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

**Julieth Patricia Castellanos Ardila, Barbara Gallina**

{julieth.castellanos, barbara.gallina}@mdh.se

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

**Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems**

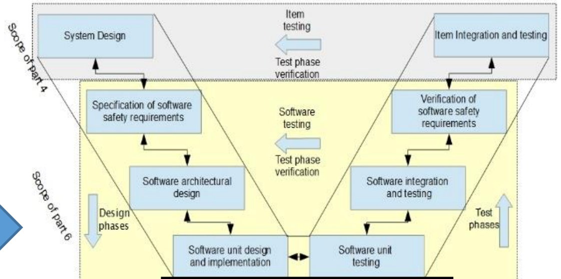
Certifiable Evidences & Justification Engineering

**TeReCom-13 December 2017, Luxembourg**

# Context and motivation

ISO 26262

Evidence from process perspective



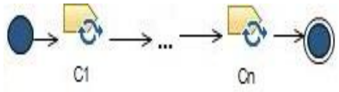
Scope of part 4
System Design
Item testing
Test phase verification
Item Integration and testing
Specification of software safety requirements
Software testing
Test phase verification
Verification of software safety requirements
Scope of part 6
Software architectural design
Software integration and testing
Design phases
Test phases
Software unit design and implementation
Software unit testing

Safety plan

+

Confirmation review

## Automatic compliance checking



Finite state model of the process

Safety requirement 1
…
Safety requirement n

Permissible states

FCL

Requires skills that can not be taken for granted!!!

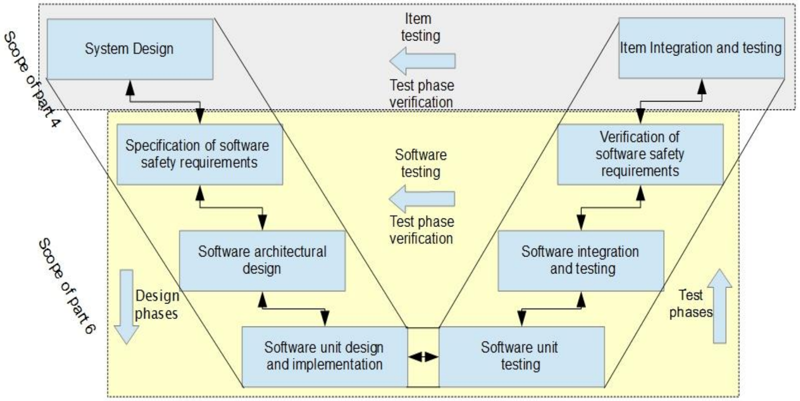
Safety Compliance Patterns

# Talk outline

- Background
  - ISO 26262
  - Specification Patterns
  - Formal Contract Logic (FCL)
- Safety Compliance Patterns
  - Our definition of safety compliance pattern
  - ISO-26262-related compliance patterns identification
  - ISO-26262-related compliance patterns definition/instantiation
- Conclusions and future work

# Background (1)

## ISO 26262 [1]



Item testing

Test phase verification

System Design

Item Integration and testing

Specification of software safety requirements

Software testing

Test phase verification

Verification of software safety requirements

Software architectural design

Software integration and testing

Software unit design and implementation

Software unit testing

Design phases

Test phases

Scope of part 4

Scope of part 6

Adapted from **ISO 26262-6:2011**: Reference phase model for the software development

☑ Confirmation review, including compliance checking of the safety plan: MANDATORY!

The safety plan can be [2]:

- Strictly planned
- Flexibly planned **(Tailoring)**

a) tailoring shall be defined in the S.P,
b) a rationale shall be provided

### Structure:

a) Divided into parts/clauses
b) Alternative methods (ASIL)
c) Disjoint alternatives
d) Frequently recurring expressions (e.g., in accordance with)

### Software unit design and implementation

| Requirements IS0 26262:6-8 | |
|---|---|
| R1 | The software unit design and implementation phase start |
| R2 | Specify software units in accordance with the architectural design and the associated safety requirements. |
| R3 | The detailed design will be implemented as a model or directly as source code. |
| R4 | The software unit design shall be described using specific notations, which are listed as alternative methods. |

[1] ISO 26262, "**Road Vehicles-Functional Safety**. International Standard." 2011.

[2] B. Gallina, "How to increase efficiency with the certification of process compliance," in *The 3rd Scandinavian Conference on Systems & Software Safety.*, 2015.

# Background (2)

## Specification patterns[3]

"Generalized descriptions of commonly occurring requirements on the permissible state sequence of a finite state model of a system."

| 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. |

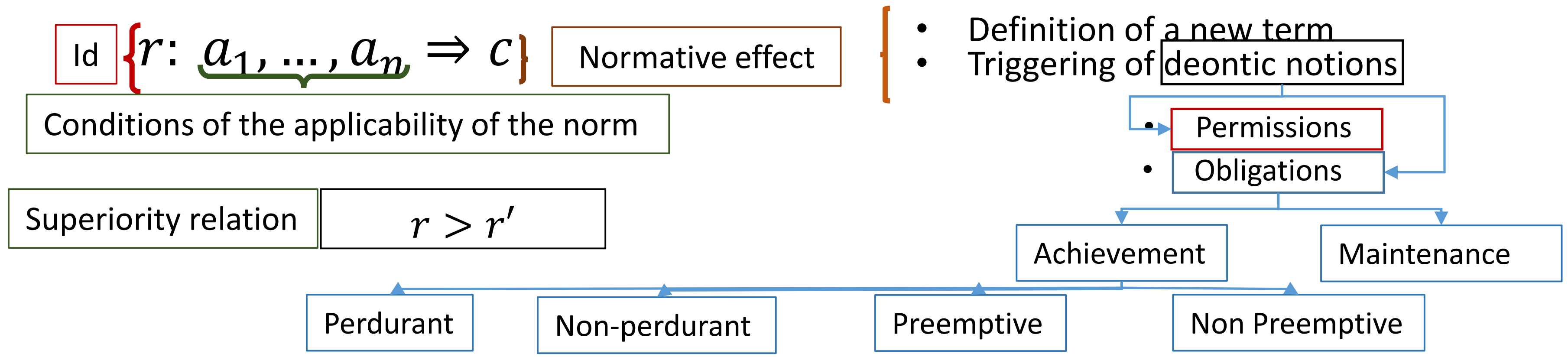
**Scope**: "The extend of the program execution over which the pattern must hold"

**a)** **Global**, which represent the entire program execution.
**b)** **After** which includes the execution after a given state.

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

# Background (3)

## Formal Contract Logic (FCL)[4] → Regorous[5]



Id $\{r: \underbrace{a_1, \ldots, a_n} \Rightarrow c\}$ Normative effect

Conditions of the applicability of the norm

Superiority relation $r > r'$

- Definition of a new term
- Triggering of deontic notions

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

[4] G. Governatori, "**Representing business contracts in RuleML**," *Int. J. Coop. Inf. Syst.*, vol. 14, no. 02n03, pp. 181–216, 2005.

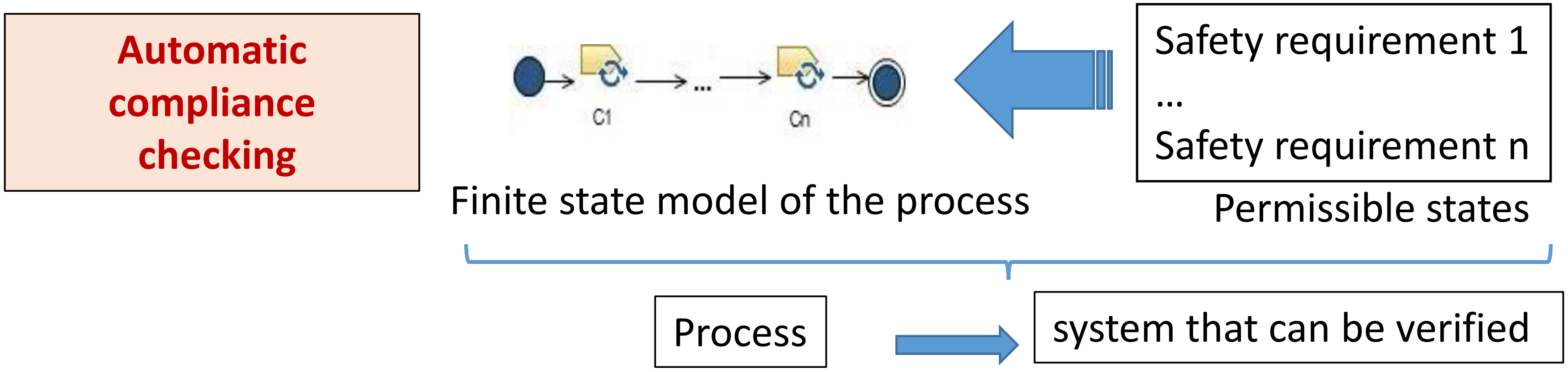
[5] https://research.csiro.au/data61/regorous/ .

# Talk outline

- Background
  - ISO 26262
  - Specification Patterns
  - Formal Contract Logic (FCL)
- Safety Compliance Patterns
  - Our definition of safety compliance pattern
  - ISO-26262-related compliance patterns identification
  - ISO-26262-related compliance patterns definition/instantiation
- Conclusions and future work

# Safety compliance patterns (1)

## Our definition of safety compliance pattern



Automatic compliance checking

Finite state model of the process

Safety requirement 1
…
Safety requirement n

Permissible states

Process

system that can be verified

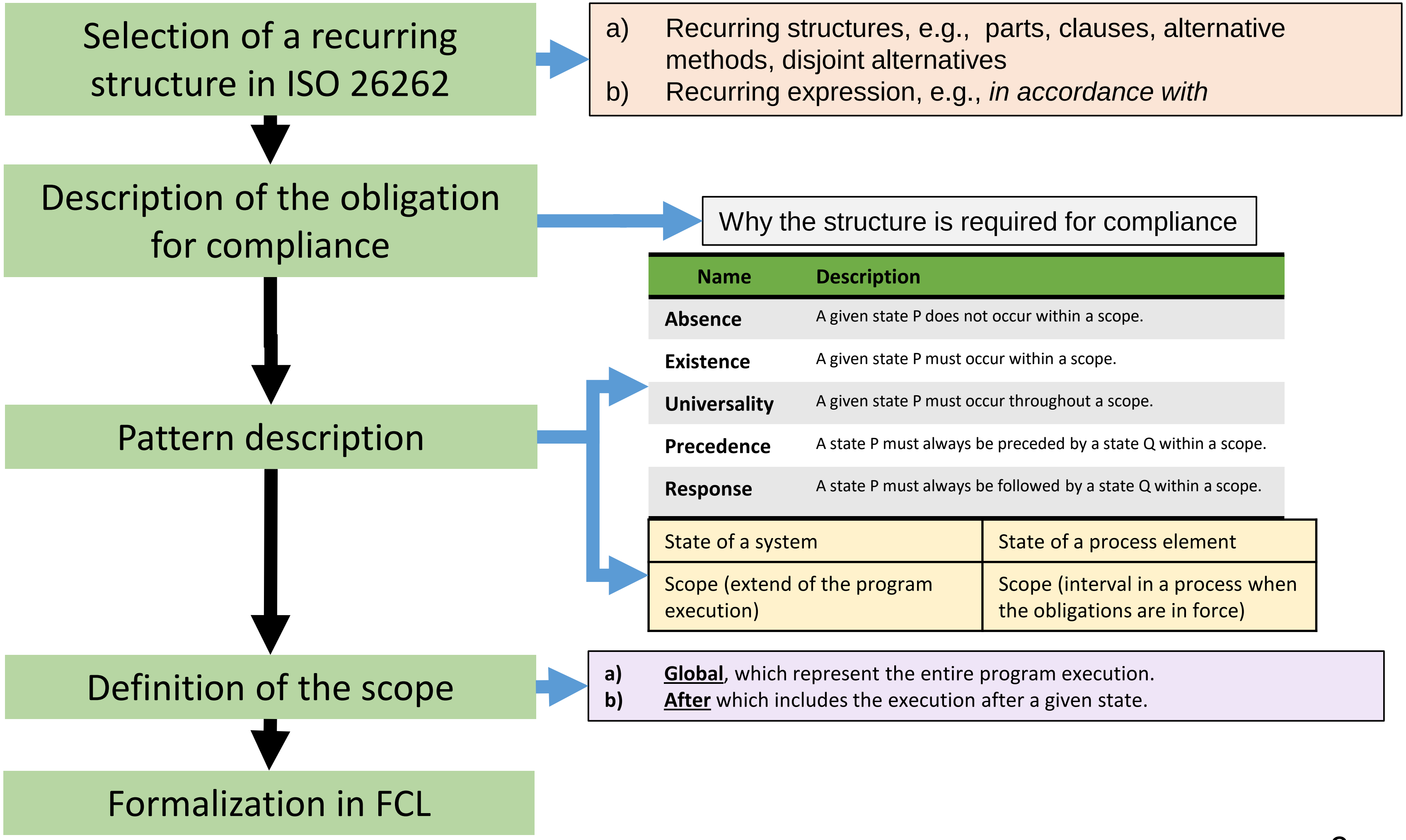
**”<u>Safety Compliance Patterns</u>** are patterns that describe commonly occurring **<u>normative safety requirements</u>** on the permissible state sequence of a finite state **<u>process model</u>**”

Specification Pattern / Safety Compliance Pattern

| | |
|---|---|
| State of a system | State of a process element |
| Scope (extend of the program execution) | Scope (interval in a process when the obligations are in force) |

# Safety compliance patterns (2)

## ISO 26262-related compliance patterns identification



Selection of a recurring structure in ISO 26262

a) Recurring structures, e.g., parts, clauses, alternative methods, disjoint alternatives
b) Recurring expression, e.g., *in accordance with*

Description of the obligation for compliance

Why the structure is required for compliance

Pattern description

| 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. |

| | |
|---|---|
| State of a system | State of a process element |
| Scope (extend of the program execution) | Scope (interval in a process when the obligations are in force) |

Definition of the scope

a) Global, which represent the entire program execution.
b) After which includes the execution after a given state.

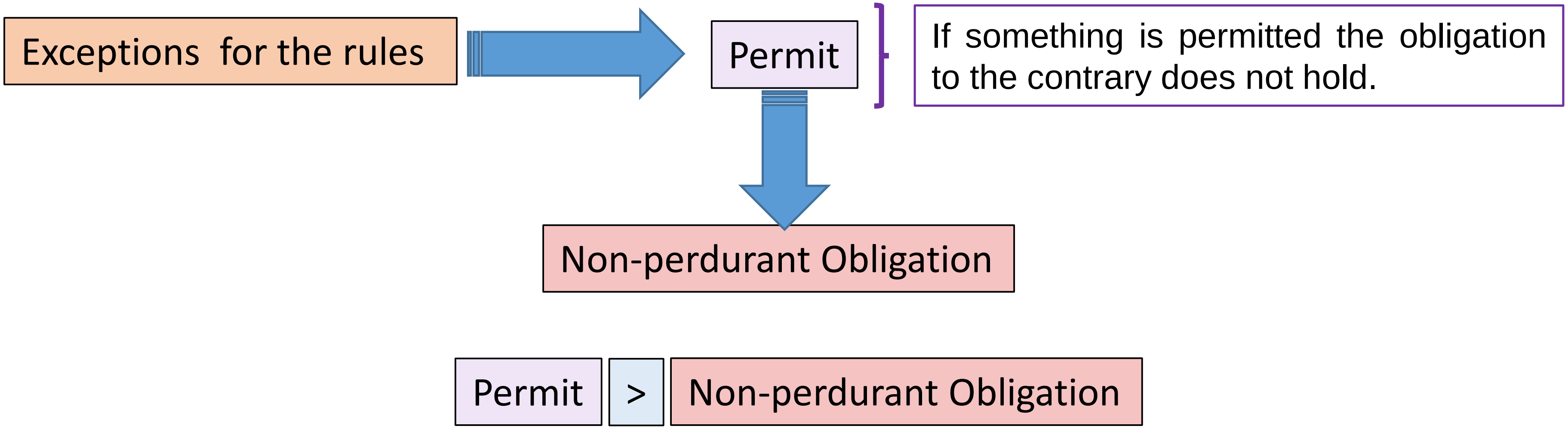
Formalization in FCL

# Safety compliance patterns (3)

## ISO 26262-related compliance patterns identification

Formalization in FCL

| **Specification patterns** | **FCL** |
|---|---|
| Global scope | Maintenence obligation |
| After scope | Non-preemptive obligation |



Exceptions for the rules

Permit

If something is permitted the obligation to the contrary does not hold.

Non-perdurant Obligation

# Safety compliance patterns (4)

## ISO 26262-related compliance patterns definition/instantiation

| Pattern | Address Phase |
|---|---|
| **Structure** | Phase |
| **Obligation** | Every phase proposed by the safety model must be addressed. A phase can be omitted if tailoring is performed and a rationale is provided |
| **Description** | (Universality + absense):A phase must occur throughout a scope. Not addressing the phase requires its tailoring and the provision of a rationale. |
| **Scope** | Global |

FCL formalization

$$\left\{\begin{array}{l} r: \{optionalTriggeringObligation\} \Rightarrow [OM]address\{Phase\} \\ r': tailor\{Phase\}, rationaleForOmmiting\{Phase\} \Rightarrow [P] - address\{Phase\} \\ r' > r \end{array}\right.$$

Pattern Instantiation →

R1 The software unit design and implementation phase start

$$r_1: \Rightarrow [OM]addressSwUnitDesingAndImplementation$$

$$r_1': tailorAddressSwUnitDesingAndImplementation, rationaleForOmmitingAddressSwUnitDesingAndImplementation$$
$$\Rightarrow [P] - addressSwUnitDesingAndImplementation$$

$$r_1' > r_1$$

# Safety compliance patterns (5)

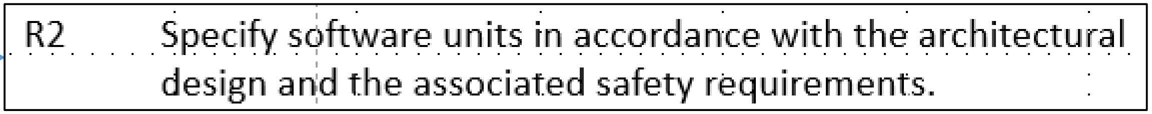
## ISO 26262-related compliance patterns definition/instantiation

| Pattern | Perform Preconditions |
|---|---|
| **Structure** | The structure implicit in the expression “***in accordance with***.” |
| **Obligation** | A task is prohibited until the preconditions are performed. |
| **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 formalization

$$
\begin{cases}
r\!: \Longrightarrow \{TriggeringObligation\} \Longrightarrow [OANPNP] - perform\{Task\} \\
r'\!: perform\{Precondition\} \Longrightarrow [P]perform\{Task\} \\
r' > r
\end{cases}
$$

Pattern Instantiation →



R2 Specify software units in accordance with the architectural design and the associated safety requirements.

$$r_2\!: addressSwUnitDesignAndImplementation \Longrightarrow [OANPNP] - performSpecifySwUnit$$

$$r_2'\!: performProvideSoftwareArchitecturalDesign, performProvideSafetyRequirements \Longrightarrow [P]performSpecifySwUnit$$

$$r_2{}' > r_2$$

# Conclusion and future work

## We have

- Use Dwyers et at.'s specification patterns to provide our definition of safety compliance pattern.
- Identify ISO 26262-specific FCL compliance patterns, extracted from implicit and explicit recurring structures.
- Instantiate the defined patterns to illustrate their applicability

## We plan to:

- Examine other ISO 26262 clauses to apply the proposed patterns and discover additional ones.
- With a complete catalog of patterns, we plan to provide a more elaborated guideline for their instantiation.
- Combine this work with previous work, regarding the provision of a framework to increase efficiency and confidence in safety process compliance management

18-21 SEPTEMBER, VÄSTERÅS, SWEDEN

SAFECOMP 2018

37TH INTERNATIONAL CONFERENCE ON COMPUTER SAFETY, RELIABILITY, & SECURITY

Thank you for your attention!

Discussion time…

# Safety compliance patterns (6)

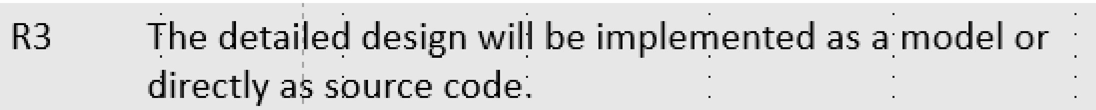
## ISO 26262-related compliance patterns definition/instantiation

| Pattern | Disjoint methods |
|---|---|
| **Structure** | The structure implicit in the word “***or***.” when it is used to list two methods |
| **Obligation** | Only one method can be selected from a list of two. |
| **Description** | (Existence + absence): A given method is selected within a scope. The presence of a second method derogates the selection of the first method.. |
| **Scope** | After. |

FCL formalization

$$\begin{cases} r\colon \Longrightarrow \{TriggeringObligaiton\} \Longrightarrow [OANPNP]select\{Method1\} \\ r'\colon select\{Method2\}, \Longrightarrow [P] - select\{Method1\} \\ r' > r \end{cases}$$

Pattern Instantiation



R3 The detailed design will be implemented as a model or directly as source code.

$$r_3\colon implementingSwUnit \Longrightarrow [OANPNP]selectImplementingAsASourceCode$$

$$r_3'\colon selectImplementingAsAModel \Longrightarrow [P] - selectImplementingAsASourceCode$$

$$r_3{}' > r_3$$

# Safety compliance patterns (7)

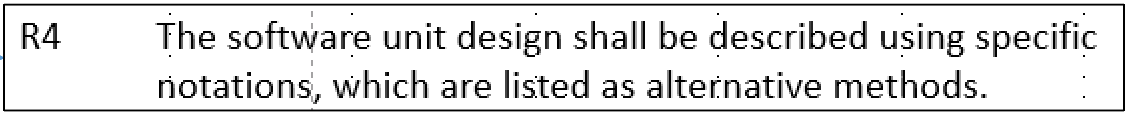
## ISO 26262-related compliance patterns definition/instantiation

| Pattern | Select alternative methods |
|---|---|
| **Structure** | Alternative methods given in tables. |
| **Obligation** | Methods should be selected according to ASIL/recommendation levels. Alternative methods can be selected if a rationale is provided |
| **Description** | (Response + absence): A given obligation has to occur. The provision of a rationale grants the permission to derogates the obligation |
| **Scope** | After. |

FCL formalization

$$\left[\begin{array}{l} r\!: \Longrightarrow \{TriggeringObligaiton\} \Longrightarrow [OANPNP]select\{MandatoryMethods\} \\ r'\!: provideRationaleForNotSelect\{MandatoryMethods\} \\ \qquad\qquad \Longrightarrow [P] - select\{MandatoryMethods\} \\ \ r' > r \end{array}\right.$$

Pattern Instantiation →



R4 The software unit design shall be described using specific notations, which are listed as alternative methods.

$$r_4\!: performSpecifySoftwareUnit \Longrightarrow [OANPNP]selectMandatoryNotationsForSwDesign$$

$$r_4'\!: provideRationaleForNotSelectMandatoryNotationsForSwDesig \Longrightarrow [P] - selectMandatoryNotationsForSwDesign$$

$$r_4{}' > r_4$$