Reliability Engineering and System Safety 180 (2018) 111–123

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Modeling patterns for reliability assessment of safety instrumented systems

Huixing Meng*,[a], Leïla Kloul[b], Antoine Rauzy[c]

[a] Laboratory of Computer Science, École Polytechnique, Paris, France
[b] DAVID, Université de Versailles St-Quentin-en-Yvelines, Versailles, France
[c] Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology, Trondheim, Norway

ARTICLE INFO

Keywords:
Modeling patterns
Reliability assessment
Safety instrumented systems
ISO/TR 12489

ABSTRACT

Safety Instrumented Systems (SIS) act as crucial safety barriers for preventing hazardous accidents in the industrial systems. It is therefore of primary importance to study their reliability, i.e. eventually to design probabilistic reliability assessment models. SIS have common behaviors such as the periodic test policies to reveal the dangerous undetected failures. These common behaviors can be captured in models via modeling patterns. By reusing modeling patterns, the modeling process can be simplified and made more efficient.

In this paper, we propose a versatile set of modeling patterns implemented in AltaRica 3.0 language. We apply them to assess the reliability of SIS described in ISO technical report ISO/TR 12489. Comparisons are performed between the results obtained from AltaRica models and those reported in ISO/TR 12489. We show that the set of proposed modeling patterns can serve as an effective tool to model SIS in a modular way.

1. Introduction

Safety Instrumented Systems (SIS) act as crucial safety barriers for preventing hazardous accidents in the industrial systems. These systems are composed of sensors, logic solvers, and final elements. Logic solvers translate signals transmitted from sensors into decisions made on final elements. SIS have attracted tremendous attention from various industrial sectors. Associated standards are proposed in several industries, such as the process industry [1], the nuclear power industry [2], the machinery industry [3,4], the automotive industry [5], and the railway industry [6–8]. The main standard is IEC 61508 [9]. The sound performance of SIS is crucial for the industrial systems.

It is therefore of primary importance to study the reliability of SIS, i.e. eventually to design probabilistic reliability assessment models. Reliability studies of SIS have been conducted extensively (see e.g., [10–13]) including proof tests [14–16], k-out-of-n voting structures [17–20], common cause failures [21–24], spurious failures [25,26], human and organizational factors [27,28], uncertainty [29–32], and optimization issues [33,34].

Modeling experience is expected to be capitalized. Otherwise, the modeling activity is unlikely to be profitable. Patterns can be utilized for reusing stabilized knowledge. However, few studies have been conducted on modeling patterns for reliability assessment of SIS.

Patterns were first formally proposed in civil engineering [35]. They have been adopted in software engineering subsequently as design patterns, which are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context [36]. A design pattern promotes design reuse, conforms to a literary style, and defines a vocabulary for discussing design [37].

A modeling pattern is a general means allowing to capture the frequently recurrent component and subsystem behaviors. Some researchers try to provide a general framework of reusing patterns. The Pattern Based System Engineering (PBSE) was proposed to develop configurable and reusable system models [38]. A PBSE procedure includes the pattern definition and the system development with patterns [39].

The reuse of systems and subsystems is a common practice in safety-critical systems engineering [40]. To reuse system behaviors, we need to standardize the representation of reusable components and clarify the way they exchange information [41]. The whole point of a pattern is thus to reuse, rather than to reinvent [37].

An advantage of high-level modeling languages, like AltaRica [42], is to *reuse* models of components or even systems [42]. The AltaRica modeling language is introduced in IEC 61508 as a technique for calculating probabilities of hardware failures in SIS [9]. The language is also referred in ISO/TR 12489 [10]. AltaRica has become a defacto European industrial standard for model-based safety assessment [43].

In this study, we propose a set of modeling patterns for reliability assessment of SIS. We classify the proposed modeling patterns into three categories. We implement these modeling patterns with the

* Corresponding author.
E-mail addresses: Huixing.Meng@hotmail.com (H. Meng), Leila.Kloul@uvsq.fr (L. Kloul), Antoine.Rauzy@ntnu.no (A. Rauzy).

https://doi.org/10.1016/j.ress.2018.06.026
Received 27 December 2017; Received in revised form 20 June 2018; Accepted 25 June 2018
Available online 20 July 2018

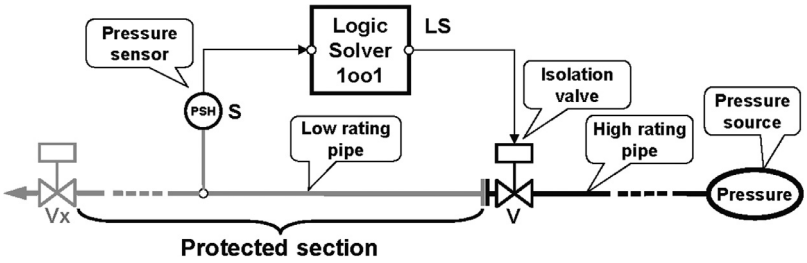


Fig. 1. An overpressure protection system with a single channel [10].

AltaRica 3.0 language. We apply these modeling patterns on all SIS in ISO/TR 12489. Preliminary results of this study have been presented at a symposium [44].

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 introduces the SIS described in ISO/TR 12489. Section 4 is dedicated to present the modeling patterns extracted from the above SIS. Section 5 discusses the modeling patterns in the framework of guarded transition systems, i.e. the mathematical background of AltaRica 3.0 language. A methodology of reusing modeling patterns is proposed in Section 6. Section 7 is devoted to show the experimental studies we conducted via modeling patterns. Eventually, Section 8 concludes this work.

2. Related works

In the RAMS (Reliability, Availability, Maintainability, and Safety) domain, patterns have been discussed [45]. Accident analyses are carried out in traffic domain [46] and industrial plants [47]. These studies apply statistical methods to discover patterns of accident causes. The dependability pattern is proposed in [39]. It is defined as the description of a particular recurring dependability problem that arises in specific contexts and presents a well-proven generic scheme for its solution. Resilience design patterns are raised to meet the demand of extreme-scale high-performance computing systems [48].

From the modeling experience of several aircraft systems using AltaRica Data-Flow language, Safety Architecture Patterns (SAP) are proposed to simplify modeling missions [49]. SAP are component assemblies used to ensure the architecture safety [49]. The application of SAP can be found in the avionics domain [49,50]. Unlike their work [49]: First, we use the AltaRica 3.0 language, which has a different mathematical foundation. Mathematical backgrounds of AltaRica Data-Flow and AltaRica 3.0 are mode automata [51] and guarded transition systems [52], respectively. Second, we propose patterns for modeling SIS in the process industry. However, their work is primarily applied in the aviation industry. Third, they mainly proposed the structured collection of redundancy-based architecture patterns. But we describe the behavioral, flow propagation, and coordination characteristics of SIS with modeling patterns.

In a recent work [53], we propose a set of modeling patterns for production-performance analysis. We apply these modeling patterns on a practical offshore installation. The two sets of modeling patterns (in [53] and this article) share some patterns, i.e. CorrectiveMaintenance, SERIES, PARALLEL, and KooN. However, most patterns are different, such as the ad hoc patterns for performance analysis of production systems and patterns for reliability assessment of SIS.

Few studies related to patterns of SIS have been conducted. Related works can be found in [10,54], where the Reliability Block Diagram (RBD) driven Petri Nets (PN) are proposed for reliability analyses. The readability of PN is improved by means of RBD. FT patterns are proposed to model safety mechanisms of automotive electric and electronic functions [55]. FT patterns include second order Safety Mechanisms (SM2) representation, maintenance, periodic tests, and the scenario without SM2.

3. Safety instrumented systems in ISO/TR 12489

We choose the SIS in ISO/TR 12489 as running examples. This is because these architectures are general enough to cover most safety systems [10]. In addition, these systems are representatives of most reliability studies of SIS performed in petroleum, petrochemical, and natural gas industries as well as in other industries [10].

Three assumptions have been made for all systems in ISO/TR 12489:

- Detected and undetected dangerous failures of a given component are independent, with exception of systems #3-2 and #3-3.
- Failure rates are constant.
- Components are as good as new after repairs.

In the following, we recall the SIS in ISO/TR 12489.

3.1. System #1: an overpressure protection system with a single channel

A basic architecture of a SIS is illustrated in Fig. 1. It is composed of a pressure sensor (S), a logic solver (LS), and an isolation valve (V). This system is applied for common safety loops with low to moderate reliability requirements (Safety Integrity Level: SIL1 to SIL2). When the pressure exceeds the predefined threshold, the sensor sends a signal to the logic solver, which in turn commands the isolation valve to close. According to different assumptions, there are four SIS generated from the system in Fig. 1. They are enumerated from #1-1 to #1-4.

The assumptions made for system #1-1 are:

- Periodic tests are perfect and performed simultaneously.
- Installation (protected section) is stopped during repairs and periodic tests.

The assumptions applied for system #1-2 are identical to system #1-1 except that:

- Periodic tests of components are not performed with the same interval.
- Two kinds of periodic tests are performed on the isolation valve:
  - Partial stroking tests to check if the valve is able to move or not;
  - Full stroking tests to check if the valve is tight after closure.

The assumptions assigned for system #1-3 are the same as #1-1 except that:

- The installation is not shut down during the repair of the sensor and of the logic solver.
- The sensor is periodically tested offline. It is no longer available for its safety function during the periodic test.

The assumptions made for system #1-4 are the same as for #1-1, except that coverages of the periodic tests are not 100%. This means that part of the Dangerous Undetected (DU) failure is not covered by periodic tests, and therefore cannot be detected.

3.2. System #2: an overpressure protection system with dual channels

System #2 is an overpressure protection system with dual channels. It is a structure with two channels (S1, LS1, and V1; S2, LS2, and V2) working in parallel. Such dual-channel architecture is commonly used for conforming to high reliability requirements (SIL2 to SIL4). According to various assumptions, four systems (#2-1, #2-2, #2-3, and #2-4) are generated from system #2.

The assumptions used for system #2-1 are the same as #1-1, except that failure rates have been split into an independent part and a common cause failure part (Beta-factor model [11,12] is applied).

Compared with system #2-1, additional assumptions are added to system #2-2. It is the same way as systems #1-1 and #1-2.

Extra assumptions are assigned to system #2-3 when compared with system #2-1. It is the same way as systems #1-1 and #1-3.

The assumptions made for system #2-4 are identical to those for #2-1 except that:

- Periodic tests of sensors and valves are staggered.
  - The pressure sensor S2 is periodically tested in the middle of the periodic test interval of S1.
  - The isolation valve V2 is periodically tested in the middle of the periodic test interval of V1.
- Each periodic test of a component provides an opportunity to detect the related potential common cause failures.

3.3. System #3: an overpressure protection system with redundant architecture

System #3 is an overpressure protection system with redundant architecture. This system is a popular architecture of SIS. It is composed of three parts in series: three pressure sensors (S1, S2, and S3) in a 2oo3 configuration, one logic solver (LS) and two parallel channels of final elements (the solenoid valve SV1 and isolation valve V1; SV2 and V2). Such an architecture is used in process industry for common safety loops with conforming to high reliability requirements (SIL3 to SIL4). Based on different assumptions, three systems (#3-1, #3-2, and #3-3) are generated from system #3.

The assumptions used for system #3-1 are identical to those for system #1-1. The assumptions assigned for system #3-2 are:

- Perfect periodic tests are performed simultaneously.
- The production is not shut down during repairs of the sensor, the logic solver, and the solenoid valves.
- This system is regarded as a subsea High Integrity Pressure Protection System (HIPPS). A maintenance rig (carrying the repair crew) is required to be mobilized for repair operations and the production is not shut down while waiting for the maintenance rig.
- Sensors and solenoid valves are periodically tested offline and are no longer available for their safety function during periodic tests.
- The production is paused during the maintenance of isolation valves.

The assumptions considered for system #3-3 are the same as those made for #3-2, except that when a dangerous failure of one sensor is detected, remaining sensors are reorganized from 2oo3 to 1oo2. This extra assumption makes system #3-3 more available than #3-2 when a dangerous failure is detected. The reason is elaborated in Eq. (4) of Section 4.2.

3.4. System #4: a multiple safety system

Fig. 2 represents a multiple safety system. It comprises two subsystems working in a predetermined sequence. The first one (S1, LS1, SV1, and V1) can be a safety loop of the BPCS (Basic Process Control System). The second one (S2, S3, LS2, SV2, SV3, V1, and V2) can be a safety loop of the ESD (Emergency Shut Down system) or a HIPPS.

The assumptions made for system #4 are:

- Perfect periodic tests are performed simultaneously.
- Installation is stopped during repairs of valves.
- Periodic test durations are negligible.

3.5. System #5: an emergency depressurization system

Fig. 3 illustrates an emergency depressurization system of a hydrocracking unit. It comes from the downstream oil and gas industry. This system is composed of two groups of temperature sensors (S1a, S1b, and S1c; S2a, S2b, and S2c) grouped in 2oo3, one logic solver (LS) and two parallel isolation valves (V1 and V2) organized in parallel and piloted by two corresponding solenoid valves (SV1 and SV2). This system aims to quickly depressurize the reactor when the temperature reaches a predetermined threshold, thus to avoid a runaway of the exothermic chemical reaction.

The assumptions used for system #5 are:

- Periodic tests are performed when the reactor is stopped.
- Installation is paused during the repair of DU failures.
- Installation is shut down during periodic tests and repair of the logic solver.
- Failures that are not covered by periodic tests will not be detected.
- The 2oo3 logic of a group of sensors is switched to 1oo2 in case of one dangerous detected failure in the group.

4. Modeling patterns

A Modeling Pattern (MP) is a general means allowing to capture the frequently recurrent component and subsystem behaviors. Modeling patterns can be categorized according to their purpose, which reflects the function of a modeling pattern. They can be classified as:

- Behavioral Patterns (BP) describe basic behaviors of components. For instance, the repairable behavior is regarded as a basic characteristic in SIS.
- Flow Propagation Patterns (FPP) depict the information or physical flows circulating components/subsystems.
- Coordination Patterns (CP) represent cooperations or synchronization in systems, such as the running scheme between a repairable unit and a repair crew.

Regarding a system, it is usually composed of subsystem(s). Similarly, a subsystem is comprised of component(s). We can implement elements (i.e., components, subsystems, and the system itself) of a system by using corresponding categories of modeling patterns, as is shown in Fig. 4. Components can be modeled by behavioral patterns. Subsystems can therefore be constructed by connecting components via flow propagation patterns. A system can thus be modeled by combining subsystems with coordination patterns.

We can take a repairable system, including a k-out-of-n (KooN) structure and a repair crew, as an example to illustrate the implementing scheme in Fig. 4. In terms of specific components in the KooN structure and teams in repair crews, we can model them by leveraging relevant behavioral patterns. Consequently, we can model this KooN subsystem and repair crew subsystem by using flow propagation patterns. Eventually, we can model the whole system by integrating the KooN and repair crew subsystems via coordination patterns.

We extract eleven (11) modeling patterns from the SIS in ISO/TR 12489. Details of these modeling patterns are illustrated in the following.

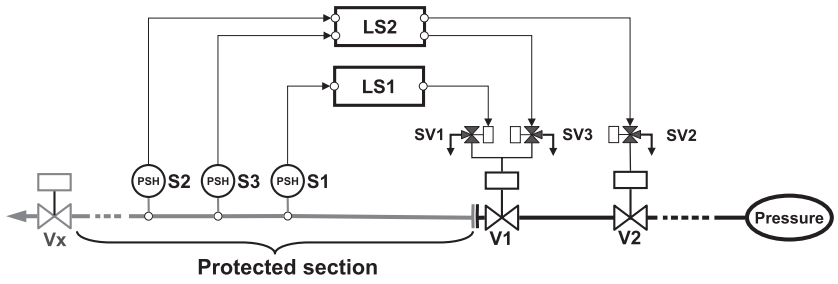


Fig. 2. A multiple safety system [10].

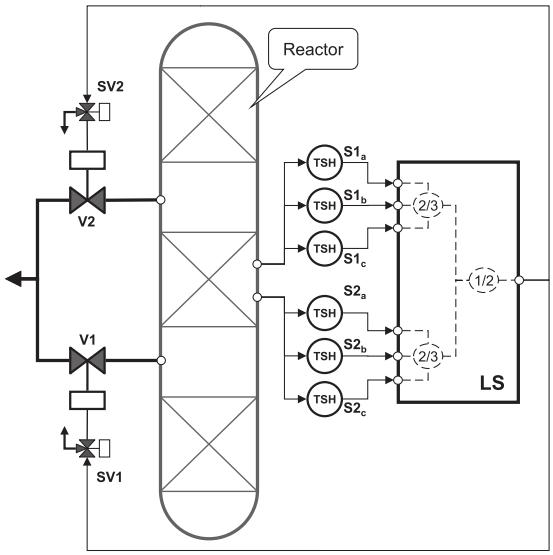


Fig. 3. An emergency depressurization system of a hydrocracking unit [10].

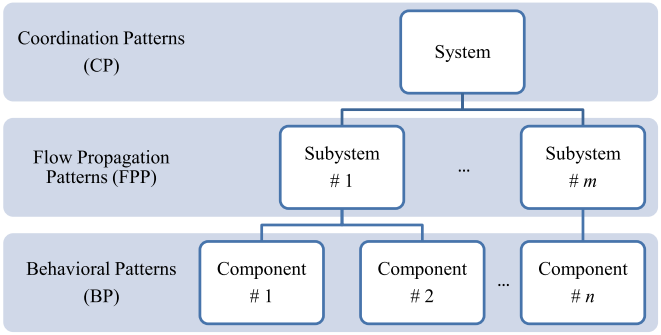


Fig. 4. Implementing elements of a system using corresponding categories of modeling patterns.

4.1. Behavioral patterns

In this part, we introduce five BP, which capture shared component behaviors.

• `NonRepairable` pattern (Fig. 5): it models components which cannot be repaired after failure. The component is initially in the `OK` state. Once a failure occurs, the component becomes `FAILED`. In SIS, DU failures are preventing activation on demand and can be revealed only by periodic tests (i.e., proof tests) [11]. Part of DU failures cannot be covered by imperfect periodic tests (i.e., the proof test coverage $<$ 100%), such as uncovered DU failures in systems ♯1-4 and ♯5, which can be modeled using this pattern. The rest part of DU failures can be covered by periodic tests.

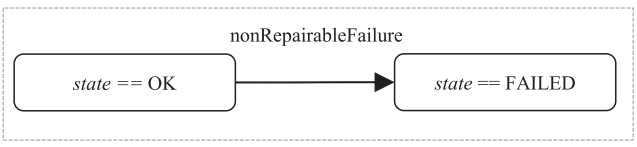


Fig. 5. Modeling pattern (MP1): `NonRepairable`.

• `CorrectiveMaintenance` pattern (Fig. 6): it models components which can be repaired after failure. The component is initially working (*state* = = `OK`). Once a failure occurs, the component falls into `FAILED` state. If the corrective maintenance team is available, the component state becomes `UNDER_REPAIR`. Finally, the component returns to the initial state once the repair operation is finished. In systems ♯1-3, ♯2-3 and ♯4, this pattern is used to model DD failures. They can be detected a short time after their occurrence by automatic diagnostic testing [11].

• `PeriodicTest` pattern (Fig. 7): it models the periodic test which can detect DU failures. Periodic tests are conducted at predefined intervals and durations. This pattern can be used to model all SIS in ISO/TR 12489.

• `StaggeredPeriodicTest` pattern (Fig. 8): it models the staggered periodic test, which is able to obtain higher availability than simultaneous tests. Compared with a reference periodic test, the duration of the first test interval in the staggered periodic test is different from the duration of following test intervals. Initially, *startStaggeredTest* is triggered. Subsequently, the rest of the pattern architecture becomes similar to the `PeriodicTest` pattern. This pattern is employed to model the staggered periodic test in system ♯2-4.

• `RevealUndetectedFailure` pattern (Fig. 9): it models the process to detect DU failures. It is based on the `PeriodicTest` and `CorrectiveMaintenance` patterns. DU failures can only be discovered when *state* = = DU and *phase* = = TEST. That is, when *state* = = DU, the periodic test can be completed only after DU failures are revealed. In the following, the component evolves as CorrectiveMaintenance pattern. This pattern can model the behaviors of revealing DU failures in all systems in ISO/TR 12489.

4.2. Flow propagation patterns

Flow Propagation Patterns (FPP) depict flow propagations inside and between components. In the following, we illustrate five FPP, which capture shared behaviors at the subsystem level.

• `SERIES` pattern (MP6) describes the series structures. It models series connection of several behavioral patterns. The average unavailability of the `SERIES` pattern $\overline{U}_{\text{SERIES}}$ is:

$$\overline{U}_{\text{SERIES}} = 1 - (1 - \overline{u}_1)(1 - \overline{u}_2)\cdots(1 - \overline{u}_n) \quad (1)$$

where $\overline{u}_1, \overline{u}_2, \cdots, \overline{u}_n$ are average unavailabilities of components $C_1, C_2, \cdots, C_n$, respectively. This pattern is used to model series structures in all systems in ISO/TR 12489.

• `PARALLEL` pattern (MP7) depicts the parallel structures. It models the parallel connection of several behavioral or `SERIES` patterns. The average unavailability of the `PARALLEL` pattern $\overline{U}_{\text{PARALLEL}}$ is:

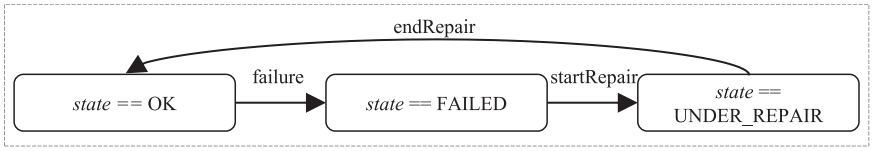


Fig. 6. Modeling pattern (MP2): `CorrectiveMaintenance`.

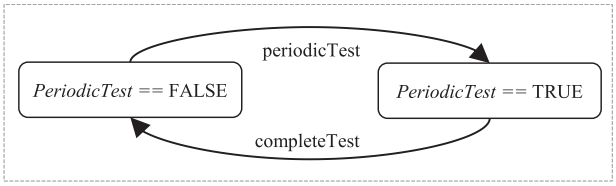


Fig. 7. Modeling pattern (MP3): `PeriodicTest`.

$$\overline{U}_{\text{PARALLEL}} = \overline{u}_1\overline{u}_2\cdots\overline{u}_n \tag{2}$$

where $\overline{u}_1, \overline{u}_2, \cdots, \overline{u}_n$ are average unavailabilities of components $C_1, C_2, \cdots, C_n$, respectively. This pattern is employed to model parallel structures in all systems in ISO/TR 12489, except the system ♯1.

• `KooN` (k-out-of-n: G) pattern (MP8) describes the structure which works when at least k of the total number n of items must be functioning. The average unavailability of the `KooN` pattern $\overline{U}_{\text{KooN}}$ is:

$$\overline{U}_{\text{KooN}} = 1 - \sum_{x=k}^{n} \binom{n}{x}(1-\overline{u})^x\overline{u}^{n-x} \tag{3}$$

where components in KooN are usually identical, and $\overline{u}$ is the average unavailability of each component. Some typical configurations of KooN structure are 1oo1, 1oo2, 2oo2, and 2oo3 [9]. `KooN` pattern is employed to model 2oo3 structures in systems ♯3-1, ♯3-2, ♯3-3, and ♯5.

• `SwitchKooN` pattern (MP9) depicts the behavior of switching a KooN structure into (K-1)-out-of-(N-1) structure when a DD or DU failure occurs. Once the failure is repaired, the structure is restored to KooN structure.

The switched configuration, (K-1)-out-of-(N-1), can increase the system availability. If there is no such a switch, the structure is supposed to work as a K-out-of-(N-1) structure after a failure. According to Eq. (3), we have:

$$\overline{U}_{\text{(K-1)-out-of-(N-1)}} - \overline{U}_{\text{K-out-of-(N-1)}} = -\binom{n-1}{k-1}(1-\overline{u})^{k-1}\overline{u}^{n-k} \tag{4}$$

where $\overline{U}_{\text{(K-1)-out-of-(N-1)}}$ and $\overline{U}_{\text{K-out-of-(N-1)}}$ are unavailabilities of (K-1)-out-of-(N-1) and K-out-of-(N-1) structures, respectively. Since $-\binom{n-1}{k-1}(1-\overline{u})^{k-1}\overline{u}^{n-k}$ is a negative value, i.e. $\overline{U}_{\text{(K-1)-out-of-(N-1)}} < \overline{U}_{\text{K-out-of-(N-1)}}$. Therefore the availability of the (K-1)-out-of-(N-1) structure is higher than that of the K-out-of-(N-1) structure.

This pattern is used in systems ♯3-3 and ♯5. If a dangerous failure (DD or DU) occurs in a 2oo3 structure, the logic solver changes the policy from 2oo3 to 1oo2.

There is a relationship between the logic policy in IEC 61508 [9] and the proposed `SwithKooN` pattern in this study. Since the safety level of the structure increases after changing the logic, this pattern has been recommended by IEC 61508. We concreted the scheme in detail by implementing it with a specific modeling language (i.e. AltaRica 3.0). In addition, we proposed mathematical formula to uncover the mechanism of this `SwithKooN` pattern.

• `SequentialWork` pattern (Fig. 10) depicts the multiple SIS which work in a sequential order, such as in system ♯4. The failed state of the previous subsystem $i-1$ triggers the successive subsystem i. This one is initially out of work (*subSystemState* $==0$). If the trigger action (startDemand) from subsystem $i-1$ is perfect, the subsystem i becomes working (*subSystemState* $==1$). If the subsystem i fails (*subSystemState* $==2$), it can be used to trigger the working of subsystem $i+1$. Note that if the trigger action is perfect, `SequentialWork` is equivalent to the `PARALLEL` pattern.

4.3. Coordination patterns

Coordination Patterns (CP) represent cooperation or synchronization in a system. Here we exhibit a coordination pattern used in ISO/TR 12489.

• `Repairable unit/Repair crew Coordination` pattern (Fig. 11): it models limited repair crews in SIS. The working state of the repair crew (*RepairCrewWork*) is FALSE initially. If the number of busy repair crews (*numberBusyCrew*) is lower than the total number of repair crews (*totalNumberCrew*) and a repair is required, the repair is started. Simultaneously, 1 is added to *numberBusyCrew*. Adversely, 1 is decreased to *numberBusyCrew* when the repair is completed. This pattern is employed to model the limited repair crews working in systems ♯3-2 and ♯3-3.

5. Implementation of modeling patterns

We implement the proposed modeling patterns in the framework of Guarded Transition Systems (GTS). GTS are the mathematical background of the AltaRica 3.0 language. They are used to implement modeling patterns throughout this work.

5.1. Guarded transition systems

A GTS is a quintuple $\langle V, E, T, A, \iota\rangle$, where V is a finite set of variables, E is a finite set of events, T is a finite set of transitions, A is an assertion, and ι is the initial assignment of variables. Further details of GTS and AltaRica 3.0 can be found in [42,52,56].

(i) V is the disjoint union of the set S of state variables and the set F of flow variables: $V = S \uplus F$. Each variable $v \in V$ takes its value from a domain denoted by $domain(v)$. Variables can be Boolean, Integers, Floating point numbers, members of finite sets of symbolic constants or anything convenient for the modeling purpose.

A variable assignment is a function from V to $\Pi_{v\in V} domain(v)$. A

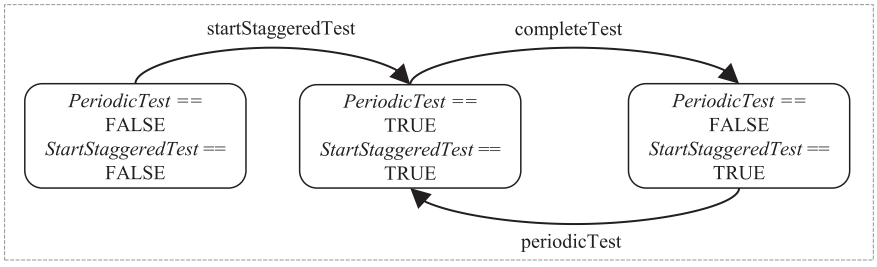


Fig. 8. Modeling pattern (MP4): `StaggeredPeriodicTest`.

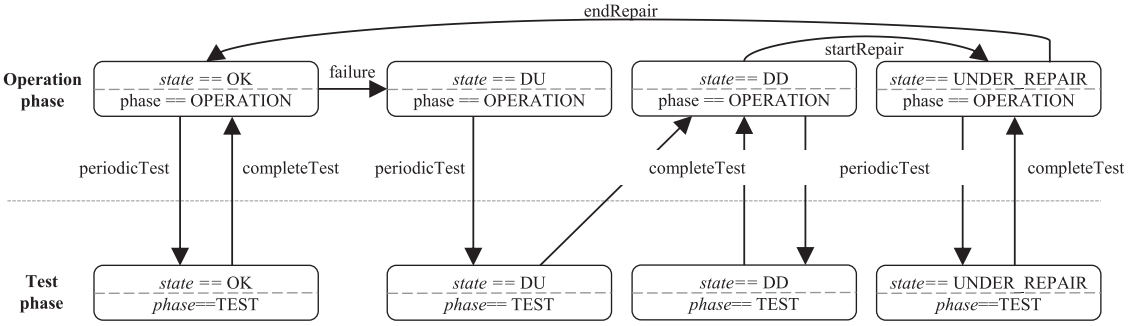


Fig. 9. Modeling pattern (MP5): `RevealUndetectedFailure`.

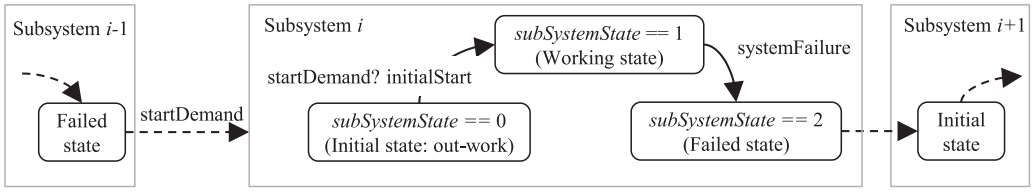


Fig. 10. Modeling pattern (MP10): `SequentialWork`.

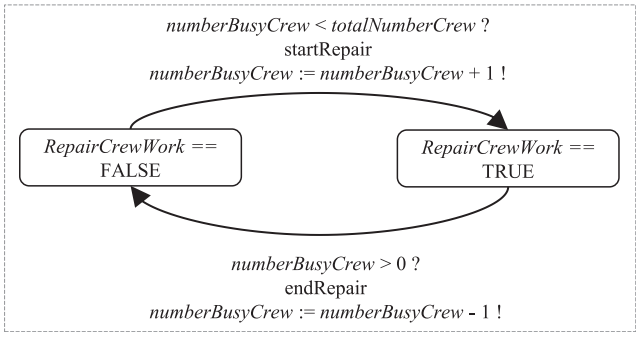


Fig. 11. Modeling pattern (MP11): `Repairable unit/Repair crew Coordination`.

variable update is a function from $\Pi_{v \in V} domain(v)$ into itself. It is a function that transforms a variable assignment into another one.

(ii) Each event $e \in E$ is associated with:

– A monotonically increasing and invertible function $delay_e$ from [0, 1] into $\Re^+$, the set of positive real numbers.

– A weight (a real number) $weight_e$ (by default, $weight_e = 1.0$).

(iii) Each transition $t \in T$ is a triple $\langle e, g, a \rangle$, denoted by $g \xrightarrow{e} a$, where e is an event in E, g is a Boolean condition (guard) over the variables in V and a is an instruction over the variables of V, that is a variable update. a is called the action of the transition.

(iv) The assertion A is an instruction over the variables of V.

Let σ be a variable assignment and t: $g \xrightarrow{e} a$ be a transition which is potentially fireable in σ, such that $\sigma(g) = true$. Firing t updates σ into the assignment $\rho = A(a(\sigma))$, which means applying on σ the update of a first, then the update A (the global assertion).

We say that a variable $v \in V$ is impacted by the update of σ into ρ if $\rho(v) \neq \sigma(v)$. By extension, we say that the transition $g' \xrightarrow{e'} a'$ is affected by this variable update if at least one of the variables occurring in g' is impacted by the update.

Let t: $g \xrightarrow{e} a$ be a transition in T. By extension, we define $weight_t$ as $weight_e$. If the two transitions can be fired at the same time, then the weight is used to choose randomly among them.

A Petri Net (PN) is a bipartite graph with two kinds of nodes, places and transitions, and directed arcs, to model local states and local events, respectively [57]. Generalized Stochastic Petri Nets (GSPN) extend Petri nets by associating with each transition a random variable representing its duration [58]. This random variable is assumed either to be constantly equal to 0 (immediate transitions) or to be exponentially distributed (with a given transition rate). GSPN have been widely applied. Efcient algorithms and tools are applicable to further study these models. Nevertheless, despite their capabilities, these formalisms share a major drawback: models designed with these formalisms are far from the functional architecture of the system under study. Consequently, models are difficult to design and to maintain throughout the life cycle of systems. A small change in the specifications may require a complete revisit of the safety models, which is both resource consuming and error prone [59].

GSPN and AltaRica can be compared according to their ways of constructing models (e.g. events, compositions, hierarchies, remote interactions between components, graphical representations) and analyzing models (available assessment tools and interpretation of time). GSPN and AltaRica can represent the state space in an implicit way [60].

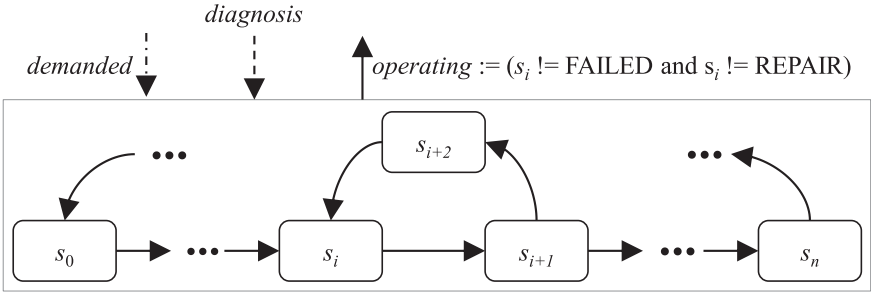


Fig. 12. Guarded transition system for behavioral patterns.

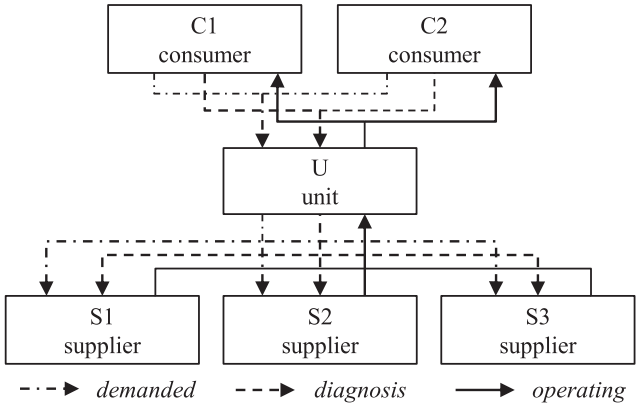


Fig. 13. Flow propagations.

The GSPN model can be extended by introducing transitions with a deterministic firing time (Deterministic Stochastic Petri nets; DSPNs) [61,62]. Petri nets-based high level formalisms, especially GSPN and DSPN, can model complex processes (including static/dynamic time-dependent behavior and stochastic processes) [63]. They have understandable graphical representation, and powerful mathematical representation. However, when they become large, they can be very confusing for users.

GTS generalize most of the formalisms used for probabilistic safety analyses, including static fault trees, reliability block diagrams and GSPN [64]. The definition of a semantic for dynamic fault trees involves actually multi-state components, immediate and timed (stochastic) transitions as in GSPN [58] (these concepts are not available in block diagrams or static fault trees), as well as block-wise construction and remote value propagation as in reliability block diagrams (these concepts are not available in generalized stochastic Petri nets) [64].

In our work, we applied the GTS, which is the mathematical background of AltaRica 3.0. The GTS framework is both more expressive than SPN with assertions and predicates and cleaner from a mathematical standpoint [65]. Moreover, this generalization and clarification is obtained at no algorithmic cost [65]. The pattern approach makes models not only much easier to design and to debug, but also to share with stakeholders and to maintain throughout the lifecycle of systems [65]. Stochastic Petri nets with assertions and predicates are quite difficult to master when the system under study gets complex. Models tend to be hardly readable [65]. A way to improve the modeling methodology is to design models by tailoring and composing modeling patterns [65]. Stochastic guarded transition systems provides a clear and powerful mathematical framework for that purpose [65].

5.2. Implement modeling patterns using GTS

5.2.1. Behavioral patterns

Behavioral patterns capture the phenomena of internal state transitions at the component level. GTS for a generic BP is shown in Fig. 12. A BP is typically comprised of the following elements:

– State variables: s_i indicates the internal state of the component. s_0 stands for the initial state. Several applicable state variables could identify component phases (e.g. the periodic test).
– Flow variables: *demanded, diagnosis*, and *operating*. *demanded* indicates the activation demand of a component. *diagnosis* indicates the diagnostic test used to discover Dangerous Detected (DD) failures of a component. *operating* indicates whether the component works or not.
– Events: changes between states with immediate or stochastic delays.
– Assertions: since the left member of an assertion is a flow variable, *demanded, diagnosis*, and *operating* are assigned in assertions.
– Input variables: *demanded* and *diagnosis*.
– Output variables: *operating*.

According to whether the failures can be detected (almost) immediately or not, failures in SIS can be categorized into dangerous detected (DD) failures and dangerous undetected (DU) failures. There are two types of corresponding tests for discovering SIS failures, i.e., diagnostic tests and proof tests. Diagnostic tests are usually conducted

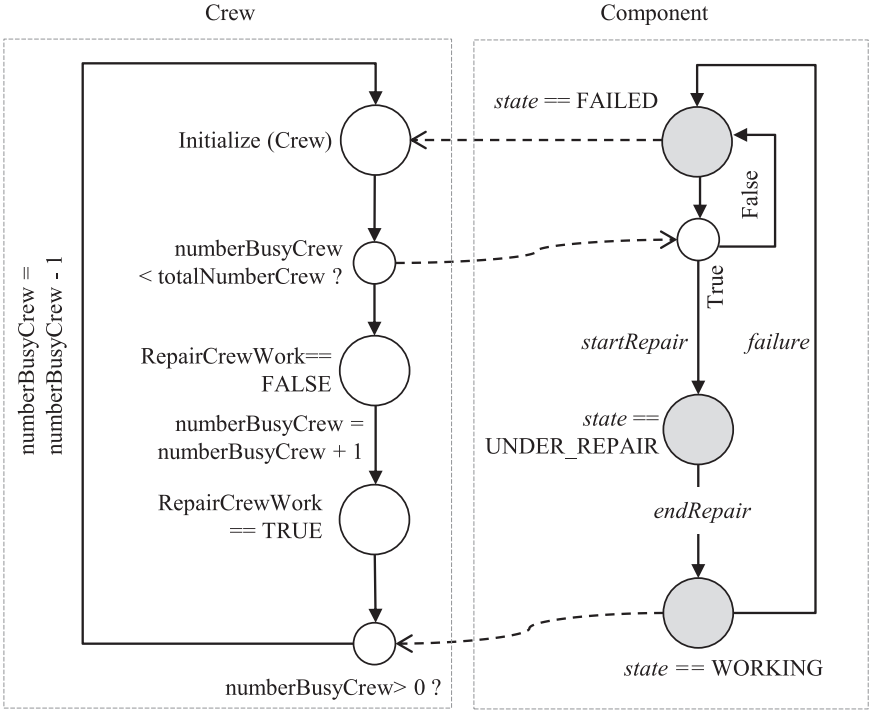


Fig. 14. Synchronization in RRC.

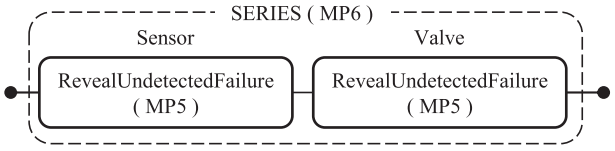
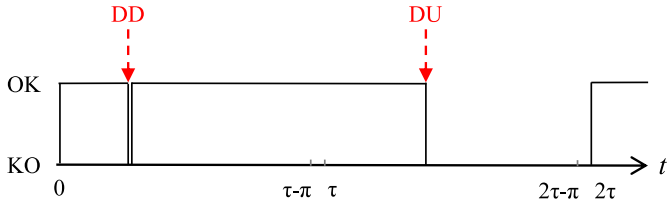


Fig. 15. Pattern-based model of safety instrumented system ♯1-1.



τ: Proof test interval. π: Proof test duration.

Fig. 16. The working scheme of the sensor in system ♯1-1.

with a quite high frequency (i.e., very short test interval). DD failures can be discovered immediately by diagnostic tests. After being detected, behaviors of DD failures can be modeled by the proposed `CorrectiveMaintenance` pattern. Proof tests are normally carried out with a relative low frequency (i.e., long test interval), which intend to reveal DU failures. In ISO/TR 12489 [10], the periodical test refers to proof test. In this study, *diagnosis* flow refers to diagnostic tests.

Consider for instance the `CorrectiveMaintenance` pattern, it is composed of the following elements:

– A state variable: *state*. It takes its value from the enumeration {OK, FAILED, UNDER_REPAIR}. The initial state is OK.
– A Boolean flow variable: *operating*.
– Three events: *failure, startRepair*, and *endRepair*. *startRepair* is immediate. *failure* and *endRepair* are stochastic.
– Three transitions:
  – *failure: state* = = WORKING → *state* := FAILED
  – *startRepair: state* = = FAILED → *state* := UNDER_REPAIR
  – *endRepair: state* = = UNDER_REPAIR → *state* := WORKING
– An assertion: *operating* := (*state* = = WORKING)

5.2.2. Flow propagation patterns

Flow propagation patterns capture flow-circulating behaviors between components. A FPP is typically comprised of the following elements:

– State variables: the current states are assigned by BP.
– Flow variables: *demanded, diagnosis*, and *operating*.
– Events: since the transitions are mainly between state variables, there are few events for FPP.
– Assertions: a set of assignments for flow variables.
– Input variables: *demanded* and *diagnosis*.
– Output variables: *operating*.

Consider the flow propagations in a system depicted in Fig. 13, unit U has three suppliers (upstream units S1, S2, and S3) and two consumers (downstream units C1 and C2). The unit can be a single component or a macro component (i.e. composed of several components). The *demanded* flow of each unit is composed of *demandedIn* and *demandedOut*. Likewise, each block is associated with *operatingIn* and *operatingOut*. This scheme can be extended to more suppliers and consumers. Regarding unit U, we have:

$$
\begin{aligned}
U.\,demandedIn &:= C1.\,demandedOut \text{ or } C2.\,demandedOut\\
U.\,demandedOut &:= U.\,demandedIn\\
S1.\,demandedIn &:= U.\,demandedOut\\
S2.\,demandedIn &:= U.\,demandedOut\\
S3.\,demandedIn &:= U.\,demandedOut\\
U.\,operatingIn &:= S1.\,operating \text{ or } S2.\,operating \text{ or } S3.\,operating\\
U.\,operatingOut &:= U.\,operatingIn \text{ and } U.\,s \text{ !=FAILED and}\\
&\quad\ U.\,s \text{ ! = REPAIR and } U.\,diagnosis
\end{aligned}
$$

5.2.3. Coordination pattern

Fig. 14 depicts the synchronization in the `Repairable unit/`

```
// ------------------------------------------------------------------------------
domain ComponentState {OK, DU, DD, UNDER_REPAIR} //States of component
domain Phase {OPERATION, TEST} //Phases of component
// ------------------------------------------------------------------------------
class RevealUndetectedFailure //Modeling pattern (MP5)
    ComponentState varState ( init = OK );
    Phase phase ( init = OPERATION );
    Boolean ComponentAvailable ( init = true ); //Component availability
    parameter Real LambdaDU = 3.0e-7; //Dangerous Undetected (DU) failure rate
    parameter Real Mu       = 1.0e-1; //Repair rate
    parameter Real PeriodicTestInterval = 8760.0; //Intervals between tests
    parameter Real PeriodicTestDuration = 0.0; //Installation stopped during tests
    event failure      ( delay = exponential ( LambdaDU ) );
    event startRepair  ( delay = 0 );
    event endRepair    ( delay = exponential ( Mu ) );
    event periodicTest ( delay = PeriodicTestInterval );
    event completeTest ( delay = PeriodicTestDuration );
    transition
        failure:     varState == OK and phase == OPERATION and ComponentAvailable == true
                  -> {varState := DU; ComponentAvailable := false;}
        startRepair: varState == DD and phase == OPERATION -> varState := UNDER_REPAIR;
        endRepair:   varState == UNDER_REPAIR and phase == OPERATION ->
                    {varState := OK; ComponentAvailable := true;}
        periodicTest: phase == OPERATION -> phase := TEST;
        completeTest: phase == TEST
                   -> {phase := OPERATION; if varState == DU then varState := DD;}
end
// ------------------------------------------------------------------------------
```

Fig. 17. AltaRica 3.0 model of the `RevealUndetectedFailure` pattern.

```
// ------------------------------------------------------------------------------
block system //Safety Instrumented System #1-1
    //"endRepair.delay = 0.0": Installation stopped during repair
    RevealUndetectedFailure PressureSensor ( endRepair.delay = 0.0 );
    RevealUndetectedFailure IsolationValve ( endRepair.delay = 0.0, LambdaDU = 2.9e-6 );
    Real SystemAvailable ( reset = 0.0 );
    observer Real AverageUnavailability = not SystemAvailable; //Unavailability
    // Average dangerous failure frequency per hour
    observer Real AverageDangerousFailureFrequency = SystemCounter*2/CalculationTime;
    Integer SystemCounter ( init = 0 ); // Counter of dangerous failures
    Boolean SystemState ( init = true );
    parameter Real CalculationTime = 87600.0; // 87600.0 hours = 10 years
    event system_Available2Unavailable ( delay = 0 );
    event system_Unavailable2Available ( delay = 0 );
    transition
        system_Available2Unavailable: (PressureSensor.varState != OK or
              IsolationValve.varState != OK) and SystemState == true  ->
              {SystemState := false; SystemCounter := SystemCounter + 1;}
        system_Unavailable2Available: (PressureSensor.varState == OK and
              IsolationValve.varState == OK) and SystemState == false ->
                                                SystemState := true;
    assertion
        SystemAvailable := PressureSensor.ComponentAvailable and
                           IsolationValve.ComponentAvailable;
end
// ------------------------------------------------------------------------------
```

Fig. 18. AltaRica 3.0 model of safety instrumented system #1-1.

`Repair crew Coordination` pattern.

RRC is composed of the following elements:

– Two state variables: *Component.state* and *Crew.RepairCrewWork*. *Component.state* takes its value from the enumeration {OK, FAILED, UNDER_REPAIR}. Its initial state is OK. *Crew.RepairCrewWork* is a Boolean variable with the initial value FALSE.
– A Boolean flow variable: *Component.operating*.
– Seven events: *Component.failure, Component.startRepair, Component.endRepair, Crew.startRepair, Crew.endRepair, Syn_startRepair*, and *Syn_endRepair*. *Component.failure, Component.endRepair*, and *Syn_endRepair* are stochastic. *Component.startRepair, Crew.endRepair*, and *Syn_startRepair* are immediate. *Crew.startRepair* is determinsitic.
– Seven transitions:
  – *Component.failure: Component.state* = = WORKING → *Component.state* := FAILED
  – *Component.startRepair: Component.state* = = FAILED → *Component.state* := UNDER_REPAIR
  – *Component.endRepair: Component.state* = = UNDER_REPAIR → *Component.state* := WORKING
  – *Crew.startRepair: Crew.numberBusyCrew* < *Crew.total NumberCrew* → *Crew.numberBusyCrew* := *Crew.numberBusyCrew* + 1
  – *Crew.endRepair: Crew.numberBusyCrew* > 0 → *Crew.numberBusyCrew* := *Crew.numberBusyCrew* - 1
  – *Syn_startRepair*: ! *Component.startRepair* & ! *Crew.startRepair*
  – *Syn_endRepair*: ! *Component.endRepair* & !*Crew.endRepair*. Operator "!" means that the associated events are forced to be firable.
– An assertion: *Component.operating* := (*Component.state* = = WORKING)

6. Reuse of modeling patterns

A methodology is required to model SIS with reusing modeling patterns. We propose such a methodology with four steps: classification, pattern-based model, AltaRica 3.0 model, and experimental results. We take system #1-1 in ISO/TR 12489 as an example to illustrate this methodology. This system is made up of a pressure sensor, a logic solver, and an isolation valve working in series.

(1) *Classification*: In this step, we identify units to be modeled and recognize corresponding modeling patterns. The target system is initially decomposed into components and subsystems. We identify corresponding modeling patterns that are required to construct these components and subsystems.

Two components are modeled in system #1-1, where the protected system is shut down during repairs and periodic tests. Thus activities related to the maintenance/repair are negligible when calculating the system unavailability. Since the system unavailability herein is only generated by DU failures, thus the logical solver (which can fail as DD failures) has not been considered. The pressure sensor and isolation valve are modeled by the `RevealUndetectedFailure` pattern (MP5). These two components work in series, thus `SERIES` pattern (MP6) is used as well.

(2) *Pattern-based model*: Based on above classification results, a pattern-based model can be obtained. This model is illustrated by means of a schematic diagram. Associated modeling patterns are assigned for each component/subsystem in the diagram. The pattern-based model simplifies the task of constructing the AltaRica 3.0 model.

The pattern-based model of system #1-1 is shown in Fig. 15. It can be used to establish a concrete model with a modeling language.

The working scheme of the sensor in system #1-1 is shown in Fig. 16.

Since the periodical tests are assumed to be perfect in ISO/TR 12489, thus all DU failures can be revealed (i.e., test coverage is 100%). DD failures can be discovered by diagnostic tests. The test duration of the diagnostic test is negligible since it is quite short. In system #1-1, the installation (protected system) is stopped during the repair (specially for DD failure). In this scenario, the unavailable state of the SIS does not have any effect on the installation. That is, only when the SIS is KO (i.e., not available) during the running of the installation, the unavailability of the SIS is counted. Therefore, the repair delay has not been considered for calculating the unavailability of the SIS. Therefore DD failures are not considered in system #1-1.

DU failures are discovered by proof tests. In system #1-1, the installation is stopped during periodical tests (intended for DU failure). Hence the proof test duration has not been considered. In this situation, installation runs until the DU failure is discovered. It is therefore in

Table 1
Modeling patterns classification for the safety systems in ISO/TR 12489.

System	Components/subsystems	Modeling patterns
#1-1	S, V	MP5
	{S, V}	MP6
#1-2	S, V	MP5
	{S, V}	MP6
#1-3	S	MP2, MP5
	LS	MP2
	V	MP5
	{S, LS, V}	MP6
#1-4	S, V	MP1, MP5
	{S, V}	MP6
#2-1	S1, S2, V1, V2	MP5
	{S1, V1},{S2, V2}	MP6
	{{S1, V1},{S2, V2}}	MP7
#2-2	S1, S2, V1, V2	MP5
	{S1, V1},{S2, V2}	MP6
	{{S1, V1},{S2, V2}}	MP7
#2-3	S1, S2	MP2, MP5
	LS1,LS2	MP2
	V1, V2	MP5
	{S1, LS1, V1},{S2, LS2, V2}	MP6
	{{S1, LS1, V1},{S2, LS2, V2}}	MP7
#2-4	S1, V1	MP5
	S2, V2	MP4
	{S1, V1},{S2, V2}	MP6
	{{S1, V1},{S2, V2}}	MP7
#3-1	S1, S2, S3, SV1, V1, SV2, V2	MP5
	{S1, S2, S3}	MP8
	{SV1, V1},{SV2, V2}	MP6
	{{SV1, V1},{SV2, V2}}	MP7
#3-2	S1, S2, S3, SV1, V1, SV2, V2	MP2, MP5
	LS	MP2
	{S1, S2, S3}	MP8
	{SV1, V1},{SV2, V2}	MP6
	{{SV1, V1},{SV2, V2}}	MP7
#3-3	S1, S2, S3, SV1, V1, SV2, V2	MP2, MP5
	LS	MP2
	{S1, S2, S3}	MP8, MP9
	{SV1, V1},{SV2, V2}	MP6
	{{SV1, V1},{SV2, V2}}	MP7
#4	S1, S2, S3	MP2, MP5
	LS1,LS2	MP2
	SV1, SV2, SV3, V1, V2	MP5
	{SV1, LS1, V1}	MP6
	All components	MP2
#5	S1a, S1b, S1c, S2a, S2b, S2c	MP1, MP2, MP5
	{S1a, S1b, S1c}, {S2a, S2b, S2c}	MP8, MP9
	{S1, S2}	MP7
	LS, SV1, V1, SV2, V2	MP1, MP5
	{SV1, V1}, {SV2, V2}	MP6

Table 2
Experimental results in system #1.

Systems	Approaches	Unavailability	Failure frequency(h^{-1})
#1-1	Formulae [10]	1.4E−2	3.2E−6
	Fault tree [10]	1.39E−2	3.16E−6
	Markovian [10]	1.39E−2	3.16E−6
	Petri net [10]	1.38E−2	3.15E−6
	AltaRica 3.0	1.39E−2	3.15E−6
#1-2	Formulae [10]	1.06E−2	–
	Fault tree [10]	1.05E−2	3.17E−6
	Markovian [10]	1.05E−2	3.17E−6
	Petri net [10]	1.05E−2	3.17E−6
	AltaRica 3.0	1.05E−2	3.18E−6
#1-3	Formulae [10]	1.47E−2	–
	Fault tree [10]	1.46E−2	1.39E−4
	Petri net [10]	1.46E−2	1.442E−4
	AltaRica 3.0	1.45E−2	1.35E−4
#1-4	Formulae [10]	2.09E−2	–
	Fault tree [10]	2.08E−2	3.158E−6
	Petri net [10]	2.07E−2	3.13E−6
	AltaRica 3.0	2.07E−2	3.13E−6

Table 3
Experimental results in system #2.

Systems	Approaches	Unavailability	Failure frequency(h^{-1})
#2-1	Formulae [10]	9.37E−4	–
	Fault tree [10]	9.33E−4	2.39E−7
	Markovian [10]	9.20E−4	2.34E−7
	Petri net [10]	9.30E−4	2.41E−7
	AltaRica 3.0	9.29E−4	2.37E−7
#2-2	Formulae [10]	6.26E−4	–
	Fault tree [10]	6.45E−4	–
	Petri net [10]	6.46E−4	–
	AltaRica 3.0	6.45E−4	2.19E−7
#2-3	Fault tree [10]	1.19E−3	–
	Petri net [10]	1.19E−3	–
	AltaRica 3.0	1.18E−3	1.05E−4
#2-4	Fault tree [10]	4.9E−4	–
	Petri net [10]	4.94E−4	–
	AltaRica 3.0	4.94E−4	2.36E−7

system #1-1, the unavailability of the SIS is only caused by DU failures.

In system #1-1, the installation is stopped during periodical tests and repair. Thus the activities (e.g., de-energize) related to the maintenance/repair are not considered when calculating the system unavailability. Since the system unavailability is only generated by DU failures, thus the Logical solver (which only has DD failures), has not been taken into consideration.

In system #1-1, the repair should be Mu = Dirac (0), rather than exponentially distributed, this is because the installation is stopped during repair. The assumption is equivalent to the situation that the component is repaired immediately. It differs from Mu = exponential (0), where the latter can lead to higher unavailability of a component in experiments.

In the pattern of periodically tested components, the availability is a state variable. This is because the availability is assigned in the transitions. State variables can occur as the left member of an assignment only in the action of a transition. Flow variables can occur as the left member of an assignment only in the assertion [66]. Sensor and Valve share the same behaviors, where they can be instantiated with `RevealUndetectedFailure` pattern. The unavailability of system #1-1 is calculated by using observer and assertion.

(3) *AltaRica 3.0 model*: In this step, we translate the pattern-based model into corresponding AltaRica 3.0 model. Modeling patterns are firstly presented in the AltaRica environment. An AltaRica 3.0 implementation of the `RevealUndetectedFailure` pattern is shown in Fig. 17. Subsequently, the AltaRica 3.0 model is constructed with identified modeling patterns. An AltaRica 3.0 implementation of the system #1-1 is illustrated in Fig. 18.

(4) *Experimental results*: The obtained AltaRica 3.0 model is first translated into a GTS model. Subsequently, experimental results are acquired by analyzing the GTS model with the stochastic simulator (see e.g. [59,67,68]).

The AltaRica model of a system is constructed by the AltaRica language [42,56,66]. On the basis of the GTS model (intermediate one obtained from AltaRica model), we can analyze them with different tools, such as the Markov process, fault tree analysis and stochastic simulator. Among them, stochastic simulator is the most powerful assessment tool of AltaRica. It is therefore we deploy stochastic simulator to obtain numerical results. The background of stochastic simulator is to conduct statistics of multiple times of repeated experiments.

From Table 2, we find that results of the pattern-based AltaRica model agree well with those reported in ISO/TR 12489.

7. Experimental study

We identify modeling patterns for modeling SIS in ISO/TR 12489, as shown in Table 1. We take system #5 as an example to illustrate the

results.

In system #5, we employ a 2oo3 structure (S1a, S1b, and S1c) as an example to elaborate the results. Since DD and DU failures of the component are assumed to be independent in EDP system, the DD failure of a component (e.g. S1a) can be modeled using the `CorrectiveMaintenance` (MP2) pattern. Since the uncovered DU failure cannot be repaired, it is constructed with the `NonRepairable` (MP1) pattern. The covered DU failure by periodic tests is considered to be modeled with the `RevealUndetectedFailure` (MP5) pattern. The subsystem composed of these three components, {S1a, S1b, S1c}, can be modeled by both `KooN` (MP8) pattern and `SwitchKooN` (MP9) pattern. Two groups of 2oo3 structures, {S1, S2}, in the EDP system can be modeled with `PARALLEL` (MP7) pattern. Note that S1 and S2 stand for 2oo3 subsystems. The rest of classification results can be interpreted in a similar way.

The SIS in ISO/TR 12489 are modeled using proposed modeling patterns. The obtained AltaRica models are analyzed with stochastic simulations. The mission time (length of histories) of all SIS is 87,600 h (10 years) except that of system #5 is 131,400 h (15 years). The number of Monte Carlo simulations (number of histories) of all systems is 10^6.

The experimental results are listed in Tables 2–8. The unavailability in each table refers to the average unavailability for the sake of simplicity. The failure frequency in tables is the average dangerous failure frequency. The formulae, fault tree, Markovian, and Petri net approaches are used in ISO/TR 12489. The more complex the systems become, the fewer approaches can be utilized. For example, because of the state explosion problem, the Markovian approach is solely used in systems #1-1, #1-2, and #2-1.

7.1. System #1: an overpressure protection system with a single channel

Experimental results for system #1 are tabulated in Table 2. In general, results from AltaRica 3.0 models agree well with those reported in ISO/TR 12489.

With regard to system #1-1, AltaRica 3.0 results meet well with those provided in ISO/TR 12489. Among these four systems, system #1-1 serves as a reference for systems #1-2, #1-3, and #1-4.

The average unavailability of system #1-2 is lower than the one of system #1-1. This is because two kinds of periodic tests are conducted on the isolation valve in system #1-2: the full stroking and partial stoking. The partial stoking is regarded as the main form of periodic test, and the corresponding periodic test interval decreases from 8760 h to 4380 h. Because DU failures can place a SIS in a down state for a long period until a periodic test is conducted, DU failures are always main contributors to the unavailability of a SIS [16]. Thus DU failures in system #1-2 can be discovered timely and less unavailability is generated.

The average unavailability of system #1-3 is slightly higher than that of system #1-1 due to the EUC is not stopped during the repair of the sensor and the logic solver. Thus DD failures of the sensor and the logic solver are considered. In addition, the sensor is periodically tested offline, therefore the periodic test duration is taken into account. DD failures, repair time, as well as the periodic test duration bring slightly more unavailability of the system.

The average unavailability of system #1-4 is higher than the one of system #1-1 because imperfect periodic tests are taken into account. Uncovered DU failures will not be detected.

7.2. System #2: an overpressure protection system with dual channel

Experimental results of system #2 can be found in Table 3. There is a good agreement for results from AltaRica 3.0 models and ISO/TR 12489. Since system #2 is a dual-channel SIS, unavailabilities of system #2 are significantly lower than those of system #1.

According to the system description and assumptions, relationships between systems #2-1, #2-2, and #2-3 are similar to those between systems #1-1, #1-2, and #1-3.

The staggered test is applied in system #2-4. Redundant components (e.g., S1 and S2, V1 and V2) in system #2-4 are tested with the same periodic test interval but not simultaneously. This policy decreases the risk that redundant components are unavailable concurrently in the test. Thus the average unavailability of system #2-4 is significantly lower than that of system #2-1.

7.3. System #3: an overpressure protection system with redundant architecture

Experimental results of system #3-1 are listed in Table 4. Average unavailabilities obtained using the AltaRica model and reported in ISO/TR 12489 give very similar results.

Experimental results of system #3-2 are tabulated in Table 5. Average unavailabilities obtained using AltaRica models are in good agreement with those in ISO/TR 12489.

Table 4
Experimental results in system #3-1.

Approaches	Unavailability
Formulae [10]	9.0E−4
Fault tree [10]	9.0E−4
Petri net [10]	9.0E−4
AltaRica 3.0	9.1E−4

Table 5
Unavailability results in system #3-2.

Mobilization time (h)	Periodic test duration (h)	Petri net [10]	AltaRica 3.0
0	0	9.76E−4	9.78E−4
0	2	1.14E−3	1.17E−3
720	0	3.95E−3	3.93E−3
720	2	4.09E−3	4.14E−3

Table 6
Experimental results in system #3-3.

Approaches	Unavailability
Petri net [10]	2.70E−3
AltaRica 3.0	2.74E−3

Experimental results of system #3-3 are shown in Table 6. The average unavailability obtained from the AltaRica model agrees well with that reported in ISO/TR 12489. Note that in system #3-3, the mobilization time of the maintenance rig is 720 h and the periodic test duration is 2 h. Because of applying the `SwitchKooN` pattern, the average unavailability of system #3-3 is lower than the one of #3-2.

7.4. System #4: a multiple safety system

Table 7 shows that results obtained using AltaRica agree rather well with those reported in ISO/TR 12489.

Table 7
Experimental results in system #4.

Approaches	Unavailability	Failure frequency(h^{-1})
Fault tree [10]	2.96E−4	4.02E−7
Petri net [10]	2.95E−4	4.0E−7
AltaRica 3.0	2.95E−4	3.98E−7

7.5. System ♯5: an emergency depressurization system

Results comparison of system ♯5 can be found in Table 8. The AltaRica 3.0 and ISO/TR 12489 give almost same results.

Table 8
Experimental results in system ♯5.

Approaches	Unavailability
Fault tree [10]	3.50E−4
AltaRica 3.0	3.46E−4

8. Conclusion

In this paper, we presented an approach to pattern-based reliability assessment of Safety Instrumented Systems (SIS). First, based on a series of SIS provided in ISO/TR 12489, we proposed a set of eleven (11) modeling patterns. We categorized these modeling patterns into behavioral patterns, flow propagation patterns, and coordination patterns. We interpreted them by means of Guarded Transition Systems (GTS). Moreover, we implemented these modeling patterns with the AltaRica 3.0 language. Eventually, we applied modeling patterns to evaluate the reliability of SIS described in ISO/TR 12489. Corresponding AltaRica models were developed to assess reliabilities of these systems. Experimental results obtained from AltaRica models using modeling patterns are in good agreement with those reported in ISO/TR 12489. It is concluded that the proposed modeling patterns are capable of constructing target systems in a modular way.

The raised modeling patterns in this paper are based on a limited set of SIS, albeit they are declared to cover most reliability studies [10], these patterns can be improved with new system behaviors. In addition, the current work investigates SIS and EUC (Equipment Under Control) separately. Future research can treat EUC as an integral part of SIS. New modeling patterns are therefore expected with such integration. Furthermore, the proposed methodology of reusing modeling patterns has proven to be applicable to a finite set of SIS. However, more experiments are required to further validate the solidity of the proposed methodology.

In our modeling patterns-based reliability analysis, we obtained final results by using stochastic simulation. Stochastic simulation has been broadly applied to predict the system performance and estimate the reliability in complex engineered systems [69]. However, stochastic simulation has limitation in evaluating performance of highly reliable systems. Once the failure rates are low, the simulation results share high randomness. When a target SIS is highly reliable, we need to improve the number of simulations or apply subset simulation [70] for obtaining reasonable simulation results. Subset simulation treats the small failure probability as a product of larger conditional probabilities of some intermediate events [70].

We need to conduct further comparison of models, not only the results, but also the computing resources, ease of modeling, and uncertainty bound on the results. The computational resources are computation time, the number of steps necessary to solve a problem, and memory space, the amount of storage required while solving the problem. We can also consider the ease of modeling: how to measure complexity of modeling, the time required for modeling, or less explicitly, by describing the complexities of other formalism compared with AltaRica models. By using the pattern-based modeling approach, we could model SIS in a modular way. We also need to consider the uncertainty bound on results: we need to quantify the uncertainty of the results, which are quite useful for such simulation.

Acknowledgment

The authors would like to express their deepest gratitude to the anonymous reviewers, as well as the Editor-in-chief, Professor Carlos Guedes Soares, for their insightful comments that greatly help us to improve the quality of this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ress.2018.06.026.

References

[1] IEC. IEC 61511 functional safety - safety instrumented systems for the process industry sector. 2016.
[2] IEC. IEC 61513 nuclear power plants - instrumentation and control important to safety - general requirements for systems. 2011.
[3] IEC. IEC 62061 safety of machinery - functional safety of safety-related electrical, electronic and programmable electronic control systems. 2005.
[4] ISO. ISO 13849-1 safety of machinery - safety-related parts of control systems - Part 1: general principles for design. 2015.
[5] ISO. ISO 26262 road vehicles - functional safety. 2011.
[6] IEC. IEC 62278 railway applications - specification and demonstration of reliability, availability, maintainability and safety (RAMS). 2002.
[7] IEC. IEC 62279 railway applications - communication, signalling and processing systems - software for railway control and protection systems. 2015.
[8] IEC. IEC 62425 railway applications - communication, signalling and processing systems - safety related electronic systems for signalling. 2007.
[9] IEC. IEC 61508 functional safety of electrical/electronic/programmable electronic safety-related systems. 2010.
[10] ISO. ISO/TR 12489 petroleum, petrochemical and natural gas industries: reliability modelling and calculation of safety systems. 2013.
[11] Rausand M. Reliability of safety-critical systems: theory and applications. John Wiley & Sons; 2014.
[12] Rausand M, Høyland A. System reliability theory: models, statistical methods, and applications. John Wiley & Sons; 2004.
[13] Dutuit Y, Rauzy A, Signoret J-P. A snapshot of methods and tools to assess safety integrity levels of high-integrity protection systems. Proc Inst Mech Eng, Part O 2008;222:371–9.
[14] Kumar M, Verma AK, Srividya A. Modeling demand rate and imperfect proof-test and analysis of their effect on system safety. Reliab Eng Syst Safety 2008;93:1720–9.
[15] Cacheux P-J, Collas S, Dutuit Y, Folleau C, Signoret J-P, Thomas P. Assessment of the expected number and frequency of failures of periodically tested systems. Reliab Eng Syst Safety 2013;118:61–70.
[16] Liu Y, Rausand M. Proof-testing strategies induced by dangerous detected failures of safety-instrumented systems. Reliab Eng Syst Safety 2016;145:366–72.
[17] Torres-Echeverria A, Martorell S, Thompson H. Modeling safety instrumented systems with MooN voting architectures addressing system reconfiguration for testing. Reliab Eng Syst Safety 2011;96:545–63.
[18] Oliveira LF, Abramovitch RN. Extension of ISA TR84. 00.02 PFD equations to KooN architectures. Reliab Eng Syst Safety 2010;95(7):707–15.
[19] Jahanian H. Generalizing PFD formulas of IEC 61508 for KooN configurations. ISA Trans 2015;55:168–74.
[20] Cai B, Liu Y, Fan Q. A multiphase dynamic Bayesian networks methodology for the determination of safety integrity levels. Reliab Eng Syst Safety 2016;150:105–15.
[21] Lundteigen MA, Rausand M. Common cause failures in safety instrumented systems on oil and gas installations: implementing defense measures through function testing. J Loss Prev Process Ind 2007;20(3):218–29.
[22] Jin H, Rausand M. Reliability of safety-instrumented systems subject to partial testing and common-cause failures. Reliab Eng Syst Safety 2014;121:146–51.
[23] Liu Z, Liu Y, Cai B, Zhang D, Zheng C. Dynamic Bayesian network modeling of reliability of subsea blowout preventer stack in presence of common cause failures. J Loss Prev Process Ind 2015;38:58–66.
[24] Hauge S, Hokstad P, Håbrekke S, Lundteigen MA. Common cause failures in safety-instrumented systems: using field experience from the petroleum industry. Reliab Eng Syst Safety 2016;151:34–45.
[25] Lundteigen MA, Rausand M. Spurious activation of safety instrumented systems in the oil and gas industry: basic concepts and formulas. Reliab Eng Syst Safety 2008;93:1208–17.
[26] Jigar AA, Liu Y, Lundteigen MA. Spurious activation analysis of safety-instrumented systems. Reliab Eng Syst Safety 2016;156:15–23.
[27] Schönbeck M, Rausand M, Rouvroye J. Human and organisational factors in the operational phase of safety instrumented systems: a new approach. Saf Sci 2010;48:310–8.
[28] Rahimi M, Rausand M. Monitoring human and organizational factors influencing common-cause failures of safety-instrumented system during the operational phase. Reliab Eng Syst Safety 2013;120:10–7.
[29] Innal F, Chebila M, Dutuit Y. Uncertainty handling in safety instrumented systems according to IEC 61508 and new proposal based on coupling Monte Carlo analysis and fuzzy sets. J Loss Prev Process Ind 2016;44:503–14.
[30] Jin H, Lundteigen MA, Rausand M. Uncertainty assessment of reliability estimates for safety-instrumented systems. Proc Inst Mech Eng, Part O 2012;226(6):646–55.
[31] Cai B, Liu H, Xie M. A real-time fault diagnosis methodology of complex systems

using object-oriented Bayesian networks. Mech Syst Signal Process 2016;80:31–44.
[32] Cai B, Xie M, Liu Y, Liu Y, Feng Q. Availability-based engineering resilience metric and its corresponding evaluation methodology. Reliab Eng Syst Safety 2018;172:216–24.
[33] Torres-Echeverria A, Martorell S, Thompson H. Modelling and optimization of proof testing policies for safety instrumented systems. Reliab Eng Syst Safety 2009;94:838–54.
[34] Longhi AEB, Pessoa AA, de Almada Garcia PA. Multiobjective optimization of strategies for operation and testing of low-demand safety instrumented systems using a genetic algorithm and fault trees. Reliab Eng Syst Safety 2015;142:525–38.
[35] Alexander C. A pattern language: towns, buildings, construction. Oxford University Press; 1977.
[36] Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional; 1995.
[37] Gamma E. Design patterns – ten years later. Software pioneers. Springer; 2002. p. 688–700.
[38] Cook D, Schindel WD. Utilizing MBSE patterns to accelerate system verification. Insight 2017;20(1):32–41.
[39] Hamid B, Perez J. Supporting pattern-based dependability engineering via model-driven development: approach, tool-support and empirical validation. J Syst Softw 2016;122:239–73.
[40] Ruiz A, Juez G, Espinoza H, de la Vara JL, Larrucea X. Reuse of safety certification artefacts across standards and domains: a systematic approach. Reliab Eng Syst Safety 2017;158:153–71.
[41] Kajtazovic N, Preschern C, Höller A, Kreiner C. Towards pattern-based reuse in safety-critical systems. Proceedings of the 14th European Conference on Pattern Languages of Programs. ACM; 2014.
[42] Prosvirnova T. AltaRica 3.0: a model-based approach for safety analyses. Palaiseau, France: Ecole Polytechnique; 2014.
[43] Bozzano M, Cimatti A, Lisagor O, Mattarei C, Mover S, Roveri M, et al. Safety assessment of AltaRica models via symbolic model checking. Sci Comput Program 2015;98:464–83.
[44] Meng H., Kloul L., Rauzy A.. A pattern-based methodology for reliability assessment of safety instrumented systems. In: The Proceedings of the 2017 IEEE International Symposium on Systems Engineering (ISSE). Vienna, Austria; 2017a, p. 438–443.
[45] Preschern C, Kajtazovic N, Höller A, Kreiner C. Pattern-based safety development methods: overview and comparison. Proceedings of the 14th European Conference on Pattern Languages of Programs. ACM; 2014.
[46] Guo Q, Xu P, Pei X, Wong S, Yao D. The effect of road network patterns on pedestrian safety: a zone-based bayesian spatial modeling approach. Accident Anal Prevent 2017;99:114–24.
[47] Verma A, Khan SD, Maiti J, Krishna O. Identifying patterns of safety related incidents in a steel plant using association rule mining of incident investigation reports. Saf Sci 2014;70:89–98.
[48] Hukerikar S, Engelmann C. Resilience design patterns - a structured approach to resilience at extreme scale. Tech. Rep.. Oak Ridge National Laboratory, Tennessee, US; 2016.
[49] Kehren C. Motifs formels d'architectures de systèmes pour la sûreté de fonctionnement. Toulouse, France: Ecole nationale superieure de l'aeronautique et de l'espace; 2005.
[50] Morel M. Model-based safety approach for early validation of integrated and modular avionics architectures. In: Ortmeier F, Rauzy A, editors. Model-based safety and assessment: 4th international symposium, IMBSA 2014, Munich, Germany, October 27–29, 2014, proceedings. Springer; 2014. p. 57–69.
[51] Rauzy A. Mode automata and their compilation into fault trees. Reliab Eng Syst Safety 2002;78(1):1–12.
[52] Rauzy A. Guarded transition systems: a new states/events formalism for reliability studies. Proc Inst Mech Eng, Part O 2008;222(4):495–505.
[53] Meng H, Kloul L, Rauzy A. Modeling patterns for performance analyses of offshore production systems. The Proceedings of the 27th International Ocean and Polar Engineering Conference. San Francisco, USA: International Society of Offshore and Polar Engineers; 2017. p. 1199–205.
[54] Signoret J-P, Dutuit Y, Cacheux P-J, Folleau C, Collas S, Thomas P. Make your Petri nets understandable: reliability block diagrams driven Petri nets. Reliab Eng Syst Safety 2013;113:61–75.
[55] Cherfi A. Toward an Efficient Generation of ISO 26262 Automotive Safety Analyses. Palaiseau, France: Ecole Polytechnique; 2015.
[56] Batteux M, Prosvirnova T, Rauzy A. Altarica 3.0 assertions: the whys and wherefores. Proc Inst Mech Eng, Part O 2017:1–10.
[57] IEC. IEC 62551 analysis techniques for dependability - Petri net techniques. 2012.
[58] Ajmone-Marsan M, Balbo G, Conte G, Donatelli S, Franceschinis G. Modelling with generalized stochastic Petri nets. Bognor Regis, West Sussex PO22 9SA, England: John Wiley and Sons; 1995. ISBN-10: 0471930598. ISBN-13: 978-0471930594
[59] Lipaczewski M, Ortmeier F, Prosvirnova T, Rauzy A, Struck S. Comparison of modeling formalisms for safety analyses: SAML and AltaRica. Reliab Eng Syst Safety 2015;140:191–9.
[60] Brameret P-A, Rauzy A, Roussel J-M. Automated generation of partial Markov chain from high level descriptions. Reliab Eng Syst Safety 2015;139:179–87.
[61] Marsan MA, Chiola G. On petri nets with deterministic and exponentially distributed firing times. In: Rozenberg G, editor. Advances in Petri Nets. Berlin, Heidelberg: Springer Berlin Heidelberg; 1987. p. 132–45.
[62] Marsan MA, Chiola G, Fumagalli A. Improving the efficiency of the analysis of DSPN models. In: Rozenberg G, editor. Advances in Petri Nets 1989. Berlin, Heidelberg: Springer Berlin Heidelberg978-3-540-46998-8; 1990. p. 30–50.
[63] Rausand M. Risk assessment: theory, methods, and applications. John Wiley & Sons; 2011.
[64] Rauzy A. Towards a sound semantics for dynamic fault trees. Reliab Eng Syst Safety 2015;142:184–91.
[65] Meng H, Kloul L, Rauzy A. Production availability analysis of floating production storage and offloading (FPSO) systems. Appl Ocean Res 2018;74:117–26.
[66] Batteux M, Prosvirnova T, Rauzy A. AltaRica 3.0 language specification. Tech. Rep.. AltaRica Association; 2015.
[67] Batteux M., Rauzy A.. Stochastic simulation of AltaRica 3.0 models. In: Proceedings of the European Safety and Reliability Conference, ESREL Amsterdam, Netherlands; 2013, p. 1093–1100.
[68] Aupetit B, Batteux M, Rauzy A, Roussel J-M. Improving performances of the AltaRica 3.0 stochastic simulator. Proceedings of the European Safety and Reliability Conference, ESREL. Zurich, Switzerland. 2015. p. 1815–23.
[69] Mullins J, Mahadevan S. Variable-fidelity model selection for stochastic simulation. Reliab Eng Syst Safety 2014;131:40–52. https://doi.org/10.1016/j.ress.2014.06.011.
[70] Zio E, Pedroni N. Reliability estimation by advanced Monte Carlo simulation. London: Springer London; 2010. p. 3–39.