Identifying critical failure-propagation in function models of complex systems

Yann Guillouët

Institute for the Protection of Maritime Infrastructures, German Aerospace Center (DLR), Germany.
E-mail: yann.guillouet@dlr.de

Frank Sill Torres

Institute for the Protection of Maritime Infrastructures, German Aerospace Center (DLR), Germany.
E-mail: frank.silltorres@dlr.de

Complex interconnected systems have high demands on meaningful analysis of the impact of failures on the actual service provision. This includes the study of obvious and high probable events, but also failures that are difficult to anticipate, e.g. due to cascading effects or combined events. This work introduces a framework for failure analysis that enables the exhaustive identification of combined failures with the strongest impact on the functionality of a system. The framework consists of two principal elements: a method for capturing the propagation of failures in complex systems that are represented via function models, and algorithms for solving the identification problem, which is formulated as combinatorial optimization problem. The feasibility of the approach is verified at hand of a function model of an Offshore Wind Farm (OWF). Both algorithms are then applied to the model of an offshore wind farm in order to identify the failure combinations with the strongest impact on the functionality.

Keywords: Reliability engineering, Function modelling, Failure propagation, Combinatorial optimization, Fault tree analysis, Complex system, Combinatorial optimization.

1. Introduction

Failures are a common threat for complex systems and should be considered in all phases of the system life cycle. This includes the analysis of known and expected failures, but also cascading and multiple ones. Furthermore, related solutions should be able to model systems in an appropriate way and to assess in exhaustive manner the impact of all possible failure combinations.

The reliability engineering community provides a vast amount of powerful solutions for failure analysis in technical systems, with Function Mode and Effect Analysis (FMEA) and Fault Tree Analysis (FTA) being the most representative examples Peeters et al. (2018). These classical methods, however, may fall short when it comes to exhaustive analyses, as they are time-consuming and often require a considerable manual effort Kabir (2017). To circumvent this challenge, Rocco et al. Rocco et al. (2020) propose the application of the formal analysis of concept, which is a mathematical method for data analysis that focuses on the relationship between sets of objects and attributes. This method, though, is rather restricted for exhaustive failure analysis of complex interconnected systems. Sun et al. Sun et al. (2019) propose the system theoretic formal analysis method to identify interactive failures, which only occur in case of two or more simultaneously failing components. The presented approach uses a model checking tool for formal analysis of the system that is described as state machine. Thus, exhaustive analysis is possible. However, this method is rather restricted for complex systems. Ortmeier and Schellhorn Ortmeier and Schellhorn (2007) discuss the use of formal FTA for systems described as state charts. Despite promising results, the authors conclude that the most critical aspect is the actual formalization of the FTA nodes.

The application of Model-Based System Engineering (MBSE) is of gaining for complex system development Cameron and Adsit (2020). MBSE can be defined as formalized application of modeling to support the design, analysis and verification of systems in a model-based context INCOSE (2014); Kaiser et al. (2016). The early availability of system models, often in form of modeling languages like SysML, motivated several researchers to explore the applicability of formal methods, e.g. for verification purposes Dickerson et al. (2016). In case of failure analysis, though, published works mainly focus on the automatic generation of fault trees Dickerson et al. (2018) or the combination with FTA Rambikur et al. (2017) and FMEA Huang et al. (2017). The proposed methods, though, appear less applicable for complex systems and an exhaustive analysis of failure propagation.

Analysis of cascading failures is a common subject in the context of network-like systems. These are commonly described as complex networks consisting of nodes that can share their loads among each other Zio and Sansavini (2011); Sicanica and Vujaklija (2020). Cascading failure

Proceedings of the 31st European Safety and Reliability Conference
Edited by Bruno Castanier, Marko Cepin, David Bigaud, and Christophe Berenguer
Copyright © ESREL 2021.*Published by* Research Publishing, Singapore.
ISBN: 978-981-18-2016-8; doi:10.3850/978-981-18-2016-8_090-cd

analysis can be conducted based on the inherent load distribution inside the grid using time-discrete models. Despite their compelling applicability for networks, these methods are less applicable for heterogeneous systems.

Kurtoglu et al. Kurtoglu et al. (2010) and van Bossuyt et al. Van Bossuyt et al. (2019) apply and extend the Function Failure Identification and Propagation (FFIP), which reasons about critical failure flows in systems. The presented solutions employ a functional view on systems, quantify individual risk of system elements and are able to consider failure flows induced by multiple failures. However, the proposed methods are based on discrete-time simulations, and thus, are less applicable for exhaustive failure analysis if it comes to complex systems.

To close this gap, this work introduces a framework to exhaustively analyze all possible combinations of failures and their impact on the functionality of a system of interest. This is achieved by extending function models by a system dynamic, deterministically describing the propagation of failing elements over time. Based on this framework, we propose algorithms to identify the failure combination with the worst possible outcome considering each possible scenario.

This work is structured in four main sections: Section 2 reviews the current state of the respective fields, introduces related definitions and concepts and motivates this work. The following Section 3 introduces a method to describe the propagation of failures in a system. Section 4 uses this method to formulate a combinatorial optimization problem that aims at the identification of failure combination with the highest impact on the system. Two algorithms are presented to solve that problem, and both are discussed regarding their complexity. Section 5 analyses the application of the framework to the model of an offshore wind farm. Finally, Section 6 concludes this work.

2. Preliminary

This Section introduces basic information and motivates this work.

2.1. *Function models*

Function Models (FM) are a common approach for the graphical representation of complex systems. In FM, the elements of the system are represented by the functions they contribute to the functioning of the system (cf. Beitz et al. (2007); Hollnagel (2012); Yildirim et al. (2017)). These functions can be physical devices, processes, or conceptual functions, like the safety of workers Erden and et al. (2008). The dependencies between these elements are then represented by arrows that connect the corresponding functions. These arrows can mean flows of mass, energy or information, or just dependencies, like the safety of workers depending on a functioning firefighting.

2.2. *Failures and failure propagation*

The notion of failure is defined in the European standard on maintenance technology European Committee for Standardization (2010) as:

Definition 2.1. The **failure** of an item is the event of the “termination of the ability of an item to perform a required function”.

To understand the propagation of failures within complex system, the following distinction is important:

Definition 2.2.

(1) **Primary failures** have their root cause in the respective item itself.
(2) **Secondary failures** are caused due to the failure of other items.

After a failure, the item is in **faulty state** until **recovery**.

2.3. *Fault tree analysis*

There are a variety of methods to analyze the interaction of primary and secondary failures, with Fault Tree Analysis (FTA) being one of the most representative solutions Peeters et al. (2018). FTA is a graphical deductive method for failure analysis that searches for possible explanations why given failures, referred to as top event, could occur. Therefore, all events that possibly lead to top event are arranged in a tree-like structure US Nuclear Regulatory Commission (1981). The leaves of that tree are referred to as basic events and can be considered the causes of the failure.

FTA is widely used for assessing probabilities to a given event, e.g. Distefano and Puliafito (2009); Halme and Aikala (2012), by computing the set of (minimal) cut sets to the top event of interest. These can be defined as follows Rausand (2014):

Definition 2.3. A **cut set** (CS) is a set of basic events whose occurrences ensure the top event to occur. A **minimal cut set** (MCS) is a cut set, from which no event can be removed such that it still ensures the top event to occur.

By attributing probabilities to each MCS, one can calculate the overall probability of the top event to occur.

2.4. *Origin of the approach*

While most methods of failure analysis focus on the most likely failures, or only on the failure of specific elements, we aim to identify the scenario with the biggest overall impact. This requires (1) a model that is able to describe the system in appropriate manner, including the propagation of failures, and (2) an algorithm for analysis of all possible failure scenarios.

The first requirement is satisfied by the application of function models (FM) that are extended by a system dynamic that describes in deterministic manner the propagation of failing elements over time. The following Section 3 details this method. As it regards the second requirement, a formal representation of the problem and related algorithms are employed. This is discussed in detail in Section 4.

3. Failure propagation modeling

As described above, a system dynamic for modeling the propagation of failures is required. This dynamic shall map a given set of primary failures to the resulting secondary failures, so that the state of the system can be determined over a time period of interest.

Let us therefore consider a system, represented by a FM in the form of a directed graph $G = G(V, E)$, where the vertices V and edges E represent the functions and their interdependencies, respectively. Each vertex can be in one of two states, namely functioning or faulty. To model the behavior of the system over time, consider a discrete time period $T = [0, t_{end}] \subset \mathbb{Z}$. The state of the system at a given time $t \in T$ can then be described by the following sets:

Let $x(t) \subseteq V$ be the set of vertices corresponding to functions suffering a primary failure at time t, due to reasons outside the boundaries of the model. Let $x = ((x(0), x(1), ..., x(t_{end}))$ be the sequence of sets of primary failures. Let $y(t) \subseteq V$ be the set of vertices in faulty state at time t, due to either primary or secondary failures. Let $y = ((y(0), y(1), ..., y(t_{end}))$ be the sequence of sets faulty elements.

To model the occurrence of secondary failures in the system, let us introduce the concept of breakers:

Definition 3.1. For a given vertex v, let a **breaker** $b \subseteq V$ be a set of parent vertices of v, such that v turns to faulty state in the next time step if all vertices in b are faulty:

$$b \subseteq y(t) \Rightarrow v \in y(t+1) \tag{1}$$

v then is a **target** of b.

Let $B(v) = \{b \subseteq V \,|\, b \text{ targets } v\}$ be the set of all breakers of v. $B(v)$ has to be defined beforehand for each vertex along with the function model, to specify how the functions depend on each other. At a given time $t \in T$, a vertex v can then be in faulty state for one of three different reasons:

(1) It was in faulty state already: $v \in y(t-1)$
(2) It suffers a primary failure: $v \in x(t)$
(3) It suffers a secondary failure. That means, all the elements of at least one of its breakers have to be faulty at time $(t-1)$: $\exists b \in B(v) : b \subseteq y(t-1)$

The system dynamic $\mathcal{M}$, mapping x to y, then reads as follows:

$$y(0) = x(0) \tag{2a}$$

$$\forall v \in V, t \in T, t > 0 :$$

$$v \in y(t) \iff \begin{cases} v \in y(t-1) \\ v \in x(t) \\ \exists b \in B(v) : b \subseteq y(t-1) \end{cases} \tag{2b}$$

Given a sequence of primary failures, the secondary failures and can now be computed iteratively for all time steps.

Definition 3.2. Let us call the pair $\varepsilon := (x, y)$ a **failure-sequence**.

An exemplary FM, serving as example throughout this work, is depicted in Figure 1, along with the corresponding breakers.

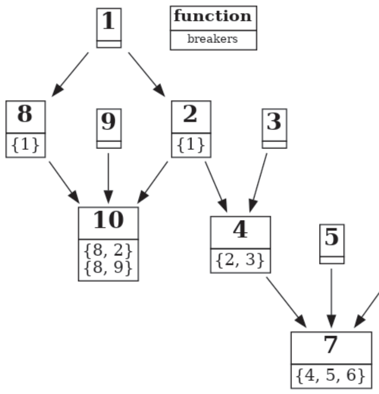


Fig. 1. Exemplary function model. The numbers represent the functions of the respective system. The sets below these functions represent their breakers. The edges represent the dependencies between the functions.

4. Optimizing the impact of failures

With the model defined, we are now interested in finding the worst-case-scenario, namely the combination of primary failures leading to the biggest impact on the system. We define the notion 'big' as the number of faulty vertices, weighted with a cost function $c(v) \geq 0 \; \forall v \in V$ to represent their importance to the system:

$$f(\varepsilon) = \sum_{v \in Y} c(v_i) \tag{3}$$

where $Y := \bigcup_{t \in T} y(t)$ denotes the set of all affected vertices in the failure sequence ε.

4.1. *Problem formulation*

Note that as items remain faulty once they suffered a failure, it makes no difference at what time a primary failures occur. Thus, by introducing $X = \bigcup_{t \in T} x(t)$, it is sufficient to identify which items may suffer primary failures, without paying attention to the timing. For simplicity, let us therefore call the pair $\mathcal{E} := (X, Y)$ the **failure-set** corresponding to ε, and $\mathcal{M}_{\mathcal{S}}$ the operator mapping X to Y.

To control the kind of resulting failure sets, and to exclude trivial cases such as $X = V$, where all elements just suffer primary failures, obviously leading to the biggest possible impact, it makes sense to take into account the overall likelihood of the failure sequences. If the primary failures are considered to happen independently, the probability of a failure sequence to occur is given by the product of the probabilities of the primary failures to occur $p(v) \in [0, 1]$:

$$p(\mathcal{E}) = \prod_{v \in X} p(v) \tag{4}$$

By limiting the probability of the failure sequences to consider by a limit probability $p_0 \leq p(\mathcal{E})$, the analyst can define in what types of failure sequences (in terms of likelihood) he is interested. The corresponding optimization problem then reads:

$$\max_{X \in \mathcal{P}(V)} \quad \sum_{v \in Y} c(v_i) \tag{5a}$$

$$\text{s.t.} \quad y = \mathcal{M}_{\mathcal{S}}(X), \tag{5b}$$

$$\prod_{v \in X} p(v) \geq p_0. \tag{5c}$$

4.2. *Naive approach*

Problem (5) is a combinatorial optimization problem (COP), as its input only allows distinct combinations from the set of vertices V. Solving such problems can be achieved by enumerating all potential solutions $\mathcal{U}$, identifying the feasible ones $\mathcal{L}$ and evaluating their objective function. In this case, all possible combinations of primary failures can be considered potential solutions, and those satisfying Constraint (5c) can be considered the feasible ones (with $\mathcal{P}$ the powerset):

$$\mathcal{U} = \{\mathcal{E} = (X, Y) \,|\, X \in \mathcal{P}(V),\, Y = \mathcal{M}_{\mathcal{S}}(X)\} \tag{6}$$

$$\mathcal{L} = \{\mathcal{E} \in \mathcal{U} \,|\, p(\mathcal{E}) \geq p_0\} \tag{7}$$

Enumerating the potential solutions is commonly achieved by recursively arranging them in the form of a tree. In this case, a natural way to built such a search tree is to order the functions and to successively add them to the already generated solutions. As $p(v) \in [0, 1]$, $p(\mathcal{E})$ then only decreases along the branches of the tree. Once the probability of a scenario falls below p_0, there can not be any further feasible solutions on that branch, and the entire branch can be pruned.

The naive search tree corresponding to the FM of Figure 1 is depicted in Figure 2, with the probabilities set such that any feasible solution consists of two primary failures at most.

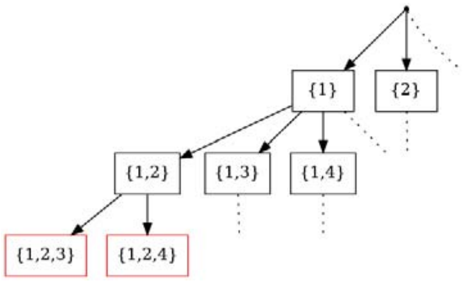


Fig. 2. Naive search tree to the FM of Fig. 1. Each box represents a potential solution, identified by its primary failures. Red solutions exceed the limit probability and the branch is therefore pruned.

4.3. *Dual Function Graph*

Depending on the size of the system and the limit probability, the naive search tree may grow extremely fast, eventually exceeding available calculation power. A suitable way to reduce the number of feasible solutions to test is therefore desirable. For that purpose, two observations are of interest:

(1) Primary failures have no further impact if they do not group up to breakers.
(2) Failing breakers may have an even bigger impact if they interact.

By assuming that the worst case scenario only consists of interacting breakers, any combination of failures with no common effect can be discarded from the set of potential solutions. This makes even more sense, if we consider the analyst to be well-aware of 'small', local failures without further effect. In that case, the analyst is indeed rather interested in those combinations of failures with non-trivial outcome.

It is therefore useful to investigate the interactions between the breakers. A breaker b_1 can interact with another breaker b_2 in two different ways:

(1) b_1 and b_2 share elements: $b_1 \cap b_2 \neq \emptyset$

(2) b_1 targets elements of b_2: $\exists v \in b_2 : b_1 \in B(v) \iff \text{target}(b_1) \cap b_2 \neq \emptyset$

These interactions can be represented in another directed graph, whose vertices represent the breakers of the system, and whose edges represent their interactions. Let us call this graph $G_D := (V_D, E_D)$ the **Dual Function Graph (DFG)**, and define it as follows:

$$V_D := \bigcup_{v \in V} B(v) \tag{8}$$

$$\begin{aligned} E_D &:= \{(b_i, b_j) \mid b_i \cap b_j \neq \emptyset \vee \text{target}(b_i) \cap b_j \neq \emptyset\} \\ &= \{(b_i, b_j) \mid (b_i \cup \text{target}(b_i)) \cap b_j \neq \emptyset\} \end{aligned} \tag{9}$$

The DFG to the FM of Figure 1 is depicted in Figure 3.

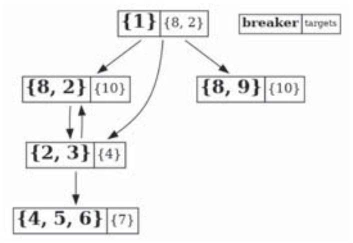


Fig. 3. DFG to the FM of Fig. 1. The vertices of the graph represent the breakers of the system, and the edges represent their interactions. The targets are represented next to each breaker.

Based on the DFG, a different search tree, only containing interacting failures, can be built in the following way: Starting from a given breaker, only neighboring (and therefore interacting) breakers are added in each recursion step. For the resulting set of breakers B, the primary failures of the corresponding failure set can be computed by taking into account the targets of the breakers:

$$Y_B = \bigcup_{b \in B} (b \cup \text{target}(b)) \tag{10}$$

$$X_B = Y_B \setminus \bigcup_{b \in B} \text{target}(b) \tag{11}$$

The resulting search tree corresponding to the FM of Figure 1 is depicted in Figure 4.

Due to the implicit computation of the primary failures, adding a parent breaker to a set of breakers however may replace primary failures and therefore increase the probability of the solution to occur. To ensure that this probability continues to only decrease along the branches of the tree, the set of breakers to add has to be reduced to the set of child breakers of the solution. To nevertheless take into account parent breakers we make use of the concept of MCSs from FTA: Each feasible solution generated is a cut set to the breakers it contains. By storing the minimal ones coupled to the respective breakers, they can be taken into account whenever that breaker is added to a solution. This procedure is described as pseudo code in Algorithm 1.

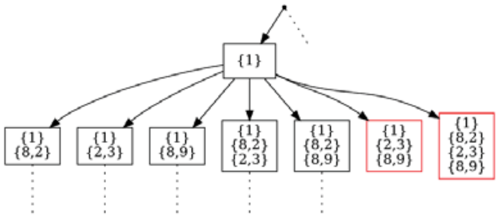


Fig. 4. DFG search tree to the FM of Fig. 1. Each box represents a potential solution, identified by the breakers it consists of. Red solutions exceed the limit probability and the branch is therefore pruned.

Algorithm 1: Generating the DFG-based search tree

```
Data: $G_D$, $p_0$
1
2  Def ADDBREAKER($\mathcal{E}_B$, $B_{new}$):
3      $B_c = G_D.children(B_{new}) \setminus B$
4      $combs$ = GENCOMBINATIONS($B_c$)
5      for $comb \in combs$ do
6          $\mathcal{E}'_B$ =FAILURESET($B \cup comb$)
7          if $p(\mathcal{E}'_B) \geq p_0$ then
8              save $\mathcal{E}'_B$
9              ADDMCS($\mathcal{E}'_B$)
10             $B'_{new} = B' \setminus B$
11             ADDBREAKERS($\mathcal{E}'_B$, $B'_{new}$)
12         end
13     end
14
15 Def MAINLOOP:
16     for $b \in V_D$ do
17         $\mathcal{E}_B$ =FAILURESET($b$)
18         ADDBREAKER($\mathcal{E}_B$, $\{b\}$)
19     end
```

Starting from every breaker once, the function ADDBREAKERS takes the children of that breaker (line 3) and generates all possible combinations of these, including their MCSs (line 4). Each resulting combination is added to the current failure set (line 6) and checked for compliance with the limit

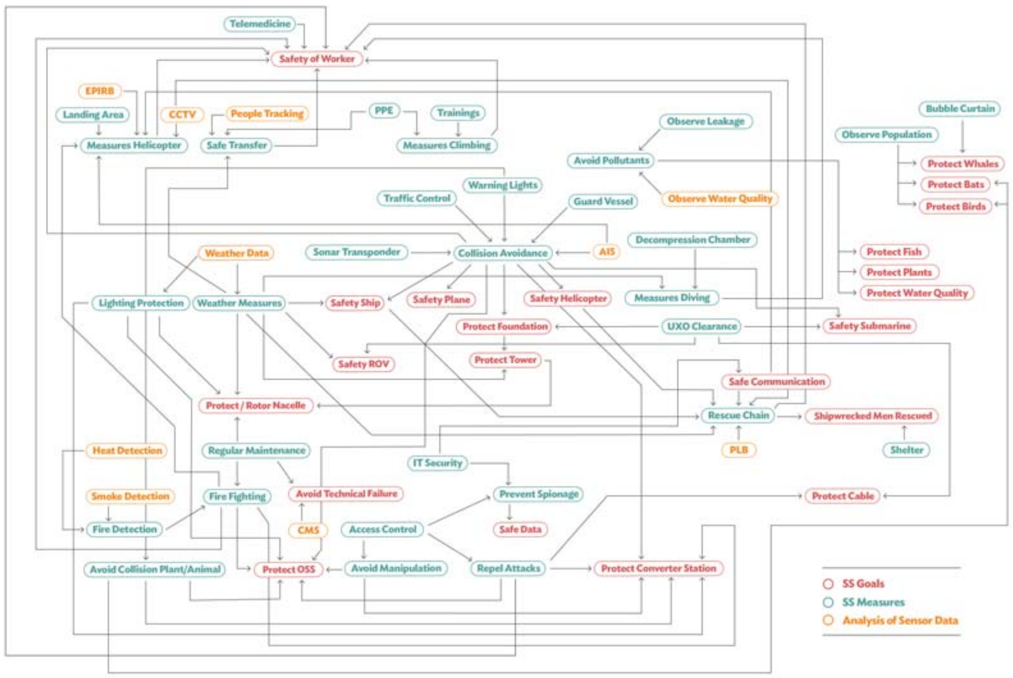


Fig. 5. Function model regarding safety and security in an OWF (adapted from Ramirez Agudelo et al. (2020))

probability (line 7). If it complies, the solution is stored to the set of feasible solutions, and also added to the set of MCSs of every breaker in B' to which it is minimal. Then, the next recursion step starts. Note that ADDBREAKERS takes the additional parameter B_{new}, which represents the breakers added in the last step. Only their child breakers are of interest, as the children of the remaining breakers were already taken into account in the previous steps.

With all the feasible solutions generated, their costs can now be computed following Eq. (3) to identify the optimal one. Here, further methods from combinatorial optimization, like greedy or branch-and-bound-algorithms, can be used.

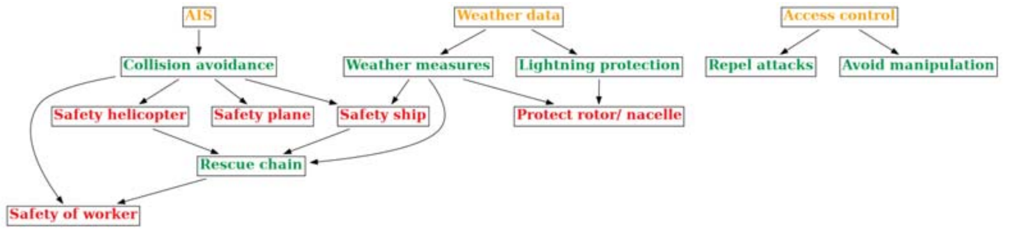
4.4. *Comparison of the algorithms*

Given a uniform FM, where each element has the same probability of failure p_v, the maximal number of primary failures k of a feasible solution is given by:

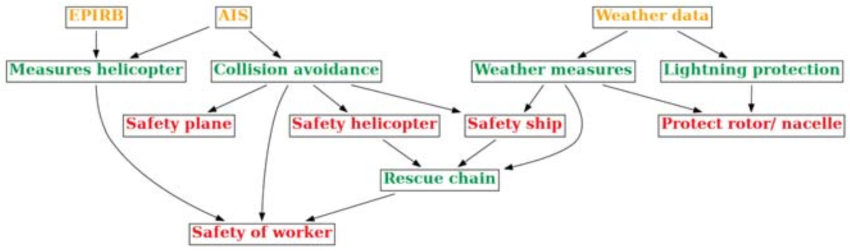
$$k = \frac{\ln(p_0)}{\ln(p_v)} \tag{12}$$

Consequently, the number of feasible solutions, and thus the number of scenarios to test in the naive approach is of magnitude $\mathcal{O}\left(N^k\right)$. The DFG-approach, on the other hand, depends on the total amount of breakers $|V_D|$. After the first iteration step, however, only combinations of child breakers are considered (together with their MCSs), leaving $\mathcal{P}(\delta_D) = 2^{\delta_D|MCS|}$ further possibilities, with δ_D the number of edges of a breaker in the DFG and $|MCS|$ its number of MCSs. Assuming that only one further primary failure is added to the scenario in each iteration step, it takes up to k steps to reach the limit probability, leading to a complexity of $\mathcal{O}\left(|V_D|\left(2^{\delta_D|MCS|}\right)^k\right) = \mathcal{O}\left(|V_D|\,2^{k\,\delta_D|MCS|}\right)$. Consequently, while the complexity of the naive approach grows exponentially with the size of the FM, the DFG-approach only grows lineally. Instead of the overall system size, the decisive parameter here is rather the inter-connectivity of the breakers in the DFG.

Also recall that the results of the algorithms differ: While the naive approach combines all functions and therefore generates the complete set of feasible solutions, the DFG-algorithm only generates those consisting of interacting breakers. Consequently, the latter does not necessarily find the optimal solution to Problem (5), if its failures may occur independently. As discussed in Section 4.3, it depends on the analyst if such solution are of actual interest.



(a) Resulting failure set from the naive algorithm, with primary failures of the functions `AIS, Weather data` and `Access control`



(b) Resulting failure set from the DFG algorithm, with primary failures of the functions `EPIRB,AIS` and `Weather data`

Fig. 6. Resulting failure sets in the OWF model. While the solution generated by the naive algorithm leads to higher costs, it consists of two distinct set of failures, which can occur independently.

5. Application

For comparison, both approaches are applied to the FM of an offshore platform. In Köpke et al. (2019), the authors propose a FM to represent the assurance of safety, security and resilience (SSR) of an Offshore Wind Farm (OWF), depicted in Figure 5. The functions represented in the model are classified in three different types:

(1) SSR-goals are the purposes of the system, representing the different safety-related parts of the OWF whose functioning are to guarantee.
(2) SSR-measures are the measures put in place to achieve these SSR-goals.
(3) Sensors generate data necessary to operate the SSR-measures.

5.1. *Extending the FM*

The additional attributes breakers, probability of primary failure and costs of failure have to be defined for each element to generate failure sets from this FM. The breakers are set such that any combination of two parent functions is a breaker to the respective function (for functions with only one parent function, this one is the only breaker). Due to its high importance for navigation in the maritime domain, the function `AIS` is excluded from this rule, and set to be a breaker on its own to the function `Collision avoidance`. Probabilities and costs of failures are attributed based on the functions type, as specified in Table 1.

Table 1. Attributes of the OWF model.

type	probability of primary failure	costs of failure
SSR goal	0	100
SSR measure	0.01	10
sensor	0.1	1

5.2. *Results*

With a limit probability set to $p_0 = 10^{-3}$, the resulting failure sequences leading to the highest costs, computed with the respective approaches, are depicted in Figure 6.

The resulting failure sets differ: The naive approach indicates that the primary failure of the functions `AIS, Weather data` and `Access control` lead to the worst outcome, with five SSR-goals suffering secondary failures and total costs of 563. However, this solution consists of two distinct set of vertices, and is therefore not generated by the DFG approach. The DFG approach instead returns the primary failures of `Emergency Position-Indicating Radio Beacon (EPRIB)` instead of `Access control` as single biggest failure sequence, setting only five SSR-measuress faulty and leading to total costs of 553.

While the first one is the actual solution to the

COP as formulated in Eq. (5), it is not generated by the DFG-algorithm, as it consists of two distinct failure sets, which can occur independently. As discussed in Section 4.3, it depends on the analyst if this solution is of actual interest.

6. Conclusion

This work introduced a new method for failure analysis, focusing on the overall impact of failing elements in a system. This is achieved by defining a system dynamic for the propagation of failures in function models. Based on this dynamic, the problem of identifying the worst-case scenario was formulated as a combinatorial optimization problem. Two algorithms were proposed to solve that problem: a naive one that generates each possible solution, and the DFG-algorithm that makes use of the dual function graph to reduce the number of solutions to test. While their solutions slightly differ, both algorithms can be of interest depending on the use-case.

References

Beitz, W., J. Feldhusen, K.-H. Grote, and G. Pahl (2007). *Engineering Design: A Systematic Approach* (3rd ed.). Springer-Verlag London.

Cameron, B. and D. M. Adsit (2020). Model-based systems engineering uptake in engineering practice. *IEEE Trans. on Engineering Management 67*(1), 152–162.

Dickerson, C. E., S. Ji, and R. Roslan (2016). Formal methods for a system of systems analysis framework applied to traffic management. In *SoS Engineering Conf. (SoSE)*, pp. 1–6.

Dickerson, C. E., R. Roslan, and S. Ji (2018). A formal transformation method for automated fault tree generation from a uml activity model. *IEEE Trans. on Reliability 67*(3), 1219–1236.

Distefano, S. and A. Puliafito (2009). Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees. *IEEE Trans. on Dep. and Se. Computing 6*(1), 4–17.

Erden, M. S. and et al. (2008). A review of function modeling: Approaches and applications. *Artifi. Intell. for Engineering Design, Analysis and Manufact.*.

European Committee for Standardization (2010). *Maintenance - Maintenance terminology*. European Committee for Standardization.

Halme, J. and A. Aikala (2012, may). Fault tree analysis for maintenance needs. *Journal of Physics: Conference Series 364*, 012102.

Hollnagel, E. (2012). *FRAM: The functional resonance method*. Ashgate Publishing Limited.

Huang, Z., S. Swalgen, H. Davidz, and J. Murray (2017). Mbse-assisted fmea approach — challenges and opportunities. In *Annual Reliability and Maintainability Symposium*, pp. 1–8.

INCOSE (2014). Incose systems engineering vision 2025.

Kabir, S. (2017). An overview of fault tree analysis and its application in model based dependability analysis. *Expert Systems with Applications 77*, 114–135.

Kaiser, L., C. Bremer, and R. Dumitrescu (2016). Exhaustiveness of systems structures in model-based systems engineering for mechatronic systems. *Procedia Technology 26*, 428–435.

Köpke, C., J. Schäfer-Frey, E. Engler, and C. P. Wrede (2019). A joint approach to safety, security and resilience using the functional resonance analysis method. In *REA Symposium*.

Kurtoglu, T., I. Y. Tumer, and D. C. Jensen (2010). A functional failure reasoning methodology for evaluation of conceptual system architectures. *Research in Engin. Design 21*(4), 209–234.

Ortmeier, F. and G. Schellhorn (2007). Formal fault tree analysis - practical experiences. *Electronic Notes in Theoretical Computer Science 185*, 139–151.

Peeters, J., R. Basten, and T. Tinga (2018). Improving failure analysis efficiency by combining fta and fmea in a recursive manner. *Reliability Engineering & System Safety 172*, 36–44.

Rambikur, S. A., K. Giammarco, and B. O'Halloran (2017). Increasing system failure analysis effectiveness through architecture modeling. *Procedia CS 114*, 4–13.

Ramirez Agudelo, O. H., C. Köpke, and F. Sill Torres (2020). Bayesian network model for accessing safety and security of offshore wind farms. In *ESREL/PAM*.

Rausand, M. (2014). *Reliability of Safety-Critical Systems: Theory and Applications*.

Rocco, C. M., E. Hernandez-Perdomo, and J. Mun (2020). Introduction to formal concept analysis and its applications in reliability engineering. *Rel. Engin. & System Safety 202*, 107002.

Sicanica, Z. and I. Vujaklija (2020). Resilience to cascading failures. a complex network approach for analysing the croatian power grid. In *Int. Conv. on Information, Communication and Electronic Technology*, pp. 918–922.

Sun, R., D. Zhong, and W. Li (2019). Formal system interactive failure analysis method based on systems theoretic process analysis model. *Engineering Failure Analysis 106*, 104141.

US Nuclear Regulatory Commission (1981). *Fault Tree Handbook*. US Nuclear Regulatory Commission.

Van Bossuyt, D. L., B. M. O'Halloran, and R. M. Arlitt (2019). A method of identifying and analyzing irrational system behavior in a system of systems. *Systems Engineering 22*(6), 519–537.

Yildirim, U., F. Campean, and H. Williams (2017). Function modeling using the system state flow diagram. *Artif. Intelligence for Engin. Design, Analysis and Manufacturing 31*(4), 413–435.

Zio, E. and G. Sansavini (2011). Modeling interdependent network systems for identifying cascade-safe operating margins. *IEEE Trans. on Reliability 60*(1), 94–101.