners who participate in safety-related process compliance checking. In this paper, we present the results of this survey. In particular, practitioners indicated the methods used and their challenges, as well as their interest in a novel method that could permit them to move from manual to automated process compliance checking. These results contribute to systematizing the knowledge about process compliance checking and finding methods and tools for facilitating this practice.

The rest of the paper is organized as follows. In Section 9.2, we present essential background. In Section 9.3, we present the research method used to conduct the survey. In Section 9.4, we present the survey results. In Section 9.5, we discuss our findings. In Section 9.6, we examine related work. Finally, in Section 9.7, we conclude our work and present future work.

## 9.2 Background

This section presents essential background.

### 9.2.1 Facilitating Process Compliance Checking

In the context of the European project AMASS (Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems)[1], we proposed a process-centered planning-time method for safety-related process compliance checking [4, 5]. The method requires users to create artifacts in a SPEM 2.0 (Systems & Software Process Engineering Metamodel)[2] reference implementation supported with Eclipse Process Framework (EPF) Composer[3] (see Figure 9.1), as follows. (1) Method content, which are elements that are part of a process, i.e., roles, tasks, work products, and guidance. (2) A knowledge base of compliance information based on the formalization of standard requirements in Formal Contract Logic (FCL) [6]. FCL is a defeasible deontic logic, i.e., it supports the modeling of norms representing obligations and permissions in a normative context that can be defeated by evolving knowledge. 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. For this, SPEM 2.0 guidance elements are customized as requirements, FCL rules, and compliance effects (which correspond to the propositions of the rules). (3) Compliance effects are annotated in the process tasks. As compliance effects describe the concrete actions prescribed by the standard requirements, users need to evaluate each task action and define its effects in the overall process compliance to make the annotation. For example, the task *Start software Unit Design Process* indicates that the process is performed and has two inputs. Thus, the annotated compliance effects are *addressSwUnitDesignProcess*, *ProvideSwArchitecturalDesign* and *ProvideSwSafetyRequirements*. (4-a) A sequential representation of the process plan, as well as its dynamic representation (4-b), are created by using the compliance annotated tasks. The dynamic representation is used to automatically generate a compliance state representation of the process, which permits automatic compliance analysis with the compliance checker Regorous[4]. Regorous provides (5) compliance checking results, i.e., description of compliance issues, rules and elements involved, and possible resolutions. For facilitating FCL formalization, the concept of Safety Compliance Pattern (SCP) [7, 8] has been defined. An SCP describes commonly occurring normative safety requirements on the per-

[1] https://www.amass-ecsel.eu/
[2] https://www.omg.org/spec/SPEM/About-SPEM/
[3] https://www.eclipse.org/epf/
[4] https://research.csiro.au/data61/regorous/

missible state sequence of a finite state model of a process. These patterns can be instantiated from predetermined templates. EPF-C has been recently updated to Eclipse Neon 4.6.3 in the context of the AMASS project [9].

### 9.2.2 Personal Opinion Surveys

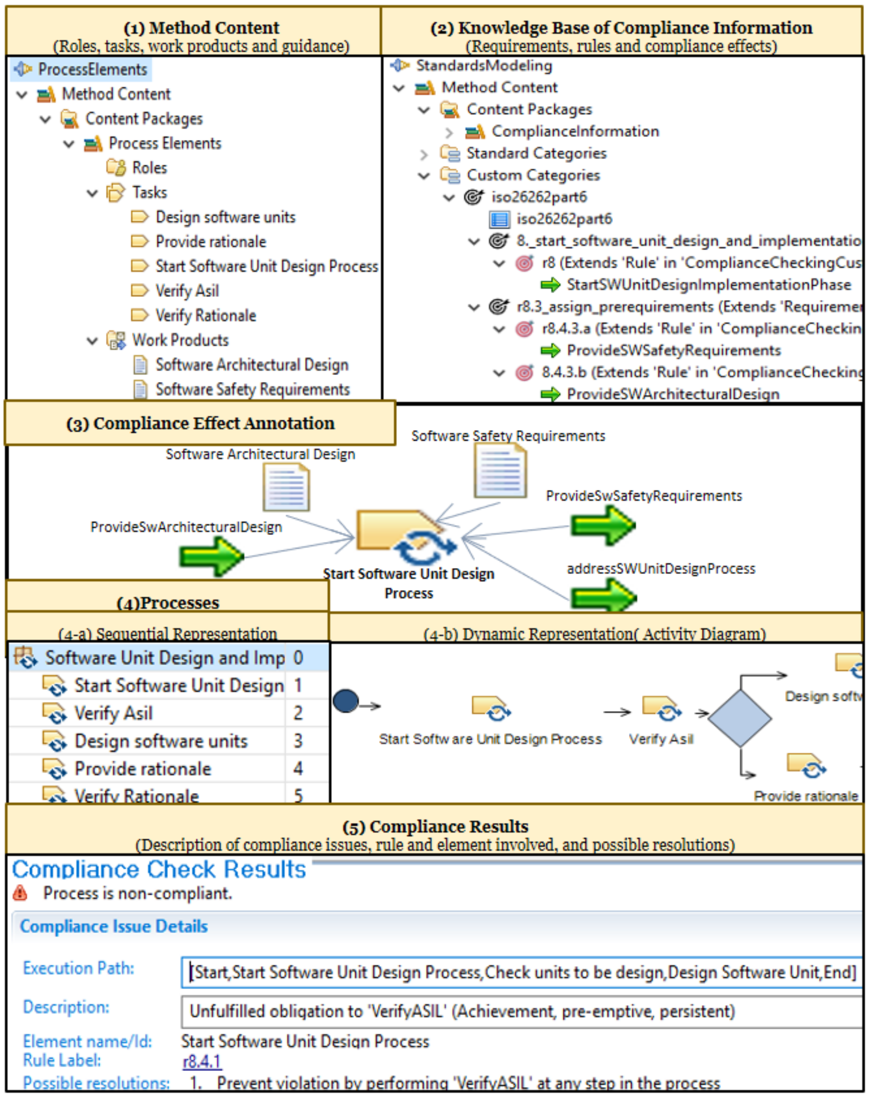
A personal opinion survey [3] is a comprehensive research method for collecting information using a questionnaire completed by subjects. When creating a survey, the first step is to define the expected outcomes. Then, the survey should be designed, e.g., cross-sectional (participants are asked for information at one fixed point in time). It is also essential to define options related to how the survey would be administered. Once designed, the survey instrument should be developed, evaluated, and applied to a sample population, from which obtained data is analyzed.

Four types of validity need to be addressed to make sure that the survey instrument is measuring what it supposes to measure [3]. 1) *Face validity* is a cursory review of items by untrained judges. 2) *Content validity* is a subjective assessment of how appropriate the instrument seems to a group of reviewers with knowledge of the subject matter. 3) *Criterion validity* is the ability of a measurement instrument to distinguish respondents belonging to different groups. 4) *Construct validity* concerns how well an instrument measures the construct it is designed to measure.

In the creation of surveys, Likert Scales [10] are widely used. Likert Scales are psychometric response scales, e.g., a five-point scale ranging from ”Strongly Disagree” to ”Strongly Agree,” used to ask respondents to indicate their level of agreement with a given statement. On a Likert scale, each specific question can have its response analyzed separately, or have it summed with other related items to create a score for a group of statements. Individual responses are generally treated as ordinal data because although the response levels do have a relative position, we cannot presume that participants perceive the difference between adjacent levels to be equal.

### 9.2.3 Technology Acceptance Model

The Technology Acceptance Model (TAM) [11] provides general determinants of computer acceptance. TAM is capable of explaining user behavior across a broad range of end-user computing technologies and user populations, while at the same time being theoretically justified. TAM focuses on three main facets of user acceptance. The first is the degree to which a person believes that using


**(1) Method Content**
(Roles, tasks, work products and guidance)

ProcessElements
- Method Content
  - Content Packages
    - Process Elements
      - Roles
      - Tasks
        - Design software units
        - Provide rationale
        - Start Software Unit Design Process
        - Verify Asil
        - Verify Rationale
      - Work Products
        - Software Architectural Design
        - Software Safety Requirements

**(2) Knowledge Base of Compliance Information**
(Requirements, rules and compliance effects)

StandardsModeling
- Method Content
  - Content Packages
    - ComplianceInformation
  - Standard Categories
  - Custom Categories
    - iso26262part6
      - iso26262part6
      - 8._start_software_unit_design_and_implementatio
        - r8 (Extends 'Rule' in 'ComplianceCheckingCus
          - StartSWUnitDesignImplementationPhase
      - r8.3_assign_prerequirements (Extends 'Requireme
        - r8.4.3.a (Extends 'Rule' in 'ComplianceCheckin
          - ProvideSWSafetyRequirements
        - 8.4.3.b (Extends 'Rule' in 'ComplianceChecking
          - ProvideSWArchitecturalDesign

**(3) Compliance Effect Annotation**

Software Architectural Design
Software Safety Requirements
ProvideSwSafetyRequirements
ProvideSwArchitecturalDesign
Start Software Unit Design Process
addressSWUnitDesignProcess

**(4)Processes**

(4-a) Sequential Representation

| | |
|---|---|
| Software Unit Design and Imp | 0 |
| Start Software Unit Design | 1 |
| Verify Asil | 2 |
| Design software units | 3 |
| Provide rationale | 4 |
| Verify Rationale | 5 |

(4-b) Dynamic Representation( Activity Diagram)

**(5) Compliance Results**
(Description of compliance issues, rule and element involved, and possible resolutions)

Compliance Check Results
Process is non-compliant.

Compliance Issue Details

| | |
|---|---|
| Execution Path: | [Start,Start Software Unit Design Process,Check units to be design,Design Software Unit,End] |
| Description: | Unfulfilled obligation to 'VerifyASIL' (Achievement, pre-emptive, persistent) |
| Element name/Id: | Start Software Unit Design Process |
| Rule Label: | r8.4.1 |
| Possible resolutions: | 1. Prevent violation by performing 'VerifyASIL' at any step in the process |

Figure 9.1: Method for Facilitating Process Compliance Checking.

a particular method will be free of effort (Perceived Usability). The second is related to a person's subjective probability that using a particular system would enhance his/her job (Perceived Usefulness). The third is the extent to which a person intends to use a particular system (Intention to Use).

## 9.3 Research Method

In this section, we present the details regarding the creation of a personal opinion survey. We followed the guidelines recalled in Sections 9.2.2 and 9.2.3.

### 9.3.1 Research Questions

In this survey, we aim at gathering information about current industrial practices and challenges in process compliance checking, as well as the acceptance level of the method for automated compliance checking (recalled in Section 9.2.1). Within this scope, we formulate the research questions presented below.

- RQ1: How do practitioners currently perform process compliance checking?
- RQ2: What are the challenges that practitioners face when performing process compliance checking?
- RQ3: What is the level of acceptance of practitioners regarding a novel method for facilitating automated compliance checking?

### 9.3.2 Survey Design

We designed a cross-sectional web-based personal opinion survey, whose goal is to collect data relevant to answer the research questions presented in Section 9.3.1. The target population is practitioners involved in process compliance checking in the safety-related context. The final survey[5], which starts with a short introduction to the purpose of the study, is composed of 21 questions, which are organized into four parts.

1. **Demographics.** Questions 1-7 aim at gathering the background characteristics of the practitioners.

[5]https://www.dropbox.com/s/efcab84me7kxpj8/FinalSurvey.pdf?dl=0

2. **Current practices.** Questions 8-14 aim at gathering information about practitioners' experiences in compliance checking.

3. **Challenges.** Questions 15 and 16 aim at inquiring about the challenges appearing in process compliance checking. In question 15, practitioners rate the importance of 7 possible challenges by using a five-point Likert scale ranging from Unimportant to Very Important. Question 16 is an open question in which practitioners can write further challenges.

4. **Automated process compliance checking.** First, practitioners read information about the method for facilitating automated compliance checking recalled in Section 9.2.1. Then, we present the questions 17-21 as a series of claims from which we seek practitioners' degree of acceptance regarding the user acceptance aspects described in the TAM model (see Section 9.2.3), i.e., the method usefulness, usability, and user's intention to use it. To collect the answers, we use a five-point Likert Scale ranging from Strongly Agree to Strongly Disagree.

We were interested in the practitioners' overall experience. Thus, where possible, the practitioners were allowed to select more than one option to indicate their experience regarding several practices. Practitioners were also given the possibility to mention additional options or answer "Don't know" if this was the case. We consider that completing the survey would take between 20-25 minutes.

### 9.3.3 Instrument Evaluation and Data Collection

The first author created a set of initial questions. The second author helped to structure and design the survey by providing comments for cleaning ambiguity and a more in-depth analysis that led to the formulation of further questions. Then, we distributed the survey to a selected group of safety experts during the Scandinavian Conference on System & Software Safety[6]. One expert provided valuable comments that were used to improve the survey. The final evaluation was performed by both authors, improving textual explanations and questions.

The data was collected from January 22th to February 28th of 2020. The survey was distributed via personal e-mail invitations. The selection of the practitioners included industrial experts (on purpose, we discarded research institutions) that participate in European projects related to certification and

[6]http://safety.addalot.se/2019

self-assessment. We also extracted industrial-related practitioners from conferences, symposiums, and workshops related to safety assurance. In total, we obtained 15 valid responses from which 8 were received after the initial invitation letter, and 7 were received after a reminder e-mail.

### 9.3.4 Subject Characteristics and Data Analysis

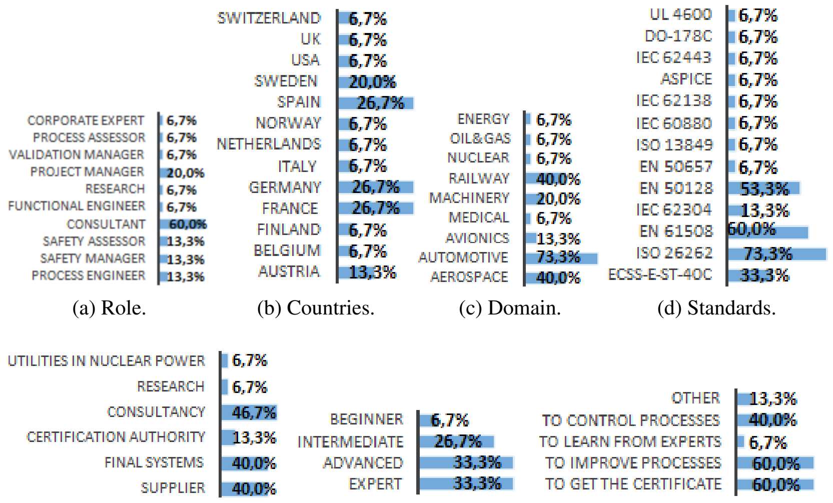
The valid answers were obtained from practitioners mostly working in the consultatory branch (see Figure 9.2a and Figure 9.2g) which have experience demonstrating process compliance checking in 13 countries (see Figure 9.2b), predominantly Europe. The practitioners have experience in 9 safety-related domains (see Figure 9.2c) and 13 standards (see Figure 9.2d), where automotive is the most represented. The major interest of the practitioners, which shows higher levels of expertise (see Figure 9.2f) in process compliance checking, is to get the compliance certification and improve processes (see Figure 9.2g). The analysis of our survey was adjusted with the information provided in the ”Others” option.

### 9.3.5 Survey Validity

The four types of validity of the survey instrument (recalled in Section 9.2.2) were addressed as follows. To avoid *face validity*, we perform a careful review of our survey instrument in several stages and with experts in the field of safety certification. *Content validity* was assured by doing a careful literature review on the topic and validating as well with experts. Regarding *criterion validity*, we assure that the practitioners' background was related to the type of expertise we were looking by making a careful selection process. For reducing the *construct validity*, we allow the practitioners to include the ”Others” option. Thus, the threat of providing an incomplete list of options is minimized. Additionally, to avoid evaluation apprehension, we guaranteed confidentiality and anonymity of the responses.

## 9.4 Survey Results

In this section, we present the results of the survey by answering the research questions presented in Section 9.3.1.



(a) Role. (b) Countries. (c) Domain. (d) Standards.

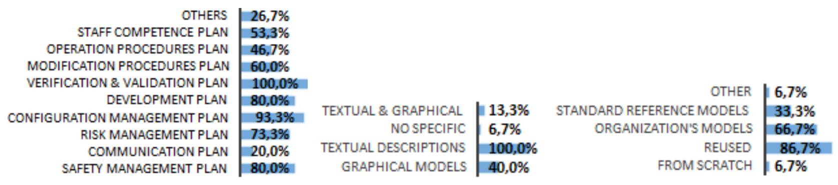
(e) Company Type. (f) Expertise. (g) Checking Reasons.

Figure 9.2: Demographic Results.

### 9.4.1 Current Practices (RQ1)

Figure 9.3a shows the 9 process plans (recalled in the introductory part) provided as alternatives in the questionnaire in the vertical axis, and the percentage of respondents, who selected each type in the horizontal axis. Figure 9.3a shows that practitioners have performed compliance checking mostly on the Verification and Validation, Configuration Management, Safety Management, Development, Risk Management, and Modification Procedure Plans. The remaining plans listed were less considered as part of the practitioners' compliance checking duties. In the ”Others” option, practitioners mentioned the Software Quality Assurance, Safety Assessment, Documentation, and Cybersecurity Plans. Current practices indicate that processes are mostly represented with only text, but graphical representations are also relevant (see Figure 9.3b). Moreover, process plan reuse is a common practice (see Figure 9.3c).

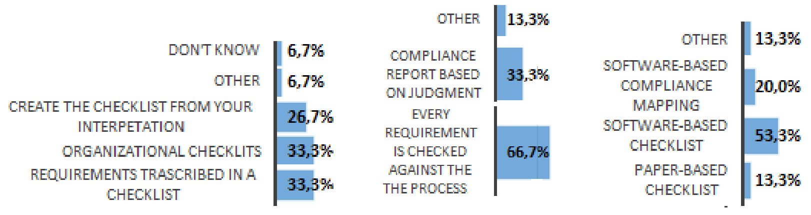
Regarding checklist preparation, we found that the three alternatives given in the questionnaire are almost equally used (see Figure 9.4a). The practice of compliance checking is done in different ways. Most commonly, practitioners take every requirement and check it against the information provided by the



(a) Plan Types. (b) Representation. (c) Creation.

Figure 9.3: Information Regarding Processes.

process specification (see Figure 9.4b). Practitioners also base the compliance assessment on other practices, such as the use of points of compliance, and the assessment of strengths and weaknesses of the findings. It is common that practitioners use software tools for performing compliance management tasks (see Figure9.4c). Rational doors, Microsoft suite (e.g., Word, Excel, and MS project), opencert, verification studio, engineering studio, stages (for modeling processes) were the tools mentioned by practitioners in the survey.



(a) Preparation. (b) Checking. (c) Mechanism.

Figure 9.4: Information Regarding Compliance Checking.

### 9.4.2 Challenges (RQ2)

Figure 9.5 presents a set of challenges that could appear during process compliance checking to which we ask respondents to rate them from very important to unimportant. The results shows that one of the challenges that was considered very important by the practitioners is that *”it is common to miss requirements”*. Important challenges are: *”Check process-based compliance requires that many people are involved”*, *”Check the compliance of a process requires many interactions”*, *”Check process-based compliance requires many hours of* *work”* , and *”It isn’t easy to determine the kind of information that should be provided as evidence from the process perspective.”* The practitioners considered the other challenges moderately important. The practitioners also have the option to list their challenges to which they answer that *”Sometimes there is no access to the evidence”*, *”Sometimes the safety assessor could have different interpretations”*, and *”It is difficult to check the user acceptance of the defined processes.”*

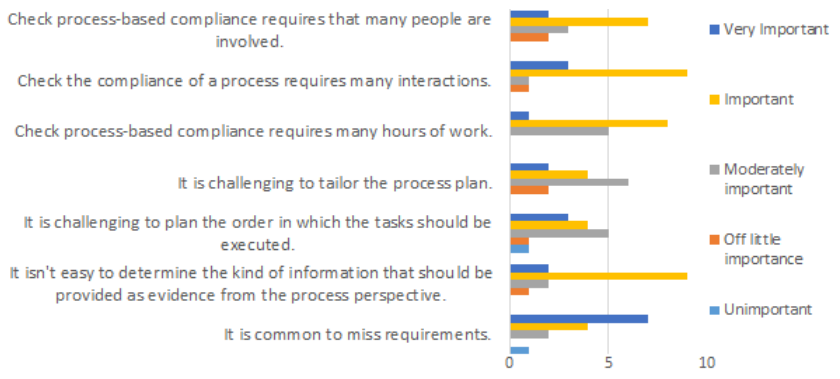


Figure 9.5: Challenges in Process Compliance Checking.

## 9.4.3 Automatic Process Compliance Checking (RQ3)

This part of the survey gathered data regarding the user acceptance level of the method for facilitating automated process compliance checking (recalled in Section 9.2.1). Initial evaluation is performed on FCL, which is the logic used to formalize the requirements prescribed by the standards. Practitioners somewhat agree that the formalization of standard requirements could be facilitated with FCL since it provides the compliance concepts and there are safety compliance patterns to instantiate (see Figure 9.6). Practitioners also somewhat agree that FCL can be used to support the creation of the tailoring rules. However, most of the practitioners are neutral whether the analysis required to formalize process requirements could help them to understand their intention.

Regarding the ability of the method to represent processes and compliance information (see Figure 9.7) we found that the majority of the practitioners somewhat agree with the statements regarding the provision of graphical rep-

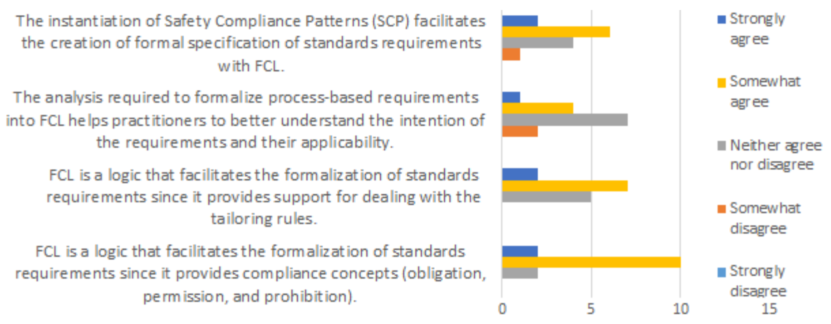


Figure 9.6: The Ability to Formalize Requirements with FCL.

resentations. In particular, graphical representation of the compliance information, as well as process plans, facilitate their understanding and documentation. Similarly, the majority of the practitioners somewhat agree that this aspect also would facilitate compliance management.

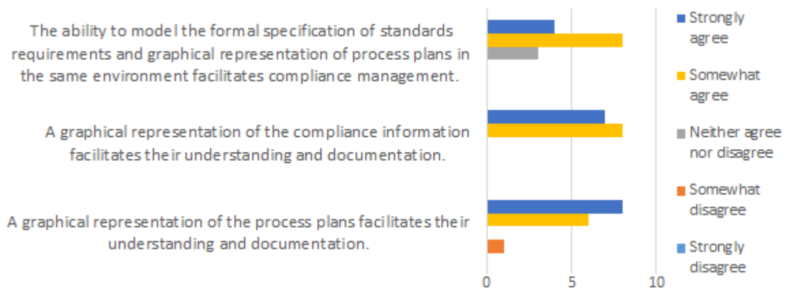


Figure 9.7: The Ability to Represent Processes and Compliance Information.

Then, we focused on the ability of the method to perform automated compliance checking (see Figure 9.8). As the figure depicts, the ability to perform automated compliance checking is seen by the majority of the practitioners as favorable. In particular, practitioners somewhat agree that the iterative application of automated compliance checking can help them to reach process plans with compliant states. Moreover, the majority of the practitioners strongly agree that modifying a compliant process plan to define a new process reduces the work that needs to be done. Finally, traceability could be facilitated with a hierarchically organized knowledge-based of compliance artifacts. Such an organization helps to understand the source of compliance problems.

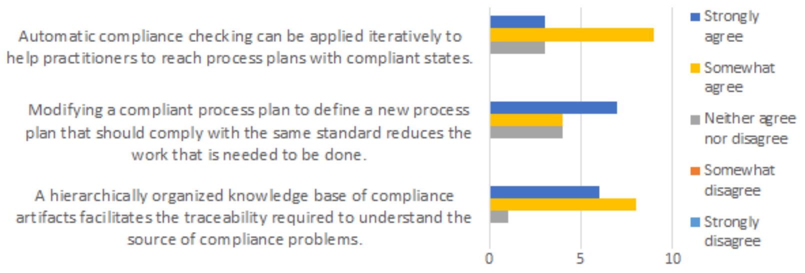


Figure 9.8: The Ability to Perform Automated Compliance Checking.

Figure 9.9 shows the results regarding the perceived usability aspect of the method. Practitioners do not strongly agree or strongly disagree with any of the questionnaire's options. However, there are two statements that practitioners somewhat agree: it is easy to 1) trace uncompliant situations and 1) graphically model process elements.

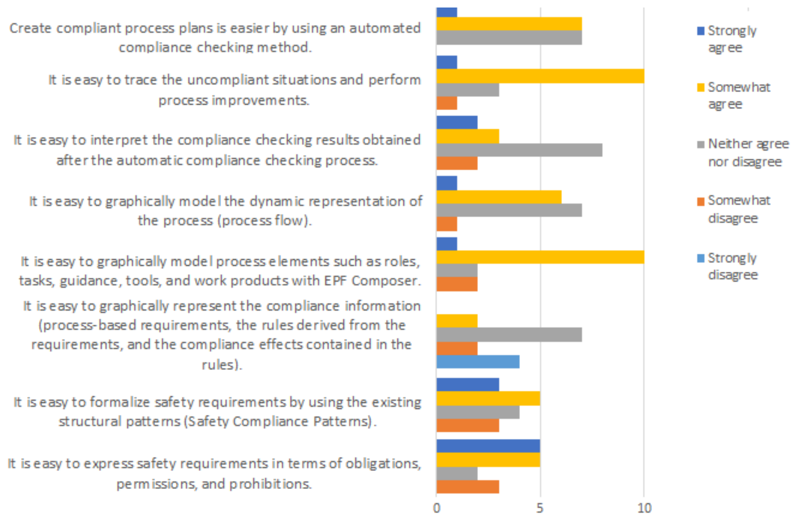


Figure 9.9: Perceived Usability Aspect of the Method.

Finally, one question was asked to the practitioners about their intention to use the method. As Figure 9.10 depicts 67% of the practitioners indicated that they would use the method for facilitating automated compliance checking if it were made available. In contrast, 13% of the practitioners do not know, and 20% would not do it.

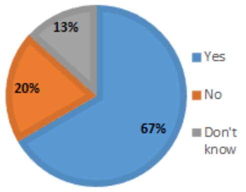


Figure 9.10: Intention to Use.

## 9.5 Discussion

In this section, based upon the result of the survey, we discuss our findings.

**Current Practices:** Given the characteristics of the subjects, presented in Section 9.3.4, we consider our sample to be representative of the European safety-critical context. For this kind of population, process compliance checking is not only the way towards a safety certificate but also a mechanism for process improvement (see Figure 9.2g). Their current practices include the checking of a variety of process plans (see Figure 9.3a). Additional plan types respect to the ones described in the introductory part were considered necessary in the safety-critical context, i.e., Software Quality Assurance Plan, Safety Assessment Plan, Documentation Plan, and Cybersecurity Plan (see Section 9.4.1). Thus, it seems that compliance management from the process perspective is a growing area. Practitioners also create process plans mostly by reusing previous processes or their elements (see Figure 9.3c). This aspect indicates that support for reusability is significant in process compliance management. We also could see that there are different ways to create checklists (see Figure 9.4a). It is interesting to see that most of the time, the practitioners receive the checklist from the organization (which is based on the organization's experience in the domain) or transcribe the actual requirements provided by the standard direct into a checklist. In those cases, there is not additional intellectual work included in the preparation of the checklist, and the provision of a general, widely accepted checklist could be useful for minimizing such initial work. Finally, most of the practitioners use software tools to perform compliance checking to support their activities (see Figure 9.4c). Thus, it is not expected that the introduction of more sophisticated software tools would generate extreme distortions in their daily job. However, it would be good to revise the ways to introduce them smoothly.

**Challenges:** Practitioners are faced with several challenges when performing compliance checking, as presented in Section 9.4.2. In general, practitioners consider that compliance checking is prone-to-error. For them, it is possible to miss requirements. Moreover, they consider that it is not easy to determine the kind of information that should be provided as evidence (or there is no access to evidence), and that there are different possible interpretations provided by the assessors. In addition, practitioners consider that compliance checking is time-consuming since it requires many hours of work and several iterations. Finally, many people in the organization are needed making it also resource-consuming. Thus, there is a need for solutions that provide more confidence and efficiency in process compliance checking.

**Automated compliance checking:** User acceptance is a major for any technological endeavor. In general, as we presented in Section 9.4.3, there are advantages regarding automated process compliance checking. In particular, as depicted in Figure 9.6, there is a good degree of acceptance for the characteristics provided by FCL, which is the formal approach used for requirements representation. However, there is some hesitation regarding its usage, as expected with formal methods. In particular, practitioners do not see how the analysis required to formalize process requirements would help them to understand their intention. For this reason, it is necessary to explain further the formalization part of the method by providing more guidance and examples. As presented in Figure 9.7 and Figure 9.8, the ability to represent processes and compliance information graphically and the ability to automatically check compliance also have a good degree of acceptance. Thus, the method has high acceptability potential, and its graphical representations are considered the strongest advantage. Finally, as presented in Figure 9.9, two aspects regarding the method are considered easy to use, i.e., graphically represent process models and trace uncompliant situations. However, we need to provide mechanisms for improving the tool usability in terms of compliance information representation, which appears to be not easy to use by practitioners. In addition, we need to improve the representation of checking results. For facilitating these aspects, we can provide more specific graphical representations of the compliance artifacts and, after backpropagating the results of Regorous into EPF Composer, present them in a suitable user interface that provides detailed explanations. Finally, practitioners show a willingness to use the method, which could be helpful for evolving from the current manual practices to automated practices via compliance checking.

## 9.6 Related Work

Nair et al. [2] performed in-depth interviews with 7 safety-related practitioners, which show the importance of checklists in safety assessment. In [12], a personal opinion survey was applied to 53 experts to study safety evidence management practices. Our survey also analyzed the use of the process plans analyzed in [12], and found that additional process-related plans are required in safety assessment. In [13], the authors present the results of interviews with practitioners regarding change impact analysis, which is essential during safety assessment. De la Vara et al. [14] surveyed safety evidence, particularly the circumstances under which it is created, the tool support used, and the challenges faced. In contrast to the works previously mentioned [2, 12, 13, 14] our focus is to investigate the currently used methods and its challenging aspects in process compliance checking, as well as the practitioner's interest in novel methods for facilitating the automation of this task. The work conducted by Diebold and Scherr [15] reports industrial practices regarding the use of software process descriptions. In particular, the survey shows that companies use different process representations, i.e., graphical, table-based, or structured text notations. It also shows that the use of formal models and their advantages are highly desirable by practitioners. Our study differs from [15] in that we also include aspects regarding the use of formal descriptions of processes for compliance checking.

## 9.7 Conclusions and Future Work

In this paper, we presented the results of a personal opinion survey conducted among practitioners who participate in process compliance checking in the safety-critical context. The practitioners indicated that they mostly represent process plans and standard requirements by using software-based tools. Thus, software-based compliance checking aids are not new for them. However, practitioners consider that process compliance checking is prone-to-error; e.g., missing requirements is a common problem. Process compliance checking also requires many hours of work and several people. Finally, the practitioners show a favorable position regarding automated process compliance checking based on SPEM 2.0-like artifacts. They also indicated usability aspects regarding the formalization of requirements that we need to revisit and improve.

Future work will include more empirical research with the use of interviews and observations to see, for instance, how practitioners carry out their compliance checking in real settings. In addition, the usability aspects will be revisited, in order to provide more guidance and improve the representation of compliance artifacts and checking results. Finally, the tool support will be concretized to facilitate evaluations in terms of efficiency through industrial case studies.

# Bibliography

[1] S. Nair, J. De La Vara, M. Sabetzadeh, and L. Briand, "An extended systematic literature review on provision of evidence for safety certification," *Information and Software Technology*, vol. 56, no. 7, pp. 689–717, 2014.

[2] S. Nair, T. Kelly, and M. Jørgensen, "A Report on the State-of-the-Practice of Safety Evidence Assessment," tech. rep., 2014.

[3] B. Kitchenham and S. Pfleeger, "Personal Opinion Surveys," in *Guide to Advanced Empirical Software Engineering*, ch. 3, pp. 63–92, Springer Science & Business Media, 2008.

[4] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, "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.

[5] J. P. Castellanos Ardila, B. Gallina, and F. UL Muram, "Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance," in *18th International SPICE Conference*, 2018.

[6] G. Governatori, "Representing business contracts in RuleML," *International Journal of Cooperative Information Systems*, vol. 14, no. 02n03, pp. 181–216, 2005.

[7] J. P. Castellanos Ardila and B. Gallina, "Formal Contract Logic Based Patterns for Facilitating Compliance Checking against ISO 26262," in *1st Workshop on Technologies for Regulatory Compliance*, pp. 65–72, 2017.

[8] J. Castellanos Ardila, B. Gallina, and G. Governatori, "Lessons Learned while formalizing ISO 26262 for Compliance Checking," in *2nd Workshop on Technologies for Regulatory Compliance*, pp. 1–12, 2018.

[9] M. Javed and B. Gallina, “Get EPF Composer back to the future: a trip from Galileo to Photon after 11 years,” in *EclipseCon*, 2018.

[10] D. Bertram, “Likert Scales Are the Meaning of Life. CPSC 681-Topic Report,” 2006.

[11] F. Davis, “A technology acceptance model for empirically testing new end-user information systems: Theory and results.,” *Massachusetts Institute of Technology*, 1985.

[12] S. Nair, J. De La Vara, M. Sabetzadeh, and D. Falessi, “Evidence management for compliance of critical systems with safety standards: A survey on the state of practice,” *Information and Software Technology*, vol. 60, pp. 1–15, 2015.

[13] M. Borg, J. de la Vara, and K. Wnuk, “Practitioners’ perspectives on change impact analysis for safety-critical software – A preliminary analysis,” in *International Conference on Computer Safety, Reliability, and Security*, pp. 346–358, 2016.

[14] J. De La Vara, M. Borg, K. Wnuk, and L. Moonen, “An Industrial Survey of Safety Evidence Change Impact Analysis Practice,” *IEEE Transactions on Software Engineering*, vol. 42, no. 12, pp. 1095–1117, 2016.

[15] P. Diebold and S. Scherr, “Software process models vs descriptions: What do practitioners use and need?,” *Journal of Software: Evolution and Process*, vol. 29, no. 11, pp. 1–13, 2017.

# Chapter 10

# Paper D: Compliance-aware Engineering Process Plans: The case of Space Software Engineering Processes

Julieth Patricia Castellanos Ardila, Barbara Gallina, Guido Governatori.
Journal of Artificial Intelligence and Law. 2021

## Abstract

Safety-critical systems manufacturers have the duty of care, i.e., they should take correct steps while performing acts that could foreseeably harm others. Commonly, industry standards prescribe reasonable steps in their process requirements, which regulatory bodies trust. Manufacturers perform careful documentation of compliance with each requirement to show that they act under acceptable criteria. To facilitate this task, a safety-centered planning-time framework, called ACCEPT, has been proposed. Based on compliance-by-design, ACCEPT capabilities (i.e., processes and standards modeling, and automatic compliance checking) permit to design Compliance-aware Engineering Process Plans (CaEPP), which are able to show the planning-time allocation of standard demands, i.e., if the elements set down by the standard requirements are present at given points in the engineering process plan.

In this paper, we perform a case study to understand if the ACCEPT produced models could support the planning of space software engineering processes. Space software is safety and mission-critical, and it is often the result of industrial cooperation. Such cooperation is coordinated through compliance with relevant standards. In the European context, ECSS-E-ST-40C is the defacto standard for space software production. The planning of processes in compliance with project-specific ECSS-E-ST-40C applicable requirements is mandatory during contractual agreements. Our analysis is based on qualitative criteria targeting the effort dictated by task demands required to create a CaEPP for software development with ACCEPT. Initial observations show that the effort required to model compliance and processes artifacts is significant. However, such an effort pays off in the long term since models are, to some extend, reusable and flexible. The coverage level of the models is also analyzed based on design decisions. In our opinion, such a level is adequate since it responds to the information needs required by the ECSS-E-ST-40C framework.

## 10.1 Introduction

Safety-critical systems manufacturers have the duty of care[1] [2], i.e., they should follow accepted practices of reasonable care, usually found in industry standards [3]. Failure or inadequate compliance with such standards could lead to legal risks, i.e., penalties [4] and prosecutions [5]. For example, in 2015, The Volkswagen "Dieselgate" scandal [6], i.e., emissions levels of the cars were not complying with emission standards, resulted in huge lost to the company [7]. Compliance with industry standards is relevant evidence for a jury to consider in a product liability action [8]. In England, the Health and Safety Executive has used compliance with IEC 61508 [9] as a guideline for bringing legal actions if harm is caused by safety-critical systems [2].

Industry standards demand documented evidence of responsibilities and agreements [10]. Usually, they place requirements on engineering processes [11], which should be planned at the beginning of the engineering activities [12]. Compliant engineering process plans are used to coordinate and track engineering progress, support contractual relationships between partners and agreements with certification bodies. In the context of the project AMASS [13, 14], a safety-centered planning-time framework, called ACCEPT (Automated Compliance Checking of Engineering Process plans against sTandards) [15, 16], has been proposed to facilitate process compliance checking tasks. ACCEPT is based on Compliance-by-design [17], an approach aimed at integrating compliance requirements at design time, permitting to resolve compliance violations in engineering process plans before they are executed. ACCEPT is supported by rules-based technologies to automatically check if a compliance-aware engineering process plan (CaEPP) is designed, i.e., if the elements set down by the requirements (e.g., tasks, personnel, work products, techniques, and tools, as well as their properties) are present at given points in the engineering process plan. A CaEPP can show how and when the evidence will be produced, taking into account all the process-related requirements or their tailoring (i.e., adapted to the specific project conditions in a compliant form). A CaEPP is able to demonstrate intentional compliance [18], i.e., planning-time allocation of responsibilities, such that if every actor fulfills its duties, then the compliance is ensured.

ACCEPT uses Formal Contract Logic (FCL) [19], which provides a framework that unambiguously represents normative knowledge, i.e., obligations,

[1]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 [1].

prohibitions, and permissions. ACCEPT also uses the compliance checker Regorous [20], which provides an algorithm that determines whether an annotated process model is compliant with a specific set of FCL rules. The annotated process models required by Regorous, i.e., process enriched with compliance effects through annotations representing the formalized requirements, is provided via SPEM 2.0 (Systems & Software Process Engineering Metamodel) [21]. We chose SPEM 2.0, as opposed to other process modeling notations, for several reasons. 1) SPEM 2.0 is a standardized language, based on the Unified Modeling Language (UML) [22]. 2) SPEM 2.0-like artifacts can be captured freely via Eclipse Process Framework Composer (EPF-C) [23] (recently ported to Eclipse Neon 4.6.3 [24]). 3) SPEM 2.0 has the ability to capture several types of information. 4) SPEM 2.0 provides variability mechanisms that can be exploited for flexible process derivation. Such mechanisms are currently tool-supported via the composition of EPC-C with BVR (Base Variability Management Tool [25]) [26] included in the AMASS tool platform [27]. 5) SPEM 2.0 elements can also be customized to permit the definition of a variety of artifacts. All these characteristics facilitate the modeling of process-related compliance artifacts, i.e., engineering processes and their elements, as well as standards requirements and their derived rulesets, annotated process plans, and workflows representations, which can be also reused, tailored and explicitly documented. EPF-C models can be ported to other tools, via model-driven transformations. Finally, SPEM 2.0 is widely accepted by the research community [28] and industry [29].

In this paper, we perform a case study to understand if the ACCEPT produced models could support space manufacturers' needs in planning space software engineering processes. Space software is safety-critical since a failure could cause a mission disaster leading to financial losses, environmental pollution, and people's endangerment in case of manned missions [30]. Moreover, space software production is frequently the result of industrial cooperation. For example, the European space context consists of space agencies often acting as customers in projects, and companies, which act as suppliers, or as intermediate customers for subcontractors [31]. Meeting the highest levels of industry standards helps to coordinate such cooperation. In this context, ECSS-E-ST-40C is the de-facto standard for space software production. Thus, the planning of processes in compliance with project-specific ECSS-E-ST-40C applicable requirements is mandatory during contractual agreements. We have selected a portion of the ECSS-E-ST-40C [32] related to the design of the software items to perform our analysis, which is based on a set of well-defined qualitative criteria defined in [33]. In particular, we target the effort dictated by task demands required to create a CaEPP for software development with ACCEPT. Initial observations show that the effort required to model compliance and processes artifacts is significant. However, such an effort pays off in the long term since models are, to some extend, reusable and flexible. The coverage level of the models is also analyzed based on design decisions. In our opinion, such a level is adequate since it responds to the information needs required by the ECSS-E-ST-40C framework.

The paper is organized as follows. In Section 10.2, we provide essential background. In Section 10.3, we present the case study design. In Section 10.4, we present the data collection. In Section 10.5, we present the case study analysis. In Section 10.6, we discuss the findings. In Section 10.7, we present related work. Finally, In Section 10.8, we present the conclusion and future work.

## 10.2 Background

In this section, we present essential background required in this paper.

### 10.2.1 Compliance with Industry Standards

Industry standards offer frameworks that encompass adequate practices refined by experts from historically successful experiences [34] as well as knowledge and awareness of public policy, societal norms, and preferences [35]. Organizations comply with industry standards (sometimes augmented with internal guidelines) to minimize legal risks [36] since compliance is the demonstration that the organization acts under well-defined and acceptable criteria. In some industries, a compliance certification is mandatory to be able to sell products on a specific market, e.g., medical devices [37]. Compliance is also a mark that customers trust. For example, in space, standards requirements are intended to support the contractual negotiation by helping customers to formulate their requirements and suppliers to prepare their responses and to implement the work [38]. Contracts are legally binding documents in which development freedom becomes limited. Thus, non-compliance is harmful to the success of organizations. In the remaining part of this section, we recall essential information regarding software process standards, and we focus on the software engineering standard that regulates the European space context.

### Software Process Standards

In the past, software companies vacate liability for software errors by licensing it to a user that agreed that the company would not be liable for damages caused by errors in the code [39]. This policy contributed to enforce the computer revolution. However, the software was limited to provide simple tasks and sometimes computational power for complex systems. Nowadays, the software is used to control most systems (including physical) involving potentially large and even catastrophic loses [40]. Consequently, software projects are becoming critical in terms of legal aspects, e.g., software not delivered in time or with ill-defined functionality could lead to legal claims [41]. In the safety-critical context, legal aspects are also related to each activity performed in its production [42, 43]. The reason is that a well-defined process would make it difficult to exclude significant aspects of the software engineering aspects. Examples of inadequate software engineering process practices have been considered as one of the factors that cause Therac-25 radiotherapy machine's massive overdose [44] and the failed launch of the ARIANE 5 [45]. Choices seem not to be either deliberately planned in the definition of the features created to force the plane BOEING 737-MAX to nose down, causing fatal accidents [46]. Sound engineering processes present a structured collection of practices [47]. Companies that follow the process-related frameworks prescribed by industry standards tend to achieve more consistent results [48]. Legal risk can also be prevented since proofs of compliance can demonstrate that companies have taken correct steps while performing acts that could foreseeably harm others [42, 36]. Software process standards do not restrict organizations from using a particular development lifecycle. Instead, the process framework focuses on what needs to be done. Sometimes, who should be involved in the process and the recommended techniques and tools to be used to achieve desirable results are also prescribed. Route maps may be indicated, but exact specifications on how the process should be done usually are not provided. In addition, software with high requirements, such as safety, requires detailed documentation according to regulations, which may imply the creation of very formal software processes [41]. For this reason, a software process engineer is responsible for the selection, composition, and correct documentation of adequate software process elements aimed at achieving the required process goals [49].

### ECSS Standards: Focus on Software Engineering

The European Cooperation for Space Standardization (ECSS) developed a set of standards for use in all European space activities. The ECSS standard system includes three branches, i.e., Management (M), Engineering (E), and Product Assurance (Q). Handbooks (HB) guide the application of the requirements. The software engineering handbook, ECSS-E-HB-40A [50], states that in a space software project, a customer-supplier business agreement should be established. The customer shall produce the project requirements documentation, which could be produced by using the ECSS Applicability Requirements Matrix (EARM). The EARM should have the list of applicable ECSS requirements with identifiers, applicability condition, i.e., applicable without change (A), applicable with modification (M), not applicable (D), and new generated requirement (N). The supplier responds with the ECSS Compliance Matrix (ECM), indicating the compliance for each requirement provided in the EARM. Partial compliance needs to be detailed, such that the customer can assess the extent to which the objective of the ECSS is covered. Non-compliance also needs to be investigated in terms of feasibility and acceptability in the scope of the project. When a space project starts, the supplier has to identify a suitable software lifecycle process. Thus, discussions about the technical specifications based on the requirements baseline must start early in the lifecycle process [51].

In space software development, the requirements prescribed by the standard ECSS-E-ST-40C [32], which determines mission (non-safety) requirements on how the goals can be achieved, should be applied. Such requirements could be tailored, i.e., adapted for the characteristics of the project. For example, ECSS-E-ST-40C-Annex R, provides a pretailoring based on safety criticality categories, which rank from catastrophic to negligible (prescribed in ECSS-Q-ST-40C [52]). Thus, mission requirements have an inherent relationship with safety issues. Further tailoring should be analyzed in the scope of the project and its consequences assessed and documented. If requirements are tailored out, the associated expected outputs are also tailored out. Table 10.1 recalls a set of requirements from the phase *5.5. Software Design and Implementation Engineering Process*, particularly the activity *5.5.2. Design of Software Items*. The inputs of this activity are the Technical Specification of the Software Components (TSSC), the Architectural Design (AD) the Design Justification (DJ), and the Preliminary Design Review (PDR). During the *detailed design review* the expected items of every requirement are revised and compiled in eight work products, i.e., the DDF (Design Definition File), SDD (Software Design Document), CDR (Critical Design Review), TS (Technical Specification), ICD (Interface Control Document), SUM (Software User Manual), DJF (Design Justification File) and SUITP (Software unit-integration Test Plan).

Table 10.1: Activity 5.5.2: Design of the Software Items.

| **Id.** | **Requirement** | **Expected Item** |
|---|---|---|
| **5.5.2.1** | **Detailed design of software component** | Software components design document (Scdd) |
| a. The supplier shall develop a detailed design for each component of the software and document it.<br>b. Each software component shall be refined into lower levels<br>c. It shall be ensured that all the software requirements are allocated to software units. | | |
| **5.5.2.2** | **Development and documentation of the software interfaces detailed design** | External and internal interfaces design (Eid /Iid). |
| a. The supplier shall develop and document a detailed design for the interfaces (external and internal). | | |
| **5.5.2.3** | **Production of the detailed design model** | Software static-dynamic and -behavioral design model (Ssdm, Sddm, Sbdm) |
| a. The supplier shall produce the detailed design model of the software components (static, dynamic and behavioural aspects). | | |
| **5.5.2.4** | **Software detail design method** | Software design method (Sdm). |
| a. The supplier shall use a design method to produce the detailed design including software units, their interfaces, and relationships. | | |
| **5.5.2.5** | **Detailed design of real–time software** | Real-time software dynamic design model (R-tddm). |
| a. The dynamic design model shall be compatible with the computational model of the architectural design.<br>b. The supplier shall document and justify all timing and synchronization mechanisms.<br>c. The supplier shall document and justify all the design mutual exclusion mechanisms.<br>d. The supplier shall document and justify the use of dynamic allocation of resources.<br>e. The supplier shall ensure protection during the use of dynamic allocation of resources. | | |
| **5.5.2.6** | **Utilization of description techniques for the software behaviour** | Software behavioural design model techniques (Sbdmt). |
| a. The behavioural design of the units shall be described by means of techniques (i.e. automata and scenarios.) | | |
| **5.5.2.7** | **Determination of design method consistency for real–time software** | Compatibility of real-time design methods with the computational model (CR-tdm). |
| a. It shall be ensured that all the methods utilized for different item of the same software are, from a dynamic stand–point, consistent among themselves and consistent with the selected computational model. | | |
| **5.5.2.8** | **Development and documentation of the software user manual** | Software user manual (Sum) |
| a.The supplier shall develop and document the software user manual. | | |
| **5.5.2.9** | **Definition and documentation of the software unit test requirements and plan** | Software unit test plan (Sutp). |
| a. The supplier shall define and document responsibility and schedule, control procedures, testing approach, test design and test case specification for testing software units. | | |
| **5.5.2.10** | **Conducting a detailed design review** | |
| a. The supplier shall conduct a detailed design review. | | |

## 10.2.2 SPEM 2.0

SPEM 2.0 (Software and Systems Process Engineering Metamodel) [21] is a modeling language that defines the elements required to plan engineering processes. An engineering process is a sequence of units of work (e.g., tasks) that consume resources (e.g., employee time) to transform inputs (e.g., data, raw material) into outputs (e.g., products) [53]. SPEM 2.0 concepts are defined in separated UML [22] packages that are interrelated. For example, the meta-class *TaskDefinition*, which belongs to the package *MethodContent* is used to describe assignable units of work. Instances of *Task Definition* can be applied in a process breakdown structure by defining a proxy with a *TaskUse*, a meta-class that belongs to the package *ProcessWithMethods* (both meta-classes are highlighted with red in Figure 10.1). The same approach is used for the definition and use of roles and work products. Instead, a *tool* definition is used to specify the tool's participation in a Task Definition. *Guidance*, which belongs to the package *Managed Content*, is a describable element that provides additional information to other elements. There are different guidance kinds, e.g., concept and reusable asset. A *Delivery Process*, which belongs to the package *Process Structure*, describes an approach for performing a specific project. A *Category* is used to group elements in a recursive way. SPEM 2.0 supports variability management on breakdown structures representing processes as well as in content elements. In particular, we recall the variability mechanism called *extends*, in which the method content element that extends the base method element inherits the attributes of the extended base element.

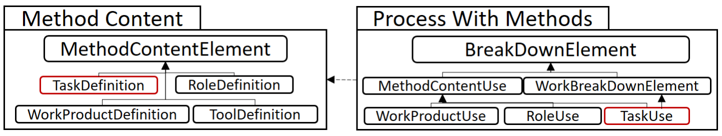
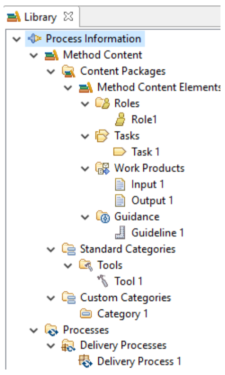
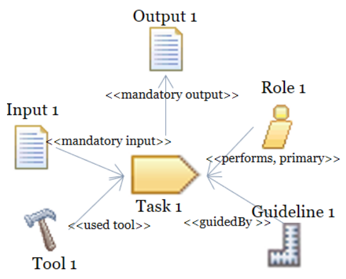


Figure 10.1: Partial Representation of SPEM 2.0 Taxonomy.

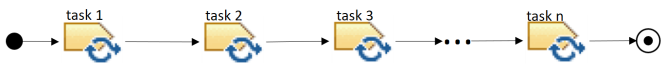
SPEM 2.0-like concepts can be modeled with an open-source tool, called Eclipse Process Framework Composer(EPF-C) [23]. In particular, EPF-C provides a *Method Authoring*, which is used to describe roles, tasks, work products, and guidance. EPF-C also has a *Process Authoring*, which is used to organize reusable process building blocks in the form of delivery processes. EPF-C implements the method plugin package, which defines capabilities for modularization and extensibility. Such plugins, which can contain libraries of method content and processes, are reusable (see Figure 10.2a). Conceptually, a task can be represented as a synergy between different process elements (see Figure 10.2b). In EPF-C, the process's partial execution semantics can be modeled with UML activity diagrams (see Figure 10.2c).



(a) EPF-C Plugin.



(b) The Task Concept.



(c) Activity Diagram Representing a Process Workflow.

Figure 10.2: EPF-C Environment.

### 10.2.3 FCL

FCL (Formal Contract Logic) [19] is a language that permits the formalization of normative requirements. An FCL rule has the form $a_1, ..., a_n \Rightarrow c$, where r is the unique identifier of the rule, $a_1, ..., a_n$ are the propositions that represent the conditions of the applicability of such a rule, and $c$ is the conclusion. The conclusion characterizes normative deontic effects, such as obligations, prohibitions, or permissions. FCL does not support contradictory conclusions but seeks to resolve conflicts. For instance, if it is sustainable support to conclude both c and $-c$, FCL does not conclude any of them. However, if the support for c has priority to the support of $-c$, then c is concluded. Thus, an FCL rules designer has to identify pairs of rules with incompatible literals and define superiority relations, as follows:

$$r : a_1, ..., a_n \Rightarrow c, \quad \text{and} \quad r' : b_1, ..., b_n \Rightarrow -c, \quad \text{then} \quad r' > r$$

Obligations and prohibitions are constraints that limit the behaviour of processes. As such, they can be violated. Permissions, which cannot be violated, can be used to determine that there are no obligations or prohibitions to the contrary. Hashmi et al. [54] proposes the foundations for the normative requirements that constraint processes, which considers different types of obligations (based on the temporal validity of norms and the effects of violating these obligations). Thus, an obligation is in force if the obligation is activated at a particular time point in a time interval. An obligation is considered to be non-persistent if it remains in force until it is terminated. Such obligation should be obeyed for the instant it is in force. In opposition, an obligation is considered persistent if it remains in force until it is removed. When a persistent obligation needs to be obeyed for the whole duration within the interval in which it is in force, it is called *maintenance obligation*. If achieving the content of the obligation at least once is enough to fulfill it, it is called *achievement obligation*. An achievement obligation is *preemptive* if it could be fulfilled even before the obligation is in force. Otherwise, it is *non-preemptive*. An achievement obligation is *perdurant* if, after being violated, the obligation is still required to be fulfilled. Otherwise is *non-perdurant*. A *prohibition* corresponds to the negation of the content of an achievement obligation. The types mentioned above are adopted, and notated in FCL, as presented in Table 10.2.

Table 10.2: FCL Rule Notations.

| Notation | Description |
|---|---|
| [P]P | A proposition P is permitted |
| [OM]P | There is a maintenance obligation for the proposition P |
| [OM]-P | There is a prohibition for proposition P |
| [OAPP]P | There is an achievement, preemptive, and non-perdurant obligation for the proposition P |
| [OANPP]P | There is an achievement, non-preemptive and perdurant obligation for the proposition P |
| [OAPNP]P | There is an achievement, preemptive and non-perdurant obligation for the proposition P |
| [OANPNP]P | There is an achievement, non-preemptive and non-perdurant obligation for proposition P |

### 10.2.4 Regorous

Regorous [20] is a process compliance checker that implements compliance by design [55], i.e., check requirements that are propagated into models of process plans. Regorous requires two specifications: 1) a rule base representing the regulation in FCL (recalled in Section 10.2.3), and 2) a state representation of a process, i.e., a process enriched with semantic annotations. Semantic annotations on process elements are literals that record data, resources, and other information used by machines to refer, compute, and align information. The recorded information, which represents the effects caused by the tasks, is used by Regorous to perform compliance analysis. Two types of semantic annotations are necessary. The first one is *State (t,n)*, which semantically annotates the set of facts in the computation to determine which rules fire (get active) for the n-th element in a trace t. A trace is a sequence of tasks in which a process can be executed. Consequently, obligations are in force after rules fire. The second one is *Force(t,n+1)*, which contains the obligations that are in force but are not terminated in n-th element in the trace t. An obligation can be terminated if the deadline is reached, the obligation has been fulfilled, or if the obligation has been violated and is not perdurant. A process is fully compliant if all obligations are fulfilled, or if violated, they are compensated). For example, Figure 10.3, shows a fictional FCL rule base and a compliance annotated process. As the figure depicts, the ruleset in FCL contains four rules. The first rule, r1, implies the obligation of providing A. The second rule, r2, implies the obligation of B given the provision of A. The third rule, r3, implies the permission to not provide C given the provision of B. And r4 implies the obligation of D given the provision of B. From the FCL rule base, we have four compliance effects, i.e., A, B, C, and D. As seen, the compliance effects are extracted from the formulas composing the rules. The tasks in the process are annotated with the effects as follows. T1 is annotated with effect A, T2 is annotated with effect B, T3 is annotated with effect C, and T4 is annotated with effect D. To check compliance, we use the functions *State* and *Force*, as previously described. The State of the start point is empty because we have not defined any effect. After the start point, the compliance checking process is activated. Thus, the first rule is in force. The first task is expected to provide the effect A since there is the obligation to provide A. Then, we check the State after the task T1. As we see in the figure, T1 produces the effect A. So, the rule is fulfilled. Then, providing A forces the provision of B in T2. In the figure, we can see that T2 provides effect B. So, the second rule is also fulfilled. After B is provided, it implies two normative effects. The first one is the permission to not providing C in T3. Second, it implies the obligation of providing D in T3. When checking T3, we can see that it provides the effect C. However, having C as the produced effect does not imply a violation of rule r3 because the force function has a permit, not an obligation. However, in T3, we should have D, and the tasks T3 is not providing E. If the obligation of providing D is a Maintenance or achievement preemptive, we have a violation. A violation means that the process is not compliant. If the obligation is achievement non-preemptive, it can be fulfilled in T4. In this case, there is no violation, and the process is compliant.

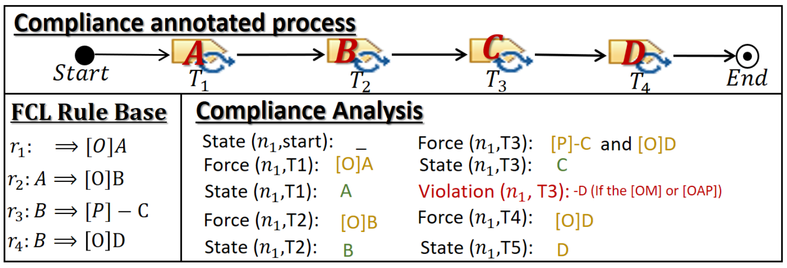


Figure 10.3: Analysis of Compliance.

## 10.2.5 Process Compliance Hints and Patterns

Skillful FCL ruleset design can be reached by applying computational thinking resources, in particular, design hints and patterns [39]. Hints are rules of thumb found in previous FCL formalization experiences, while patterns indicate common situations an FCL designer is likely to encounter. Both process compliance hints and patterns aim at facilitating the formalization of process-related requirements into FCL rules. In the remaining part of this section, we recall these resources in more detail.

### Process Compliance Hints

The divide-and-conquer strategy, adopted in software engineering as a principle to manage complexity [56], is a hint that can be applied in the formalization of process-related requirements, as presented in [57]. In particular, the aspects that requirements in standards regulate are the tasks, their specific order, the mandatory in/outputs of the tasks, roles performing the tasks, and the tools/recommended techniques 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 be qualified. 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. To facilitate the creation of compliance effects, which later can be used to form the propositions of the rules in FCL (recalled in section 10.2.3), two aspects are proposed (see Figure 10.4). The first aspect is the customization of icons, which describe the targeted elements. The second aspect is the definition of templates that facilitate compliance effects creation (fragments between {}, should be replaced by the specific element or its property). Both, icons and templates are based on the concepts described in SPEM 2.0 (recalled in Section 10.2.2, specifically in Figure 10.2). Once created, the compliance annotations are performed in the elements that carry out their compliance responsibility.

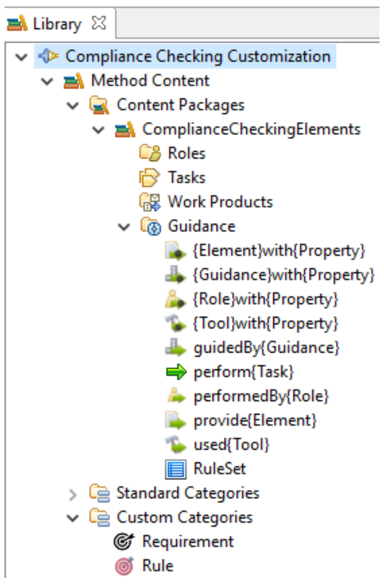

- Library
  - Compliance Checking Customization
    - Method Content
      - Content Packages
        - ComplianceCheckingElements
          - Roles
          - Tasks
          - Work Products
          - Guidance
            - {Element}with{Property}
            - {Guidance}with{Property}
            - {Role}with{Property}
            - {Tool}with{Property}
            - guidedBy{Guidance}
            - perform{Task}
            - performedBy{Role}
            - provide{Element}
            - used{Tool}
            - RuleSet
      - Standard Categories
      - Custom Categories
        - Requirement
        - Rule

Figure 10.4: Elements Customization.

**Process Compliance Patterns**

Process Compliance Patterns (PCP) [58] are commonly occurring normative requirements on the permissible state sequence of a finite state model of a process. The PCPs description is based on similar (or a combination of) behaviors described for the property specification patterns [59], which are mapped to the notations provided in FCL (recalled in Table 10.2). A global scope, which represents the entire process model execution, is defined as a [OM]P. A before scope, which includes the execution of the process model up to a given state, is mapped to a partial [OAP ]. An after scope, which includes the execution of the process model after a given state, is mapped to a partial [OANP ]. If an obligation admits an exception, e.g., tailoring, the part of the pattern corresponding to the exception is described as [P] since if something is permitted, the obligation to the contrary does not hold. The excepted obligation is modeled as non-perdurant, since the permission is not a violation of the obligation. Thus, the obligation does not persist after the permission is granted. In principle, all the requirements could be tailored. Thus, obligations are modeled as [OAPNP] or [OANPNP]. In this case, obligation and permission have contradictory conclusions, but the permission is superior since it represents an exception. Table 10.3 presents the templates of the PCPs. In all templates $\{\#\}$ should be replaced with the number that identifies the requirement in the standard. When it is described as $\{\#.i\}$, the *i* should be replaced by a, b, ..., n, where n is the number of sub-items, e.g., if there is a requirement with two parts that is identified with the number 5, the rules' identifiers are 5.a and 5.b. Following, we present a more detailed description of the patterns.

**Tailoring requirements (PCP 1a and 1b).** Tailoring means to adapt (omit or perform differently) the requirements to a specific project in a compliant form. Tailoring requires a rationale (or justification). For being valid, a rationale should always be verified by an expert. The rationale is an input element, and its verification is a property. An expert with specific qualifications should also be appointed. Thus, we use the templates for definitional and property-based propositions described in Figure 10.4 for in/output elements and roles, i.e., *provide*$\{Rationale\}$, $\{Rationale\}$*withVerificationByExpert*, *performedBy*$\{Expert\}$ and $\{Expert\}$*with*$\{Qualification\}$. Providing those four conditions permit to omit the requirement (in other words, permit not to perform the requirement). Any of the definitional and property-based propositions present in Figure 10.4 could be the target of such omission. For explanations purposes, we consider omitting a requirement that imposes the definition

Table 10.3: PCP Templates.

| **PCP** | **FCL notation** |
|---|---|
| 1a | **r$\{\#\}$.Omitted:** *provide*$\{Rationale\}, \{Rationale\}withVerificationByExpert\}$ , performedBy$\{Expert\}, \{Expert\}with\{Qualification\}$ $\Rightarrow [OANPP] - perform\{Task\}$ |
| 1b | **r$\{\#\}$.ChangedRule:** *-perform*$\{Task\} \Rightarrow [OANPNP]perform\{DifferentTask\}$ |
| 2 | **r**$\{\#.i\}$**:**$\{optionalTrigeringObligation\} \Rightarrow [OAPNP]provide\{prerequisite.i\}$ |
| 3a | **r**$\{\#\}$**:** provide$\{Prerequisite1\}..., provide\{Prerequisite.i\}$ $\Rightarrow [OAPNP]perform\{TitleClause\}$ |
| 3b | *perform*$\{Task\} \Rightarrow [OANPNP]perform\{FollowingTask\}$ |
| 4 | $\{\#.i\}$**:** perform$\{Task\}, \{Guidance\}with\{Property\}$ $\Rightarrow [OAPNP]guidedBy\{Guidance\}$ |
| 5 | **r**$\{\#.i\}$**:**$\{providePreviousObligations\}, \{WorkProduct\}with\{Property\}$ $\Rightarrow [OANPNP]provide\{WorkProduct\}$ |

of a task ($\Rightarrow [P] - perform\{Task\}$). In PCP 1a, $\{Rationale\}$ should be replaced with the title of the required justification. $\{Task\}$ should be replaced with the name of the task that will be omitted. Finally, $\{Expert\}$ should be replaced with the role required and $\{Qualification\}$ with the necessary qualifications. A second rule, i.e., PCP 1b, is included in case the task is done in a different way, where *[OANPNP]perform*$\{DifferentTask\}$ corresponds to the new task replacing the previous one.

**Provide a prerequisite (PCP 2).** A prerequisite is an obligatory input element, which should be fulfilled before it is in force. $\{prerequisite\}$ should be replaced with the name of the prerequisite. If a previous rule triggers the prerequisite, its conclusion is included in the $\{optionalTrigeringObligation\}$, e.g., when the prerequisite is produced by a previous task. Prerequisite could have properties. In this case, the $\{optionalTrigeringObligation\}$ could be a list of such properties, using the template $\{Element\}$*with*$\{Property\}$. Otherwise, it is left empty.

**Perform a unit of work (PCP 3a and 3b).** Template PCP 3a represents the performance of a unit of work that can be prescribed in a process (i.e., phase/activity/task). It considers the prerequisites, if any, as the conditions of the applicability of the rule, which normative conclusion is performing a unit of work (e.g., a phase). It could be preemptive ([OAPNP]), if the prerequisites and the task are provided at the same time. It can be non-preemtive ([OANPNP]) as in template 3b, if the prerequisite is another task, that have to be done first. In the example of PCP 3a, $\{TitleClause\}$ should be replaced with the specific clause title.

**Provide guidance (PCP 4).** Guidance elements may not be required during standards compliance auditing. However, internal policies in a company may impose guidance elements. In that case, guidance elements should be provided at the moment the element guided is created. We create the propositions by using the template for guidance provided in Figure 10.4, i.e., *guidedBy*$\{Guidance\}$ and $\{Guidance\}$*with*$\{Property\}$. Guidance can be defined for any element in the process (tasks, work product, tool, or role). For explanation purposes, we consider *perform*$\{Task\}$ (see PCP 4).

**Provide a work product (PCP 5).** Work products are the result of certain requirements. Thus, these requirements are presented as antecedents that oblige the provision of the related work product. PCP 5 presents this aspect in FCL, where $\{providePreviousObligations\}$ should be replaced with the conditions that oblige the work product's production, usually the execution of a task (*perform*$\{Task\}$). Work product properties may be also required, i.e., $\{WorkProduct\}$*with*$\{Property\}$, where $\{WorkProduct\}$ is the work product's name and $\{Property\}$ is the corresponding property.

### 10.2.6 ACCEPT

ACCEPT (Automatic Compliance Checking of Engineering Processes against sTandards) [60, 61], is a safety-centered planning-time framework aimed at facilitating the analysis of the tradeoffs associated with the planning of compliant processes in the safety-critical context. ACCEPT uses state-of-the-art tools and methodologies (see Figure 10.5).

In particular, ACCEPT is based on compliance by design [17], a preventive approach aimed at integrating compliance requirements into process plans. Such an approach requires the definition of two specifications. The first one is the FCL (recalled in Section 10.2.3) based standards requirements. FCL provides a framework that unambiguously represents the deontic notions required for compliance analysis. The second one is the process plan enriched with compliance effects, which is provided via SPEM 2.0-like artifacts in EPF-C (recalled in Section 10.2.2). SPEM 2.0 is flexible, i.e., concepts can be customized and extended to permit not only the creation artifacts related to processes but

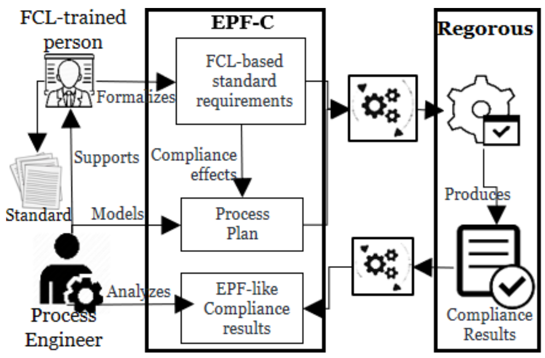


Figure 10.5: ACCEPT Framework.

also compliance checking artifacts, such as standard requirements, rules and annotated process plans. SPEM 2.0-like artifacts are also reusable since capabilities for modularization and extensibility are implemented in EPF-C (i.e., plugins). With the composition of EPF-C and the Base Variability Management Tool [26], tailoring of compliance artifacts and reuse is also facilitated. ACCEPT is equipped with guidance regarding process compliance hints and patterns (recalled in Section 10.2.5) that ease the creation of the required specifications. ACCEPT uses Regorous (recalled in Section 10.2.4), which provides a sound algorithm for the analysis of FCL rules that automatically check if a compliance-aware engineering process plan (CaEPP) is designed, i.e., if the elements set down by the requirements (e.g., tasks, personnel, work products, techniques, and tools, as well as their properties) are present at given points in the engineering process plan. The approach consists of five methodological steps, as shown in Figure 10.6.

**Step 1: Formalization of Requirements.** Standard requirements are formalized in FCL by and FCL-trained person supported by a process engineer (or an FCL-trained process engineer). Three inputs are required: the standard requirements, the compliance hints and patterns guidance, and the EPF-C plugin with the customized compliance checking artifacts (see Figure 10.4). First, the requirements should be classified in terms of the process elements they target and their properties to create the rules' propositions (see Section 10.2.5). Then, PCPs are used to create the FCL rules (see section 10.2.5). The output is an EPF-C plugin with the FCL-based ruleset containing information about the

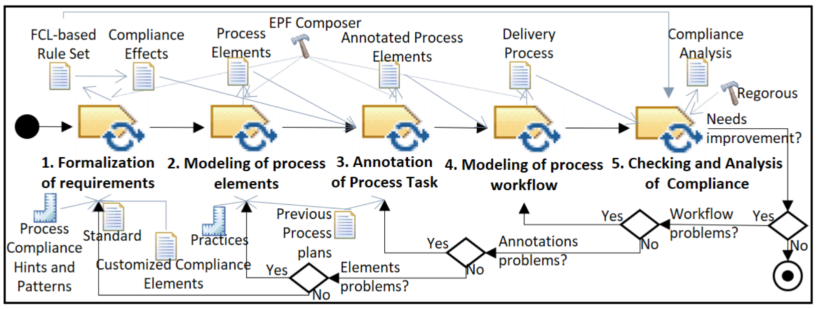

FCL-based Rule Set
Compliance Effects
Process Elements
EPF Composer
Annotated Process Elements
Delivery Process
Compliance Analysis
Regorous
Needs improvement?
1. Formalization of requirements
2. Modeling of process elements
3. Annotation of Process Task
4. Modeling of process workflow
5. Checking and Analysis of Compliance
Process Compliance Hints and Patterns
Standard
Customized Compliance Elements
Practices
Previous Process plans
Yes
No
Elements problems?
Annotations problems?
Workflow problems?

Figure 10.6: Methodological Steps Required for Using ACCEPT.

standard, their requirements, the rules derived from the requirements, and the separated set of propositions composing the rules.

**Step 2: Modeling of Process Elements.** Capturing process plan elements is a task performed by the process engineer. The required input is information about process plans, which could steam from the organization's practices and previous process plans. The output is the representation of the process elements in EPF-C, as depicted in Figure 10.2a. Detailed guidance regarding the creation of content elements in EPF-C is provided in [62]).

**Step 3: Annotation of Process Tasks.** The annotation process, which a process engineer performs manually, consists of assigning the compliance effects to the elements that fulfill them as presented in Figure 10.7. For this, the compliance effects modeled in the FCL ruleset (created in step 1) and the process elements (created in step 2), or previous process plans, are the inputs of this step. The output is the annotated process elements in EPF-C.

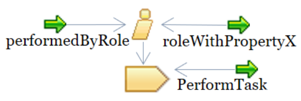


Figure 10.7: Annotation.

**Step 4: Modeling of Process Workflow.** The process engineer uses the compliance annotated process elements resulting from step 3 to model the workflow (see Figure 10.2c). The output is the delivery process in EPF-C, which contains the process plan checkable for compliance, i.e., the compliance state representation of the process plan.

**Step 5: Checking and Analysis** Checking and analyzing compliance is a task performed by the process engineer. The required inputs are the FCL-based ruleset and the delivery process. The output is the compliance analysis, which contains information regarding the rules violated by the process, their reparation policies, and the rules that were not activated during the compliance checking analysis. Such information is used to improve the process plans to be checked iteratively. Reasons for such improvements could be workflow problems (error in the placement of tasks), failure in the annotation process (errors in the assignment of the compliance effects), failure in the selection of process elements (e.g., missed elements), or FCL ruleset errors (not applicable rules due to tailoring or standards evolution).

## 10.3 Case Study Design

In this section, we present the essential details regarding the case study design.

### 10.3.1 Rationale for the Case Study

In the European context, space software production is often the result of industrial cooperation. Such cooperation is coordinated using the de-facto standard ECSS-E-ST-40C (recalled in Section 10.2.1). Such a standard provides requirements that help customers formulate their project-specific requirements by using the EARM matrix. Suppliers need to prepare their responses by using the ECM matrix, which will help them implement the work. ECSS-E-ST-40C is a process-related standard. Thus, the planning of software engineering processes in compliance with project-specific ECSS-E-ST-40C applicable requirements is mandatory during contractual agreements. Moreover, the tailoring decisions, i.e., A, M, D, and N, should be documented. Thus, we wonder if the current status of the models produced by ACCEPT could support space manufacturers' needs. For this, we perform a case study, according to the guidelines provided in [63]. In particular, we consider the selected portion of ECSS-E-ST-40C requirements related to the software items' design (recalled in Table 10.1). Our case study is descriptive (i.e., it portrays the current status of ACCEPT) and exploratory (i.e., it seeks future ACCEPT improvements). The data collected is qualitative involving models and their descriptions. The criteria used for the analysis is described in [33]. In particular, we analyze the *effort to model* (needed to establish a model for managing compliance), the *effort to comprehend* (processes and standards models), the *effort to document* *compliance*, (needed to verify whether process models comply with standards models), and the *effort to manage evolution*, (needed to find potential instances of non-compliance when standards change). Moreover, we take into account the *level of coverage for the model* (which shows how much of the requirements and engineering processes can be modeled), the *level of coverage for compliance documentation* (which examines the level of success of the approach in terms of documenting the compliance), and the *level of coverage for the evolution management* (which examines the approach's success in handling the changes).

## 10.3.2 Goal and Research Questions of the Case Study

As presented in Section 10.3.1, we want to analyze ACCEPT in the context of space software engineering processes planning in compliance with ECSS-E-ST-40C. We have also selected specific criteria, which essentially consider two variables: effort and coverage level. The effort, according to [64], is a variable that could be estimated during task performance in two ways: the actual effort (determined by task demands) and the perception of effort (relative to a subject's capacity to recognize the effort). In this case study, our analysis is based on actual effort (from now on called effort) since, in theory, it can be used to determine the intent to complete a task a priori, independently of any conscious actor. The coverage level is analyzed considering how the models respond to the information that needs to be required by the ECSS-E-ST-40C framework, i.e., information regarding standards, process plans, and compliance (i.e., EARM and ECM matrices). Thus, our goal is **to qualitatively analyze the current effort required to model a CaEPP in ACCEPT for software development processes in compliance with ECSS-E-ST-40C and the coverage level of such models**. Based on this goal, we derive the following research questions:

**RQ1: How could we consider the effort required in designing a CaEPP with ACCEPT for software development?** The answer of this question will be supported by answering the following subquestions:

**RQ1.1:** How could we consider the effort required to create models?

**RQ1.2:** How could we consider the effort required to comprehend the models?

**RQ1.3:** How could we consider the effort required to document compliance?

**RQ1.4:** How could we consider the effort required to manage evolution?

**RQ2: How could we consider the coverage level of a CaEPP for software development created with ACCEPT?** The answer of this question will be supported by answering the following subquestions:

**RQ2.1:** How could we consider the coverage level of the models?

**RQ2.2:** How could we consider the coverage level of the documentation?

**RQ2.3:** How could we consider the coverage level of the evolution management?

### 10.3.3 Unit of Analysis and Method

To support our goal (defined in Section 10.3.2), we model a CaEPP for space software engineering processes. The standard requirements involved in our models are the ECSS-E-ST-40C, focus on software design (recalled in Section 10.2.1). In general, ECSS-E-ST-40C determines mission-critical requirements that have an inherent relationship with safety issues since a software failure could lead to mission loss that could have catastrophic consequences. Thus, such requirements belong to the safety-critical context. The portion selected provides a view to the general structure of such a standard, i.e., prescribes requirements that impose the presence of process elements that can be tailored in a process plan. Thus, we consider such a portion representative of the whole standard. The method selected for conducting the case study is described in the steps required for facilitating automated compliance checking of engineering processes against standards (see Figure 10.6). Such a method permits us to collect the data required for the analysis.

### 10.3.4 Validity of the study

Case studies in software engineering are conducted to increase knowledge and bringing change in the studied phenomenon [63]. Researchers must consider issues that may diminish the results' trustworthiness by demonstrating the extent to which the researchers' subjectiveness does not bias the results. In this study, we consider a scheme of four aspects of validity in case studies in software engineering defined in [63]. 1) *Construct validity* reflects the extent to which the research represents the theoretical concepts used in the study. 2) *Internal validity* is of concern when causal relations are examined. 3) *External validity* addresses the ability of the research to be generalized. 4) *Reliability* is concerned with the extent to which the data and analysis are dependent on specific researchers. Addressing these four validity aspects is essential since it permits an accurate account of the research by selecting and using acceptable methodological practices that guarantee correct steps for collecting and analyzing the data. The concrete ways in which we addressed the mentioned validity aspects are listed below.

**Construct validity:** To avoid construct validity, we established a chain of evidence by rigorously following our defined methodology (see Figure 10.6), reporting the results consistently with such a methodology. However, designed methodologies may be biased. For mitigating this aspect, we review our assumptions against theoretical foundations several times during several sessions to avoid oversimplifications that may confirm our preconceptions. We also got external reviews in previous phases of our work, as well as initial stages of this case study. We use such reviews to improve our methodology, its presentation, and the definition of the case study itself.

**Internal validity:** The manual formalization of the FCL rules may imply a internal validity threat, due to the possibility of typos in the syntax and inconsistencies in the rules statements. For mitigating this aspect, we instantiated the process compliance hints and patterns (see Section 10.2.5) and performed manual syntactic corrections of the FCL specification. In the future, we plan to develop tools for supporting the process of writing and verifying rules.

**External validity:** We have performed an ACCEPT analysis on a limited portion of a software process standard. It is a single case study, but it is not trivial. It shows dilemmas and design choices that are typical in safety-related engineering process plans. In particular, ECSS-E-ST-40C determines mission-critical requirements that have an inherent relationship with safety issues since mission loss could lead to catastrophic consequences. Thus, they belong to the safety context. Moreover, the ECSS-E-ST-40C portion selected contains all the characteristics regarding process-based standards, i.e., the definition of work units, in/outputs, elements properties, and other process-related elements such as guidance, which have the possibility to be tailored. Such characteristics are presented in whole standard. Thus, the selected requirements are representative and can be generalized to the complete ECSS-E-ST-40C standard. However, the outcome of this case study applies to a CaEPP for software development that respond to the mentioned characteristics. Additional challenges may arise when analyzing standards beyond safety and software that also apply in the safety-critical context. Thus, to generalize our framework capabilities, we must carry out case studies beyond the ones we have already performed.

**Reliability:** Reliability threats were mitigated by involving the researchers in peer debriefing, i.e., iterative review of research artifacts (formalization of standard requirements, EPF-C models) during all the process.

## 10.4 Data Collection

In this section, we collect the data required for the case study.

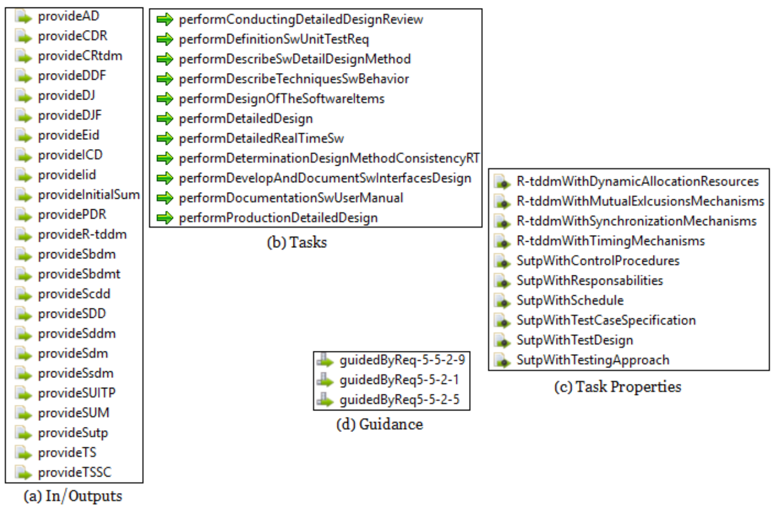
### 10.4.1 Formalization of ECSS-E-ST-40C requirements

As described in Section 10.2.6, we classify the requirements in terms of the process elements and their properties (see Table 10.4), and create the rules' propositions (see Figure 10.8) based on process compliance hints (see Section 10.2.5).

Table 10.4: Process Elements Required by ECSS-E-ST-40C.

| **Element** | **Description** |
|---|---|
| Inputs | TSSC, AD, DJ, PDR. |
| Outputs | Expected items of every requirement are: Scdd, Eid, Iid, Ssdm, Sddm, Sbdm, Sdm, R-tddm (with timing, synchronization and mutual exclusion mechanisms, and dynamic allocation of resources), Sbdmt CR-tdm, Sum, and Sutp (with responsibility, schedule, control procedures, testing approach, test design and test case specification). Final outputs: DDF, SDD, CDR, TS, ICD, SUM, DJF and SUITP. |
| Tasks | Phase Design of the software items which contains the following tasks (and their properties): 1) Detailed design of each software component, 2) Development and documentation of the software interfaces detailed design, 3) Production of the detailed design model, 4) Software detail design method, 5) Detailed design of real–time software, 6) Utilization of description techniques for the software behaviour, 7) Determination of design method consistency for real–time software, 8) Development and documentation of the software user manual, 9)Definition and documentation of the software unit test requirements and plan, and 10) Conducting a detailed design review |
| Guide | Guidance for Scdd (req-5-5-2-1), R-tddm (req-5-5-2-5), Sutp (req-5-5-2-9) |

The initial part of the ruleset defines the provision of requirements TSSC, AD, DJ, PDR, which are needed to start the activity described in Table 10.1. The PCP 2 is used to create the rules mandating the requirements (rules r5.5.2.a to r5.5.2.d) and PCP 3a to create the rule mandating the phase definition (rule



provideAD
provideCDR
provideCRtdm
provideDDF
provideDJ
provideDJF
provideEid
provideICD
provideId
provideInitialSum
providePDR
provideR-tddm
provideSbdm
provideSbdmt
provideScdd
provideSDD
provideSddm
provideSdm
provideSsdm
provideSUITP
provideSUM
provideSutp
provideTS
provideTSSC

(a) In/Outputs

performConductingDetailedDesignReview
performDefinitionSwUnitTestReq
performDescribeSwDetailDesignMethod
performDescribeTechniquesSwBehavior
performDesignOfTheSoftwareItems
performDetailedDesign
performDetailedRealTimeSw
performDeterminationDesignMethodConsistencyRT
performDevelopAndDocumentSwInterfacesDesign
performDocumentationSwUserManual
performProductionDetailedDesign

(b) Tasks

R-tddmWithDynamicAllocationResources
R-tddmWithMutualExlcusionsMechanisms
R-tddmWithSynchronizationMechanisms
R-tddmWithTimingMechanisms
SutpWithControlProcedures
SutpWithResponsabilities
SutpWithSchedule
SutpWithTestCaseSpecification
SutpWithTestDesign
SutpWithTestingApproach

(c) Task Properties

guidedByReq-5-5-2-9
guidedByReq5-5-2-1
guidedByReq5-5-2-5

(d) Guidance

Figure 10.8: Rules Propositions.

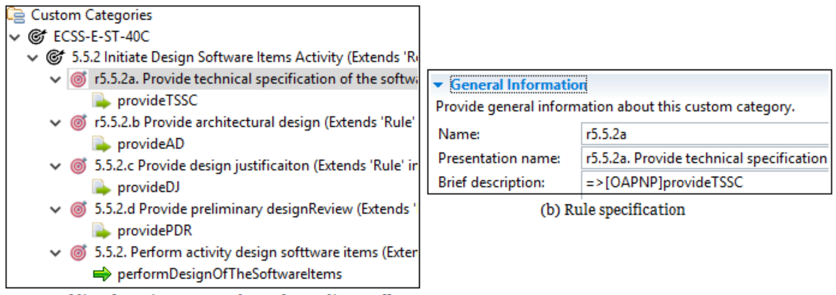
r5.5.2) as follows:

**r5.5.2.a:** $\Rightarrow [OAPNP]provideTSSC$
**r5.5.2.b:** $\Rightarrow [OAPNP]provideAD$
**r5.5.2.c:** $\Rightarrow [OAPNP]provideDJ$
**r5.5.2.d:** $\Rightarrow [OAPNP]providePDR$
**r5.5.2:**$provideTSSC, provideAD, provideDJ, providePDR$
$\Rightarrow [OAPNP]performDesignOfTheSoftwareItems$

We define a custom category in EFP-C (ECSS-E-ST-40C) to create the rule set. For each requirement, we nest a category. In each category, we nest the rule. We assign the compliance effect (the conclusion of the rule) to each rule (see Figure 10.9). All requirements and rules are modeled in a similar way.

We use PCP 3b formalize the requirements that define the first task *Detailed design of each software component* (see rule 5.5.2.1), which prerequisite is the activity definition (see rule 5.5.2). Then, we use PCP 5 to define the expected item (ei), which is the work product of this task (see rule r5.5.2.1.ei). Then we used the PCP 4 to define the guidance (see rule r5.5.2.1.guide).

**r5.5.2.1:**$performDesignOfTheSoftwareItems \Rightarrow [OANPNP]performDetailedDesign$
**r5.5.2.1.ei:**$performDetailedDesign \Rightarrow [OANPNP]provideScdd$
**r5.5.2.1.guide:**$performDetailedDesign \Rightarrow [OANPP]guidedByReq5-5-2-1$


Custom Categories
ECSS-E-ST-40C
5.5.2 Initiate Design Software Items Activity (Extends 'R
r5.5.2a. Provide technical specification of the softw
provideTSSC
r5.5.2.b Provide architectural design (Extends 'Rule'
provideAD
5.5.2.c Provide design justificaiton (Extends 'Rule' ir
provideDJ
5.5.2.d Provide preliminary designReview (Extends '
providePDR
5.5.2. Perform activity design softtware items (Exter
performDesignOfTheSoftwareItems

(a) Nested list of requirements, rules and compliance effects

General Information
Provide general information about this custom category.

| | |
|---|---|
| Name: | r5.5.2a |
| Presentation name: | r5.5.2a. Provide technical specification |
| Brief description: | =>[OAPNP]provideTSSC |

(b) Rule specification

Figure 10.9: Rules Definition.

Requisite 5.5.2.2 is the definition of the task *Development and documentation of the software interfaces detailed design*, which produces two expected items (ei) *Eid* and *Iid*. As two ei are created, we further identify the rules by adding a and b to the rules (see rules r5.5.2.2.ei.a and r5.5.2.2.ei.b).

$$\begin{aligned}
&\textbf{r5.5.2.2:} performDetailedDesign \\
&\Rightarrow [OANPNP]performDevelopAndDocumentSwInterfacesDesign \\
&\textbf{r5.5.2.2.ei.a:} performDevelopAndDocumentSwInterfacesDesign \\
&\Rightarrow [OANPNP]provideEid \\
&\textbf{r5.5.2.2.ei.b:} performDevelopAndDocumentSwInterfacesDesign \\
&\Rightarrow [OANPNP]provideEid
\end{aligned}$$

Requisite 5.5.2.3 is the definition of the task *Production of the detailed design model* (see rule r5.5.2.3) and three items are expected (see rules r5.5.2.3.ei.a, r5.5.2.3.ei.b and r5.5.2.3.ei.c).

$$\begin{aligned}
&\textbf{r5.5.2.3:} performDevelopAndDocumentSwInterfacesDesign \\
&\Rightarrow [OANPNP]performProductionDetailedDesign \\
&\textbf{r5.5.2.3.ei.a:} performProductionDetailedDesign \Rightarrow [OANPNP]provideSsdm \\
&\textbf{r5.5.2.3.ei.b:} performProductionDetailedDesign \Rightarrow [OANPNP]provideSddm \\
&\textbf{r5.5.2.3.ei.c:} performProductionDetailedDesign \Rightarrow [OANPNP]provideSbdm
\end{aligned}$$

Requisite 5.5.2.4 is the definition of the task *Software detail design method* (see rule r5.5.2.4) and one item is expected (see rule r5.5.2.4.ei).

$$\begin{aligned}
&\textbf{r5.5.2.4:} performProductionDetailedDesign \\
&\Rightarrow [OANPNP]performDescribeSwDetailDesignMethod \\
&\textbf{r5.5.2.ei:} performProductionDetailedDesign \Rightarrow [OANPNP]provideSdm
\end{aligned}$$

Requisite 5.5.2.5 is the definition of the task *Detailed design of real–time software* (see rule r5.5.2.5) and one item is expected . However, this expected item has several properties, which are included in the antecedent of the rule r5.5.2.5.ei. Additionally, guidance is defined for this task. So, we use PCP 4 to define the mandatory guidance (see rule r5.5.2.5.guide).

**r5.5.2.5:**$performDescribeSwDetailDesignMethod$
$\Rightarrow [OANPNP]performDetailedRealTimeSw$
**r5.5.2.5.ei:**$performDetailedRealTimeSw, R - tddmWithDynamicAllocationResources,$
$R - tddmWithMutualExlcusionsMechanisms,$
$R - tddmWithSynchronizationMechanisms, R - tddmWithTimingMechanisms$
$\Rightarrow [OANPNP]provideR - tddm$
**r5.5.2.5.guide:**$performDescribeSwDetailDesignMethod$
$\Rightarrow [OAPNP]guidedByReq5 - 5 - 2 - 5$

Requisite 5.5.2.6 is the definition of the task *Utilization of description techniques for the software behaviour* (see rule r5.5.2.6) and one item is expected (see rule r5.5.2.6.ei).

**r5.5.2.6:**$performDetailedRealTimeSw$
$\Rightarrow [OANPNP]performDescribeTechniquesSwBehavior$
**r5.5.2.6.ei:**$performDescribeTechniquesSwBehavior$
$\Rightarrow [OANPNP]provideSbdmt$

Requisite 5.5.2.7 is the definition of the task *Determination of design method consistency for real–time software* (see rule r5.5.2.7) and one item is expected (see rule r5.5.2.7.ei).

**r5.5.2.7:**$performDescribeTechniquesSwBehavior$
$\Rightarrow [OANPNP]performDeterminationDesignMethodConsistencyRT$
**r5.5.2.7.ei:**$performDeterminationDesignMethodConsistencyRT$
$\Rightarrow [OANPNP]provideCRtdm$

Requisite 5.5.2.8 is the definition of the task *Development and documentation of the software user manual* (see rule r5.5.2.8) and one item is expected (see rule r5.5.2.8.ei).

**r5.5.2.8:**$performDeterminationDesignMethodConsistencyRT$
$\Rightarrow [OANPNP]performDocumentationSwUserManual$
**r5.5.2.8.ei:**$performDocumentationSwUserManual$
$\Rightarrow [OANPNP]provideInitialSum$

Requisite 5.5.2.9 is the definition of the task *Definition and documentation of the software unit test requirements and plan* (see rule r5.5.2.9). One item, with several properties is expected (see rule r5.5.2.9.ei) as well as guidance (see rule r5.5.2.9.guide).

**r5.5.2.9:**$performDocumentationSwUserManual$
$\Rightarrow [OANPNP]performDefinitionSwUnitTestReq$
**r5.5.2.9.ei:**$performDefinitionSwUnitTestReq, SutpWithControlProcedures,$
$SutpWithResponsabilities, SutpWithSchedule, SutpWithTestCaseSpecification,$
$SutpWithTestDesign, SutpWithTestingApproach$
$\Rightarrow [OANPNP]provideSutp$
**r5.5.2.9.guide:**$performDocumentationSwUserManual$
$\Rightarrow [OAPNP]guidedByReq - 5 - 5 - 2 - 9$

Finally, requisite 5.5.2.10 is the definition of the task *Conducting a detailed design review* (see rule r5.5.2.10), Eight items are expected after this task (see rules r5.5.2.10.ei.a to r5.5.2.10.ei.h).

**r5.5.2.10:**$performDefinitionSwUnitTestReq$
$\Rightarrow [OANPNP]performConductingDetailedDesignReview$
**r5.5.2.10.ei.a:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideDDF$
**r5.5.2.10.ei.b:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideSDD$
**r5.5.2.10.ei.c:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideCDR$
**r5.5.2.10.ei.d:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideTS$
**r5.5.2.10.ei.e:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideICD$
**r5.5.2.10.ei.f:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideSUM$
**r5.5.2.10.ei.g:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideDJF$
**r5.5.2.10.ei.h:**$performConductingDetailedDesignReview \Rightarrow [OANPNP]provideSUITP$

Requirements tailored as omitted (not applicable (D) in the ECSS Applicability Requirements Matrix (EARM)) can be formalized using PCP 1a. For example, it is defined that requirement 5.5.2.10, which is the definition of the task *Conducting a detailed design review* is omitted (see rule r5.5.2.10.Ommited). If we do not perform the review, their work products are also not required.

**r5.5.2.10.Ommited:**$provideJustificationNotPerformConductingDetailedDesignReview,$
$JustNotPerformConductingDetailedDesignReviewWithVerificationByExpert,$
$performedByAssesor, AsessorWithExperienceECSS - E - ST - 40 - C$
$\Rightarrow [P] - performConductingDetailedDesignReview$
$r5.5.2.10.Ommited > r5.5.2.10$

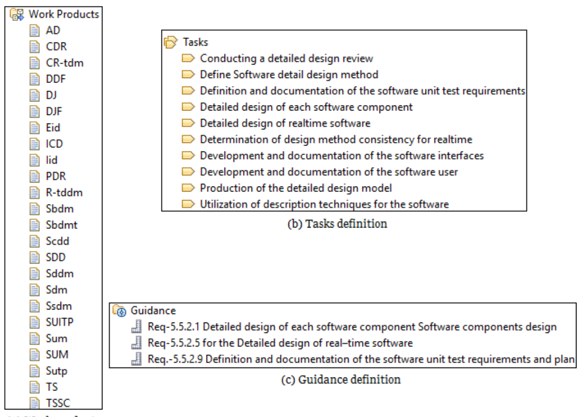
We use the PCP 1b, if the same task is defined in the EARM as applicable with modification (M), or new generated requirement (N). For illustration purpose, we consider that a simple review could be performed instead of the detailed review (see rule r5.5.2.10.ChangedRule).

**r5.5.2.10.ChangedRule:** $- performConductingDetailedDesignReview$
$\Rightarrow [OANPNP]performSimpleReview$

Propositions used in rules r5.5.2.10.Ommited and r5.5.2.10.ChangedRule should be added to the list of rule propositions presented in Figure 10.8.

## 10.4.2 Modeling of Process Elements

Initial process plan elements are extracted from the standard ECSS-E-ST-40C, specifically, the requirements presented in Table 10.1. The result is process elements depicted in Figure 10.10, which contains work products, tasks and guidance artifacts.



Work Products
- AD
- CDR
- CR-tdm
- DDF
- DJ
- DJF
- Eid
- ICD
- Iid
- PDR
- R-tddm
- Sbdm
- Sbdmt
- Scdd
- SDD
- Sddm
- Sdm
- Ssdm
- SUITP
- Sum
- SUM
- Sutp
- TS
- TSSC

(a) Work products definition

Tasks
- Conducting a detailed design review
- Define Software detail design method
- Definition and documentation of the software unit test requirements
- Detailed design of each software component
- Detailed design of realtime software
- Determination of design method consistency for realtime
- Development and documentation of the software interfaces
- Development and documentation of the software user
- Production of the detailed design model
- Utilization of description techniques for the software

(b) Tasks definition

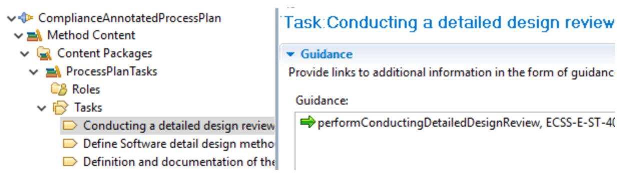
Guidance
- Req-5.5.2.1 Detailed design of each software component Software components design
- Req-5.5.2.5 for the Detailed design of real–time software
- Req.-5.5.2.9 Definition and documentation of the software unit test requirements and plan

(c) Guidance definition

Figure 10.10: Process Elements Plugin.

## 10.4.3 Annotation of Process Tasks

A copy of the process elements defined in Section 10.4.2 (depicted in Figure 10.10) is created in a new plugin, which we called *ComplianceAnnotatedProcessPlan*. These process elements are also extended to the original by using the content variability type *Extends*. With this extension, we ensure that the information previously defined is also included, such as the assignment of in/output or guidance to the tasks. Then, every process element is annotated with the compliance effect they produce by adding the guidance elements that contains the effect (see Figure 10.11). The complete compliance annotation of process elements is presented in Table 10.5.



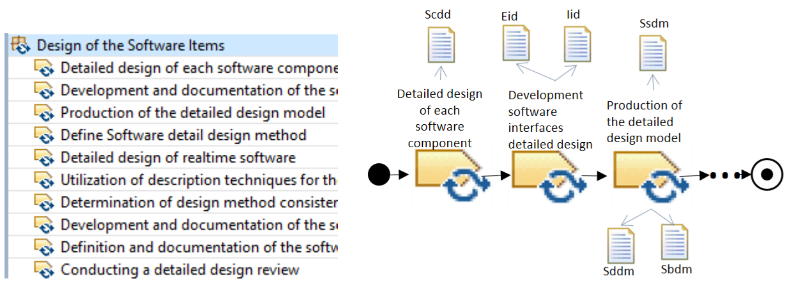
ComplianceAnnotatedProcessPlan
Method Content
Content Packages
ProcessPlanTasks
Roles
Tasks
Conducting a detailed design review
Define Software detail design metho
Definition and documentation of the

Task:Conducting a detailed design review
Guidance
Provide links to additional information in the form of guidanc
Guidance:
performConductingDetailedDesignReview, ECSS-E-ST-4(

Figure 10.11: Annotation of Tasks.

## 10.4.4 Modeling of Process Workflow

The tasks annotated in Section 10.4.3 are used to create the delivery process (see Figure 10.12).



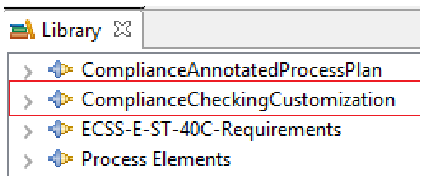
Design of the Software Items
Detailed design of each software compone
Development and documentation of the s
Production of the detailed design model
Define Software detail design method
Detailed design of realtime software
Utilization of description techniques for th
Determination of design method consister
Development and documentation of the s
Definition and documentation of the softw
Conducting a detailed design review

Scdd
Eid
Iid
Ssdm
Detailed design of each software component
Development software interfaces detailed design
Production of the detailed design model
Sddm
Sbdm

(a) Breakdown Structure. (b) Activity Siagram.

Figure 10.12: Delivery Process.

As depicted in Figure 10.13, ACCEPT involves the modeling of four separated models, which are concretized in EPF Composer as plug-ins. The *ComplianceCheckingCustomization* (highlighted in red in the figure) is provided in the method and the process engineer only needs to use it.



Library
ComplianceAnnotatedProcessPlan
ComplianceCheckingCustomization
ECSS-E-ST-40C-Requirements
Process Elements

Figure 10.13: EPF-C Plugins.

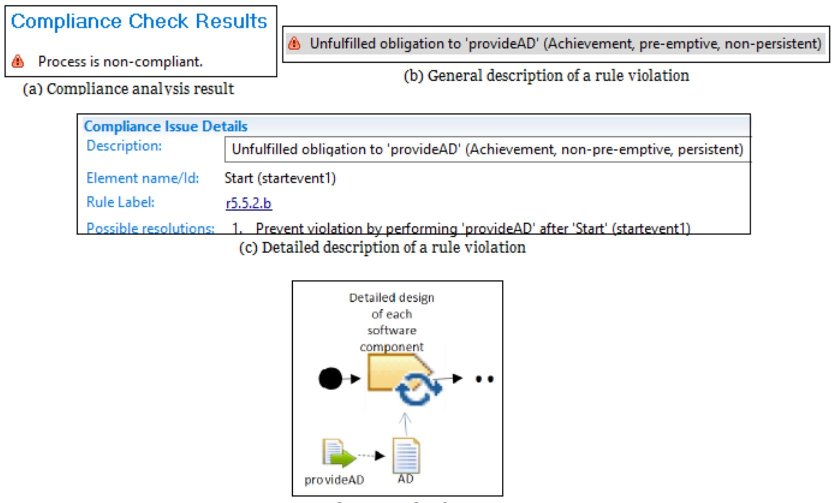
Table 10.5: Compliance Effects Annotation on Process Elements.

| **Process element** | **Compliance Effect** |
|---|---|
| Detailed design of each software component | performDetailedDesign |
| Scdd | provideScdd |
| Development/documentation of the software interfaces | performDevelopAndDocumentSwInterfacesDesign |
| Eid | provideEid |
| Iid | provideIid |
| Production of the detailed design model | performProductionDetailedDesign |
| Ssdm | provideSsdm |
| Sddm | provideSddm |
| Sbdm | provideSbdm |
| Define Software detail design method | performDescribeSwDetailDesignMethod |
| Sdm | provideSdm |
| Detailed design of real-time software | performDetailedRealTimeSw |
| R-tddm | provideR-tddm |
| Utilization of description techniques for the software | performDescribeTechniquesSwBehavior |
| Sbdmt | provideSbdmt |
| Determination of design method consistency for realtime | performDeterminationDesignMethodConsistencyRT |
| CRtdm | provideCRtdm |
| Development and documentation of the software user manual | performDocumentationSwUserManual |
| Sum | provideInitialSum |
| Definition and documentation of the software unit test requirements and plan | performDefinitionSwUnitTestReq |
| Sutp | provideSutp |
| Conducting a detailed design review | performConductingDetailedDesignReview |
| Req-5.5.2.1 Detailed design of each software component Software components design | guidedByReq5-5-2-1 |
| Req-5.5.2.5 for the Detailed design of real–time software | guidedByReq5-5-2-5 |
| Req.-5.5.2.9 Definition and documentation of the software unit test requirements and plan | guidedByReq-5-5-2-9 |

## 10.4.5 Checking and Analysis of Compliance

The ruleset formalized in Section 10.4.1, and the workflow modeled in Section 10.4.4 (located in the plugins *ECSS-E-ST-40C-Requirements* and *Compli-*

*anceAnnotatedProcessPlan* in Figure 10.13, respectively) are the two specifications required by Regorous (recalled in Section 10.2.4) to perform automatic compliance checking. The results of the checking are presented in Figure 10.14.


Compliance Check Results

Process is non-compliant.

(a) Compliance analysis result

Unfulfilled obligation to 'provideAD' (Achievement, pre-emptive, non-persistent)

(b) General description of a rule violation

| Compliance Issue Details | |
|---|---|
| Description: | Unfulfilled obligation to 'provideAD' (Achievement, non-pre-emptive, persistent) |
| Element name/Id: | Start (startevent1) |
| Rule Label: | r5.5.2.b |
| Possible resolutions: | 1. Prevent violation by performing 'provideAD' after 'Start' (startevent1) |

(c) Detailed description of a rule violation

(d) Improved task

Figure 10.14: Compliance Checking Results.

As Figure 10.14(a) depicts, *the process is non-compliant*. Thus, the plan needs improvement. Such results could point to compliance problems on the workflow, in the annotation process, or missing characteristics in the process plans (e.g., absent tasks or work products). The problems related to the rules (e.g., wrong formalization) need to be analyzed in the context of specific standard with experts, such as safety assessors. For example, there is a violation regarding provideAD (see Figure 10.14(b)). It turns out that rule r5.5.2.b is violated (see Figure 10.14(c)). The reparation policy suggests to *prevent the violation, by performing provideAD after 'Start'*. This means that in the first task, which is called *detailed design of each software component*, we need to add the input *AD* and its corresponding compliance effect *provideAD* (see Figure 10.14(d)). When the compliance checking results are positive, namely the result is *the process is compliant*, it does not mean that there is no further improvement. Instead, we need to perform the analysis, taking into account the rules that did not fire. Once the process is improved for compliance, compliance checking is performed iteratively until the process plan is deemed compliant by Regorous.

## 10.5 Case Study Analysis

In this section, we analyse the case study results presented in Section 10.4 by answering the research questions defined in Section 10.3.2.

### 10.5.1 Effort designing a CAEPP for software development (RQ1)

We judge the effort determined by task demand, which is based on design choices. In theory, such effort can be used to determine the intent to complete a task independently of any conscious actor. We refer in this analysis to the effort required to create and comprehend the models and document and manage evolution.

#### Effort required to create the models (RQ1.1)

When using ACCEPT for creating a CaEPP for software projects, three models (the *ComplianceCheckingCustomization* is provided in the method), which are concretized in EPF-C as plugins, are required (see Figure 10.13). To populate the *ECSS-E-ST-40C-Requirements* plugin, we needed to formalize requirements in FCL. Performing a formalization process is, in general, a difficult task that requires skills, which cannot be taken for granted. Moreover, due to the sheer size of the standards, this work requires time and focus. An advantage of FCL is that it provides a valid set of understandable concepts to the users of the standards, i.e., obligations, prohibitions, and permission (see Section 10.2.3). Moreover, ACCEPT provides process compliance hints and patterns (recalled in Section 10.2.5), which facilitate the identification and modeling of compliance artifacts, i.e., requirements and their corresponding rules as well as compliance effects (see Section 10.4.1). However, the instantiation of hints and patterns is done manually in a process that is repetitive and prone-to-error. Defining the *ProcessElements plugin* required in a process is a task that is not difficult to perform since EPF-C has graphical representations of the elements that can be modeled in a well-defined structure (see Section 10.4.2). However, a specific effort in terms of time is also required. ACCEPT states that the elements required in the *ComplianceAnnotatedProcessPlan* plugin are linked to the elements in other models in two ways (see Section 10.4.3). 1) By performing extensions between elements in the method content, to inherit the characteristics of the process elements. 2) By performing the compliance annotation process, which is the method that guarantees that the model is checkable for compliance. Currently, compliance annotations are performed manually, based on the domain expert's knowledge about the engineering process.

In summary, there is a need to manually (and iteratively) formalize requirements, graphically model compliance and process artifacts, and extend and annotate compliance effects. Thus, the effort required to create models is significant and may increase with the continued attempting to do the same tasks repeatedly. However, the effort required to model the *ECSS-E-ST-40C-Requirements* plugin is only significant during the first time. The reason is that such a model can be used several times in different CaEPPs that are modeled in compliance with the same standard (until new versions of the standard are released). Similar situations could occur with the other plugins, but they need to be evaluated in project-specific circumstances.

#### Effort required to comprehend processes and standards models (RQ1.2)

The method uses artifacts that are systematically organized in a hierarchical and visual structure that permits the identification of compliance information. In particular, standard requirements and their elements are arranged in a nested list of compliance artifacts (see Figure 10.9(a)). Moreover, process elements are created in particular structures that differentiate, e.g., work products from tasks (see Figure 10.10). The abstract association of elements within process tasks (depicted in Figure 10.2b) permits the comprehension of the required compliance information provided by the compliance effects. This abstraction provides an approach for direct requirements allocation into process models. Thus, once the models are created, there is a required low effort to comprehend the information they contain. In summary, the models created in EPF-C have a specific structure that facilitates the visualization of their artifacts and their use, proving models that have an advantage over, e.g., text-based approaches.

#### Effort required to document compliance (RQ1.3)

The ability to provide means to document method content and processes in SPEM 2.0-like elements was exploited in ACCEPT. Having an structural, hierarchical representation of the standards (see Figure 10.9(a)) with descriptive information (see (see Figure 10.9(b)), as well as content elements organized according to their function (see Figure 10.10) helps to have a written record of the artifacts required for compliance. This structural representation, originally provided by SPEM 2.0, may facilitate the work of a third party (independent) assessor in case the parties decide to include additional certainty to their assessment schema. Thus, the required high modeling effort results in lower compliance documentation effort.

**Effort required to manage evolution (RQ1.4)**

The compliance information in ACCEPT created in Section 10.4.1 could be seen as an initial frozen specification of the standard. However, such specification does not need to be bypassed altogether, when obsolescence no longer stands the strain of being frozen, i.e., a new version of the standard is released. In normal conditions, only some requirements change, and some get deprecated, but the majority remain. For example, the ECSS-E-ST-40-C has a log, which explicitly describes few adjustments regarding previous versions (ECSS-E-40 Part 1B, released on 28 November 2003, and the ECSS-E-40 Part 2B, released on 31 March 2005). In ACCEPT, such changes can be embraced. First, as specifications are reusable, a copy of the requirements can be performed and saved with the new version name. Second, as the requirements model is hierarchically designed, the changes can be absorbed in an orderly way. Once a new version of the standards is defined, changes in the rules may impact the compliance status. Thus, it becomes necessary to re-check the process plans. However, as the rulesets are executable, the checking is easier, and process plans can be improved according to the new version of the ruleset (as described in Section 10.4.5). However, standards evolution would require some human intervention. In particular, there is a need for monitoring the changes in the standards and maintain accuracy in the rulesets. EFP-C provides textual descriptions regarding, i.e., versioning or revisions, which can be used to maintain a log of information between users facilitating further revision work.

### 10.5.2 Coverage level of a CaEPP for software development (RQ2)

We judge the models' coverage level, taking into account how the information provided by the CaEPP models fit in the information required by the ECSS-E-ST-40C framework. We refer to the coverage level of the models, the compliance documentation, and the evolution management.

### Level of coverage of the models (RQ2.1)

The models used in ACCEPT cover several aspects required in process compliance. First, standard artifacts (see Figure 10.9(a)) are represented by a structure that covers the textual description of the requirements, their respective FCL rules, and compliance effects. Second, the method content provided (see Figure 10.10) covers the elements required to describe detailed process plans. Third, the compliance effects annotation (see Figure 10.11) covers the requirements allocating into process plans, which permits us to understand the explicit relationships between artifacts. Finally, the compliance analysis provided by Regorous (see Figure 10.14) covers the compliance status of the process plan, the compliance violations, and possible resolutions that facilitate the compliance analysis.

### Level of coverage of the compliance documentation (RQ2.2)

As previously described, the models provide all the required information for documenting compliance. Moreover, the compliance analysis is detailed enough to describe compliance status (full or non-compliance), the compliance violations, and the inactive rules. For our particular case study, this approach is sufficient. On the one side, as recalled in Section 10.2.1, a customer of a space software project needs to provide an ECSS Applicability Requirements Matrix (EARM), which can be extracted from the model that contain the requirements, i.e., the ECSS-E-ST-40C-Requirements plugin (see Figure 10.9(a)). As the figure depicts, the plugin contains the requirements identifier and the text. Moreover, the applicability status can be obtained from the description of the identifier in the rules. For example, tailored out requirements are identified with the particle *Ommited* (see rule r5.5.2.10.Ommited). In contrast, modified ones are identified with the particle *ChangedRule* (see rule r5.5.2.10.ChangedRule). On the other side, the supplier needs to respond with the ECSS Compliance Matrix (ECM), which should be done at the level of each requirement (as opposed to a global statement of compliance) in order to allow the customer to detect early enough in the project the non or partial compliance. The information required in the ECM can be extracted from the compliance checking results (see Figure 10.14(b) and (c)). Such results will identify compliance violations to the rules that belong to the requirements explicitly defined by the customer in the EARM. An analysis of the violations may lead to modification of the requirements upon agreements between the parties when compliance has become excessively demanding or unreachable (due to unpredictable or changing conditions in the project).

**Level of coverage of the evolution management (RQ2.3)**

ACCEPT is defined in an authoring platform that permits the organization the compliance information as standards evolve. Thus, successive models that represent the evolution of the standards can be defined and stored in EPF-C plugins as a library of reusable knowledge (see Figure 10.13). Administrative directives from the organization that apply the standards could also be defined and included as FCL rules (as presented in Section 10.4.1). Consequently, the process engineer, whose expertise may be limited by specific knowledge, could find the hints that facilitate applying the specific standard version. Moreover, the process engineer could also include his/her knowledge (or lessons learned) after performing compliance practices, as part of the documentation that is permitted by EPF-C.

## 10.6 Discussion

As presented in Section 10.2.1, a software process engineer is responsible for the composition and documentation of compliant software engineering processes plans. In general, the planning of engineering processes, which criteria could be initially abstracted from ad-hoc practices, needs to be concretized to support manufacturers in achieving goals. Specifically, in the space context, baseline criteria for software process planning are defined by the de-facto standard ECSS-E-ST-40C (recalled in Section 10.2.1). ECSS-E-ST-40C proposes reference models that prescribe artifacts related to planning activities, i.e., a set of units of work necessary to engineer systems. ECSS-E-ST-40C also contains process-related requirements, which prescribe properties for the activities, e.g., the prerequisites for performing activities, the work products to be produced, and specific guidance (see Table 10.4). Guidance elements may not be required for compliance auditing. However, internal policies in a company may impose the need to have guidance that facilitates the process's execution. All those requirements need to be specified in the project-related documents, e.g., the EARM, after careful selection by the customer. The requirements specification should contain the definition of one party's obligations towards the other and the authorization from the customer to the supplier to deviate from the standard requirements. The specification of the customer's requirements is an input for software project-specific contractual agreements with the supplier, who use them to define a CaEPP to perform the job. Thus, for defining contractual obligations regarding software projects, the discussions about the technical specifications based on the requirements baseline provided by ECSS-E-ST-40C must be carried out early in the lifecycle process. Selected requirements must be correctly adopted in the software engineering process plan. Otherwise, they may constitute a legal cause of action for breaching the contractual agreements.

Manually checking software process plans compliance with the EARM requirements is a common practice in this context. Indeed, the ECSS secretariat provides the EARM matrix with all requirements [2] for that purpose. Filled checklists highlight specific defects in the process (e.g., missed tasks) respect the defined requirements, which could be the source of compliance risks (as well as legal risks) during process execution. Besides, these checklists provide hints to improve processes performance and re-negotiate requirements if full compliance has become too demanding or unnecessary for the specific project. However, performing manual checks could be overwhelming. In particular, the knowledge included in ECSS-E-ST-40C is abundant (656 requirements), and their complexity (there are connections between different requirements and standards) have a direct implication in the correctness of the resulting process plans, i.e., the sequencing of process tasks and the definition of the properties of such tasks. Moreover, standards evolve (new versions are frequently released). Extensive process plans, which typically have a high number of states and transitions, are difficult to verify against industry standards' changing nature. Thus, the lack of methodological support for dealing with compliance management could involve unstructured practices, uncertain outcomes, compliance, and legal risks. Due to the fact that we are performing a single-case study, no firm conclusions should be done for the results. However, the data collected (see Section 10.4) and its analysis results (see Section 10.5) can be used as indications to guide the shaping of future designs and prototypes. In the remaining part of this section, we present a specific discussion regarding the case study insights as well as the challenges and potential improvements that could be done to enhance ACCEPT.

### 10.6.1 Case study insights

When planning a software engineering process plan, the challenge is to understand how many process elements should be specified and their order. In the case study conducted, we defined a model of a software process plan by the book, i.e., we extract the process elements suggested by the selected portion of ECSS-E-ST-40C standard without any tailoring. It is called CaEPP since such process elements are enriched with compliance information. In case of deviation (e.g., tailoring), we can also know if the requirements are tailored out or

[2]https://ecss.nl/standards/downloads/doors-download/

modified (as done for rules r5.5.2.10.Ommited and r5.5.2.10.ChangedRule in Section 10.4.1). ACCEPT states that creating a CaEPP requires several models, which design is a process that is not free of effort, as presented in Section 10.5.1. However, initial observations have shown that the effort required to comprehend processes and standards models (see Section 10.5.2) and document models (see Section 10.5.1) is significantly less. The reason is that formal specifications are accompanied by informal explanations that clarify their meaning and place them in context. Moreover, the visual approach adopted allows for more focused reviews. It is clear that organizations may depart from normative practices (not creation process plans by the book) for project-inherent reasons that can be justified. In such cases, logic-based requirements representations can be effortlessly superseded (as analyzed in Section 10.5.1). In addition, the level of coverage of the models is higher (as analyzed in Section 10.5.2). Thus, we can take good advantage of such an initial effort in the long term. Specifically, the models are created in an authoring environment that permits a well-defined organization of compliance-related artifacts in a hierarchical, visual, and enriched structure, which can be reused. This modeling strategy minimizes the distance between the specification of the requirements' normative intention and the process elements that should respond to such requirements. We could also include the possible exceptions that are derived from the deviations. Once the models are created, the process plans' validity can be established by doing automatic reasoning about the standard conditions. In particular, compliance violations could be drafted better since failure to requirements is connected to textual sources. Therefore, the comprehension of processes, standards, and their relationships is more natural, and the documentation of compliance and the management of evolution get better support than in manual checklists. These features are a valuable gain since once industry standards and process plans are formalized, process engineers do not need to expend valuable hours on reading regulatory documentation to infer the actions that must be taken to maintain compliance.

### 10.6.2 Challenges and Potential Improvements

A key challenge in the use of ACCEPT is that standards are currently written in natural language, and formalizing them is an intimidating and fairly sophisticated task. The reason is that the number of requirements in a standard is significant and context-specific. Thus, their interpretation requires expertise. However, FCL has a limited set of constructs, which provide the expressivity required for formalizing requirements. Such constructs also provide a framework for thinking about the requirements in terms of deontic notions and exceptions, which could simplify their interpretation. Therefore, showing process engineers the FCL potential and its easy to use aspect may strive the interest for its exploitation.

The work to be done when creating a CaEPP for software projects in the space context has the tendency to be repetitive. Repetition could cause a drop in a subject's capacity to perform the modeling task (i.e., disinterest, boredom, fatigue), making relative task demands greater than necessary. Further automation of such tasks might reduce the absolute demands, and thus the actual effort. For example, the manual creation of the compliance effects is a repetitive task that has to be done for each effect. In this case, we repeated this task 48 times (see Figure 10.8). It was also prone to error since the effects' names have similarities (e.g., almost all the tasks' effects have the word design). Thus, we needed to review our design several times and manually track the information we were writing in EPF-C. However, this task is systematic and supported by templates. As such, it could be automatized by using a domain-specific language that permits an adequate characterization of the specific compliance effects and their production.

In general, different mechanisms can be defined to determine the meaning of context-dependent situations that could affect rules' formalization. In particular, patterns facilitate the recognition of relevant requirements, improving efficiency and consistency when producing rules. Our current selection of compliance patterns is limited to general situations, and they are also manually instantiated. Still, they can provide some assistance. Moreover, the process compliance hints could be used to establish conceptual relationships between the elements composing process plans and their compliance effects in the general compliance status. Thus, automated formalization of requirements could also be provided by performing an intermediate translation step into controlled English. For the compliance annotation of processes, programming scripts that examine the semantic similarity between process elements and compliance effects can be created. The modeling part could also be facilitated by providing general-purpose process model repositories to process engineers. Indeed, EPF-C offers such kind of repositories with libraries that can be downloaded and assemble in specific projects [3].

[3]https://www.eclipse.org/epf/downloads/praclib/praclib_downloads.php

## 10.7 Related Work

ECSS standards are difficult to manage since they involve hundreds of pages containing around 25.000 requirements for the development and operations of European Space Systems. These standards are available in the form of documents (Word and PDF). In an effort for helping organizations, The European Space Agency (ESA) [4] provides an Excel document that contains the ECSS Applicability Requirement Matrix (EARM), which is useful for selecting requirements and document tailoring procedures. However, process engineers need to check the applicable requirements one by one. In addition, the data model requirements are specified in the ECSS digital Requirements Management System (E-RMS) conceptual data model [65], which guarantees the persistence of the ECSS requirements, but it does not have facilities for process modeling processes. A more sophisticated approach is presented in [66], which proposes a persistent connection via relational databases to word processor documents that contain the work products required in the standard. With this approach, compliance checking results may be incomplete, as not all activities produce work products, leaving mandatory activities out of the checking scope. In contrast, ACCEPT provides compliance checking of the process workflow, which not only takes into account the results of the tasks (e.g., work products). For this reason, it is more useful at planning stages. Moreover, ACCEPT is not only domain-specific as the ESA Excel document. In principle, any process-based requirements catalog can be formalized, uploaded and applied to a variety of safety-critical related software process plans.

Compliance-related artifacts modeling has also been the target of some research efforts. For example, in [67], the authors provide a model of the process concepts, via the UML [22] class diagrams. Automated rule checking with OCL (Object Constraint Language) constructs [5] is also suggested (but not implemented). In [68], the authors introduce SafetyMet. SafetyMet is a generic metamodel that includes the concepts and relationships common to different safety standards and project practices. With SafetyMet, mapping standards models and project information is also possible. In [69], the authors describe a SPEM 2.0 extension, which incorporates process requirements, guidelines, and their properties. The extension is used to generate an ontological representation that can be visualized with the Semantic Media Wiki (SMW) [6]. In contrast to the work presented in [67, 68, 69], we consider SPEM 2.0 (with-

[4]https://ecss.nl/

[5]https://wiki.eclipse.org/OCL

[6]https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki

out performing any extension), an Object Management Group specification [7] that is well-documented, mature, and open, and permits to model not only the processes and their related library but also the artifacts required for performing compliance checking.

Logic-based approaches have also provided a suitable framework to represent and reasoning upon normative knowledge. In [70], the authors propose a document schema specification in UML. The properties of the documents prescribed by the standards are formalized in first-order logic (FOL). Checks are performed when there is an attempt to read/write documents during process enactment. FOL can express property specifications, but it is insufficient to express the sequence of tasks in a process plan. In [71], the authors propose an approach for compliance checking based on temporal logic. However, it does not build an operational model of the process from the beginning. Instead, it needs the extraction of knowledge from event logs provided by the systems to create traces. Traces only contain tasks, which means that other process elements are not explicitly defined. As in [70], this approach only detects uncompliant states in the process execution. Moreover, the explicit definition of process elements beyond tasks is not possible. In [72], the authors present a framework based on Natural Language Semantics and Natural Language Processing techniques for recognizing correlations between provisions in a standard and requirements in a given law. However, it does not provide logic constructs to represent process plans, which is also part or our work. Our approach, as in [70, 71, 72], uses logical-based approaches for the formulation of requirements constraints. However, it is process-centered, planning-time, which means that a process and its elements are essential inputs.

Compliance checking by design is approached in the safety-critical context. In [73], the authors propose the comparison between an initial model of the process lifecycle prescribed by the standards and the plan provided by the users. The compliance checking is the result of the matching between the two process-based models. In contrast to the approach provided by [73], other approaches provide a comparison between the process and the requirements specification. For example, the authors in [74] propose a framework that uses Linear Temporal Logic to model a specification of the reference model provided by a standard. This specification is used to check the model of a SPEM 2.0 process. The work presented in [75, 76] aims at facilitating the checking of constraints by using SWRL (Semantic Web Rule Language) [77] on a process defined in SPEM 2.0. In [78], the authors present an approach for representing

[7]https://www.omg.org/spec/

SPEM 2.0 process models in Description Logics, to provide process analysis such as reasoning and consistency checks. ACCEPT has similarities with the approaches provided in the previous works [74, 75, 76]. First, it considers that the model of the norms and the model of the process are necessary inputs for compliance checking. Second, processes are modeled with SPEM 2.0-like artifacts. However, ACCEPT uses customized SPEM 2.0-like artifacts to provide visual relationships between compliance artifacts. Moreover, ACCEPT uses FCL, which is a deontic language able to directly provide normative notions, i.e., obligation, prohibition, and permission, without the need for combining expressions. Moreover, FCL, which is also a defeasible logic, is capable of providing the management of the tailoring rules.

## 10.8 Conclusions and Future Work

ACCEPT is a framework based on a proactive strategy called compliance-by-design that permits process engineers to create compliance-aware engineering process plans (CaEPP). A CaEPP can show the planning-time allocation of standard demands, i.e., if the elements set down by the standard requirements are present at given points in the engineering process plan. A CaEPP avoids that process engineers experience the tasks related to process compliance management as reactive, i.e., it provides a risk control mechanism at planning-time that facilitates the decision-making process. Thus, a CaEPP could increase confidence in process compliance, which at the same time could reduce liability in case of an adverse event occurs. This situation is essential in the safety-critical context since the duty of care and standards compliance are typically linked together. In this paper, we performed a case study to understand if the ACCEPT produced models could support the planning of space software engineering processes. Space software is safety-critical since a software failure could cause a space mission disaster leading to financial losses, environmental pollution, and people's endangerment. Space software production is frequently the result of industrial cooperation. Such cooperation is coordinated through compliance with relevant standards. In the European space context, in which projects are share between companies that act as supplies with others that act as customers, the de-facto standard that regulated software development is the ECSS-E-ST-40C. Such a standard provides requirements that help customers formulate their project-specific requirements (ECSS Applicability Requirements Matrix or EARM) and suppliers to prepare their responses and implement the work (ECSS Compliance Matrix or ECM). For this reason, the planning of software engineering processes in compliance with project-specific ECSS-E-ST-40C applicable requirements is mandatory during contractual agreements. The sheer volume of the requirements in this specific standard, which requires tailoring and documentation, make compliance duties challenging. The case study's goal was to qualitatively analyze the current effort required to model a CaEPP in ACCEPT for software development processes in compliance with ECSS-E-ST-40C and the coverage level of such models. In particular, we analyzed actual effort, which is determined by task demands. Initial observations show that the effort required to model compliance and processes artifacts is significant. However, the effort is reduced in the long term since models are, to some extend, reusable and flexible. Thus, process engineers in the space context do not need to start from scratch in every project. Reusing artifacts in the compliance checking process may simplify the work that process engineers need to perform in every process planning. Such gain could be interpreted as a benefit in terms of resource savings since professionals' time is costly. We also analyzed the coverage level of the models based on design decisions. In our opinion, such a coverage level is adequate since it responds to the information needs required by the ECSS-E-ST-40C framework, i.e., information requested by EARM and ECM matrices, the process they regulate, and their required alignment (compliance annotations, analysis, and results).

The outcome of this case study only applies to compliance-aware software engineering processes with the characteristics demanded by ECSS-E-ST-40C. Other safety-critical engineering processes, such as safety-critical processes in chemical plants, may exhibit additional challenges. Thus, to generalize our framework capabilities, we have to perform more case studies with a broader range of standards applicable to the safety-related context. Additional case studies may also help us further improve our framework and provide more cases that facilitate its introduction in different contexts. However, the analysis performed in this case study gave us insights that could lead to additional refinements and improvements. In general, ACCEPT methodology is systematic and can be further automated. Thus, we consider adding mechanisms that facilitate the edition of rules and the use of templates for safety compliance hints and patterns. We also aim to design algorithms that facilitate the automation of the formalization of requirements and examine the semantic similarity between process elements and compliance effects to facilitate the compliance annotation of process elements. In terms of analysis, we have a further job to do. The experience of effort (or perception of effort) is a factor that is also important to analyze since it can provide feedback on task difficulty. Therefore, we plan to conduct experiments that include users perceiving effort in the modeling tasks required to create CaEPPs. Moreover, we aim to evaluate user acceptance by using frameworks, such as the technology acceptance model (TAM) [79]). We also need to specify well-defined metrics to demonstrate our approach's value-add in terms of efficiency. Finally, we could generate fitness functions that facilitate calculations regarding the adequacy of the information coverage level provided by the models, the compliance documentation, and the evolution management.

# Bibliography

[1] V. Icheku, *Understanding ethics and ethical decision-making*. Xlibris Corporation, 2011.

[2] P. B. Ladkin, "Duty of care and engineering functional-safety standards," *Digital Evidence & Elec. Signature L. Rev.*, vol. 16, p. 51, 2019.

[3] M. Generowicz, "The Easy Path to Functional Safety Compliance." 2013.

[4] M. A. Cusumano, "Who is liable for bugs and security flaws in software?," *Communications of the ACM*, vol. 47, no. 3, pp. 25–27, 2004.

[5] S. Ingolfo, A. Siena, and J. Mylopoulos, "Establishing regulatory compliance for software requirements," in *International Conference on Conceptual Modeling*, pp. 47–61, Springer, 2011.

[6] N. Walkinshaw, "Software quality assurance," *Springer International Publishing*, vol. 10, pp. 978–3, 2017.

[7] B. Blackwelder, K. Coleman, S. Colunga-Santoyo, J. S. Harrison, and D. Wozniak, "The volkswagen scandal," 2016.

[8] A. Schwartz, "Statutory interpretation, capture, and tort law: The regulatory compliance defense," *American Law and Economics Review*, vol. 2, no. 1, pp. 1–57, 2000.

[9] IEC 61508, "Functional safety of electrical/electronic/programmable electronic safety-related systems," 2010.

[10] F. Moyón, D. Méndez, K. Beckers, and S. Klepper, "How to integrate security compliance requirements with agile software engineering at scale?," in *International Conference on Product-Focused Software Process Improvement*, pp. 69–87, Springer, 2020.

[11] K. Eastaughffe, A. Cant, and M. Ozols, "A framework for assessing standards for safety critical computer-based systems," in *4th IEEE International Software Engineering Standards Symposium and Forum.'Best Software Practices for the Internet Age'*, pp. 33–44, IEEE, 1999.

[12] B. Gallina, F. U. Muram, and J. P. Castellanos Ardila, "Compliance of agilized (software) development processes with safety standards: a vision," in *19th International Conference on Agile Software Development*, pp. 1–6, 2018.

[13] A. Ruiz, B. Gallina, J. L. de la Vara, S. Mazzini, and H. Espinoza, "Amass: Architecturedriven, multi-concern, seamless, reuse-oriented assurance and certification of cpss," in *5th International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems (SASSUR), Trondheim, Norway, September, Computer Safety, Reliability, and Security (SAFECOMP), Lecture Notes in Computer Science, vol 9923*, pp. 311–321, 2016.

[14] J. L. de la Vara, E. Parra, A. Ruiz, and B. Gallina, "Amass: a large-scale european project to improve the assurance and certification of cyber-physical systems," in *International Conference on Product-Focused Software Process Improvement*, pp. 626–632, Springer, 2019.

[15] J. Castellanos Ardila, *Facilitating Compliance Checking of Processes against Safety Standards*. Licentiate thesis, Mälardalen University, Sweden, 2019.

[16] J. P. Castellanos Ardila, "Facilitating automated compliance checking in the safety-critical context," *Electronic Communications of the EASST*, vol. 78, 2019.

[17] R. Lu, S. Sadiq, and G. Governatori, "Compliance aware business process design," in *International Conference on Business Process Management*, pp. 120–131, Springer, 2007.

[18] A. Siena, J. Mylopoulos, A. Perini, and A. Susi, "From laws to requirements," in *2008 Requirements Engineering and Law*, pp. 6–10, IEEE, 2008.

[19] G. Governatori, "Representing Business Contracts in RuleML," *International Journal of Cooperative Information Systems.*, pp. 181–216, 2005.

[20] G. Governatori, "The Regorous approach to process compliance," in *IEEE 19th International Enterprise Distributed Object Computing Workshop*, pp. 33–40, 2015.

[21] OMG, "Software & Systems Process Engineering Meta-Model Specification. V. 2.0.," 2008.

[22] OMG, "Unified Modeling Language Specification V. 2.5.1," 2017.

[23] Eclipse Foundation, "Eclipse Process Framework (EPF) Composer," 2018.

[24] M. A. Javed and B. Gallina, "Get EPF Composer back to the future: a trip from Galileo to Photon after 11 years," in *EclipseCon*, 2018.

[25] SINTEF, "BVR Tool, https://github.com/SINTEF-9012/bvr," 2016.

[26] M. A. Javed and B. Gallina, "Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool," in *22nd International Systems and Software Product Line Conference*, pp. 23–28, 2018.

[27] J. L. de la Vara, E. Parra, A. Ruiz, and B. Gallina, "The amass tool platform: An innovative solution for assurance and certication of cyber-physical systems.," in *Joint Proceedings of REFSQ-2020 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 26th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2020)CEUR Workshop Proceedings, Vol-2584, urn:nbn:de:0074-2584-1*, 2020.

[28] I. Ruiz-Rube, J. M. Dodero, M. Palomo-Duarte, M. Ruiz, and D. Gawn, "Uses and applications of spem process models. a systematic mapping study," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 1, no. 32, pp. 999–1025, 2012.

[29] G. Baumgarten, M. Rosinger, A. Todino, and R. de Juan Marín, "Spem 2.0 as process baseline meta-model for the development and optimization of complex embedded systems," in *2015 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 155–162, IEEE, 2015.

[30] V. Rantala, K. Könnölä, S. Suomi, M. Isomäki, and T. Lehtonen, "Agile embedded system development versus european space standards," *International Journal of Information Systems and Social Change*, vol. 8, no. 1, pp. 1–23, 2017.

[31] A. Lill, *Definition of an Agile Software Development Process for the European Space Industry*. Master thesis, Technische Univesität München, 2018.

[32] ESA, "*ECSS-E-ST-40C – Space Engineering Software*," 2009.

[33] S. Ghanavati, D. Amyot, and L. Peyton, "Comparative Analysis between Document-based and Model-based Compliance Management Approaches," in *Requirements Engineering and Law*, pp. 35–39, 2008.

[34] N. Harkiolakis, *Assurance*, pp. 122–127. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[35] N. G. Leveson, *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[36] H. M. Kienle, D. Sundmark, K. Lundqvist, and A. Johnsen, "Liability for software in safety-critical mechatronic systems: An industrial questionnaire," in *2nd International Workshop on Software Engineering for Embedded Systems*, pp. 44–50, IEEE, 2012.

[37] U.S. FDA, "U.S. Food and Drug-Medical Devices," 1906.

[38] ESA, "*ECSS-S-ST-00C – System, Description, implementation and general requirements*," 2009.

[39] P. J. Denning and M. Tedre, *Computational Thinking*. MIT Press, 2019.

[40] N. Leveson, "Are you sure your software will not kill anyone?," *Communications of the ACM*, vol. 63, no. 2, pp. 25–28, 2020.

[41] G. Kalus and M. Kuhrmann, "Criteria for software process tailoring: a systematic review," in *International Conference on Software and System Process*, pp. 171–180, 2013.

[42] J. Cosgrove, "Software engineering and the law," *IEEE Software*, vol. 18, no. 3, pp. 14–16, 2001.

[43] L. Buglione, A. April, and R. J. Rejas-Muslera, "The need for a legal perspective in software engineering maturity models," *Intellectual Property*, vol. 4, no. 9, p. 10, 2010.

[44] N. Leveson *et al.*, "Medical devices: The therac-25," *Appendix of: Safeware: System Safety and Computers*, 1995.

[45] M. Dowson, “The ariane 5 software failure,” *ACM SIGSOFT Software Engineering Notes*, vol. 22, no. 2, p. 84, 1997.

[46] B. S. Cruz and M. de Oliveira Dias, “Crashed boeing 737-max: Fatalities or malpractice?,” *GSJ*, vol. 8, no. 1, pp. 2615–2624, 2020.

[47] SEI, “*CMMI for Development V. 1.3– Capability Maturity Model Integration*,” 2011.

[48] G. O’Regan, “Overview of software engineering,” in *World of Computing*, pp. 179–202, Springer, Cham, 2018.

[49] B. Gallina, E. Gómez-Martínez, and C. B. Earle, “Deriving safety case fragments for assessing mbasafe’s compliance with en 50128,” in *International Conference on Software Process Improvement and Capability Determination*, pp. 3–16, Springer, 2016.

[50] ESA, “*ECSS-E-HB-40C – Space engineering - Software engineering handbook*,” 2013.

[51] E. Ahmad, B. Raza, R. Feldt, and T. Nordebäck, “Ecss standard compliant agile software development: an industrial case study,” in *Proceedings of the 2010 National Software Engineering Conference*, pp. 1–6, 2010.

[52] ESA, “*ECSS-Q-ST-40C – Space Product Assurance - Safety*,” 2017.

[53] T. Boutros and T. Purdie, *The Process Improvement Handbook: A Blueprint for Managing Change and Increasing Organizational Performance*. McGraw-Hill Education, 2014.

[54] M. Hashmi, G. Governatori, and M. Wynn, “Normative Requirements for Business Process Compliance,” *Lecture Notes in Business Information Processing*, vol. 177, pp. 100–116, 2013.

[55] S. Sadiq, G. Governatori, and K. Namiri, “Modeling Control Objectives for Business Process Compliance,” in *International Conference on Business Process Management*, pp. 149–164, 2007.

[56] D. Smith, “The Design of Divide and Conquer Algorithms,” *Science of Computer Programming*, vol. 5, pp. 37–58, 1985.

[57] J. P. Castellanos Ardila and B. Gallina, “Separation of concerns in process compliance checking: Divide-and-conquer,” in *27th European & Asian*

*System, Software & Service Process Improvement & Innovation*, pp. 135–147, Springer, 2020.

[58] J. P. Castellanos Ardila and B. Gallina, "Formal Contract Logic Based Patterns for Facilitating Compliance Checking against ISO 26262," in *1st Workshop on Technologies for Regulatory Compliance*, pp. 65–72, 2017.

[59] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *21st International Conference on Software Engineering*, pp. 411–420, 1999.

[60] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, "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.

[61] J. P. Castellanos Ardila, B. Gallina, and F. U. Muram, "Transforming spem 2.0-compatible process models into models checkable for compliance," in *International Conference on Software Process Improvement and Capability Determination*, pp. 233–247, Springer, 2018.

[62] B. Tuft, "Eclipse Process Framework (EPF) Composer: Installation, Introduction, Tutorial and Manual," 2010.

[63] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[64] J. Steele, "What is (perception of) effort? objective and subjective effort during task performance," *PsyArXiv, doi: 10.31234/osf.io/kbyhm*, 2020.

[65] ESA, "ECSS MasterDB - User requirements Document," 2018.

[66] O. Armbrust, A. Ocampo, and M. Soto, "Tracing process model evolution: A semi-formal process modeling approach," in *ECMDA Traceability Workshop*, pp. 57–66, 2005.

[67] R. K. Panesar-Walawege, M. Sabetzadeh, L. Briand, and T. Coq, "Characterizing the chain of evidence for software safety cases: A conceptual model based on the iec 61508 standard," in *3rd International Conference on Software Testing, Verification and Validation*, pp. 335–344, IEEE, 2010.

[68] J. L. de la Vara and R. K. Panesar-Walawege, “Safetymet: A metamodel for safety standards,” in *International Conference on Model Driven Engineering Languages and Systems*, pp. 69–86, Springer, 2013.

[69] R. Eito-Brun and A. Amescua, “Dealing with software process requirements complexity: an information access proposal based on semantic technologies,” *Requirements Engineering*, vol. 22, no. 4, pp. 527–542, 2017.

[70] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, “Managing standards compliance,” *IEEE Transactions on Software Engineering*, vol. 25, no. 6, pp. 836–851, 1999.

[71] F. Martinelli, F. Mercaldo, V. Nardone, A. Orlando, A. Santone, and G. Vaglini, “Model Checking Based Approach for Compliance Checking,” *Information Technology And Control*, vol. 48, no. 2, pp. 278–298, 2019.

[72] C. Bartolini, A. Giurgiu, G. Lenzini, and L. Robaldo, “Towards legal compliance by correlating standards and laws with a semi-automated methodology,” in *Benelux Conference on Artificial Intelligence*, pp. 47–62, Springer, 2016.

[73] P. W. Chung, L. Y. Cheung, and C. H. Machin, “Compliance flow–managing the compliance of dynamic and complex processes,” *Knowledge-Based Systems*, vol. 21, no. 4, pp. 332–354, 2008.

[74] F. Golra, F. Dagnat, R. Bendraou, and A. Beugnard, “Continuous Process Compliance Using Model Driven Engineering,” in *International Conference on Model and Data Engineering*, pp. 42–56, Springer, 2017.

[75] D. Rodriguez, E. Garcia, S. Sanchez, and C. R.-S. Nuzzi, “Defining software process model constraints with rules using owl and swrl,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 20, no. 4, pp. 533–548, 2010.

[76] M. Valiente, E. García-Barriocanal, and M. Sicilia, “Applying Ontology-Based Models for Supporting Integrated Software Development and IT Service,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 42, no. 1, pp. 61–74, 2012.

[77] I. Horrocks, P. Patel-schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML,” *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.

[78] S. Wang, L. Jin, and C. Jin, “Represent Software Process Engineering Metamodel in Description Logic,” in *World Academy of Science, Engineering and Technology*, vol. 11, pp. 109–113, 2006.

[79] F. D. Davis, *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. PhD thesis, Massachusetts Institute of Technology, 1985.

# Chapter 11

# Paper E: Reusing (Safety-oriented) Compliance Artifacts while Recertifying

Julieth Patricia Castellanos Ardila, Barbara Gallina.
Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development. Scitepress Digital Library - Volume 1: Modelsward. 2021

## Abstract

Revisions of safety-related standards lead to the release of new versions. Consequently, products and processes need to be recertified. To support that need, product line-oriented best practices have been adopted to systematize reuse at various levels, including the engineering process itself. As a result, Safety-oriented Process Line Engineering (SoPLE) is introduced to systematize reuse of safety-oriented process-related artifacts. To systematize reuse of artifacts during automated process compliance checking, SoPLE was conceptually combined with a logic-based framework. However, no integrated and tool-supported solution was provided. In this paper, we focus on process recertification (interpreted as the need to show process plan adherence with the new version of the standard) and propose a concrete technical and tool-supported methodological framework for reusing (safety-oriented) compliance artifacts while recertifying. We illustrate the benefits of our methodological framework by considering ISO 14971 versions, and measuring the enabled reuse.

## 11.1 Introduction

Revisions of safety-related standards lead to the release of new versions. Adjustments resulting from adding, deleting, or modifying requirements change the compliance status of organizations. Consequently, products and processes need to be recertified. To maintain processes compliance back in line, manufacturers perform a gap analysis between standards versions. A gap analysis permits manufacturers to understand what can be reused in terms of process information and process compliance demonstration [1]. In general, by reading the requirements of prescriptive standards, it is possible to identify similarities regarding tasks, work products, and other process-related artifacts, which are candidates for reuse. Based on product line-oriented best practices, reuse can be systematized at various levels, including the engineering process itself. As a result, Safety-oriented Process Line Engineering (SoPLE) [2] is introduced to systematize reuse of safety-oriented process-related artifacts.

To increase confidence in process compliance via compliance proofs and efficiency via systematic reuse [3], SoPLE was conceptually combined with a logic-based framework. The initial logic-based framework was adapted to be used with safety-related software processes [4, 5]. As such, it requires users to model process plans checkable for compliance in EPF-C [6] (recently migrated from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 [7]), which provides the environment for modeling SPEM 2.0 (Systems & Software Process Engineering Metamodel) [8]-like artifacts. Process models of this type are composed of artifacts enriched with compliance information through annotations representing formalized standards requirements in FCL (Formal Contract Logic) [9]. FCL, a logic used to interpret and model normative knowledge, can be formally verified with Regorous [10], a compliance checker created in the business and legal context. The addition of SoPLE was meant to extend systematic reuse to the automated compliance checking artifacts included in such models. However, no integrated and tool-supported solution was provided.

In this paper, we focus on showing process plan adherence with new versions of standards and propose a concrete technical and tool-supported methodological framework for reusing (safety-oriented) compliance artifacts while recertifying. In particular, we include the tool support for variability management offered by BVR-T (Base Variability Resolution Tool [11]), included in the tool-chain EPF-C $\circ$ BVR-T [12]. EPF-C $\circ$ BVR-T was developed in the context of the AMASS project [13] and was used in the space domain [14]. Systematic reuse of compliance checking artifacts is done in four steps. 1) Initial compliance checking of single process plans is performed, via EPF-C and

Regorous. 2) The resulting models are used as the base for evaluating commonalities and variabilities while adding standards of the same family, e.g., different versions of the same standard. The artifacts that vary are modeled in EPF-C. 3) The analysis of the compliance status of the standard-specific artifacts that are part of the variability is performed by taking into account the annotated compliance information. The compliance status can be analyzed by using Regorous. 4) The tool-chain EPF-C $\circ$ BVR-T is used to model the families included in the compliance checking process, pre-check the choices at variation points and deliver the concrete standard-specific (safety-oriented) compliance checking artifacts, i.e., process models, rulesets denoting formalized requirements from standards, and compliance annotated process artifacts. We illustrate the benefits of our tool-supported methodological framework by considering the evolution (i.e., new versions of the standard resulting from revisions) of ISO 14971 [15]-process for risk management to medical devices. In particular, when published, ISO 14971:2007 [16] was internationally endorsed. In contrast, EN ISO 14971:2012 [17] is harmonized with EU directives for the European market. The latest version, ISO 14971:2019 [18], is internationally endorsed again. Thus, ISO 14971-related compliance is challenging for manufacturers of medical devices, who need to find approval from regulatory bodies within and outside the EU. By measuring the enabled reuse, we answer the question *To what extent process-related compliance artifacts can be reused?*

The paper is organized as follows. In Section 11.2, we provide essential background. In Section 11.3, we present our methodological framework for compliance checking artifacts reusability. In Section 11.4, we illustrate our methodological framework by considering ISO 14971 versions, and measure the enabled reuse. In Section 11.5, we discuss our findings. In Section 11.6, we present related work. Finally, in Section 11.7, we conclude our work and present future work.

## 11.2 Background

In this section, we provide basic information on which we base our work.

### 11.2.1 ISO 14971 and its evolution

ISO 14971 [15] specifies the process required to identify hazards, estimate, evaluate, control, and monitor the risk of medical devices during its lifecycle. The content of ISO 14971 has been evolving over the years [19], incorporating consensus-based modifications and refinements. As a result, different versions have been published, i.e., ISO 14971:2007 [16], EN ISO 14971:2012 [17] and ISO 14971:2019 [18]. Relevant concepts that are used in the following sections are presented in italics. In particular, the risk analysis phase in ISO 14971:2007 and EN ISO 14971:2012 corresponds to *clause 4* and requires the planning of three tasks, i.e., 1) *Define use/safety characteristics*, 2) *Estimate risks* and 3) *Identify hazards*. In contrast, the same phase corresponds to *clause 5* in ISO 14971:2019 and the task *Define use/safety characteristics* should be divided into two. For ISO 14971:2007, *the manufacturer shall discard the negligible risk*. Annexes of EN ISO 14971:2012 and ISO 14971:2019 provide a deviation, i.e., *the manufacturer shall consider all risks*. In all versions, *the manufacturer* is the role in charge, *the risk management plan* is the prerequisite of the clause, and the work products are *risk analysis document* and *risk management file*. The risk analysis document requires information regarding the *medical device description and identification, the identification of the person and organization, the scope, date, the intended use, and reasonably foreseeable misuse, the qualitative/quantitative safety characteristics of the medical device, known and foreseeable hazards associated with the medical device, fault conditions, reasonably foreseeable sequences of events,* and *the resulting hazardous situation*. Additional information is prescribed by ISO 14971:2019, i.e., *intended medical indication, patient population, part of the body/tissue, user profile,* and *operating principle*.

### 11.2.2 Automated Compliance Checking

Our logic-based framework for automated compliance checking [4] requires process engineers to model process plans enriched with compliance annotations (see Figure 11.1), which are extracted from formalized standards requirements. An expert in FCL (Formal Contract Logic) [9] performs the required formalization. FCL is a logic that supports the modeling of norms representing obligations (**[O]**) and permissions (**[P]**) in a normative context that can be defeated by evolving knowledge. In FCL, norms are implications in which the antecedent represents the conditions for the requirements' applicability, and the conclusion represents compliance effects. Compliance effects express the concrete behavior of the process elements that adhere to standards requirements. Regorous receives the models automatically transformed from EPF-C (see [5]) and perform the automated compliance analysis. The process engineer uses compliance results (which have the potential to be transformed back into EPF-C-like formats) to perform compliance analysis and improve the process compliance iteratively.

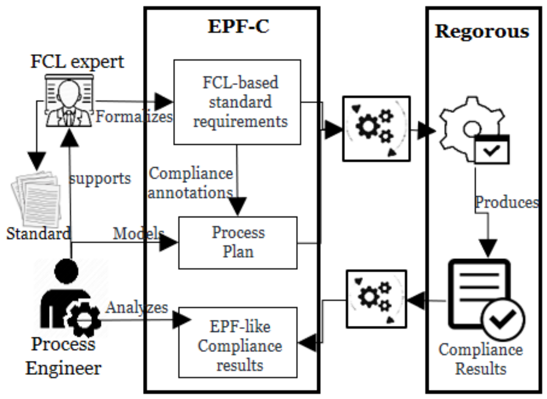


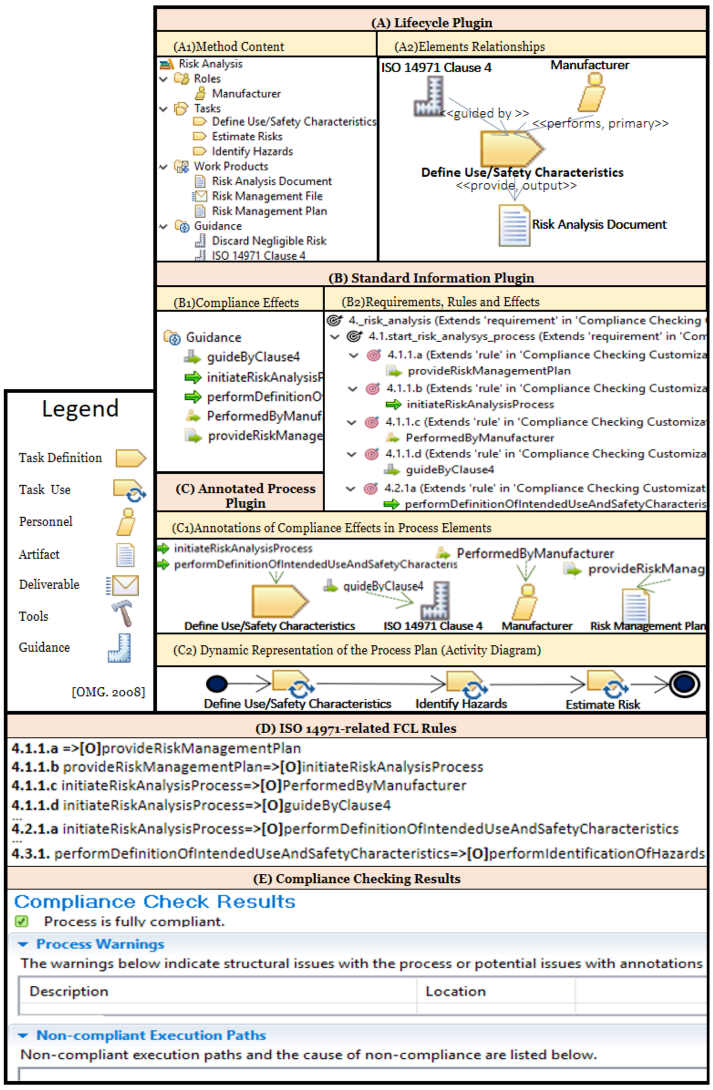
Figure 11.1: Process Compliance Checking Framework.

More concretely, EPF-C is used to model the base process and its related library (see Figure 11.2-$(A_1)$). A *role* represents who does a *task*. *Work products* identify a type of artifacts resulting from a task. *Guidance* represents free-form documentation that can be attached to process elements. A task is related to other elements as depicted in Figure 11.2-$(A_2)$. For performing automated compliance checking, the process engineer needs to model three plugins[1] in EPF-C (see Figure 11.2-$(A)$, -$(B)$ and -$(C)$).

The **Lifecycle Plugin** (see Figure 11.2-(A)) contains the method content necessary to create process plans. Figure 11.2-$(A_1)$ depicts method content required for manufacturing a medical device in compliance with ISO 14971:2007 (see [19]).

The **Standard Information Plugin** (see Figure 11.2-(B)) contains the standard requirements and their formalization in FCL. To model FCL-related information, rule propositions are modeled by using SPEM 2.0 guidance elements customized in a specific way [20]. For this, we take into account the type of process elements that are targeted by the standard requirements. As a result, process elements definition are represented with specific icons (see Figure 11.2-$(B_1)$) and the propositions are created based on templates, i.e., perform{*TaskName*}, provide{*WorkProductName*}, guidedBy{*GuidanceName*},

[1]An EPF-C plugin is a mechanism for packaging content providing modularization and extensibility


(A) Lifecycle Plugin
(A1)Method Content
(A2)Elements Relationships
Risk Analysis
Roles
Manufacturer
Tasks
Define Use/Safety Characteristics
Estimate Risks
Identify Hazards
Work Products
Risk Analysis Document
Risk Management File
Risk Management Plan
Guidance
Discard Negligible Risk
ISO 14971 Clause 4
ISO 14971 Clause 4
Manufacturer
<<guided by >>
<<performs, primary>>
Define Use/Safety Characteristics
<<provide, output>>
Risk Analysis Document
(B) Standard Information Plugin
(B1)Compliance Effects
(B2)Requirements, Rules and Effects
Guidance
guideByClause4
initiateRiskAnalysisP
performDefinitionO
PerformedByManuf
provideRiskManage
4._risk_analysis (Extends 'requirement' in 'Compliance Checking
4.1.start_risk_analysys_process (Extends 'requirement' in 'Com
4.1.1.a (Extends 'rule' in 'Compliance Checking Customiza
provideRiskManagementPlan
4.1.1.b (Extends 'rule' in 'Compliance Checking Customiza
initiateRiskAnalysisProcess
4.1.1.c (Extends 'rule' in 'Compliance Checking Customiza
PerformedByManufacturer
4.1.1.d (Extends 'rule' in 'Compliance Checking Customiza
guideByClause4
4.2.1a (Extends 'rule' in 'Compliance Checking Customizat
performDefinitionOfIntendedUseAndSafetyCharacteris
Legend
Task Definition
Task Use
Personnel
Artifact
Deliverable
Tools
Guidance
[OMG. 2008]
(C) Annotated Process Plugin
(C1)Annotations of Compliance Effects in Process Elements
initiateRiskAnalysisProcess
performDefinitionOfIntendedUseAndSafetyCharacteris
PerformedByManufacturer
provideRiskManag
guideByClause4
Define Use/Safety Characteristics
ISO 14971 Clause 4
Manufacturer
Risk Management Plan
(C2) Dynamic Representation of the Process Plan (Activity Diagram)
Define Use/Safety Characteristics
Identify Hazards
Estimate Risk
(D) ISO 14971-related FCL Rules
(E) Compliance Checking Results

```
4.1.1.a =>[O]provideRiskManagementPlan
4.1.1.b provideRiskManagementPlan=>[O]initiateRiskAnalysisProcess
4.1.1.c initiateRiskAnalysisProcess=>[O]PerformedByManufacturer
4.1.1.d initiateRiskAnalysisProcess=>[O]guideByClause4
...
4.2.1.a initiateRiskAnalysisProcess=>[O]performDefinitionOfIntendedUseAndSafetyCharacteristics
...
4.3.1. performDefinitionOfIntendedUseAndSafetyCharacteristics=>[O]performIdentificationOfHazards
```

Compliance Check Results

✓ Process is fully compliant.

Process Warnings

The warnings below indicate structural issues with the process or potential issues with annotations

| Description | Location |
|---|---|

Non-compliant Execution Paths

Non-compliant execution paths and the cause of non-compliance are listed below.

Figure 11.2: Modeling Process Plans Checkable for Compliance.

performedBy{*RoleName*}. Similarly the definition of process elements properties, i.e., $\{$*ElementName*$\}$with$\{$*Element Property*$\}$. Requirements and rules are also represented with specific customized icons (see Figure 11.2-($B_2$)). A set of FCL rules for ISO 14971-risk analysis is presented in Figure 11.2-($D$). For example, rule 4.1.1.a refers to the provision of the prerequisite, which as recalled in Section 11.2.1, is an obligation. Once provided, we have the obligation of initiate the risk analysis process (see 4.1.1.b).

The **Compliance Annotated Process Plugin** contains the process annotated with compliance effects (see Figure 11.2-($C$)). The annotation requires users to evaluate the effect that each element provide to the overall process compliance (see Figure 11.2-($C_1$)). For example, the task *DefineUse/Safety-Characteristics* is used to initiate the risk analysis process and to perform the definition of intended use and safety characteristics. Thus, we annotate it with the corresponding compliance effects. Then, a dynamic representation of the process plans is created with the annotated process elements (see Figure 11.2-($C_2$)).

**Regorous** automatically generates a compliance state representation of the annotated process plan and analyses compliance against the FCL ruleset by using two functions. The function *State(t,i)* returns the state of a *task (t)*, in the *step (i)*. The function *Force(t,i)* = $\{O\}$ associates to each *task (t)*, in the *step (i)* a set of *obligations O*. See, for example, the rules 4.1.1.a, 4.1.1.b, 4.1.1.c, 4.1.1.d. and 4.2.1.a, presented in the ruleset excerpt (see Figure 11.2-($D$)). These rules represent the obligations in force at different steps. Thus, rule 4.1.1.a forces the first obligation, i.e., Force(1,1) = [O]provideRiskManagementPlan. In a similar manner, the subsequent rules are forced, because the antecedent is getting fulfilled. *Define use/Safety Characteristics* is the first task in the workflow (see Figure 11.2-($C_2$)), and all the elements are associated to this task (Figure 11.2-($A_2$)) have their corresponding annotated compliance effects (see Figure 11.2-($C_1$)). Thus, the state representation of this task, State (1,1), contains all the compliance effects required by the force functions and the task is compliant. Regorous apply this strategy to the whole workflow and provide the compliance status of the process as well as the counterexamples in case of rules violations. When no counterexamples exist, Regorous defines that the process is compliant (see Figure 11.2-($E$)).

## 11.2.3 Compliance Proofs Reuse

A methodological framework (see Figure 11.3) for enabling reuse of compliance proofs [3] includes the combination of formal approaches with SoPLE (Safety-oriented Process Line Engineering) [2]. SoPLE manages families of processes and standards (i.e., families that exhibit several commonalities and differ via as set of managed variabilities, e.g., different versions of a standard). In SoPLE, commonalities, indeed, represent clearly reusable elements. These commonalities are defined beyond the syntactical comparison. We are interested in extracting full commonalities, i.e., whenever two elements of the same type expose only common aspects. With our methodological framework, we learned that proofs of compliance could be fully or partially reused, depending on the compliance effects produced by the variability. In Section 11.3.1, we extend the compliance analysis of such reuse.

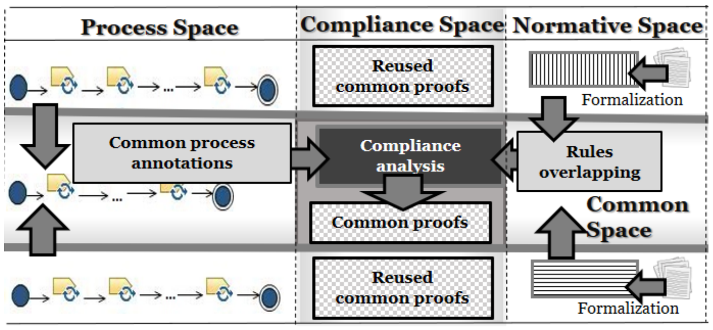


Figure 11.3: Framework for Compliance Proofs Reuse.

The framework is composed by four spaces where the process engineer perform specific actions.

1. In the process space, he/she models a Safety-oriented Process Line (SoPL). A SoPL includes manually modeling the skeleton (with commonalities and variabilities) of the process sequence.

2. In the normative space, he/she formalizes rules and models a SoPL-like structure with such rules, i.e., selects the set of rules that overlap.

3. In the common space, he/she analyzes the compliance of commonalities between the process-related SoPL with the SoPL-like rules.

4. In the compliance space, he/she analyzes the compliance effects of the tasks that contribute to the variabilities in the standard-specific process.

### 11.2.4 EPF-C ∘ BVR-T

EPF-C ∘ BVR-T [12] is a tool-chain composed by EPF (see Section 11.2.2) Composer and BVR-T (Base Variability Resolution Tool) [11] that enables SoPLE (recalled in Section 11.2.3). We focus on BVR-T. As summarized in [21], BVR-T is used to manage the variability by providing an environment in which families of different kinds, e.g., processes or products, can be modeled. A BVR model consist of three parts. The first part is the variability model, called VSpec, which permits users to model the family via a feature diagram-like fashion supplemented with constraints. Feature diagrams permit to define the distinctive user-visible aspects of the family members that are common and that vary. Table 2 recalls some basic elements. A choice represents a yes/no decision. A constraint (given in Basic Constraint Language-BCL) specifies restrictions on permissible resolution models. A group dictates the number of choice resolutions. For example, 1..1 (represents an XOR) identifies that one of the child features must be selected. Solid lines permit to link the mandatory features to a parent feature, while dashed lines permit to link optional features. Figure 11.5 depicts a VSpec diagram created with the mentioned elements.

Table 11.1: BVR Essential Modeling Elements.

| **Choice** | **Constraint** | **Group** |
|---|---|---|
| ▭ | ▱ | △ |

The second part, called the resolution, is used to allocate specific family members' values and validate such values. Thus, wrong choices violating the cross-variation points requirements designed in the VSpec can be detected. Finally, the realization permits users to bind conceptual resolutions with the concrete elements defined in EPF-C via the definition of fragment substitutions. The realization permits that specific processes are derived automatically. In this paper, we have not performed the realization part.

### 11.2.5 Reuse Measurement

A metric for reuse measurement is proposed by [22] (see below).

$$\%\ Reuse = (1 - \frac{Number\ of\ new\ objects\ built}{Total\ number\ of\ objects\ used}) * 100$$

The metric can be applied in hierarchical structures that permit the identification of the objects and the applications to which they were originally created.

This metric is expressed in terms of percentage by considering the proportion of the number of new objects built (created from scratch) and the total number of objects used (in the absence of reuse). Besides, it focuses on the total benefit attributable to reuse. Thus, objects that are reused multiple times are considered to represent multiple instances of reuse.

## 11.3 Compliance Artifacts Reusability

In this section, we present our methodological framework for compliance artifacts reusability.

### 11.3.1 Compliance Analysis

The skeleton of a family in SoPLE (as recalled in Section 11.2.3) is represented as the sequence C1-V1-C2 (see Figure 11.4). Such sequence is called the Safety-oriented Process Line or SoPL, where C1 and C2 represent the commonalities in the family and V1 represent the variability. For compliance checking (as recalled in Section 11.2.2), C1 and C2 are annotated with the compliance effects a and b, respectively. When deriving processes from the family, the variability, V1, is replaced either with R1 or R2, according to some aspect, e.g., the selection of a specific standard. Moreover, R1 is annotated with c, while R2 does not have any annotation.

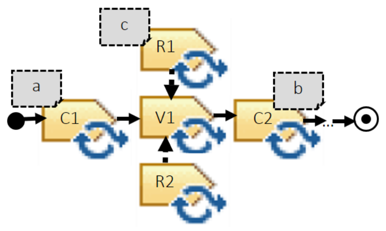


Figure 11.4: SoPL Skeleton of a Family of Processes.

The VSpec model representing the skeleton of the family C1-V1-C2 is described in VSpec as features connected to the parent feature (Checking_-Management) via solid lines (see Figure 11.5). The variability R1 and R2 are connected via dashed lines. Additional information can be modeled. In particular, the standard versions (e.g., S1 and S2) are modeled with a group element.

Moreover, BCL constraints are created to restrict the selection of the variations according to the selected standard, e.g., if S1 is selected, then R1 and its effect c become mandatory features.

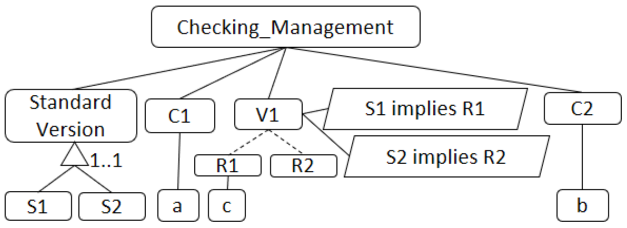


Figure 11.5: VSpec Model of the Checking Management.

The compliance state representation of the skeleton (see Figure 11.6b) is different from the derived family member, in which the replacement R1, which is annotated with the compliance information c, is replaced in V1 (see Figure 11.6d). Such representations have to comply with the respective ruleset (see Figures. 11.6a, 11.6c)

```
r1:=>[O] a
r2:a=>[O] b
```

(a)

```
State(C1)=Ann{C1}={a}
State(C2)=State{C1}UAnn{C2}={a,b}
```

(b)

```
r1:=>[O] a
r1':a=>[O] c
r2:c=>[O] b
```

(c)

```
State(C1)=Ann{C1}={a}
State(V1)=State{C1}UAnn{V1}={a,c}
State(C2)=State{C1}UAnn{C2}UAnn{C3}={a,c,b}
```

(d)

Figure 11.6: Effects/State Representation of the Variability.

Changes in the compliance status of the derived standard-specific processes depend on the normative effect of the variant. If $c = 0$, the composition of the process elements would not affect compliance since the ruleset in Figure 11.6a, would be the same that applies to the SoPLE-member. For $c \neq 0$, there are two cases. First, the effect is local to the task, i.e., the effect is triggered and fulfilled in the variant. Second, the variant effect is not triggered by a previous task and/or make a new influence in the subsequent task effect (see Figure 11.6c). In both cases, the compliance status may be affected. For these cases, we consider the separations of concerns within the regulatory space and check the structural compliance (first case) separately from the compliance of the sequence of tasks (second case). The former permits the integration of the proof in the line without affecting the general compliance status. Such checking can be performed by BVR-T, which checks the presence/absence of process elements features. The latter makes the reuse of proof conditioned to additional compliance analysis of the tasks surrounding the variant (C1 and C2 in Figure 11.4). This analysis can be performed by Regorous.

## 11.3.2 Systematic reuse of compliance artifacts

The systematic reuse of compliance artifacts requires four steps (see Figure 11.7).

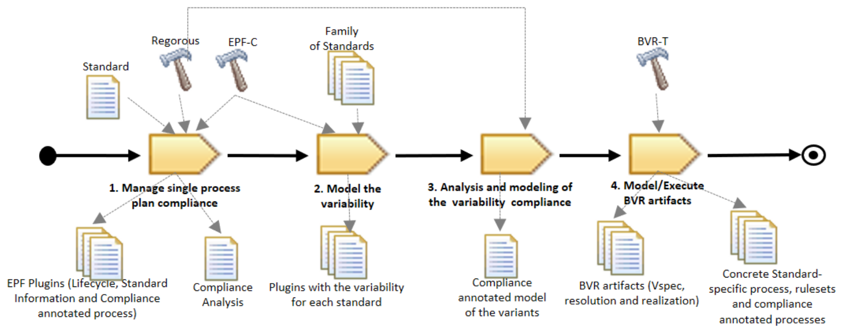

Standard
Regorous
EPF-C
Family of Standards
BVR-T
1. Manage single process plan compliance
2. Model the variability
3. Analysis and modeling of the variability compliance
4. Model/Execute BVR artifacts
EPF Plugins (Lifecycle, Standard Information and Compliance annotated process)
Compliance Analysis
Plugins with the variability for each standard
Compliance annotated model of the variants
BVR artifacts (Vspec, resolution and realization)
Concrete Standard-specific process, rulesets and compliance annotated processes

Figure 11.7: Family-oriented Compliance Checking Process.

1. **Manage single process plan compliance.** We seek for single process plan compliance by using the automated compliance checking method recalled in Section 11.2.2. Resulting artifacts are three EPF-C plugins and the compliance results issued by Regorous.

2. **Model the variability.** We evaluate the commonalities and variabilities regarding the models obtained in step 1) while adding standards of the same family, e.g., different versions of the same standard. For this, we use the method recalled in Section 11.2.3. The artifacts that vary are modeled in EPF-C. Thus, the resulting models are a lifecycle plugin and standard information plugin for each standard evolution, containing only artifacts related to the variability.

3. **Analysis and Modeling of the variability compliance.** An analysis of the changes in the compliance status of the standard-specific artifacts that

are part of the variability, as presented in Section 11.3.1, is performed by taking into account the annotated compliance information. If needed, we use Regorous. However, if the variant is small, such analysis can be done manually. The result of this step is the compliance annotated process model of the variants.

4. **Model BVR artifacts.** BVR-T is used to create the abstract representation of the families involved in compliance checking, i.e., lifecycle, standard information and compliance annotated processes. Resolution models are automatically generated from the VSpec models, and use to validate the membership of the elements according to the selected standard. In a final step, which is not part of the scope of this paper, realization models are created. Realization permits to define the replacements that should be part of the concrete standard-related artifacts that are exported back to EPF-C. Thus, in this step we use the tool-chain EPF-C $\circ$ BVR-T, recalled in Section 11.2.4.

## 11.4 Reuse within ISO 14971 evolution

In this section, we use our solution (presented in Section 11.3.2) to systematize and measure compliance artifacts reuse within the evolution of the ISO 14971 standards (recalled in Section 11.2.1).

### 11.4.1 ISO 14971 evolved artifacts

As presented in Figure 11.7, the first step consists of seeking the compliance of a process plan against an initial standard, in this case, ISO 14971:2007. The results of this step are three plugins that contain process elements, compliance rules, annotated process models (see Figures 11.2-($A$), -($B$) and -($C$)), and the compliance analysis delivered by Regorous (see Figure 11.2-($E$)), which shows that the process is compliant with the rules derived from the standard.

The second step consists of modeling the variability, i.e., we perform a gap analysis and model the additional artifacts imposed by the new standard versions. In particular, a new process element is additionally required for compliance with EN ISO 14971:2012, i.e., the guidance related to the inclusion of all risks for the treatment of negligible risk (see Figure 11.8).

In contrast, four new elements are required for compliance with ISO 14971:2019, i.e., two guidance elements (ISO 14871 clause 5 and the treatments of negli-

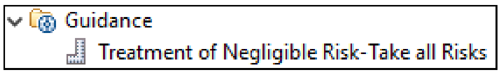

Guidance
Treatment of Negligible Risk-Take all Risks

Figure 11.8: EN ISO 14971:2012-Variable Process Elements.

gible risk), and two additional tasks, which are the result of splitting the task *Define/use safety characteristics* (see Figure 11.9).

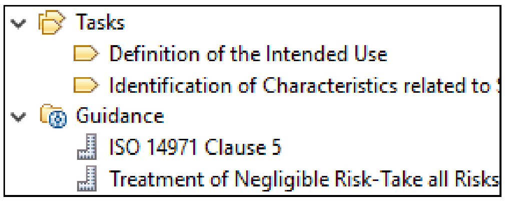

Tasks
Definition of the Intended Use
Identification of Characteristics related to
Guidance
ISO 14971 Clause 5
Treatment of Negligible Risk-Take all Risks

Figure 11.9: ISO 14971:2019-Variable Process Elements.

We also model the compliance effects. Figure 11.10 represents compliance effects extracted from ISO 14971:2007. Figure 11.11 shows 1 new compliance effect found in EN ISO14971:2012, while Figure 11.12 shows 11 new compliance effects found in ISO 14971:2019.

Figures 11.10, 11.11, and 11.12 also depict artifacts highlighted with colors. Such colors represent replacements that are necessary to be done during the standard-specific derivation. For example, performIdentificationofSafety-Characteristics and performIntendedUse, created for the ISO 14971:2019 ruleset, are meant to replace the effect performDefinitionOfIntendedUseAndSafetyCharacteristics, created for the ISO 14971:2007. In black, we highlight an artifact which contains the general information of the ruleset, which also varies with each standard. Compliance effects that are not highlighted represent artifacts that are common and can be reused.

In Step 3, the compliance analysis of the variability is performed, as presented in Section 11.3.1. In our case, we found that the compliance with EN ISO 14971:2012 requires that one new element, specifically a guidance called *Treatment of Negligible Risk-Take all Risks* (see Fig. 11.8) is annotated with a compliance effect called *guidedByTakeIntoAccountAllRisks* (see Figure 11.11). A more complex analysis is performed in the case of ISO 14971:2019. In particular, there are requirements that mandate the replacement of the task Define/Use Safety Characteristics. This implies a variation in the ruleset as

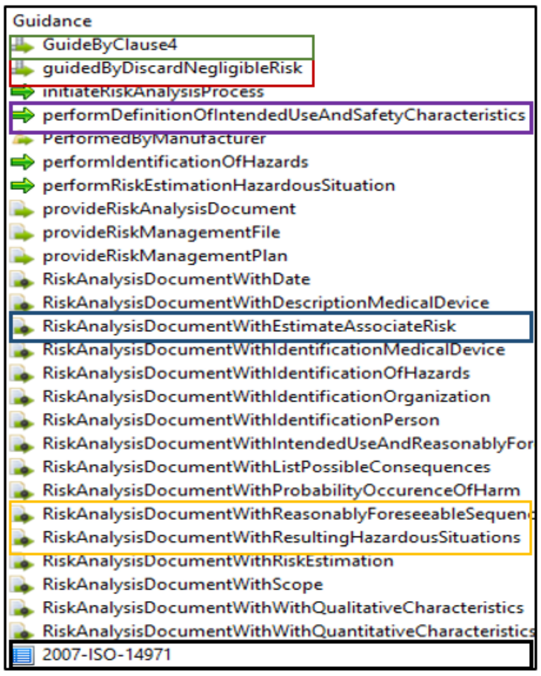

Guidance
GuideByClause4
guidedByDiscardNegligibleRisk
initiateRiskAnalysisProcess
performDefinitionOfIntendedUseAndSafetyCharacteristics
PerformedByManufacturer
performIdentificationOfHazards
performRiskEstimationHazardousSituation
provideRiskAnalysisDocument
provideRiskManagementFile
provideRiskManagementPlan
RiskAnalysisDocumentWithDate
RiskAnalysisDocumentWithDescriptionMedicalDevice
RiskAnalysisDocumentWithEstimateAssociateRisk
RiskAnalysisDocumentWithIdentificationMedicalDevice
RiskAnalysisDocumentWithIdentificationOfHazards
RiskAnalysisDocumentWithIdentificationOrganization
RiskAnalysisDocumentWithIdentificationPerson
RiskAnalysisDocumentWithIntendedUseAndReasonablyFor
RiskAnalysisDocumentWithListPossibleConsequences
RiskAnalysisDocumentWithProbabilityOccurenceOfHarm
RiskAnalysisDocumentWithReasonablyForeseeableSequen
RiskAnalysisDocumentWithResultingHazardousSituations
RiskAnalysisDocumentWithRiskEstimation
RiskAnalysisDocumentWithScope
RiskAnalysisDocumentWithWithQualitativeCharacteristics
RiskAnalysisDocumentWithWithQuantitativeCharacteristics
2007-ISO-14971

Figure 11.10: ISO 14971:2007-Compliance Effects.

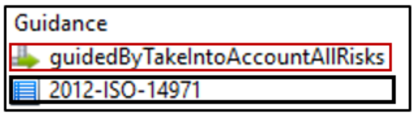
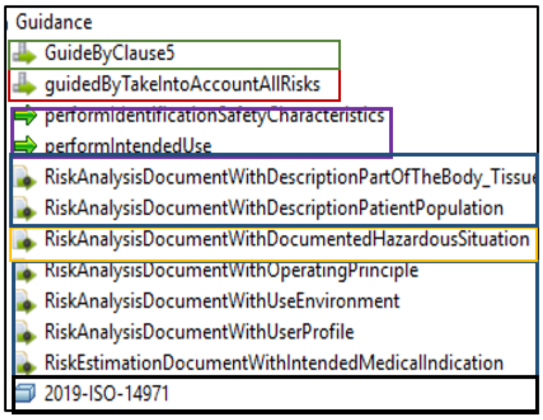

Guidance
guidedByTakeIntoAccountAllRisks
2012-ISO-14971

Figure 11.11: EN ISO 14971:2012-Effects Variability.

presented in Figure 11.13, which is evidently different from the ruleset created for ISO 14971:2007 (see Figure 11.2-$(D)$). With the introduction of these requirements, the compliance flow changes. Thus, we need to use Regorous for perform compliance checking in the first 3 tasks of the new workflow.

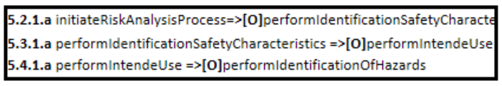
The remaining new elements (see Figure 11.9) trigger and fulfil themselves the new compliance effects (see Figure 11.12). The result of this analysis corresponds to the compliance annotations presented in Table 11.2.

The fourth step is the modeling of BVR artifacts. In this step, we model



Guidance
GuideByClause5
guidedByTakeIntoAccountAllRisks
performIdentificationSafetyCharacteristics
performIntendedUse
RiskAnalysisDocumentWithDescriptionPartOfTheBody_Tissue
RiskAnalysisDocumentWithDescriptionPatientPopulation
RiskAnalysisDocumentWithDocumentedHazardousSituation
RiskAnalysisDocumentWithOperatingPrinciple
RiskAnalysisDocumentWithUseEnvironment
RiskAnalysisDocumentWithUserProfile
RiskEstimationDocumentWithIntendedMedicalIndication
2019-ISO-14971

Figure 11.12: ISO 14971:2019-Effects Variability.



**5.2.1.a** initiateRiskAnalysisProcess=>**[O]**performIdentificationSafetyCharacte
**5.3.1.a** performIdentificationSafetyCharacteristics =>**[O]**performIntendeUse
**5.4.1.a** performIntendeUse =>**[O]**performIdentificationOfHazards

Figure 11.13: Ruleset Variation Respect ISO 14971:2019.

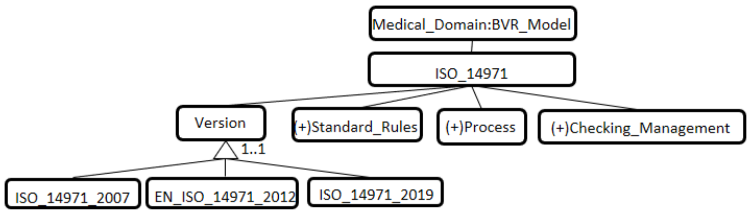


Figure 11.14: BVR VSpec.

the VSpecs of the families corresponding to the process, ruleset, and checking management. All the families are created under the same root, i.e., ISO_14971 (see Figure 11.14). A branch of the feature model tree contains the version of the standards used to make the changes at variation points.

The interested reader may refer to [19] for detailed information regarding the VSpec of the process for risk management with ISO 14971. In this paper, we focus on the VSpec model for the ruleset and the checking manage-

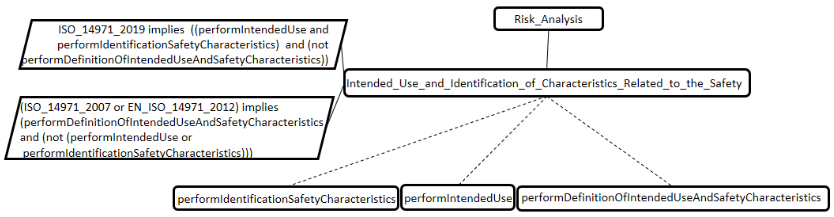

ISO_14971_2019 implies ((performIntendedUse and performIdentificationSafetyCharacteristics) and (not performDefinitionOfIntendedUseAndSafetyCharacteristics))
(ISO_14971_2007 or EN_ISO_14971_2012) implies (performDefinitionOfIntendedUseAndSafetyCharacteristics and (not (performIntendedUse or performIdentificationSafetyCharacteristics)))
Risk_Analysis
Intended_Use_and_Identification_of_Characteristics_Related_to_the_Safety
performIdentificationSafetyCharacteristics
performIntendedUse
performDefinitionOfIntendedUseAndSafetyCharacteristics

Figure 11.15: Variation Related to Compliance Effects.

Table 11.2: ISO 14971: 2019-related Annotations.

| **Element** | **Compliance Effect** |
|---|---|
| Task: Definition of the Intended Use | performIntendedUse, initiateRiskAnalysisProcess |
| Task: Identification of Characteristics related to Safety | performIdentificationSafetyCharacteristics |
| Work Product: Identification of Characteristics related to Safety | RiskAnalysisDocumentWithDescriptionPartOfTheTissue, RiskAnalysisDocumentWithDescriptionPatientPopulation, RiskAnalysisDocumentWithDocumentedHazardousSituation, RiskAnalysisDocumentWithIntendedMedicalIndication, RiskAnalysisDocumentWithIntendedUseAndReasonablyForeseeableMisuse, RiskAnalysisDocumentWithOperatingPrinciple, RiskAnalysisDocumentWithUseEnvironment, RiskAnalysisDocumentWithUserProfile |
| Guidance: ISO 14971 Clause 5 | GuideByClause5 |
| Guidance: Treatment of Negligible Risk-Take all Risks | guidedByTakeIntoAccountAllRisks |

ment. For example, Figure 11.15 depicts the representation of the compliance effects related to the requirements that impose the creation of the task Define use/safety characteristics. In particular, as presented in Section 11.2.1, such task is mandatory for ISO 14971:2007 and EN ISO 14971:2012, while in ISO 14971:2019 becomes two tasks, i.e., intended Use and Identification of Safety Characteristics. Thus, the three compliance effects (highlighted in purple in Figures 11.10 and 11.11) are modeled and two BCL constraints are created to define the variations regarding the version of the standard selected. For example, if the standard ISO_14971_2019 is selected in the branch of the version, BVR resolution will check that we select performIdentificationSafetyCharacteristics and performIntendedUse during the selection of the family-member corresponding to the ruleset of such version. The VSpec for the branch compliance checking management, contains the compliant process elements

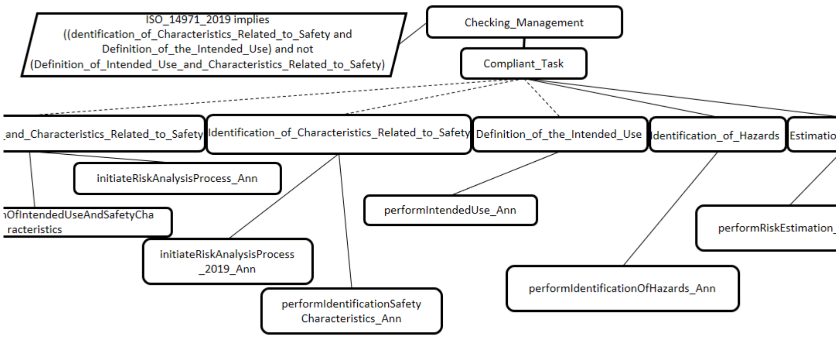

ISO_14971_2019 implies ((dentification_of_Characteristics_Related_to_Safety and Definition_of_the_Intended_Use) and not (Definition_of_Intended_Use_and_Characteristics_Related_to_Safety)
Checking_Management
Compliant_Task
_and_Characteristics_Related_to_Safety
Identification_of_Characteristics_Related_to_Safety
Definition_of_the_Intended_Use
Identification_of_Hazards
Estimatio
initiateRiskAnalysisProcess_Ann
OfIntendedUseAndSafetyCharacteristics
initiateRiskAnalysisProcess_2019_Ann
performIdentificationSafetyCharacteristics_Ann
performIntendedUse_Ann
performIdentificationOfHazards_Ann
performRiskEstimation_

Figure 11.16: Variation Related to Annotated Compliant Tasks.

grouped by their concern, i.e., tasks, role, work product, and guidance, and enriched with the compliance effects annotations. Figure 11.16 presents the branch Compliant_Task that shows the set of tasks that should appear in the process plan as well as BCL constraint that restrict the correct representation according to the standard version selected. The resolution permits the selection of correct configuration that could be exported back to EPF-C via realization models. A realization model will permit to bind the selected configuration into the concrete EPF-C related models.

## 11.4.2 Reuse Measurement

In our approach, we opt to model the full commonalities between families of standards, the process they regulate, and the compliance annotations required for automated compliance checking. Full commonalities can be guaranteed by atomizing the elements in the compliance spaces as much as possible so that only common aspects are present. For this reason, we consider that the commonalities included in the modeling of such families have the potential to be fully reusable. In that light, the percentage of reuse of compliance artifacts can be performed by using the metric defined for reuse measurement which is recalled in Section 11.2.5.

### Reuse-related to EN ISO 14971:2012

For compliance with EN ISO 14971:2012, the guidance called Treatment of Negligible Risk-Take all Risks (see Figure 11.8) was additionally required with respect to compliance established with ISO 14971:2007. In total, we used 9 process elements. Thus, the percentage of reuse is 88,9%. We also need to create 1 compliance effect and 1 ruleset (see Figure 11.11). As we used 27 artifacts, the reuse is 92,3%. The compliance effect is associated to the new guidance, which corresponds to a new compliance annotation of 26 used in total. Thus, the reuse of compliance annotations is 96,2%. (See Table 11.3).

Table 11.3: Reuse Measurement Related to EN ISO 14971:2012.

| Type of artifacts | New | Total Used | Reuse Percentage |
|---|---|---|---|
| Process | 1 | 9 | 88,9% |
| Compl. Effects | 2 | 27 | 92,3% |
| Compl. Annotations | 1 | 26 | 96,2% |

**Reuse-related to ISO 14971:2019**

For compliance with ISO 14971:2019, we need to create 4 new process artifacts (see Figure 11.9), 11 new compliance effects (see Figure 11.12) and perform 13 compliance annotations (see Table 11.2). The number of total artifacts was 10 process elements, and 32 compliance effects and compliance annotations. Thus, reuse is 60%, 61,3% and 59,4% respectively (see Table 11.4).

Table 11.4: Reuse Measurement Related to ISO 14971:2019.

| Type of artifacts | New | Total Used | Reuse Percentage |
|---|---|---|---|
| Process | 4 | 10 | 60% |
| Compl. Effects | 12 | 32 | 61,3% |
| Compl. Annotations | 13 | 32 | 59,4% |

## 11.5 Discussion

For coping with the recertification demands enforced by the new versions of standards (new requirements, jurisdictional changes) in the medical domain, process plan reconfiguration is necessary. Compliant process plan reconfiguration supported by models automatically checked for compliance is a plausible solution. Such solution involves the creation of new modeling artifacts, as presented in Section 11.4.1. However, it also involves high degrees of artifacts reuse, as presented in Section 11.4.2. In particular, Tables 11.3 and 11.4

shows a positive gain in terms of compliance checking artifacts reusability. With these percentages, the answer to the question posed in the introductory part of this paper, *to what extent process-related compliance artifacts can be reused?*, could be the following: the reuse extent in the context of medical devices is significant (the minimum gain was 59,4%). In particular, given that the manual configuration of process models checkable for compliance in EPF-C could be labor-intensive and time-consuming, the context of medical devices complying with ISO 14971 can positively benefit from the systematic reuse of compliance checking artifacts. In general, processes and standards that evince low levels of variation could be part of a family that exhibits high reuse levels in terms of compliance checking artifacts and could benefit from using our methodological framework during the required modeling task.

It is widely recognized that standards requirements are challenging to understand due to their wordiness and how they relate to each other. Their evolution is also challenging, due to the need to handle the normative changes and the recertification effort, which, as for ISO 14971, may include cross-jurisdictional spaces. When using our methodological framework, process engineers need to analyze new requirements systematically. This analysis is required to determine whether an existing compliance checking-related artifact can fulfill a specific requirement as-is or with modifications (new properties should be added/deleted), or if new artifacts have to be modeled from scratch. It also highlights problems between requirements, which may put compliance at risk, i.e., contradictory requirements, real/fake dependencies between requirements and new compliance information applicable to existing process plans. The most important is that the process engineer's analysis is recorded in graphical models, which not only provide automated checks but also automated process plan's reconfiguration. Thus, our methodological framework supports a confident reduction of the work required to be done when new instances of compliant process plans have to be modeled.

## 11.6 Related Work

The change triggered by updated standards for software process is a topic tackled from different perspectives. In [23], the authors propose a method that permits to attach change information to process documents to facilitate change understandability. However, no systematic methods to reuse modeling artifacts facilitating the changes are proposed as we do in our work. Methods for modeling the change/variation and reuse of processes result from the application of software process lines methodologies, as recently surveyed in [24]. In particular, SoPLE [2] has been exploited to provide a representation of family members with safety information, e.g., reusable process arguments used in safety cases [25] and tailoring of process models according to safety integrity levels of products [26]. In our work, we also use SoPLE to provide mechanisms to support variation knowledge reuse regarding compliance checking artifacts, which has not being yet addresses in other approaches.

Advances regarding compliance artifacts reusability exist in the business community. Some researchers tackled reuse by defining building blocks that implement compliance requirements, e.g., compliance scopes [27], and compliance fragments [28, 29]. Reuse is also approached with the use of process patterns [30], and rule patterns [31]. In contrast, we propose a holistic modeling framework for safety-related process compliance checking that permits to model artifacts, which can be automatically interleaved with evolutionary/changing artifacts originated from new versions of standards. In that way, not only building blocks that implement compliance requirements (i.e., called in our framework, process models checkable for compliance) are reusable but also process models and rulesets denoting formalized requirements from standards.

## 11.7 Conclusions and Future Work

Recertification is the consequence of the release of new versions of standards. In this paper, we focused on process recertification needs (interpreted as the need to show process plan adherence with the new version of the standard). Taking this into account, we proposed a concrete technical and tool-supported methodological framework for reusing (safety-oriented) compliance artifacts while recertifying. This methodological framework encompasses process modeling, process compliance checking, and variability management capabilities to enable systematic reuse and automatic generation of process-related compliance checking artifacts (i.e., process models, rulesets denoting formalized requirements from standards, and compliance annotated process artifacts). We illustrate our methodological framework within the family composed of the versions of the standard ISO 14971. Finally, we answer our initial question regarding the extent of reuse of process-related compliance artifacts by measuring the reuse enabled by our methodological framework. In particular, in the context of medical devices complying with different versions of ISO 14971, the reuse is significant. We concluded that processes and standards that evince low levels of variation (such as ISO 14971) could benefit from using our methodological framework during the modeling task required for compliance checking.

In the future, we intend to perform evaluations that consider the entire ISO 14971 and related standards (e.g., process improvement and security). Moreover, we plan to conduct controlled experiments to evaluate the users' perceived usefulness. We also believe that when creating/updating standards, process models, and formal representations of the requirements should also be provided. Thus, we plan to contact standardization bodies to investigate this possibility, which could reduce our approach's modeling effort and at the same time reduce undesired room for interpretation of the standards. Finally, we intend to use more elaborated measurement frameworks to provide evidence concerning our solution's efficiency in terms of time and cost reduction, as well as scalability.

# Bibliography

[1] B. Gallina, S. Kashiyarandi, H. Martin, and R. Bramberger, "Modeling a Safety- and Automotive-Oriented Process Line to Enable Reuse and Flexible Process Derivation," in *38th International Computer Software and Applications Conference*, pp. 504–509, 2014.

[2] B. Gallina, I. Sljivo, and O. Jaradat, "Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification," in *35th Annual IEEE Software Engineering Workshop*, pp. 148–157, 2012.

[3] J. P. Castellanos Ardila and B. Gallina, "Towards increased efficiency and confidence in process compliance," in *Systems, Software and Services Process Improvement*, pp. 162–174, 2017.

[4] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, "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.

[5] J. P. Castellanos Ardila, B. Gallina, and F. UL Muram, "Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance," in *18th International SPICE Conference*, 2018.

[6] Eclipse-Foundation, "*Eclipse Process Framework (EPF) Composer – EPF 1.5.2 Release*," 2018.

[7] M. Javed and B. Gallina, "Get EPF Composer back to the future: a trip from Galileo to Photon after 11 years," in *EclipseCon*, 2018.

[8] OMG, "Software & Systems Process Engineering Meta-Model Specification. Version 2.0.," 2008.

[9] G. Governatori, "Representing Business Contracts in RuleML," *International Journal of Cooperative Information Systems.*, pp. 181–216, 2005.

[10] G. Governatori, "The Regorous Approach to Process Compliance," in *19th International Enterprise Distributed Object Computing Workshop*, pp. 33–40, 2015.

[11] SINTEF, "BVR Tool, https://github.com/SINTEF-9012/bvr," 2016.

[12] M. Javed and B. Gallina, "Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool," in *22nd International Systems and Software Product Line Conference*, pp. 23–28, 2018.

[13] J. L. de la Vara, E. Parra, A. Ruiz, and B. Gallina, "AMASS: A Large-Scale European Project to Improve the Assurance and Certification of Cyber-Physical Systems," in *20th International Conference in Product-Focused Software Process Improvement*, pp. 626–632, 2019.

[14] B. Gallina, "Quantitative evaluation of tailoring within spice-compliant security-informed safety-oriented process lines," *Journal of Software: Evolution and Process*, vol. e2212, pp. 1–13, aug 2019.

[15] ISO, "*ISO 14971:2000 – Application of risk management to medical devices*," Dec. 2000.

[16] ISO, "*ISO 14971:2007 – Application of risk management to medical devices*," Mar. 2007.

[17] ISO, "*EN ISO 14971:2012 – Application of risk management to medical devices (ISO 14971:2007, Corrected version 2007-10-01)*," July 2012.

[18] ISO, "*ISO 14971:2019 – Application of risk management to medical devices*," Dec. 2019.

[19] A. Pulla and A. Bregu, "Master Thesis: Evaluating the Compliance Re-Certification Efficiency Enabled by the AMASS Platform for Medical Devices, Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden," 2020.

[20] J. P. Castellanos Ardila and B. Gallina, "Separation of concerns in process compliance checking: Divide-and-conquer," in *Systems, Software and Services Process Improvement*, pp. 135–147, 2020.

[21] B. Gallina, A. Pulla, A. Bregu, and J. Castellanos Ardila, "Process Compliance Re-Certification Efficiency Enabled by EPF-C $\circ$ BVR-T : a Case Study," in *13th International Conference on the Quality of Information and Communications Technology*, pp. 1–8, 2020.

[22] R. Banker, R. Kauffman, and D. Zweig, "Repository Evaluation of Software Reuse," *IEEE Transactions on Software Engineering*, vol. 19, no. 4, pp. 379–389, 1993.

[23] A. Ocampo and J. Münch, "Rationale Modeling for Software Process Evolution Alexis," *Software Process: Improvement and Practice*, vol. 14, no. 2, pp. 85–105, 2009.

[24] E. N. Teixeira, F. A. Aleixo, F. D. de Sousa Amâncio, E. OliveiraJr, U. Kulesza, and C. Werner, "Software process line as an approach to support software process reuse: A systematic literature review," *Information and Software Technology*, vol. 116, p. 106175, 2019.

[25] H. Martin, M. Krammer, R. Bramberger, and E. Armengaud, "Process- and product-based lines of argument for automotive safety cases," in *7th International Conference on Cyber-Physical Systems*, 2016.

[26] L. Bressan, A. L. de Oliveira, F. Campos, Y. Papadopoulos, and D. Parker, "An integrated approach to support the process-based certification of variant-intensive systems," in *International Symposium on Model-Based Safety and Assessment*, pp. 179–193, 2020.

[27] D. Schleicher, S. Grohe, F. Leymann, P. Schneider, D. Schumm, and T. Wolf, "An approach to combine data-related and control-flow-related compliance rules," in *International Conference on Service-Oriented Computing and Applications*, pp. 1–8, 2011.

[28] K. Görlach, O. Kopp, F. Leymann, and D. Schumm, "WS-BPEL extension for compliance fragments (BPEL4CFrags)," tech. rep., Institute of Architecture of Application Systems, University of Stuttgart., 2011.

[29] Z. Ma, *Process fragments: enhancing reuse of process logic in BPEL process models*. Ph.d. dissertation, University of Stuttgart, 2012.

[30] M. Kabir, Z. Xing, P. Chandrasekaran, and S. Lin, "Process Patterns: Reusable Design Artifacts for Business Process Models," *International Computer Software and Applications Conference*, vol. 1, pp. 714–721, 2017.

[31] A. Elgammal, O. Turetken, W. van den Heuvel, and M. Papazoglou, "Formalizing and applying compliance patterns for business process compliance," *Software and Systems Modeling.*, pp. 119–146, 2016.

# Chapter 12

# Paper F: Systematic Literature Review of Compliance Checking Approaches for Software Processes

Julieth Patricia Castellanos Ardila, Barbara Gallina, Faiz Ul Muram.
Technical Report, ISRN MDH-MRTC-336/2021-1-SE. Mälardalen Real-Time Research Center, Mälardalen University, June 2021. A version is submitted to a journal.

**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.

## 12.1 Introduction

Many applications and infrastructures rely on software, including the internet, warning systems, and medical and financial information systems [1]. Due to its growing use, the software is becoming a public good, and its quality is a concern for society [2]. In particular, there is a group of stakeholders, called community stakeholders [3], 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 [4]. There are also knowledgeable individuals demanding the use of standards to influence responsible behavior among industry practices [5]. 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 [7, 8, 9]. Standardized software processes are referenced in international standards, e.g., ISO/IEC 12207 [10] for software, and ISO/IEC 15504 [11] series of standards - and its evolution ISO/IEC 330xx series [12] 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 [13]. 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 [3]. For this reason, they act as process constraints and generally omit implementation-specific details [15].

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

[1]Safety-critical systems are those whose failure could lead to unacceptable consequences, e.g., death, injury, loss of property, or environmental harm [6]

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 [16]. 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 [17, 18], 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 compliance 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.

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

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 12.2, we present essential background. In Section 12.3, we present the research method. In Section 12.4, we report the results of the review. In Section 12.5, we discuss the findings. In Section 12.6, we discuss the validity of the findings. In Section 12.7, we discuss related work. Finally, In Section 12.8, we summarize the work and present future remarks.

## 12.2 Background

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

### 12.2.1 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 [19]. 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 [9]. Traceability is also a mandated requirement [20]. 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 [21]. 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 [22]. 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 [23, 24]. 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 [25]. Moreover, the translation of requirements written in natural language is complex. For this reason, precise notions are required [26]. Moreover, to be formal in a certification context, a model must have an unambiguous, mathematically defined syntax and semantics [27].

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 [28]. 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 [29]. 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 [30]. Thus, compliance is a relationship between permissions (optional) and obligations (required).

### 12.2.2 Software Processes

Software developers perform processes, which are often defined to various levels of detail [31]. According to Parnas et al. [32], the most advantageous form of a process description will be in terms of work products workflow. Lonchamp [33] highlighted the importance of organizational structures. Fuggetta [34] 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 [35, 36]. 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 [37]. According to Armour [38], 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 [37] by performing a knowledge acquisition activity [38].

Explicit descriptions of the software processes servers development to proceed in a systematic way [7], increases predictability and transparency [9]. Software process descriptions are also commonly used to convince third parties, such as customers or regulatory bodies, regarding the quality of the software [34, 9]. 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 [39]. 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 [40, 41], e.g., the Scaled Agile Framework (SAFe)[5].

Software process models help organizations preserve, repeat, analyze and reuse process information [42]. Models also can improve the understanding of compliance needs [43]. A software process model is an abstraction whose goal is to approximate the full range of characteristics and properties of an actual software process [44]. For this, a process model should [31]: 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 [9]: software life cycle models define the

[3]https://agilemanifesto.org/

[4]See for instance: https://stateofagile.com/

[5]https://www.scaledagile.com/

main steps and their sequence, while software process models provide more detailed instructions, 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 [23] 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.

### 12.2.3 Software Process-related Normative Frameworks

Normative frameworks addressing software processes prescribe requirements for their implementation. Organizations follow these documents, which are also called prescriptive [45], to facilitate the process standardization, evaluation, and improvement [46, 47]. For example, the standard ISO/IEC/IEEE 12207 [10] 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) [48] as a significant innovation [49]. Then, the Software Process Improvement and Capability Determination-SPICE (ISO/IEC 15504 [11]) 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 [7].

Several SPI context-specific frameworks exist [50], e.g., Automotive SPICE (or ASPICE) [51], the medical devices MDevSPICEe [52] and the Object-Oriented Software Process OOSPICE [53]. Recently, SPICE has been revised and replaced with the ISO/IEC 330XX series [12], e.g., ISO/IEC TS 33053 [54], 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 [55]. In particular, there is the ISO 9000 series [56], e.g., ISO 9001 [57], guidance for their application ISO/IEC 90003 [58], and ISO/IEC TR 29110 [59], 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 [55]. 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 [60].

Manufacturers of safety-critical systems have the duty of care[6] [62]. 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 [20], usually found in industry standards [1]. 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 [63]. The Health and Safety Executive in England has used compliance with IEC 61508 [64] as a guideline for bringing legal actions if harm is caused by safety-critical systems [62]. As compliance with safety standards has become essential evidence for a jury in a product liability action [65], failure or inadequate compliance could lead to legal risks, i.e., penalties [66] and prosecutions [67].

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 [68] (automotive), CENELEC EN 50128 [69] and EN 50126 [70] (railway), DO-178C [71] (avionics), and IEC 62304 [72] (medical devices), to only mention some of them. Cybersecurity handbooks (e.g., cyber-physical vehicle systems SAE J3061 [73]), standard for software development (e.g., medical devices-IEC 62304 [72] and space mission-critical software-ECSS-ST-40C [74]), risk management (e.g., ISO 14971-application of risk management to medical devices [75]), and information technology (e.g., ISO/IEC 27000 [76]), 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

[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 [61].

Data Protection Directive (EU DPD) [77], then replaced by the General Data Protection Regulation (GDPR) [78], and PIPEDA (Personal Information Protection and Electronic Documents Act) [79]. 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 [80]. Since AI is still software, its needs will probably be approached from the software process perspective [81]. Thus, practitioners have to embrace software process diversity, i.e., the adoption of multiple normative software process frameworks within single software processes [19].

## 12.3 Research Method

This section describes our research method, which is based on the guidelines for Systematic Literature Review (SLR) recommended by Kitchenham and others [17, 18]. 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 12.4.

## 12.3.1 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 12.2.2 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 12.2.3). 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 12.2.1). 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. [82]. 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 policy-makers.

### 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 12.3.1, 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 12.2.3). As presented in Section 12.2.1, 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 12.1).

### 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,

Table 12.1: 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. |

we consider the structure of the Context–Intervention–Mechanism–Outcome (CIMO) Logic [83]. The CIMO is a logic constructed as follows: if you have a problematic Context (C), use a special kind of Intervention (I) to invoke the generative Mechanism(s) (M), to deliver a specific Outcome (O). The context corresponds to the surroundings (external and internal environment) factors.

The interventions are those factors that have the potential to do some influence. The mechanisms are the means that in a specific context are triggered by the intervention. Finally, the outcomes are the intervention results in its various aspects.

Table 12.2: 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 |

As presented in Table 12.2, the factors of importance in our SLR are software processes, normative software process constraints, automation methods, the results 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 12.2. First, in Section 12.2.2, 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 12.2.3, we found sources of ”normative software process constraints” in a ”standard”, ”reference model”, ”framework”, ”regulation”, ”policy.” Third, in Section 12.2.1, 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

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

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 12.3).

Table 12.3: 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”) |

**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 [84]: 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 12.4.

[8]https://dl.acm.org/

[9]https://ieeexplore.ieee.org/Xplore/home.jsp

[10]https://link.springer.com/

[11]https://scholar.google.com/

Table 12.4: 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 12.3 is applied to the electronic databases selected in the study selection criteria. Different filtering levels are then applied to the retrieved studies to find the relevant ones for this research. Initially, we perform a title screening on the initial set of retrieved publications. In this phase, we also remove the duplicates that can be found in different databases. Then, we perform an abstract screening, from which we select the papers that would be thoroughly read. After, we perform a snowballing [85], 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 12.5.

Table 12.5: 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 12.6), based on criteria formulated by Kitchenham and others [86]. 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.

### 12.3.2 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 12.3.1. In particular, we applied the search string defined in Table 12.3 to the different databases included in the study selection criteria without trunking the dates of the search. Our search was performed between February 22 to March 15, 2021. The databases Springer Link, ACM (in which we took the option ”Expand our search to The ACM Guide to Computing Literature”), and 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 process” 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 12.1.

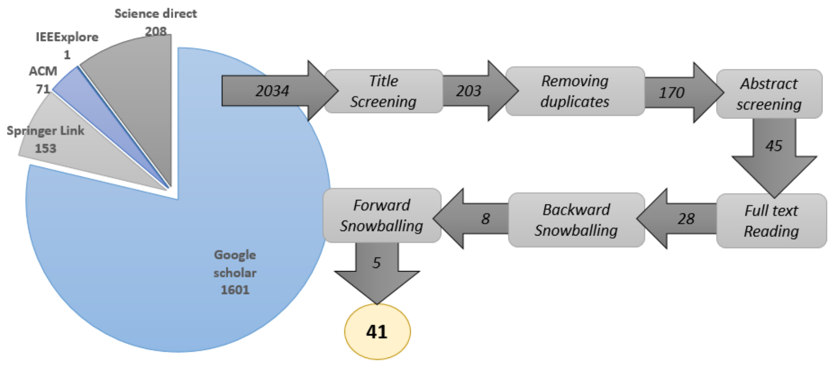


Figure 12.1: Paper Selection Process.

### Data Extraction

The selected 41 papers were carefully read, evaluated with the quality criteria (presented in Table 12.6), and compiled in Table 12.7. Then, we did a summary of every approach, which we present in Section 12.4. We also collect relevant data that could help to answer our research questions (presented in Table 12.1). To record the data for later analysis and correlation, we used Excel spreadsheets. In particular, we focused on the data presented in Table 12.5.

Table 12.6: 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. |

Table 12.7: 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 [87] | 1995 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S2 | Managing Standards compliance [88] | 1999 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S3 | Managing Process compliance [89] | 2003 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S4 | Compliance flow - managing the compliance of dynamic and complex processes [90] | 2008 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S5 | An Automatic Compliance Checking Approach for Software Processes [91] | 2009 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S6 | Supporting Qualification- Safety Standard Compliant Process Planning and Monitoring [92] | 2010 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S7 | Defining Software process model constraints with rules using OWL and SWRL [93] | 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 [94] | 2011 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S9 | NOVA Workflow: A Workflow Management Tool Targeting Health Services Delivery [95] | 2012 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S10 | Towards a process for legally compliant software [96] | 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 [97] | 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 [98] | 2015 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S13 | Cybersecurity policy verification with declarative programming [99] | 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 [100] | 2016 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |

**Table 12.7 Continued:** Selected Primary Studies.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S15 | How to Assure Correctness and Safety of Medical Software: The Hemodialysis Machine Case Study [101] | 2016 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5 |
| S16 | A Framework for Safety-Critical Process Management in Engineering Projects [102] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5 |
| S17 | Applying process mining techniques in software process appraisals [103] | 2017 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S18 | Continuous process compliance using model driven engineering [104] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S19 | Towards Increased Efficiency and Confidence in Process Compliance [105] | 2017 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S20 | Automated legal compliance checking by security policy analysis [106] | 2017 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S21 | A formalization of the ISO/IEC 15504: enabling automatic inference of capability levels [107] | 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 [108] | 2017 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S23 | Towards efficiently checking compliance against automotive security and safety standards [109] | 2017 | Workshop | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S24 | An Axiom Based Metamodel for Software Process Formalisation: An Ontology Approach [110] | 2018 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S25 | Enabling compliance checking against safety standards from SPEM 2.0 process models [111] | 2018 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S26 | Ensuring Conformance to Process Standards Through Formal Verification [112] | 2018 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S27 | Integrating formal methods into medical software development: The ASM approach [113] | 2018 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S28 | Transforming SPEM 2.0-Compatible Process Models into Models Checkable for Compliance [114] | 2018 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |

**Table 12.7 Continued:** Selected Primary Studies.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S29 | Compliance of agilized (software) development processes with safety standards: a vision [115] | 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 [116] | 2018 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S31 | Fast Compliance Checking in an OWL2 Fragment [117] | 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 [118] | 2018 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| S33 | Facilitating Automated Compliance Checking in the Safety-critical Context [119] | 2019 | Journal | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S34 | Formalising Process Assessment and Capability Determination: An Ontology Approach [120] | 2019 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S35 | Using Models to Enable Compliance Checking against the GDPR : An Experience Report [121] | 2019 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S36 | A Life Cycle for Authorization Systems Development in the GDPR Perspective [122] | 2020 | Conference | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| S37 | Co-engineering of safety and security life cycles for engineering of automotive systems [123] | 2020 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S38 | Separation of Concerns in Process Compliance Checking: Divide-and-Conquer [21] | 2020 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S39 | Reusing (Safety-oriented) Compliance Artifacts while Recertifying [124] | 2021 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S40 | Compliance-aware Engineering Process Plans: The Case of Space Software Engineering Processes [124] | 2021 | Journal | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| S41 | Supporting Quality Assurance with Automated Process-Centric Quality Constraints Checking [125] | 2021 | Conference | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |

## 12.4 Results

In this section, we report the results of the SLR. In Section 12.4.1, we present the a summary of the primary studies selected. Finally, in section 12.4.2, we present the analysis of the results in relation to the research questions of the study.

### 12.4.1 Summary of the Primary Studies

In this section, we summarize the main results obtained in the SLR. Specifically, the selected 41 papers (see Table 12.7) are categorized into 5 groups, according to the type of approach (see Table 12.8). 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 12.9, 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 12.8: 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 [126], 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) [127], 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 [128], 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) [129]. 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.

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

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 axiom 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

[13]https://alloytools.org/

[14]https://protege.stanford.edu/

[15]https://moodle.org/

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 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 trans-

[16]https://sas.cmmiinstitute.com/pars/
[17]http://www.hermit-reasoner.com/

formed 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

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

[19]https://camunda.com/

[20]http://spindle.data61.csiro.au/spindle/

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 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 $\circ$ BVR-T [130] 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 [39], an agile process for avionics [131] and Safe Scrum [132].

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.

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

**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.

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 [133]) 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

[23]https://github.com/pysmt/pysmt
[24]http://mathsat.fbk.eu.

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 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) [134], 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

[25] https://www.omg.org/spec/OCL/2.4/PDF

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 12.9: 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 |

**Table 12.9 Continued:** Summary of the Reviewed Studies.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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 |

**Table 12.9 Continued:** Summary of the Reviewed Studies.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 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]

[27] https://www.omg.org/spec/UML/
[28] https://www.w3.org/OWL/
[29] https://www.omg.org/spec/SPEM

**Table 12.9 Continued:** Summary of the Reviewed Studies.

(CWMLT) Compensable Workflow Modeling Language [135], (fUML) Foundational Subset for Executable UML Models[30],
(ASM) Abstract State Machines [134], (ASP) Answer Set Programming & (ASL) Answer Set Logic [136],
(BPMN) Business Process Management Notation[31], (SMT) Satisfiability Modulo Theories [133],
(XACML) eXtensible Access Control Markup Language, (FOL) First Order Logic, (PPST) Process Pattern Structure Tree [129],
(XSLT) Extensible Stylesheet Language Transformations[32], (SWRL) Semantic Web Rule Language[33], (OCL) Object Constraint Language[34],
(LTL) Linear Temporal Logic [137], (GAL) Governance Analysis Language [96], (DL) Description Logic[35],
(Def-L) Defeasible Logic[36], (FCL) Formal Contract Logic [138], (CTL) Computational Tree Logic, (BCL) Basic Constraint Language,
(SHACL) Shapes Constraint Language[37], (CT) Composition Trees

[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/

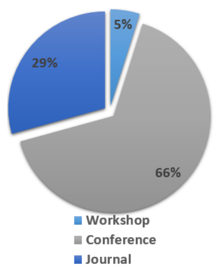
### 12.4.2 Analysis

In this section, we present the analysis of the results in relation to the addressed research questions presented in Table 12.1.

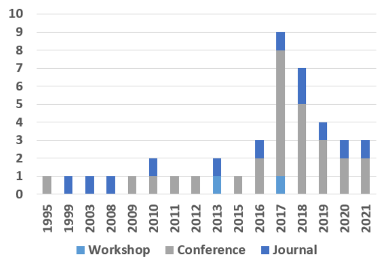
#### RQ 1. Publications Distribution

This section presents the publication distribution of the 41 primary studies resulting for the SLR (i.e., time and venue) (see Figure 12.2) and active research groups in the context automatic compliance checking of software processes (see Table 12.10). In particular, Figure 12.2a, 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.

Figure 12.2: 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 12.10

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 12.10: 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 12.2b 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 [37] 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.

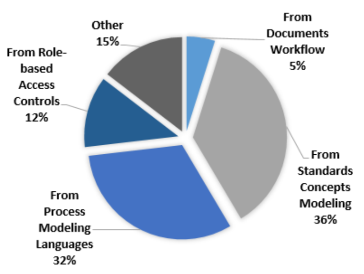
### RQ 2. Characteristics of the Methods

In this section, we present the characteristics of the methods described in the primary studies selected (summarized in Figure 12.3) by answering questions RQ 2.1, RQ 2.2, RQ 2.3 and RQ 2.4.

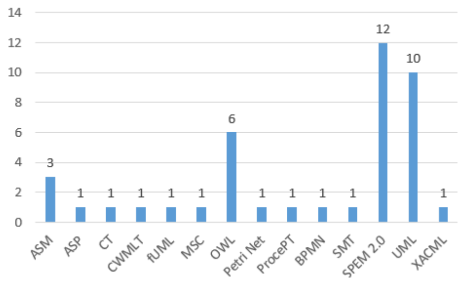
**RQ 2.1. Languages used to represent software processes entities and structures** Five types of approaches (see Table 12.8) have been used to represent the information contained in software processes. Such approaches are distributed as presented in Figure 12.3a. 36% of the primary studies, namely S3, 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 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 12.3b 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 12.9).

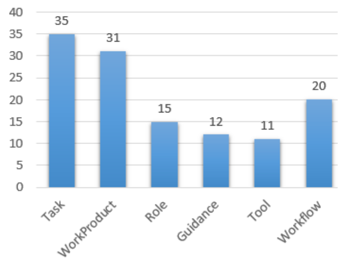
It is important to note that models created in OWL and UML could also be considered as new languages. Thus, in the end, we have more than 14 languages used for modeling software processes. We also can see in Figure 12.3c that some process elements have more importance than others in the modeling languages 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.



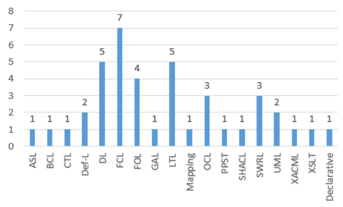
(a) General Approaches.



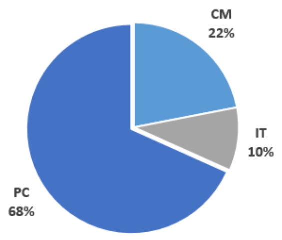
(b) Languages for Modeling Process.



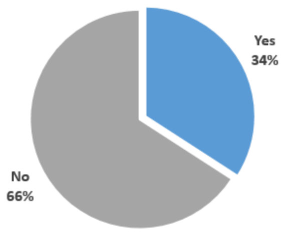
(c) Process Elements Represented.



(d) Languages for Requirements Modeling.



(e) State of the tool support.



(f) Evolution Handling Provision.

Figure 12.3: Methods Characteristics.

**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 12.3d). 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 12.4) 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 12.3e. 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 12.3f, 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 automatic compliance checking in the industry when the tool support is lacking or nonexistent.

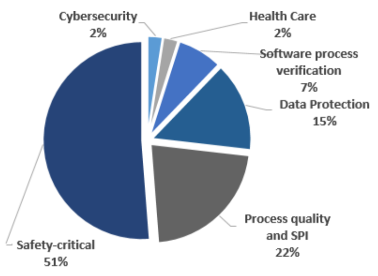
Evolution handling is a crucial aspect of process-related compliance management. However, the results of the SLR show that this aspect has been somewhat neglected. Another downside of the methods could be that the hard-coded rules could lessen the extensibility and generality and, therefore, the scope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

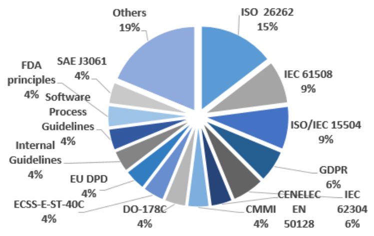
In this section, we present the potential impact of the studies in terms of application domain, normative documents targeted, illustrative Scenarios and agile support (summarized in Figure 12.4) by answering questions RQ 3.1, RQ 3.2, RQ 3.3 and RQ 3.4.

**RQ 3.1. Application domains** Several application domains are addressed in the primary studies, as presented in Figure 12.4a. 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.

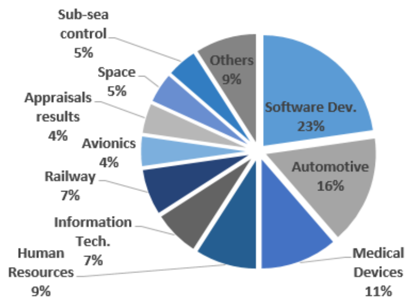
**RQ 3.2. Normative documents targeted** Different standards have been modeled and used in the experimentations or illustration results provided in the primary studies. As depicted in Figure 12.4b, the standards more used are ISO 26262 (15%), i.e., S6, S19, S23, S25, S29, S33, and S37, IEC 61508 (9%), 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



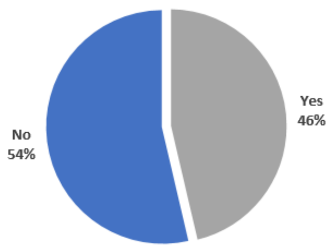
(a) Application Domain.



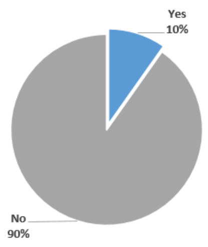
(b) Normative Frameworks Addressed.



(c) Illustrative Scenarios.



(d) Evaluation Data from Industry.



(e) Agile Support.

Figure 12.4: Potential Impact.

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 12.4c and 12.4d. 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. 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.

## 12.5 Discussion

In this section, we discuss outstanding aspects regarding automatic compliance checking of software processes based on interpretations of the authors.

### 12.5.1 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

For example, languages like SPEM 2.0 and BPMN have already defined characteristics, e.g., extensibility and reuse capabilities. 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.

### 12.5.2 Language suitability for 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.

### 12.5.3 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. [139]. 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.

### 12.5.4 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.

### 12.5.5 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.

### 12.5.6 Going beyond technological dilemmas

Article 22 of the GDPR [140] 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.

## 12.6 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 [17, 18]. 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 12.3. The first author prepared the protocol while the second and third authors (who have previously participated in research involving SLR, see for instance [141, 142]) 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 12.2) 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 12.3. 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 12.5. The first author prepared the selection criteria by considering the quality criteria presented in Table 12.6, and the research questions we intend to answer in the SLR, presented in Table 12.1. 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 12.7). The second table contains 16 columns aimed at recording the information corresponding to the research questions (see Table 12.9). 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.

## 12.7 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. [143] 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 [144] 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. [145], 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 [146] and El Kharbili et al [147], 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. [146] 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. [148] and, more recently Akhigbe et al. [149], 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. [50] is an SLR, that focuses on software process capability/maturity models. In addition, the work done by Yan et al [150] 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 [151] 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 [152], 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.

## 12.8 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 12.5). 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.

# Bibliography

[1] M. Generowicz, “The Easy Path to Functional Safety Compliance,” I&E Systems Pty Ltda, pp. 1–3, 2013. https://www.iesystems.com.au/wp-content/uploads/2015/04/Duty-of-Care-Article.pdf, Accessed March 30, 2021.

[2] I. Schieferdecker, “Responsible software engineering,” in *The Future of Software Quality Assurance*, pp. 137–146, Springer, Cham, 2020.

[3] M. Usman, M. Felderer, M. Unterkalmsteiner, E. Klotins, D. Mendez, and E. Alégroth, “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.

[4] J. P. Castellanos Ardila, B. Gallina, and G. Governatori, “Compliance-aware engineering process plans: The case of space software engineering processes,” *Artificial intelligence and law*, 2021.

[5] M. M. Rahim and S. O. Idowu, *Social Audit Regulation: Development, Challenges and Opportunities*. Springer, 2015.

[6] N. Leveson, “Safety : Why, What, and How,” *ACM Computing Surveys (CSUR)*, vol. 18, no. 2, pp. 125–163, 1986.

[7] G. Cugola and C. Ghezzi, “Software processes: a retrospective and a path to the future,” *Software Process: Improvement and Practice*, vol. 4, no. 3, pp. 101–123, 1998.

[8] P. L. Tarr and A. L. Wolf, “Introduction to “engineering of software: The continuing contributions of leon j. osterweil”,” Engineering of Software, Springer, Berlin, Heidelberg.

[9] R. Kneuper, *Software Processes and Life Cycle Models. An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes*. Springer, Cham, 2018. ISBN 978-3-319-98845-0.

[10] International Organization for Standardization/International Electrotechnical Commission, "*ISO/IEC/IEEE 12207– Systems and software engineering — Software life cycle processes*," 2017.

[11] International Organization for Standardization, "*ISO/IEC 15504 – Information technology - Process assessment*," June 2013.

[12] International Organization for Standardization - Technical Committe: ISO/IEC JTC 1/SC 7, "*ISO/IEC 330XX – Information technology - Process assessment – Concepts and Terminology.*," 2015.

[13] R. Kemp, "Regulating the safety of autonomous vehicles using artificial intelligence," *Communications Law*, vol. 24, no. 1, pp. 24–33, 2019.

[14] M. Biro, "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] I. F. Alexander, "A taxonomy of stakeholders: Human roles in system development," *International Journal of Technology and Human Interaction (IJTHI)*, vol. 1, no. 1, pp. 23–59, 2005.

[16] S. Kerrigan and K. H. Law, "Logic-based regulation compliance-assistance," Proceedings of the 9th international conference on Artificial intelligence and law, pp. 126–135, 2003.

[17] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering," Tech. Rep. 4ve, 2007.

[18] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Information and software technology*, vol. 55, no. 12, pp. 2049–2075, 2013.

[19] N. Ramasubbu, A. Bharadwaj, and G. K. Tayi, "Software process diversity: Conceptualization, measurement, and analysis of impact on project performance," *MIS Quarterly*, vol. 39, no. 4, pp. 787–808, 2015.

[20] G. Regan, M. Biro, F. Mc Caffery, K. Mc Daid, and D. Flood, "A traceability process assessment model for the medical device domain," European Conference on Software Process Improvement, pp. 206–216, Springer, 2014.

[21] J. P. Castellanos Ardila and B. Gallina, "Separation of concerns in process compliance checking: Divide-and-conquer," European Conference on Software Process Improvement, pp. 135–147, Springer, 2020.

[22] J. L. De La Vara, A. Ruiz, K. Attwood, H. Espinoza, R. K. Panesar-Walawege, Á. López, I. Del Río, and T. Kelly, "Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel," *Information and Software Technology*, vol. 72, no. C, pp. 16–30, 2016.

[23] P. Diebold and S. Scherr, "Software process models vs descriptions: What do practitioners use and need?," *Journal of Software: Evolution and Process*, vol. 29, no. 11, pp. 1–13, 2017.

[24] S. Vilkomir, J. Bowen, and A. Ghose, "Formalization and assessment of regulatory requirements for safety-critical software," *Innovations in Systems and Software Engineering*, vol. 2, no. 3-4, pp. 165–178, 2006.

[25] J. Munoz-Gama, "Conformance Checking and its Challenges," *Conformance Checking and Diagnosis in Process Mining Comparing Observed and Modeled Processes*, pp. 11–18, 2016.

[26] L. Lúcio, S. Rahman, C.-H. Cheng, and A. Mavin, "Just formal enough? automated analysis of ears requirements," NASA Formal Methods Symposium, pp. 427–434, Springer, 2017.

[27] D. Brown, H. Delseny, K. Hayhurst, and V. Wiels, "Guidance for using formal methods in a certification context," ERTS2 2010, Embedded Real Time Software & Systems, 2010.

[28] H. Harju, J. Lahtinen, J. Ranta, R. Nevalainen, and M. Johansson, "Software safety standards for the basis of certification in the nuclear domain," 7th International Conference on the Quality of Information and Communications Technology, pp. 54–62, 2010.

[29] N. Jääskinen, "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.

[30] N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky, "Checking Traces for Regulatory Conformance," Proceedings of the International Workshop on Runtime Verification, pp. 86–103, 2008.

[31] L. J. Osterweil, "Formalisms to support the definition of processes," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 198–211, 2009.

[32] D. L. Parnas and P. C. Clements, "A rational design process: How and why to fake it," *IEEE transactions on software engineering*, no. 2, pp. 251–257, 1986.

[33] J. Lonchamp, "A structured conceptual and terminological framework for software process engineering," 2nd International Conference on the Software Process-Continuous Software Process Improvement, pp. 41–53, IEEE, 1993.

[34] A. Fuggetta, "Software process: a roadmap," Conference on the Future of Software Engineering, (Orlando, Florida), pp. 25–34, 2000.

[35] W. Scacchi, "Business processes can be software too: some initial lessons learned," 3rd International Conference on the Software Process. Applying the Software Process, pp. 183–184, IEEE Computer Society, 1994.

[36] P. Henderson, "Software processes are business processes too," 3rd International Conference on the Software Process. Applying the Software Process, pp. 181–182, IEEE, 1994.

[37] L. Osterweil, "Software processes are software too," in *9th international conference on Software Engineering (ICSE 1987)*, IEE, 1987.

[38] P. G. Armour, *The Laws of Software Process: A New Model for the Production and Management of Software*. CRC Press, 2003.

[39] B. Fitzgerald, K. J. Stol, R. O'Sullivan, and D. O'Brien, "Scaling Agile Methods to Regulated Environments: An Industry Case Study," 35th

International Conference on Software Engineering (ICSE), pp. 863–872, IEEE Computer Society, 2013.

[40] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, V. Garousi, M. Felderer, K. Trektere, F. McCaffery, O. Linssen, E. Hanser, *et al.*, "Hybrid software and system development in practice: waterfall, scrum, and beyond," International Conference on Software and System Process, pp. 30–39, 2017.

[41] M. Kuhrmann, P. Diebold, J. Munch, P. Tell, K. Trektere, F. McCaffery, V. Garousi, M. Felderer, O. Linssen, E. Hanser, *et al.*, "Hybrid software development approaches in practice: a european perspective," *IEEE Software*, vol. 36, no. 4, pp. 20–31, 2018.

[42] L. A. Clarke, L. J. Osterweil, and G. S. Avrunin, "Supporting human-intensive systems," FSE/SDP workshop on Future of software engineering research, pp. 87–92, 2010.

[43] J. L. De La Vara, B. Marín, C. Ayora, and G. Giachetti, "An empirical evaluation of the use of models to improve the understanding of safety compliance needs," *Information and Software Technology*, vol. 126, p. 106351, 2020.

[44] S. Cha, R. N. Taylor, and K. Kang, *Handbook of software engineering*. Springer, 2019.

[45] N. G. Leveson, "The use of safety cases in certification and regulation," tech. rep., Massachusetts Institute of Technology. Engineering Systems Division, 2011.

[46] J. M. Fernandes and F. J. Duarte, "A reference framework for process-oriented software development organizations," *Software & Systems Modeling*, vol. 4, no. 1, pp. 94–105, 2005.

[47] M. Kuhrmann, C. Konopka, P. Nellemann, P. Diebold, and J. Münch, "Software process improvement: where is the evidence?: initial findings from a systematic mapping study," International Conference on Software and System Process, pp. 107–116, 2015.

[48] Software Engineering Institute - Carnegie Mellon University, "*CMMI for Development Version 1.3– Capability Maturity Model Integration*," 2011.

[49] M. Biro, "Open services for software process compliance engineering," SOFSEM 2014: Theory and Practice of Computer Science, (Cham), pp. 1–6, Springer International Publishing, 2014.

[50] C. G. von Wangenheim, J. C. R. Hauck, C. F. Salviano, and A. von Wangenheim, "Systematic literature review of software process capability/maturity models," International Conference on Software Process Improvement and Capabity Determination (SPICE), Pisa, Italy, 2010.

[51] Automotive SIG, "*Automotive SPICE V. 3.0 – Process Assessment/Reference Model*," 2015.

[52] P. Clarke, M. LepmetsF, F. McCaffery, A. Finnegan, A. Dorling, and D. Flood, "Mdevspice - a comprehensive solution for manufacturers and assessors of safety-critical medical device software," Software Process Improvement and Capability Determination, (Cham), pp. 274–278, Springer International Publishing, 2014.

[53] F. Stallinger, B. Henderson-Sellers, and J. Torgersson, "The oospice assessment component: Customizing," *Business Component-Based Software Engineering*, vol. 705, p. 119, 2012.

[54] ISO/IEC JTC 1/SC 7, "*ISO/IEC TS 33053 – Information technology — Process assessment — Process Reference Model (PRM) for quality management*," 2019.

[55] A. A. Khan, J. Keung, M. Niazi, S. Hussain, and A. Ahmad, "Systematic literature review and empirical investigation of barriers to process improvement in global software development: Client–vendor perspective," *Information and Software Technology*, vol. 87, pp. 180–205, 2017.

[56] International Organization for Standardization, "*ISO 9000– Quality Management Systems-Fundamentals and Vocabulary*," 2005.

[57] International Organization for Standardization, "*ISO 9001-3– Quality Management and Quality Assurance Standards - Part 3*," 1991.

[58] Internation Organization for Standardization, "*ISO/IEC 90003:2004-Software engineering – Guidelines for the application of ISO 9001:2000 to computer software*," 2004.

[59] 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.

[60] A. C. Tonini, M. D. Mesquita Spinola, and F. J. Barbin Laurindo, "Six sigma and software development process: Dmaic improvements," vol. 6 of *Technology Management for the Global Future - PICMET 2006 Conference*, pp. 2815–2823, 2006.

[61] V. Icheku, *Understanding ethics and ethical decision-making*. Xlibris Corporation, 2011.

[62] P. B. Ladkin, "Duty of care and engineering functional-safety standards," *Digital Evidence & Elec. Signature L. Rev.*, vol. 16, p. 51, 2019.

[63] N. Carroll and I. Richardson, "Software-as-a-medical device: demystifying connected health regulations," *Journal of Systems and Information Technology*, 2016.

[64] International Electrotechnical Commission, "*IEC 61508– Functional safety of electric/electronic/programmable electronic safety-related systems*," 1998.

[65] A. Schwartz, "Statutory interpretation, capture, and tort law: The regulatory compliance defense," *American Law and Economics Review*, vol. 2, no. 1, pp. 1–57, 2000.

[66] M. A. Cusumano, "Who is liable for bugs and security flaws in software?," *Communications of the ACM*, vol. 47, no. 3, pp. 25–27, 2004.

[67] S. Ingolfo, A. Siena, and J. Mylopoulos, "Establishing regulatory compliance for software requirements," International Conference on Conceptual Modeling, pp. 47–61, 2011.

[68] International Organization for Standardization - Technical Committe: ISO/TC 22/SC 32, "*ISO 26262: Road Vehicles Functional Safety*," 2018.

[69] European Committee for Electrotechnical Standardization, "*CENELEC - EN 50128. Railway Applications-Communication, Signaling and Processing Systems Software for Railway Control and Protection Systems*," 2011.

[70] European Committee for Electrotechnical Standardization, "*CENELEC - EN 50126. Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*," 2017.

[71] European Organisation for Civil Aviation Equipment & European Organisation for Civil Aviation Equipment, "*RTCA/DO-178C – Software Considerations in Airborne Systems and Equipment Certification*," 2011.

[72] Internation Organization for Standardization - Technical Committee 210, "*IEC 62304- Medical device software — Software life cycle processes*," 2006.

[73] SAE International, "*SAE J3061 – Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*," 2016.

[74] European Space Agency, "*ECSS-E-ST-40C – Space Engineering Software*," 2009.

[75] Internation Organization for Standardization, "*ISO 14971:2019 – Application of risk management to medical devices*," Dec. 2019.

[76] International Organization for Standardization - Technical Committe: ISO/IEC joint technical committee JTC 1, "*ISO/IEC 27000– Information Technology*," 2018.

[77] The European Parlament and the Council of the European Union, " *EU DPD – European Data Protection Directive*," 1995.

[78] European Parliament and Council of the European Union, "*General Data Protection Regulation (GDPR)*," 2016.

[79] Goverment of Canada, "*PIPEDA – Personal Information Protection and Electronic Documents Act*," 2000.

[80] K. Bauer, O. Hinz, W. van der Aalst, and C. Weinhardt, "Expl(ai)n it to me–explainable ai and information systems research," *Business & Information Systems Engineering*, 2021.

[81] V. Vakkuri, M. Jantunen, E. Halme, K.-K. Kemell, A. Nguyen-Duc, T. Mikkonen, and P. Abrahamsson, "Time for ai (ethics) maturity model is now," *arXiv preprint arXiv:2101.12701*, 2021.

[82] H. M. Cooper, “Organizing knowledge syntheses: A taxonomy of literature reviews,” *Knowledge in society*, vol. 1, no. 1, pp. 104–126, 1988.

[83] D. Denyer, D. Tranfield, and J. E. Van Aken, “Developing design propositions through research synthesis,” *Organization studies*, vol. 29, no. 3, pp. 393–413, 2008.

[84] H. Zhang, M. Ali-Babar, and P. Tell, “Identifying relevant studies in software engineering,” *Information and Software Technology*, vol. 53, no. 6, pp. 625–637, 2011.

[85] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10, 2014.

[86] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering–a systematic literature review,” *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[87] D. Welzel, H. Walter, and W. Schmidt, “Tailoring and conformance testing of software processes: the ProcePT approach,” Software Engineering Standards Symposium, pp. 41–49, 1995.

[88] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, “Managing Standards Compliance,” *IEEE Transactions on Software Engineering*, vol. 25, no. 6, pp. 836–851, 1999.

[89] L. Cheung, P. Chung, and R. Dawson, “Managing process compliance,” *Information management: Support systems and multimedia technology*, pp. 48–62, 2003.

[90] P. Chung, L. Cheung, and C. Machin, “Compliance Flow-Managing the Compliance of Dynamic and Complex Processes,” *Knowledge-Based Systems*, vol. 21, no. 4, pp. 332–354, 2008.

[91] X. He, J. Guo, Y. Wang, and Y. Guo, “An automatic compliance checking approach for software processes,” Asia-Pacific Software Engineering Conference, pp. 467–474, 2009.

[92] H. Jost, A. Hahn, S. Häusler, S. Köhler, J. Gačnik, F. Köster, and K. Lemmer, "Supporting qualification: Safety standard compliant process planning and monitoring," Symposium on Product Compliance Engineering, pp. 1–6, 2010.

[93] D. Rodriguez, E. Garcia, S. Sanchez, and C. R.-S. Nuzzi, "Defining software process model constraints with rules using owl and swrl," *International Journal of Software Engineering and Knowledge Engineering*, vol. 20, no. 04, pp. 533–548, 2010.

[94] R. Panesar-Walawege, M. Sabetzadeh, and L. Briand, "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.

[95] W. Maccaull and F. Rabbi, "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.

[96] W. Hassan and L. Logrippo, "Towards a process for legally compliant software," in *2013 6th International Workshop on Requirements Engineering and Law (RELAW)*, pp. 44–52, IEEE, 2013.

[97] R. Panesar-Walawege, M. Sabetzadeh, and L. Briand, "Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation," *Information and Software Technology*, vol. 55, no. 5, pp. 836–864, 2013.

[98] D.-E. Khelladi, R. Bendraou, S. Baarir, Y. Laurent, and M.-P. Gervais, "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.

[99] R. Hewett, P. Kijsanayothin, S. Bak, and M. Galbrei, "Cybersecurity policy verification with declarative programming," *Applied Intelligence*, vol. 45, no. 1, pp. 83–95, 2016.

[100] E. Kabaale, L. Wen, Z. Wang, and T. Rout, "Representing Software Process in Description Logics: An Ontology Approach for Software Process Reasoning and Verification," Software Process Improvement and Capability Determination Conference, pp. 362–376, Springer, 2016.

[101] P. Arcaini, S. Bonfanti, A. Gargantini, and E. Riccobene, “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.

[102] S. Bala, C. Cabanillas, A. Haselböck, G. Havur, J. Mendling, A. Polleres, S. Sperl, and S. Steyskal, “A Framework for Safety-Critical Process Management in Engineering Projects,” vol. 1 of *International Symposium on Data-Driven Process Discovery and Analysis*, pp. 1–27, 2017.

[103] A. M. Valle, E. A. Santos, and E. R. Loures, “Applying process mining techniques in software process appraisals,” *Information and software technology*, vol. 87, pp. 19–31, 2017.

[104] F. R. Golra, F. Dagnat, R. Bendraou, and A. Beugnard, “Continuous process compliance using model driven engineering,” in *International Conference on Model and Data Engineering*, pp. 42–56, Springer, 2017.

[105] J. Castellanos Ardila and B. Gallina, “Towards increased efficiency and confidence in process compliance,” vol. 748 of *The 24th EuroAsiaSPI Conference*, 2017.

[106] S. Ranise and H. Siswantoro, “Automated Legal Compliance Checking by Security Policy Analysis,” in *International Conference on Computer Safety, Reliability, and Security*, pp. 361–372, 2017.

[107] D. Proença and J. Borbinha, “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.

[108] P. Guarda and S. Ranise, “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.

[109] J. P. Castellanos Ardila and B. Gallina, “Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards,” 7th IEEE International Workshop on Software Certification, 2017.

[110] E. Kabaale, L. Wen, Z. Wang, and T. Rout, "An Axiom Based Metamodel for Software Process Formalisation : An Ontology Approach," vol. 2 of *International Conference on Software Process Improvement and Capability Determination*, pp. 226–240, 2017.

[111] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, "Enabling Compliance Checking against Safety Standards from SPEM 2.0 Process Models," Euromicro Conference on Software Engineering and Advanced Applications, pp. 45 – 49, 2018.

[112] E. Kabaale, L. Wen, Z. Wang, and T. Rout, "Ensuring Conformance to Process Standards Through Formal Verification," vol. 2 of *International Conference on Software Process Improvement and Capability Determination*, pp. 248–262, Springer International Publishing, 2018.

[113] P. Arcaini, S. Bonfanti, A. Gargantini, and A. Mashkoor, "Integrating formal methods into medical software development: The ASM approach," *Science of Computer Programming*, vol. 158, pp. 148–167, 2018.

[114] J. P. Castellanos Ardila, B. Gallina, and F. UL Muram, "Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance," 18th International SPICE Conference, 2018.

[115] B. Gallina, F. Ul Muram, and J. Castellanos Ardila, "Compliance of Agilized (Software) Development Processes with Safety Standards: a Vision," 4th International Workshop on Agile Development of Safety-Critical Software, pp. 1–6, 2018.

[116] D. Proença and J. Borbinha, "Formalizing ISO/IEC 15504-5 and SEI CMMI v1.3 – Enabling automatic inference of maturity and capability levels," *Computer Standards and Interfaces*, 2018.

[117] P. Bonatti, "Fast Compliance Checking in an OWL2 Fragment," in *27th International Joint Conferences on Artificial Intelligence Organization*, pp. 1746–1752, 2018.

[118] A. Bombarda, S. Bonfanti, and A. Gargantini, "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.

[119] J. Castellanos Ardila, B. Gallina, and F. Ul Muram, “Facilitating Automated Compliance Checking of Processes in the Safety-critical Context,” *Electronic Communications of the EASST*, vol. 078, pp. 1–20, 2019.

[120] E. Kabaale, L. Wen, Z. Wang, and T. Rout, “Formalising Process Assessment and Capability Determination : An Ontology Approach,” vol. 2 of *European Conference on Software Process Improvement*, pp. 594–605, Springer International Publishing, 2019.

[121] D. Torre, G. Soltana, M. Sabetzadeh, L. Briand, Y. Auffinger, and P. Goes, “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.

[122] S. Daoudagh and E. Marchetti, “A life cycle for authorization systems development in the gdpr perspective,” ITASEC, pp. 128–140, 2020.

[123] R. Bramberger, H. Martin, B. Gallina, and C. Schmittner, “Co-engineering of safety and security life cycles for engineering of automotive systems,” *ACM SIGAda Ada Letters*, vol. 39, no. 2, pp. 41–48, 2020.

[124] J. P. Castellanos Ardila and B. Gallina, “Reusing (safety-oriented) compliance artifacts while recertifying,” 9th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,, pp. 53–64, INSTICC, SciTePress, 2021.

[125] C. Mayr-Dorn, M. Vierhauser, S. Bichler, F. Keplinger, J. Cleland-Huang, A. Egyed, and T. Mehofer, “Supporting quality assurance with automated process-centric quality constraints checking,” IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021.

[126] A. Colmerauer, “An Introduction to Prolog III,” *Computational Logic*, pp. 37–79, 1990.

[127] 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.

[128] D. Cohen, “AP5 Reference Manual. https://ap5.com/doc/ap5-man.html,” 2019.

[129] J. Vanhatalo, H. Völzer, and J. Koehler, "The refined process structure tree," *Data & Knowledge Engineering*, vol. 68, no. 9, pp. 793–818, 2009.

[130] M. Javed and B. Gallina, "Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool," 22nd International Systems and Software Product Line Conference, pp. 23–28, 2018.

[131] J. Marsden, A. Windisch, R. Mayo, J. Grossi, J. Villermin, L. Fabre, and C. Aventini, "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.

[132] T. Stålhane, T. Myklebust, and G. Hanssen, "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.

[133] C. Barrett and C. Tinelli, "Satisfiability Modulo Theories," *Handbook of Model Checking*, pp. 305–335, 2018.

[134] E. Börger and R. Stark, *Abstract State Machines: A Method for High-Level System Design and Analysis*. New York, Inc.: Springer-Verlag, 2003.

[135] F. Rabbi, H. Wang, and W. MacCaull, "Compensable workflow nets," International Conference on Formal Engineering Methods, pp. 122–137, Springer, 2010.

[136] V. Lifschitz, "What is answer set programming," vol. 8 of *AAAI*, pp. 1594–7, 2008.

[137] A. Pnueli, "The temporal logic of programs," 18th Annual Symposium on Foundations of Computer Science, pp. 46–57, IEEE, 1977.

[138] G. Governatori, "Representing business contracts in RuleML," *International Journal of Cooperative Information Systems*, vol. 14, no. 02n03, pp. 181–216, 2005.

[139] A. Siena, J. Mylopoulos, A. Perini, and A. Susi, "From Laws to Requirements," in *Requirements Engineering and Law*, pp. 6–10, 2008.

[140] The European Parliament and the Council of the European Union, "*GDPR – General Data Protection Regulation*," 2016.

[141] C. Carlan, B. Gallina, and L. Soima, "Safety case maintenance: A systematic literature review," in *40th International Conference on Computer Safety, Reliability and Security*, 2021.

[142] F. u. Muram, H. Tran, and U. Zdun, "Systematic review of software behavioral model consistency checking," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–39, 2017.

[143] J. Becker, P. Delfmann, M. Eggert, and S. Schwittay, "Generalizability and Applicability of Model- Based Business Process Compliance-Checking Approaches: A State-of-the-Art Analysis and Research Roadmap," *Business Research*, vol. 5, no. 2, pp. 221–247, 2012.

[144] L. Ly, F. Maggi, M. Montali, S. Rinderle-Ma, and W. Van Der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Information Systems*, vol. 54, pp. 209–234, 2015.

[145] M. Hashmi and G. Governatori, "A methodological evaluation of business process compliance management frameworks," *Proceedings of the Asia-Pacific Conference on Business Process Management*, pp. 106–115, 2013.

[146] M. Hashmi, G. Governatori, H. Lam, and M. Wynn, "Are we done with business process compliance: state of the art and challenges ahead," *Knowledge and Information Systems*, vol. 57, no. 1, pp. 79–133, 2018.

[147] M. e. Kharbili, A. K. A. d. Medeiros, S. Stein, and W. M. van der Aalst, "Business process compliance checking: Current state and future challenges," *Modellierung betrieblicher Informationssysteme (MobIS 2008)*, 2008.

[148] G. Boella, L. Humphreys, R. Muthuri, P. Rossi, and L. Van Der Torre, "A critical analysis of legal requirements engineering from the perspective of legal practice," 7th International Workshop on Requirements Engineering and Law, pp. 14–21, 2014.

[149] O. Akhigbe, D. Amyot, and G. Richards, "A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory

compliance," *Requirements Engineering*, vol. 24, no. 4, pp. 459–481, 2019.

[150] M. Yan, X. Xia, X. Zhang, L. Xu, D. Yang, and S. Li, "Software quality assessment model: A systematic mapping study," *Science China Information Sciences*, vol. 62, no. 9, pp. 1–18, 2019.

[151] L. García-Borgoñon, M. A. Barcelona, J. A. García-García, M. Alba, and M. J. Escalona, "Software process modeling languages: A systematic literature review," *Information and Software Technology*, vol. 56, no. 2, pp. 103–116, 2014.

[152] S. Nair, J. De La Vara, M. Sabetzadeh, and L. Briand, "An extended systematic literature review on provision of evidence for safety certification," *Information and Software Technology*, vol. 56, no. 7, pp. 689–717, 2014.