Formal Contract Logic Based Patterns for Facilitating Compliance Checking against ISO 26262

Julieth Patricia Castellanos Ardila and Barbara Gallina

Mälardalen University, Box 883, 721 23 Västerås, Sweden
{julieth.castellanos,barbara.gallina}@mdh.se

Abstract. ISO 26262 demands a confirmation review of the safety plan, which includes the compliance checking of planned processes against safety requirements. Formal Contract Logic (FCL), a logic-based language stemming from business compliance, provides means to formalize normative requirements enabling automatic compliance checking. However, formalizing safety requirements in FCL requires skills, which cannot be taken for granted. In this paper, we provide a set of ISO 26262-specific FCL compliance patterns to facilitate rules formalization. First, we identify and define the patterns, based on Dwyer' et al.'s specification patterns style. Then, we instantiate the patterns to illustrate their applicability. Finally, we sketch conclusions and future work.

Keywords: ISO 26262, Confirmation review, Compliance checking, Formal Contract Logic, Safety compliance patterns.

1 Introduction

Safety critical systems designers rely on safety standards, which embody the public consensus of acceptable risk [1]. Particularly, the automotive industry has adopted the functional safety standard ISO 26262 [2], which guides the development of the safety-related systems included in a specific class of road vehicles. To claim compliance with ISO 26262 from a process perspective, necessary pieces of evidence are: *the safety plan*, which is used to manage the execution of safety activities, as well as the corresponding *confirmation review*, which includes the compliance checking of planned processes against safety requirements. In [3, 4], we have identified that automatic compliance checking of safety processes involves the definition of a finite state model of the process, where normative safety requirements provides the permissible states of the process elements. This task can be supported with available on the shelf tools. In particular, Formal Contract Logic (FCL) [5], a logic-based language stemming from business compliance, provides means to formalize normative requirements. However, formalizing safety requirements in FCL requires skills, which cannot be taken for granted. Patterns, which are *"abstractions from concrete forms which keeps recurring in specific non-arbitrary context"* [6], could represent a solution. In this paper, we follow Dwyer et al.'s specification patterns style [7], created to ease the formalization of systems requirements for finite state system model verification, to draw a general definition of safety compliance pattern. We also use specification patterns to identify and define

a set of ISO 26262-specific FCL compliance patterns. The defined patterns are instantiated to illustrate their applicability. This work will contribute to AMASS project [8], in particular, it aims at support the compliance management vision proposed, in which automotive is one of the 11 domains involved [9].

The rest of the paper is organized as follows. In Section 2, we provide essential background. In Section 3, we provide our definition of safety compliance pattern as well as our identified and defined ISO 26262-related patterns. In Section 4, we instantiate the defined safety compliance patterns. In section 5, we present related work. Finally, in Section 6, we present conclusions and future work.

2 Background

This section recalls essential information required in our work.

2.1 ISO 26262

ISO 26262 [2] addresses functional safety in automotive. The safety process influences functional safety. Thus, a confirmation review of the safety plan, which includes the compliance checking of the planned process against safety requirements is mandatory. The safety process can be either strictly planned, i.e., including all the activities provided by the reference process, or flexibly planned, i.e., by tailoring activities (omitting/performing them differently) [10]. According to ISO 26262:2, if a safety activity is tailored, *a) the tailoring shall be defined in the safety plan* and *b) a rationale as to why the tailoring is adequate and sufficient to achieve functional safety shall be available*.

From a structure perspective, ISO 26262 is divided into parts, which are subdivided into clauses. Some clauses represent phases of the safety process, which also describe activities and tasks. ISO 26262 uses Automotive Safety Integrity Levels (ASIL), which are levels to specify item's necessary safety requirements. Alternative methods to use in the planning of safety activities (e.g., tables) have to be chosen according to the higher recommendation for the ASIL assigned, but if not, a rationale shall be given that the selected methods comply with the corresponding requirement. Disjoint alternatives are also included in the text of the normative requirements. Frequently recurring expressions, which can guide the reading of the standard, can also be found, e.g., *in accordance with*. In Table 1, we recall a subset of requirements from ISO 26262:6 clause 8, which specifies the software unit design and implementation phase.

Table 1. Requirements for ISO 26262:6 clause 8.

ID	Ref	Description
R1	8	The Software unit design and implementation phase start.
R2	8.1	Specify software units in accordance with the architectural design and the associated safety requirements.
R3	8.2	The detailed design will be implemented as a model or directly as source code.
R4	8.4.2	The software unit design shall be described using specific notations, which are listed as alternative methods.

2.2 Specification Patterns

The specification patterns, formulated by Dwyer et al.'s [7], 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 2. The reader may refer to [11] 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 2. Dwyer's specification patterns [7]

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

2.3 Formal Contract Logic

Formal Contract Logic (FCL) [5] is a language designed to formalize normative requirements. FCL is implemented in Regorous, a tool developed by Data61/CSIRO in Australia[1]. An FCL rule is represented as follows:

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

$a_1, ..., a_n =$ Conditions of the applicability of the norm.

$c =$ Normative effect.

If a rule has an empty antecedent, it represents the definition of a new term. Otherwise, it represents the triggering of deontic notions, i.e., *obligations*, situations to which the bearer is legally bounded, or that should avoid, and *permissions*. If something is permitted the obligation to the contrary does not hold [12]. In the modeling of the rules, the normative effect requires a notation that clarifies the applicability of the norm (presented in Table 3). Thus, if an obligation has to be obeyed during all instants of the process interval, 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 actually in force. Otherwise, it is *non-preemptive*. If the obligation persists after being violated, it is a *perdurant obligation*, otherwise is a *non-perdurant*. A binary relation between rules ($>$) allows handling rules with conflicting conclusions.

[1] `https://research.csiro.au/data61/regorous/`.

Table 3. FCL rule notations [12]

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

3 Safety Compliance Patterns Identification and Definition

This section introduces our definition of safety compliance pattern as well as our identified and defined ISO 26262-related compliance patterns.

3.1 Our definition of Safety Compliance Pattern

As recalled in the introduction, automatic compliance checking of safety process involves the definition of a finite state model of the process, where normative safety 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 specification pattern definition (see Section 2.2) into our context as follows: ***safety compliance patterns*** *are patterns that describe commonly occurring normative safety requirements on the permissible state sequence of a finite state process model.* With this definition, we can develop a mapping between specification patterns and safety compliance patterns, as follows: the presence of a state in a system can be interpreted as the state of the obligation imposed to an element in the process, and the scope corresponds to the interval in a process when the obligations formulated by the pattern are in force. In Section 3.2, we identify the safety compliance patterns extracted from ISO 26262.

3.2 ISO 26262-related Compliance Patterns Identification

For identifying a safety compliance pattern in ISO 26262, we have delineated five methodological steps. The first step consists of the selection of a **recurring structure** in the standard since, as recalled in Section 2.1, safety requirements in ISO 26262 have implicit and explicit structures. The second step is the description of the **obligation for compliance**, namely, the reasons why the structure is required for safety compliance. The third step is the **pattern description**, based on similar (or a combination of) behaviors of the patterns described by Dwyer et al.'s (see Table 2). This description is contextualized to safety compliance, based on the mapping presented in Section 3.1. In this step, we also assign a name for the safety pattern, which reflects the related obligation for compliance. The fourth step is the definition of the **scope** of the pattern, which we also base on Dwyer et al.'s work. 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 *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, since, as recalled in Section 2.3, if something is permitted the obligation to the contrary does not hold. 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 a permission have contradictory conclusions, but the permission is superior since it represent an exception. These methodological steps have helped us to define an initial set of four ISO 26262 - related FCL compliance patterns, presented in Section 3.3. The description of our patterns has information related to the steps mentioned above. Therefore, the corresponding expressions in bold represent the elements of the pattern's description.

3.3 ISO 26262-related Compliance Patterns Definition

In what follows, we define our safety compliance patterns in ISO 26262.

Pattern: *Address Phase*. **Recurring structure:** A phase. **Obligation for compliance:** Every phase proposed by the safety model must be addressed. A phase can be omitted if tailoring is performed and a rationale is provided. **Pattern description:** *Universality + absence* - A phase must occur. Not addressing the phase requires its tailoring and the provision of a rationale. **Scope:** Global. **FCL mapping:** A maintenance obligation $address\{Phase\}$ is triggered by a previous task $\{optionalTriggeringObligation\}$, which can be empty if the phase is checked for compliance in isolation from the other phases. The permission for not addressing the phase requires two antecedents, $tailor\{Phase\}$ and $rationaleForOmiting\{Phase\}$ (See Formula 1).

$$\begin{array}{r} r : \{optionalTriggeringObligation\} \Rightarrow [OM]address\{Phase\} \\ r' : tailor\{Phase\}, rationaleForOmiting\{Phase\} \Rightarrow [P] - address\{Phase\} \\ r' > r \end{array} \quad (1)$$

Pattern: *Perform Preconditions*. **Structure:** The structure implicit in the expression *in accordance with*. **Obligation for compliance:** A task is prohibited until the preconditions are performed. **Pattern description:** *Absence + precedence* - A given task cannot occur within a scope. The task is permitted to be performed if the preconditions are performed. **Scope:** After. **FCL mapping:** A rule triggered by a previous rule $\{TriggeringObligation\}$ prohibits the performing of the task $perform\{Task\}$. The permission of performing $perform\{Task\}$ is granted after the preconditions are fulfilled $perform\{Preconditions\}$ (See Formula 2).

$$\begin{array}{r} r : \{TriggeringObligation\} \Rightarrow [OANPNP] - perform\{Task\} \\ r' : perform\{Precondition1\}, ..., perform\{PreconditionN\} \Rightarrow [P]perform\{Task\} \\ r' > r \end{array} \quad (2)$$

Pattern: *Select Disjoint Methods*. **Structure:** Structure implicit when the word *or* is used to list two methods. **Obligation for compliance:** Only one method can be selected from a list of two. **Pattern description:** *Existence + absence* - A given method can be selected within a scope. The presence of a second method derogates the selection of the first method. **Scope:** After. **FCL mapping:** A rule triggered by previous obligations $\{TriggeringObligation\}$ imposes the obligation of selecting a method $select\{Method1\}$. The selection of a second method $select\{Method2\}$, derogates the previous selection $select\{Method1\}$ (See Formula 3).

$$\begin{aligned} r &: \{TriggeringObligation\} \Rightarrow [OANPNP]select\{Method1\}] \\ r' &: select\{Method2\} \Rightarrow [P] - select\{Method1\} \\ & r'>r \end{aligned} \tag{3}$$

Pattern: *Select alternative methods*. **Structure:** Alternative methods given in tables. **Obligation for compliance:** Methods should be selected according to ASIL/recommendation levels. Alternative methods can be selected if a rationale is provided. **Pattern description:** *Response + absence* - A given obligation has to occur. The provision of a rationale grants the permission to derogates the obligation. **Scope:** After. **FCL mapping:** A rule triggered by previous obligations $\{TriggeringObligation\}$ imposes the selection of methods according to the requirements $select\{mandatoryMethods\}$. The provision of the rationale is the permission that derogates the obligation (See Formula 4).

$$\begin{aligned} r &: \{TriggeringObligation\} \Rightarrow [OANPNP]select\{mandatoryMethods\} \\ r' &: provideRationaleForNotSelect\{mandatoryMethods\} \Rightarrow [P] - select\{mandatoryMethods\} \\ & r'>r \end{aligned} \tag{4}$$

4 ISO 26262-related Compliance Patterns Instantiation

In this section, we instantiate the patterns defined in Section 3.3, using the ISO 26262 requirements presented in Table 1.

Requirement R1, which defines the phase *software unit design and specification*, can be specified by using the pattern *Address Phase*. We assume that the phase is checked in isolation from other phases (See Formula 5).

$$\begin{aligned} r_1 &:\Rightarrow [OM]addressSwUnitDesignAndImplementation \\ r_1' &: tailorSwUnitDesignAndImplementation, rationaleForOmitingSwDesignAndImplementation \\ &\Rightarrow [P] - addressSwUnitDesignAndImplementation \\ & r_1'>r_1 \end{aligned} \tag{5}$$

Requirement R2 have the expression *in accordance with*, which can be represented with the pattern *Perform Preconditions*. Specifically, the software architectural design and the associated safety requirements are preconditions to specify the software units. We assume that the triggering rule is *addressSwUnitDesignAndImplementation* (See Formula 6).

$$\begin{aligned} r_2 &: addressSwUnitDesignAndImplementation \Rightarrow [OANPNP] - performSpecifySwUnit \\ r_2' &: performProvideSwArchitecturalDesign, performProvideSafetyRequirements \Rightarrow [P]performSpecifySwUnit\{Task\} \\ & r_2'>r_2 \end{aligned} \tag{6}$$

Requirement R3 mentions the use of two disjoint implementation strategies, namely implementation as a model or directly as source code. Therefore, this requirement can be modeled using the pattern *Select Disjoint Methods*. We assume that the triggering rule is *implementingSwUnit* (See Formula 7).

$$\begin{aligned} r_3 &: implementingSwUnit \Rightarrow [OANPNP] selectImplementingAsSourceCode(X) \\ r_3' &: selectImplementingAsModel(X) \Rightarrow [P] - selectImplementingAsSourceCode(X) \\ & \qquad\qquad r_3' > r_3 \end{aligned} \tag{7}$$

Requirement R4 refers to a table with alternative entries. This requirement can be represented by using the pattern *Select alternative methods*. We assume that the triggering rule is *performSpecifySwUnit* (See Formula 8).

$$\begin{aligned} r_4 &: performSpecifySwUnit \Rightarrow [OANPNP] selectMandatoryNotationsforSwDesign \\ r_4' &: provideRationaleForNotSelectMandatoryNotationsforSwDesign \Rightarrow [P] - selectMandatoryNotationsforSwDesign \\ & \qquad\qquad r_4' > r_4 \end{aligned} \tag{8}$$

5 Related Work

Patterns for the formal specification of system safety requirements are presented in [13]. These patterns consider the cases and the terminology used in industrial automation systems to facilitate formal verification. Our patterns, instead, are restricted to process-centered requirements. Some works provide patterns for facilitating the formalization of the normative requirements for compliance checking in areas like business process compliance, e.g., the works presented in [14, 15] which extends Dwyer et al.'s specification patterns, and the work presented in [16], which uses REA (Resources, Events, and Agents) approach. A similar work, in the context of security, is presented in [17], which aims at providing a pattern structure for generating security policies for service-oriented architectures. In our work, as in some of the previously mentioned works, we use Dwyer et al.'s specification patterns as a base for providing our definition for safety compliance pattern. Also, we identified and defined the safety compliance patterns present in some structures of the standard ISO 26262. Moreover, our patterns are formalized in FCL, a formal language that is explicitly created for compliance checking, providing precise structures for modeling, e.g., deontic effects, which facilitate the expression of normative requirements in a more natural way.

6 Conclusion and Future Work

In this paper, we use Dwyer et al.'s specification patterns style to provide our definition of safety compliance patterns. Also, we identify and define set of ISO 26262-specific FCL compliance patterns, which were extracted from implicit and explicit recurring structures provided by ISO 26262. In the last part of our work, we have instantiated the defined safety compliance patterns, to illustrate their applicability.

In future, we plan to examine other clauses of ISO 26262 to apply the proposed patterns and discover additional ones. Once a complete catalogue of safety compliance patterns embracing ISO 26262 is ready, we plan to facilitate their instantiation by

providing more elaborated guidelines. Our work on safety compliance patterns is expected to be combined with previously achieved results [3, 4] regarding the provision of a framework to increase efficiency and confidence in process compliance management.

Acknowledgments. This work is supported by the EU and VINNOVA via the ECSEL JU project AMASS (No. 692474) [8].

References

1. Dunn, W.: Designing Safety-Critical Computer Systems. Computer. **36**(11) (2003) 40–46
2. ISO 26262: Road Vehicles-Functional Safety. International Standard (2011)
3. Castellanos Ardila, J., Gallina, B.: Towards Increased Efficiency and Confidence in Process Compliance. In: 24th European Conference EuroSPI. (2017) 162–174
4. Castellanos Ardila, J., Gallina, B.: Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards. In: The 7th IEEE International Workshop on Software Certification., Toulouse, France (2017)
5. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems. **14**(02n03) (2005) 181–216
6. Riehle, D., Züllighoven, H.: Understanding and using patterns in software development. Tapos **2**(1) (1996) 3–13
7. Dwyer, M., Avrunin, G., Corbett, J.: Property Specification for Finite-State Verification. In: International Conference on Software Engineering. (1998) 411–420
8. AMASS: Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems. http://www.amass-ecsel.eu/
9. AMASS: Case studies description and business impact D1.1. https://www.amass-ecsel.eu/content/deliverables. Technical report (2017)
10. Gallina, B.: How to increase efficiency with the certification of process compliance. In: The 3rd Scandinavian Conference on Systems & Software Safety. (2015)
11. Santos Laboratory: Specification Patterns. http://patterns.projects.cs.ksu.edu/
12. Hashmi, M., Governatori, G., Wynn, M.: Normative requirements for regulatory compliance: An abstract formal framework. Information Systems Frontiers. **18**(3) (2016) 429–455
13. Bitsch, F.: Safety patterns-the key to formal specification of safety requirements. Computer Safety, Reliability, and Security. (2001) 176–189
14. Namiri, K., Stojanovic, N.: Pattern-Based Design and Validation of Business Process Compliance. On the Move to Meaningful Internet Systems. (2007) 59–76
15. Elgammal, A., Turetken, O., van den Heuvel, W., Papazoglou, M.: Formalizing and applying compliance patterns for business process compliance. Software and Systems Modeling. **15**(1) (2016) 119–146
16. Karimi, V.R.: Formal Analysis of Access Control Policies for Pattern-Based Business Processes. In: World Congress on Privacy, Security, Trust and the Management of e-Business. (2009) 239–242
17. Menzel, M., Warschofsky, R., Meinel, C.: A pattern-driven generation of security policies for Service-oriented Architectures. In: IEEE International Conference on Web Services. (2010) 243–250