



8th International Symposium  
on Leveraging Applications of Formal Methods, Verification  
and Validation

-

Doctoral Symposium and Industry Day, 2018

Facilitating Automated Compliance Checking of Processes in the  
Safety-critical Context

Julieth Patricia Castellanos Ardila  
Supervised by: Barbara Gallina and Faiz UL Muram

20 pages

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

Julieth Patricia Castellanos Ardila<sup>1</sup>

Supervised by: Barbara Gallina<sup>2</sup> and Faiz UL Muram<sup>3</sup>

<sup>1</sup> julieth.castellanos@mdh.se <sup>2</sup> barbara.gallina@mdh.se, <sup>3</sup> faiz.ul.muram@mdh.se  
School of Innovation, Design and Technology, Mälardalen University, Västerås, Sweden

**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.

**Keywords:** Compliance Checking, Engineering Processes, Safety-critical Systems, Safety Standards, SPEM 2.0, Formal Contract Logic, Regorous.

## 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 [GUC18]. 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 [JAMS17]. 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 [FD14]. In particular, SPEM 2.0 (Systems & Software Process Engineering Metamodel) [OMG08] is a well-defined standard that is used to model engineering processes. In addition, SPEM 2.0 provide support for Safety-oriented Pro-

cesses Lines Engineering (SoPLE) [GSJ12], 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 [HGW12]. 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) [Gov05]. 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 [Nut01], 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 [Gov15], a compliance checker available on the shelf, which provides traceable conclusions [SGN07].

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 2 recalls essential background information. Section 3 presents related work. Section 4 presents the research summary. Section 5 presents preliminary results. Finally, Section 6 presents the concluding remarks and next steps.

## 2 Background

This section introduces essential background required by the current research.

### 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 [VBG06]. Such standards prescribe a safety lifecycle, which describes specific activities related to assuring the safety of the system [IEC15]. 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 [GUC18]. Both, safety plan and compliance justification, should be agreed upon at the beginning of the project between the regulatory body and the applicant [BBC<sup>+</sup>10] and used to manage the execution of safety activities during the engineering of safety-critical systems.







## 2.2 Safety Standard ISO 26262

ISO 26262 [ISO18] 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.

## 2.3 Software & Systems Process Engineering Metamodel 2.0

SPEM 2.0 [OMG08] is a standard that describes *Method Content* (knowledge base of reusable elements) and *Processes*. Some elements of SPEM 2.0 are depicted in Table 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* [Ecl], 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 1: Subset of icons used in SPEM 2.0 [OMG08].

Task Definition	Task Use	Work Product	Guidance	Custom Category	Delivery Process
					

## 2.4 Safety-oriented Process Line Engineering

Safety-oriented Process Line Engineering (SoPLE) [GSJ12] 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 [Ecl], which is used to model the base processes,

and Base Variability Resolution (BVR) Tool [HØ14], 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 [JG18].

## 2.5 Defeasible Logic

Defeasible logic [ABGM00] 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: a) *facts*: indisputable statements; b) *strict rules*: rules in the classical sense, whenever the premises are indisputable, so is the conclusion; c) *defeasible rules*: rules that can be defeated by contrary evidence; d) *defeaters*: rules used only to prevent conclusions; e) *superiority relation*: a relation among rules used to define priorities. Formally,  $r: A(r) \leftrightarrow C(r)$ , a rule  $r$  consists of an antecedent  $A$ , the consequence of the rule  $C$ , and the rule  $\leftrightarrow = \{\rightarrow (\text{strict}), \Rightarrow (\text{defeasible}), \text{or } \rightsquigarrow (\text{defeater})\}$ . A defeasible proof requires that we: a) Put forward a supported rule for the conclusion we want to prove; b) consider all possible reasons against the desired conclusion; and c) rebut all counterarguments, by either showing that some premises of the counterargument do not hold, or another argument defeats the argument.

## 2.6 Formal Contract Logic

Formal Contract Logic (FCL) [Gov05] is a language based on defeasible logic (described in Section 2.5) and deontic logic of violations [GR06]. An FCL rule is represented as follows:

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

where  $a_1, \dots, a_n$  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 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*.

## 2.7 Compliance by Design Approach

Compliance by design [LSG07] 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* ( $o$ ).

Table 2: FCL rule notations [Gov05]

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

## 2.8 Regorous

Regorous [Gov15] 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 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.

## 2.9 Specification Patterns

The specification patterns, formulated by Dwyer et al.'s [DAC98], 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 3. The reader may refer to [San] 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 3: Dwyer’s specification patterns [DAC98]

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

### 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 [ESM17], the authors propose a semi-automatic compliance process to support the definition of a formal specification of software requirements. In [KWR16], 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) [WTR11] and Description Logic (DL). Similarly, in [PB18] and [SK12] approaches for enabling the definition process capability levels, according to ISO/IEC 15504 and CMMI (Capability Maturity Model Integration) v1.3 [SEI10] are presented. In [Bon18], the author presents a formalization of data usage policies in a fragment of OWL (Web Ontology Language) [OWL]. 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 [Bor96], 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 [BCCG07], 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 [BCCG07] is also used in [GDBB17], to ensure process compliance during execution time. The work presented in [RGSR10] 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) [HPB<sup>+</sup>04]. The approach in [RGSR10] is also used in [VGS12], 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 [WJJ06]. 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 [JKK11]. 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 [HKR09].

Approaches for compliance checking have been widely studied in the business context. For instance, in [GMP06], the authors propose to capture high-level policies with a compliance meta-model called REALM (Regulations Expressed As Logical Models), to support the formalization of compliance requirements in Real-time Temporal Object Logic [GLM<sup>+</sup>05]. In [KRG07], 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 [DCM<sup>+</sup>09], the authors propose a Service Oriented Architecture-based compliance governance, called COMPAS, to define compliant process fragments. In [ADW08], 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 [LGRD08], 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) [Obj11] and Temporal Logics for the modeling of regulations, e.g., [STK<sup>+</sup>10], and [EI 12]. 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 [GH15].

## 4 Proposed Research

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

### 4.1 Research Methodology

Our research methodology, which was inspired in the research methodology for information systems research proposed in [PTRC07], 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.

## 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.

### 4.3 Research Goals

Given the research motivation presented in Section 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.

## 5 Preliminary Results

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

### 5.1 Conditions for Checking Compliance in the Safety-Critical Context

As presented in Section 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 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 standards requirements, to describe a model with standard-compliant states. These conditions require the association of three components as depicted in Figure 1. 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 [Cas19].

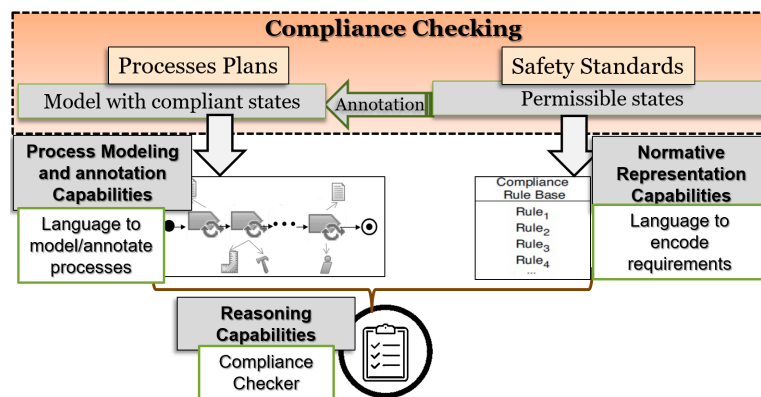


Figure 1: Required Components for Automating Compliance Checking.

## 5.2 Automated Compliance Checking Vision

Our compliance checking vision (see Figure 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 1. In particular, the vision includes the provision of a compliance rule base in FCL (recalled in Section 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 2.3), as well as a basic platform for FCL rule edition. Finally, we include Regorous (recalled in Section 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 standards 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.

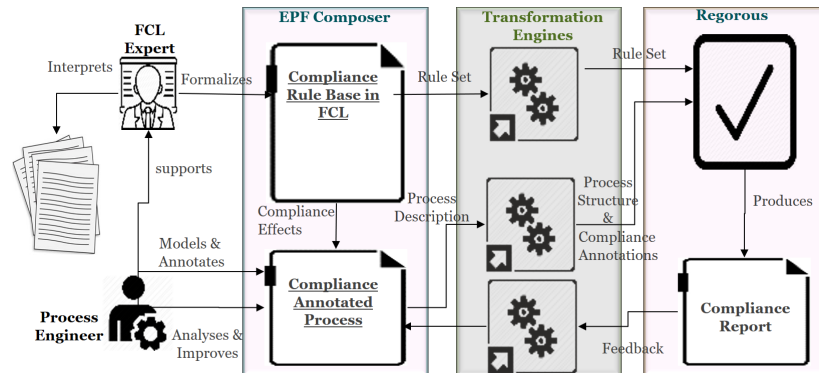


Figure 2: Automated Compliance Checking Vision [CGU18b].

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 [CGU18b, CGU18a].

### 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 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 pattern as follows: *Safety compliance patterns describe commonly occurring normative safety 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 3, by using SPEM 2.0 elements (recalled in Table 1).

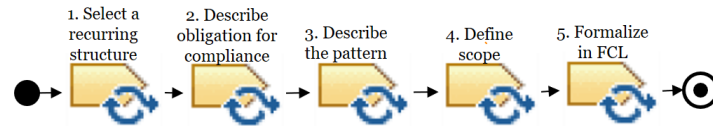


Figure 3: Methodological steps for identifying safety compliance patterns.

The first step consists of the selection of a recurring structure in the standard since, as recalled in Section 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 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 [CG17a].

#### 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 2.2) was a necessary task. The pre-processing, which is depicted in Figure 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 cars 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 [CGG18].

From the pre-processing tasks described above, we got an understanding of what to formalize



Figure 4: Pre-processing.

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 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.

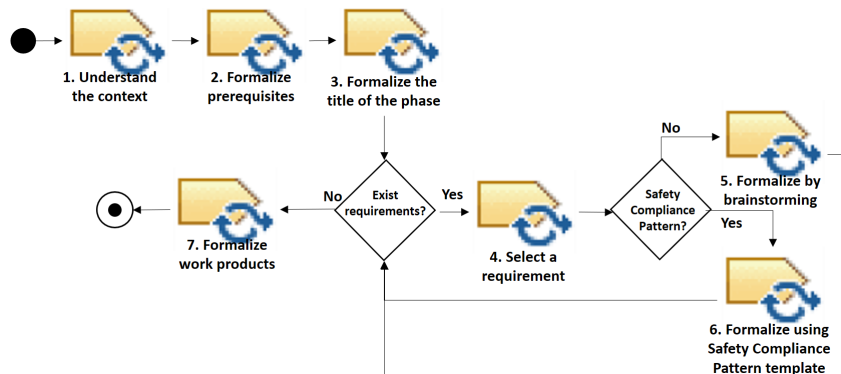


Figure 5: Methodological Guidelines.

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

Safety-oriented Process Line Engineering (SoPLE) (recalled in Section 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 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 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 [CG17c, CG17b].

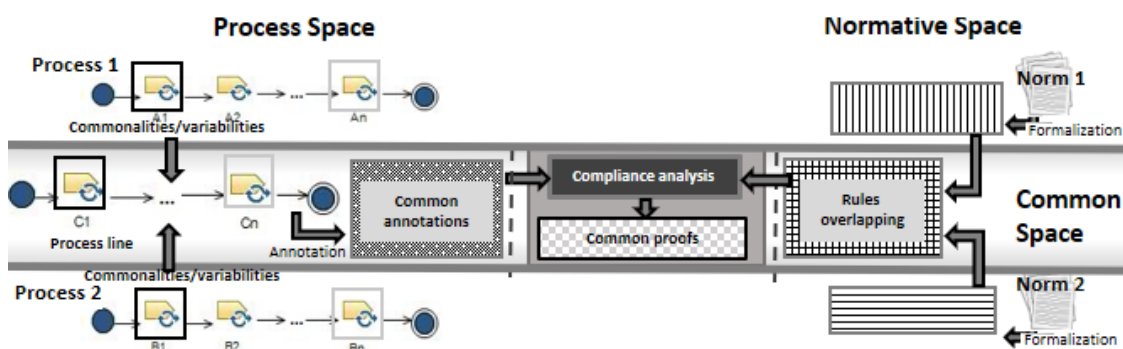


Figure 6: SoPLE&Logic-basedCM Framework.

## 6 Conclusions and Next Steps

This section present concluding remarks and next steps in the research

### 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 standards 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 standards 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.

## 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) [AMA16].

**Acknowledgements:** The work in this paper has been supported by EU and VINNOVA via the ECSEL JU project AMASS (No. 692474) [AMA16].

## Bibliography

- [ABGM00] G. Antoniou, D. Billington, G. Governatori, M. J. Maher. Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic* 2:255–287, 2000.
- [ADW08] A. Awad, G. Decker, M. Weske. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. *International Conference on Business Process Management*, pp. 326–341, 2008.
- [AMA16] AMASS. Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems. 2016.  
<http://www.amass-ecsel.eu/>
- [BBC<sup>+</sup>10] Bel V, BfS, CSN, ISTec, ONR, SSM, STUK. Licensing of safety critical software for nuclear reactors. Common position of seven European nuclear regulators and authorised technical support organisations. Technical report, 2010.
- [BCCG07] R. Bendraou, B. Combemale, X. Crégut, M. Gervais. Definition of an executable SPEM 2.0. In *14th Asia-Pacific Software Engineering Conference (ASPEC)*. Pp. 390–397. 2007.
- [Bon18] P. Bonatti. Fast Compliance Checking in an OWL2 Fragment. In *27th International Joint Conferences on Artificial Intelligence Organization (IJCAI)*. Pp. 1746–1752. 2018.
- [Bor96] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial intelligence* 82(1-2):353–367, 1996.

- [Cas19] J. Castellanos Ardila. *Facilitating Compliance Checking of Processes against Safety Standards*. Licentiate thesis, Mälardalen University, 2019.  
<http://www.diva-portal.org/smash/get/diva2:1290902/FULLTEXT02.pdf>
- [CG17a] J. P. Castellanos Ardila, 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.
- [CG17b] J. P. Castellanos Ardila, B. Gallina. Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards. In *The 7th IEEE International Workshop on Software Certification (WoSoCer)*. 2017.
- [CG17c] J. Castellanos Ardila, B. Gallina. Towards increased efficiency and confidence in process compliance. In *The 24th EuroAsiaSPI Conference*. Volume 748. 2017.
- [CGG18] J. Castellanos Ardila, B. Gallina, 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.
- [CGU18a] J. P. Castellanos Ardila, B. Gallina, 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.
- [CGU18b] J. P. Castellanos Ardila, B. Gallina, F. Ul Muram. Transforming SPEM 2.0-compatible Process Models into Models Checkable for Compliance. In *18th International SPICE Conference*. 2018.
- [DAC98] M. Dwyer, G. Avrunin, J. Corbett. Property Specification for Finite-State Verification. In *2nd Workshop on Formal Methods in Software Practice*. Pp. 7–15. 1998.
- [DCM<sup>+</sup>09] F. Daniel, F. Casati, E. Mulo, U. Zdun, S. Strauch, D. Schumm, F. Leymann, S. Sebahi, F. De Marchi, M. S. Hacid. Business compliance governance in service-oriented architectures. In *International Conference on Advanced Information Networking and Applications (AINA)*. Pp. 113–120. 2009.
- [Ecl] Eclipse Foundation. Eclipse Composer Framework.  
<https://www.eclipse.org/epf/>
- [El 12] 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.
- [ESM17] P. Engiel, J. Sampaio do Prado Leite, 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.
- [FD14] A. Fuggetta, E. Di Nitto. Software process. In *Future of Software Engineering*. Pp. 1–12. 2014.



- [GDBB17] F. Golra, F. Dagnat, R. Bendraou, A. Beugnard. Continuous Process Compliance Using Model Driven Engineering. In *International Conference on Model and Data Engineering*. Pp. 42–56. Springer, 2017.
- [GH15] G. Governatori, M. Hashmi. No Time for Compliance. *IEEE 19th International Enterprise Distributed Object Computing Workshop, (EDOCW)*, pp. 9–18, 2015.
- [GLM<sup>+</sup>05] C. Giblin, A. Liu, S. Müller, B. Pfitzmann, X. Zhou. Regulations Expressed As Logical Models (REALM). Technical report, IBM China Research Lab, 2005.
- [GMP06] C. Giblin, S. Müller, B. Pfitzmann. From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. Technical report, IBM Research Laboratory, Zurich, 2006.
- [Gov05] G. Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems* 14(02n03):181–216, 2005.
- [Gov15] G. Governatori. The Regorous Approach to Process Compliance. In *IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW)*. Pp. 33–40. IEEE, 2015.
- [GR06] G. Governatori, A. Rotolo. Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligations. *Australasian Journal of Logic* 4(4):193–215, 2006.
- [GSJ12] B. Gallina, I. Sljivo, 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.
- [GUC18] B. Gallina, F. Ul Muram, 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.
- [HGW12] M. Hashmi, G. Governatori, M. Wynn. Business process data compliance. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*. Pp. 32–46. 2012.
- [HKR09] P. Hitzler, M. Krötzsch, S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC., 2009.
- [HØ14] Ø. Haugen, O. Øgård. BVR Better Variability Results. In *International Conference on System Analysis and Modeling*. Pp. 1–15. 2014.
- [HPB<sup>+</sup>04] I. Horrocks, P. Patel-schneider, H. Boley, S. Tabet, B. Grosz, M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member submission* 21(79):1–31, 2004.
- [IEC15] IEC. Functional safety. Essential to overall safety. 2015.

- [ISO18] ISO. Road vehicles Functional safety. 2018.
- [JAMS17] J. Jiménez, J. Amelio, M. Merodio, L. Sanz. Computer Standards & Interfaces Checklists for compliance to DO-178C and DO-278A standards. *Computer Standards & Interfaces* 52:41–50, 2017.
- [JG18] M. Javed, 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.
- [JKK11] H. Jost, S. Köhler, 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.
- [KRG07] J. Küster, K. Ryndina, H. Gall. Generation of Business Process Models for Object Life Cycle Compliance. In *International Conference on Business Process Management*. Pp. 165–181. Springer, 2007.
- [KWWR16] E. Kabaale, L. Wen, Z. Wang, 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.
- [LGRD08] L. Ly, K. Göser, S. Rinderle-ma, 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 (GRCIS)*. 2008.
- [LSG07] R. Lu, S. Sadiq, G. Governatori. Compliance Aware Business Process Design. In *International Conference on Business Process Management*. Pp. 120–131. 2007.
- [Nut01] D. Nute. Defeasible Logic. In *International Conference on Applications of Prolog*. Pp. 151–169. Springer, 2001.
- [Obj11] Object Management Group. Business Process Model and Notation Version 2.0. 2011.
- [OMG08] OMG. Software & Systems Process Engineering Meta-Model Specification. Version 2.0. *OMG Std., Rev*, p. 236, 2008.
- [OWL] OWL Working Group. Web Ontology Language (OWL).
- [PB18] D. Proença, 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.
- [PTRC07] K. Peffers, T. Tuunanen, M. Rothenberger, S. Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems* 24(3):45–77, 2007.



- [RGSR10] D. Rodríguez, E. Garcia, S. Sanchez, C. Rodríguez-Solano Nuzzi. Defining software process model constraints with rules using OWL and SWRL. *International Journal of Software Engineering and Knowledge Engineering* 20(04):533–548, 2010.
- [San] Santos Laboratory. Specification Patterns. <http://patterns.projects.cs.ksu.edu/>.
- [SEI10] SEI Carnegie Mellon. CMMI® for Development, Version 1.3 CMMI-ACQ, V1.3. Technical report November, Software Engineering Institute, Carnegie Mellon, 2010.
- [SGN07] S. Sadiq, G. Governatori, K. Namiri. Modeling Control Objectives for Business Process Compliance. In *International Conference on Business Process Management*. Pp. 149–164. 2007.
- [SK12] G. Soydan, M. Kokar. A Partial Formalization of the CMMI-DEV A Capability Maturity Model for Development. *Journal of Software Engineering and Applications* 5(10):777–788, 2012.
- [STK<sup>+</sup>10] D. Schumm, O. Turetken, N. Kokash, A. Elgammal, F. Leymann, W. van den Heuvel. Business Process Compliance through Reusable Units of Compliant Processes. In *International Conference on Web Engineering*. Pp. 325–337. 2010.
- [VBG06] S. Vilkomir, J. Bowen, A. Ghose. Formalization and assessment of regulatory requirements for safety-critical software. *Innovations in Systems and Software Engineering* 2(3-4):165–178, 2006.
- [VGS12] M. Valiente, E. García-Barriocanal, 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* 42(1):61–74, 2012.
- [WJJ06] S. Wang, L. Jin, C. Jin. Represent Software Process Engineering Metamodel in Description Logic. *World Academy of Science, Engineering and Technology* 11:109–113, 2006.
- [WTR11] L. Wen, D. Tuffley, T. Rout. Using Composition Trees to Model and Compare. In *International Conference on Software Process Improvement and Capability Determination*. Volume March 2014, pp. 1–15. Springer, 2011.