Analysing Residual Risks when Introducing Monitoring and Diagnosis into Systems

Thomas Hirsch, and Franz Wotawa*
Graz University of Technology, Institute for Software Technology, Graz, Austria
thirsch@ist.tugraz.at, wotawa@ist.tugraz.at
*corresponding author

Abstract—Systems under operation come with risks, i.e., a likelihood that a fault causes unwanted events or even harm. In the case of safety-critical systems like cars or airplanes, identifying and mitigating risks is essential for avoiding such critical events. Measures for mitigation, including monitoring and property checking, also come with risks. Not being able to classify a failure correctly or coming up with errors or warnings without reason may cause trouble, too. Therefore, it is evident to analyze the remaining risks (i.e., the residual risks) and compare them with the original ones. This paper presents a framework for analyzing such risks, show their application when introducing monitoring and mitigation, and presents a case study using concrete values.

Keywords–Risk and reliability analysis; fault diagnosis and detection; dynamic fault trees

1. Introduction

Risk in a technical system, can be decreased by risk avoidance, or risk reduction [1]. The latter approach does so by reducing the probability and/or severity of a certain unwanted event and its outcome. Examples of such risk reduction measures are the introduction of redundancies and automated fault correction/mitigation strategies. After introducing risk reduction measures there is still a risk remaining, i.e., the residual risk, which should be smaller than the original risk without applying risk reduction measures.

In order to perform such automated correction or mitigation, an underlying fault has to be diagnosed or at least detected. Model-based diagnosis (see e.g., [2], [3] or more recently [4]) is one of the many research fields around detecting, isolating, and diagnosing faults in technical systems in an automated fashion. For an overview, we refer the interested reader to the literature surveys of Venkatasubramanian et al. [5] and Gao et al. [6]. Other work include monitoring for failure detection, i.e., [7], which gained attention with the increasing interest in autonomous driving. There identifying critical scenarios and mitigating their negative effects is important for preventing people from harm.

A wide variety of metrics and attributes are used in model-based diagnosis literature and research. Ranging from quantitative and well-defined metrics as for example, “accuracy”[5], [8], [9], “misdetection rate”[8], [10], “false alarm rate”[11], [8], [10]. To more abstract and qualitative attributes as for example, “coverage”[12], [8], “robustness”[5], [8] or even “diagnostic performance”[6]. However, none of these metrics and attributes intuitively translate into a quantifiable risk reduction.

Risk assessment and analysis are performed in order to identify sources of risk and to select areas and parts to be treated with risk reduction measures. Further, such analysis is then applied to evaluate and quantify the performance of implemented risk reduction measures, and their effects on residual risk. The concepts of coverage [13] and coverage models [14] are used to model risk reduction measures.

The risk analysis approach based on the coverage [13] metric combines diagnostic recall and the effectiveness of the corresponding mitigation into one parameter but is oblivious to false positives and their effects. Quantitative diagnostic performance metrics for diagnostic systems, for example, “accuracy” and “false alarm rate”, do not encompass the corresponding mitigation actions and any information on their effectiveness and associated risks.

To bridge this gap between the world of risk analysis and automated fault diagnosis we create a simple risk/reliability model to link these detection/diagnosis performance metrics to the achievable risk reduction in a quantifiable manner. We create a scalable model comprised of abstract representations of a component and its diagnostic monitor using Dynamic Fault Trees (DFT) [15]. Further, we discuss the different types and sources of risks in such a system. We use this model to analyze and catalog the various states, failure modes, and resulting risks occurring in such a system, and perform rudimentary sensitivity analysis.

We see two main use cases for our model: First, for the creation and analysis of performance requirements for diagnostic systems and corresponding mitigation actions. Second, to catalog failure modes and to serve as a starting point for in-depth risk analysis of a system incorporating automated diagnostic and mitigation.

We demonstrate how a parameterized model can be used to connect its diagnostic monitors’ performance metrics and mitigation strategies to quantifiable risk metrics. Our model is scalable and modular while still open to more detailed risk analysis once more information is available on the modeled system.

We organize this paper as follows: First, we discuss related research work in the context of reliability and diagnosis. Afterward, we discuss the background behind our work and give an example. We continue describing the modeling principles and the model. Furthermore, we outline details of an implementation and obtained results. Finally, after discussing limitations and the threats to validity, we conclude the paper.

2. Related work

Bouricius et al. [13] introduced the coverage parameter defined as $C = P(system\ recovers|fault\ occurs)$ to model the self-repair and fault tolerance qualities and capabilities of a system. Coverage, by definition, combines diagnostic performance in terms of recall and the corresponding mitigation effectiveness in a single parameter. Since then, the concept of coverage has been well adopted in the field of risk and reliability analysis, and its implications and applications have been actively researched. Dugan et al. [14] discusses different coverage models and their application to compute/estimate the coverage parameter, including sensitivity analysis of coverage regarding different error handling strategies. Doyle et al. [16], [17] discusses the application and combination of coverage models to reliability and dependability assessment approaches. Amari et al. [18] on finding optimal configurations of systems with imperfect coverage. An example of the practical application of the coverage parameter and the utilization of the coverage parameter in DFTs is shown by Ghadhab et al. [19].

However, its lack of expressiveness regarding misdetections/-false positives, and the inseparableness from mitigation effectiveness, inhibit its use in the field of automated fault detection and diagnosis.

The metrics popular in the field of automated fault detection and diagnosis, e.g. accuracy and false positive rates [5], [8], [9], [11], [10], are not directly applicable to perform risk/reliability assessment of a system, as they do not consider mitigation effectiveness or mitigation risks. For example, the papers referred to in Habibi et al.'s [20] survey on reliability improvements through model-based fault detection, deal with reliability on a rather abstract level instead of quantitative reliability analysis.

It is important to note here that the concept of "coverage" as used in the context of automated fault diagnosis [12], [8] is different and unrelated from the "coverage" parameter used in risk and reliability analysis [13].

This paper differs from the above mentioned papers in the following: We want to investigate the types of risks and sources of risks in a system using automated fault diagnosis for risk reduction. We want to bridge the gap between diagnostic performance and system residual risk, and system reliability. We want to enable quantitative analysis of system residual risk and system reliability. We do so primarily from the standpoint of fault diagnosis and detection.

3. Background

Fault trees are well known and in widespread use for risk and reliability analysis of technical systems. This is largely owed to their ease of use, resulting from intuitive representation using directed acyclic graphs to model failure propagation through a system [21].

However, static fault trees have limited modeling capabilities when it comes to dynamic and temporal aspects of a system. Examples are the order of events and the concept of dormancy. While such behaviors can be represented using Markov models, such models suffer from state explosion and are difficult to manually create [22].

Dugan et al. [22] proposed Dynamic fault trees that enable modeling dynamic and temporal behaviors using special gates in fault trees. The following gates are introduced in the original publication: **Priority-And** gate, where the output occurs only if both inputs occur, and the left input occurred before the right input. **Sequence-Enforcing** gate enforcing an order of occurrence of its inputs, in the sequence of left to right inputs. **Functional-Dependency** gate, where the dependent events are forced to occur if the trigger event occurs. **Cold-Spare** gate, where the spare unit is considered dormant, therefore can not fail, before it is activated by the trigger event on the input. Multiple spares can be connected to the same gate, becoming active only if the previous spare has failed. The output of the gate occurs once all spares have failed. It is important to point out the difference between occurrence and activation, the latter describing that the respective component is from this point on subject to its failure probability. **Hot-Spare** gate, where the spare units are active, and therefore can fail according to their given failure before the triggering event occurs. **Warm-Spare** gate, where the spare units are subject to a certain failure rate while not being activated.

Later extensions include: **Priority-Or** gate, where the output occurs if the left fails before the right input [23]. **One-shot Probabilistic Depencency** gate, as a modified functional dependency gate that allows specifying a probability for triggering the failure of the dependent events[24], [25]. **Persistent Probabilistic Depencency** gate, that increases the failure probability of the dependent events by a specified amount[25]. **Mutex** gates, to define events to be mutually exclusive. While such exclusions can be modeled using basic DFT gates [26], they are often depicted as dedicated gates for sake of clarity.

However, there are limitations to dynamic fault trees. Dugan et al. [22] based his proposed DFT modelling approach on following assumptions: Faults are considered to be random and statistically independent, failure rates are assumed constant, mission times are to be short, and repairs during system runtime are not considered. Junges et al. [26] investigated scopes, coverage, and semantics of different flavors of dynamic fault trees discussing problems such as undefined behaviors. The introduction of Priority-Or and Not gates can lead to non-coherent dynamic fault trees, in which the occurrence of an event results in situations where the output of a gate can go from True to False again [27], [26].

In this work, we limit ourselves to the use of only basic gates, as to keep our proof-of-concept model independent of implementation and semantic differences of the various flavors and tools for analyzing dynamic fault trees.

Multiple approaches to quantitative analysis of dynamic fault trees have been investigated, including, but not limited to algebraic [28], Markov chains [15], dynamic Bayesian networks [29], and Monte Carlo simulation [30]. For our quantitative analysis and simulation of our models we use the

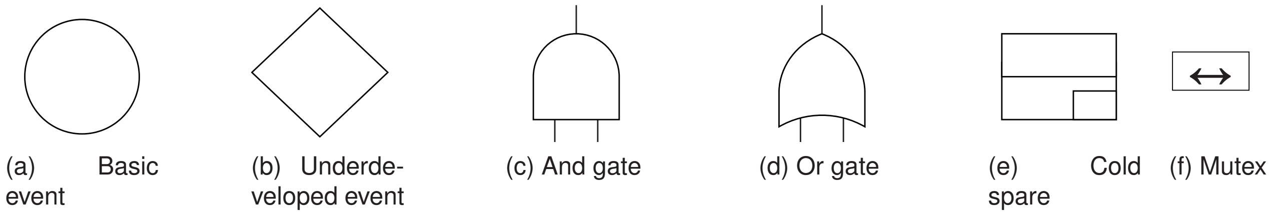


Figure 1. Gates and components of our dynamic fault trees.

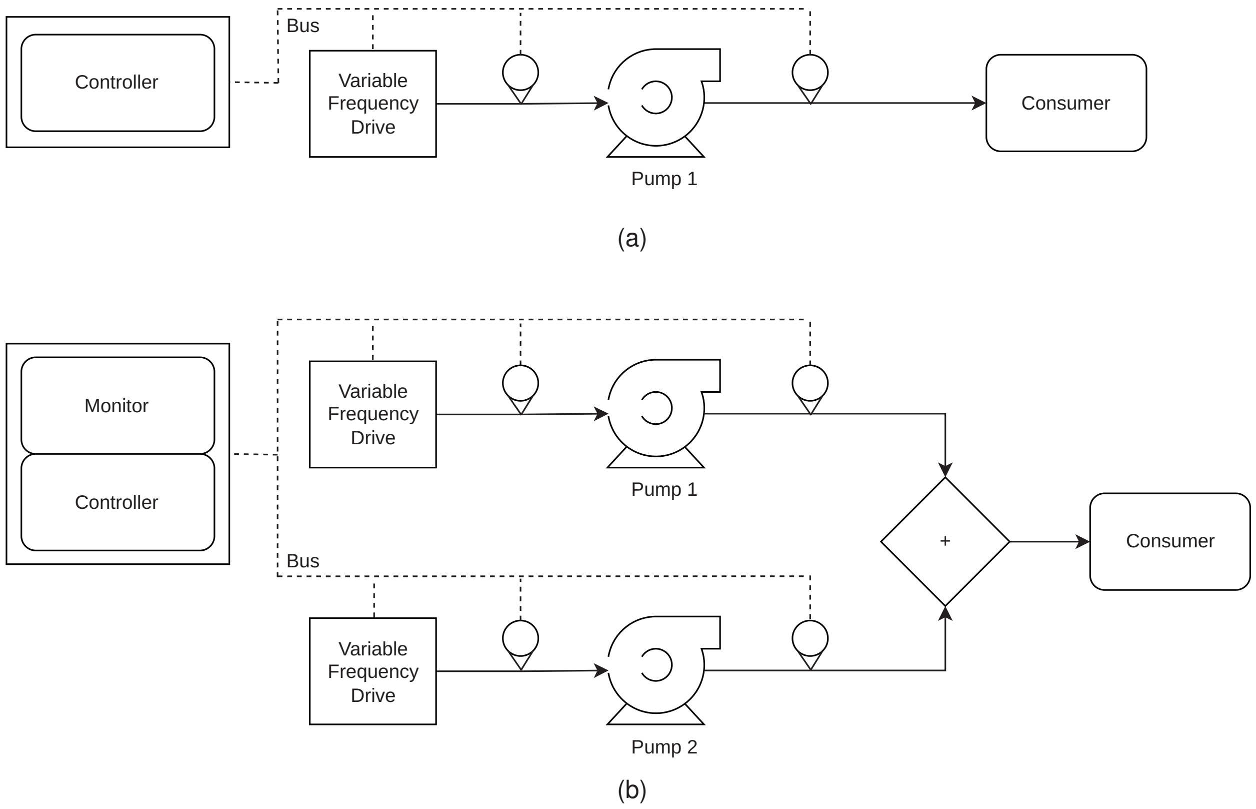


Figure 2. Exmplary system with a single pump, and with a backup pump.

probabilistic model checker *Storm* [31], due to the project's maturity level, availability, good documentation, and open source license.

4. Example

Figure 2a shows a simple system a pump and its consumer. The pump is powered by variable-frequency drivers. There are sensors on the pumps' output, measuring its transport volume, and on the output of the variable frequency drive. The controller for the system is connected to all sensors and the driver over a bus. The goal of the system is to provide a constant flow of liquid to the consumer.
We assume the system to fail by not meeting the goal, when either the pump, controller, or the driver fail.

Figure 2b extends upon the previous system by adding an identical backup pump (pump 2) and a new component we refer to as fault monitor. A fault monitor in this regard is a device performing automated fault diagnosis to detect and diagnose faults and to trigger appropriate mitigation actions in order to avert failure.

The fault monitor in our system is hosted on the same hardware as the controller and connected to the bus. If the fault monitor detects a fault in the pump 1 subsystem, the backup pump 2 is switched on in order to avoid missing the system goal.

Given a perfect monitoring device, the overall probability of not meeting the system goal is cut in half, as now both pump subsystems have to fail in order to reach this failure state. However, this assumption of a perfect monitoring device does not hold in practice, as perfect detection performance is deemed unrealistic and increased system complexity comes at a price.

Fault detection performance: If the monitor does not detect the failure of pump 1, the system will still fail. Success therefore strongly depends on the fault monitor's True Positive Rate (TPR, recall, or sensitivity), where a low TPR leaves us with a similar risk as the unmonitored system, and with a perfect monitor providing a $TPR = 1$.

However, the system is also sensitive to the monitor's False Positive Rate (FPR, or fallout), as uncalled switching to the backup pump does not contribute to our initial goal of risk reduction.

A theoretical fault monitor with perfect $TPR = 1$ but a $FPR = 1$ will always directly switch to the backup pump, which will leave us again with a vulnerable single-pump system. A high FPR could further increase the overall system risk if the mitigation action brings additional risks. For example, a backup pump and connected components having a shorter lifespan than the main pump would lead to an overall shorter time to failure with a high FPR than the unmonitored single pump system.

Further, an uncalled mitigation action being performed on an otherwise faultless system could introduce additional risk. For example, the startup process of a pump could be more demanding on its components than uninterrupted operation, therefore the act of switching itself introducing new risk, or even extreme scenarios such as overloading the piping and consumer due to starting up the backup pump on top of a running main pump.

Increased complexity: The introduction of the monitoring device can have negative effects on the component that it is monitoring, depending on what functional parts it shares with this component. In our example, this would be the shared computing platform with the systems controller. The monitoring software could lead to a slowdown, memory corruption, crash, or freeze of the computing platform that also runs the controller software.

While all above discusses the effects of the monitor on its component, there is yet another factor to be considered - shared structures that may be used by other, maybe unrelated, components in a bigger system. If the bus in our pump example is used by other components, they could fall victim to our monitor device becoming a babbling idiot, rendering the bus unusable for other participants. All shared structures that are connected to the monitor have to be examined and the resulting risks evaluated. This ranges from shared power supplies, computing and communication structures, to physical structures, for example, additional stresses or corrosion due to sensors integrated solely for the sake of monitoring.

5. Model

In order to support risk analysis and decision-making around the integration of a fault monitoring device, we want to formalize and model the effects occurring in our previous example using DFTs.

TABLE I
MONITORED COMPONENT STATES

	Component working X	Component fault X
Monitor negative X	TN_X	FN_X
Monitor positive X	FP_X	TP_X

A monitored component consists of a functional component and its corresponding monitoring device. This monitoring device can detect and diagnose faults in the component, and perform corresponding mitigation in order to avoid component and/or system failure.

5.1. Single fault model

We assume that the component is either in working condition or subject to a fault X. We further assume that the states are sticky, for example, faults cannot be repaired, and a monitor positive for a certain fault will stay in this positive state.

Monitor detection performance: The monitoring device's detection performance can be defined by its True Positive Rate (TPR, recall, or sensitivity) and False Positive Rate (FPR or fall-out) for each fault X.

Modeling component/monitor states: We define our BEs (Basic Events) to be *true positive*, *false positive*, *false negative*, representing the status of the monitored system. Creating a BE for *true negative* is superfluous as it represents the nominal fault-free state.

Table I lists the monitored components' states in form of true positive for fault X (TP_X), false positive for fault X (FP_X), and corresponding negatives (TN_X and FN_X), depending on the underlying component state and the monitor's output.

We mutually exclude TP_X, FP_X, and FN_X as only one of them can occur simultaneously, shown in Figure3.

In order to model the effects of monitoring performance on resulting risk we need to consider the next step in the monitoring process described above, the triggered mitigation actions. Positive monitor output for fault X triggers a corresponding mitigation action. Such mitigation action has specific risks attached to it.

A mitigation action can fail, or may even be harmful to a healthy system not exhibiting the fault. We model these failure paths as abstract underdeveloped events representing the mitigation being performed on a faulty component (true positive), and the mitigation being performed on a healthy component (false positive).

These underdeveloped events are activated by their corresponding monitored components state through a spare gate.

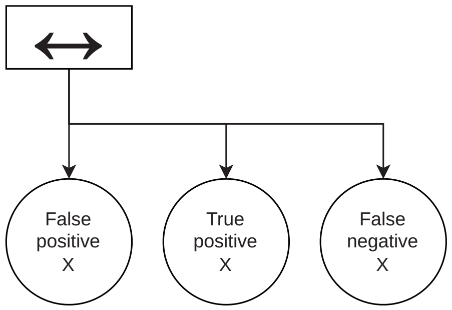


Figure 3. Mutex for monitor/component states.

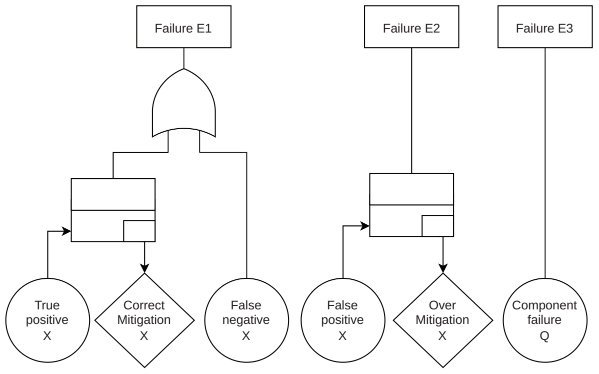


Figure 4. Failure trees E1-E3 resulting from component and monitor state

Combining all of the above-defined DFTs and their events we can model the monitored components' fault modes and their resulting failure modes. We define five different failures $[E1, E2, E3, E4, E5]$ in order to accommodate for different severity measures for each failure.

The DFT for Failure $E1$ shown in Figure 4, is where a reduction in risk is achieved over the un-monitored component by the introduction of the monitor. Failure $E1$ requires either a false negative of the monitor for fault X or a true positive for fault X of the monitor in conjunction with the corresponding mitigation action failing. In an un-monitored system, Failure $E1$ would be a direct cause of component fault X.

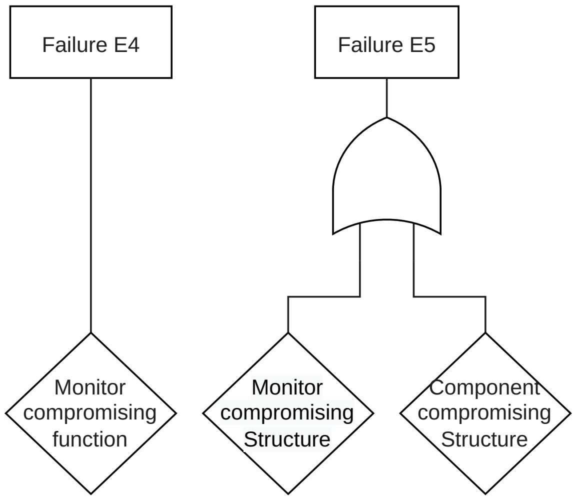
The DFT for Failure $E2$, see Figure 4, models the possibility of an unjustified mitigation action having a negative impact on the system. Failure $E2$ does not exist in the un-monitored system.

The DFT for Failure $E3$, see Figure 4, models the component faults that are not detectable by the monitoring device. The risks resulting from Failure $E3$ are the same for the monitored and un-monitored versions of the system.

Monitoring device-based risks: In addition to the aforementioned monitor performance-based risks, the introduction of a monitoring device itself adds risks because of increased system complexity and required resources.

Failure $E4$ (see Figure 5) models the possibility of the monitoring device compromising or deteriorating the components' function. Reasons for such disturbances of component function can be, for example, the monitoring device sharing computational resources with the component, allowing the monitor to corrupt or slow down the shared resource.

Failure $E5$ models the effects of the monitoring device on structures shared with other components. For example, data buses, electric supply, or even mechanical structures. Given a system composed of multiple components, monitor devices, and structures, each structure is to be modeled as such a tree, incorporating all components and monitors connected to this structure. Figure 5 shows a minimal DFT for Failure $E5$ for a system comprising only one component and its monitor. Given a structure that is critical to multiple components, and shared by those components and their monitors, all of the involved components have to be incorporated. This DFT serves as an illustration and approximation, while in practice it may be replaced by risks and probabilities determined using structural reliability analysis.



Failure E4
Failure E5
Monitor compromising function
Monitor compromising Structure
Component compromising Structure

Figure 5. Failure trees E4-E5, monitor compromising function and structure.

Multiple faults and multiple components Assuming not only the independence of faults but also the independence of their corresponding mitigation actions or misdetections, as well as the exclusion of emergent or combined effects - a multi-component, multi-fault system can be modeled using simple replication of the DFTs described above. In order to do so, the sub DFTs for $E1$ and $E2$, (see Figure 4) have to be replicated for each specific fault of a single component.

TABLE II
MULTI FAULT MONITORED COMPONENT STATES

	FP X	FN X	TP X	TN X
FP Y	Complete overmitigation	Wrong mitigation	Partial overmitigation	SFT
FN Y	Wrong mitigation	Complete undermitigation	Partial undermitigation	SFT
TP Y	Partial overmitigation	Partial undermitigation	Complete mitigation	SFT
TN Y	SFT	SFT	SFT	SFT

The failure trees $E3$ (see Figure 4) as well as $E4$ (see Figure 5). have to be replicated for each component in the system. The failure tree $E5$ (see Figure 5) is to be replicated for each structure in the system.

5.2. True multi fault model

Considering such emergent and combinatory effect, while allowing multiple simultaneous faults to occur, increases model complexity significantly. For the following discussion we therefore limit the number of simultaneously occurring faults to two, in order to provide a manageable scope.
The occurrence of multiple simultaneous faults constitutes sources of risks, including emerging or combinatory effects from simultaneous component faults, mitigation actions, and their possible combinations. Table II provides an overview of the newly introduced states for combining two faults. Each state in Table II is defined by severity and manifestation probability.
The bottom row and right column, of Table II contains the states already covered by the single-fault trees discussed in the previous section (see 5-A).
The states/failure modes described in Table II can be modeled using cold spare gates, activating the abstract events representing combinatory and emerging issues. Figure 6 shows the DFT for *Complete overmitigation* as an example.
This results in nine additional states, comprised of six different types of failure modes that we briefly discuss:
Complete overmitigation describes the application of two simultaneous mitigation actions to a healthy system and includes any additional risks introduced by such combination over the risks are already modeled in the false positive branches of failure tree $E2$ in Figure 4.
Complete undermitigation describes two simultaneous unmitigated faults and incorporates the additional risk from this combination over their standalone risks as modeled in failure tree $E1$ in Figure 4.
Complete mitigation describes two simultaneous mitigated faults and contains additional risks that may arise from such a combination.
Wrong mitigation describes the combination of a false positive and false negatives and models the risks of this specific mitigation action being applied on a component in another fault mode beyond the risks already modeled in $E1$ and $E2$ in Figure 4.

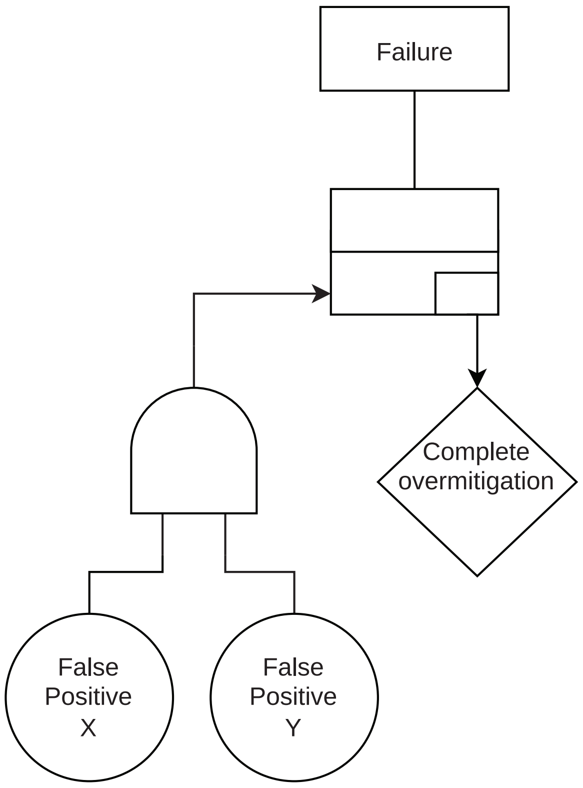


Figure 6. DFT for *Complete overmitigation* in a system with two monitored faults X and Y.

Partial overmitigation and partial undermitigation model the states in which either two different mitigation actions are applied to a single fault, or only one mitigation action is applied to two faults. Again, the risks arising from the specific combinations beyond their standalone effects, already modeled in $E1$ and $E2$ in Figure 4, are to be considered.
The resulting matrices can be translated again into DFTs representing underlying states. These fault/failure matrices can of course be sparse. When generating the DFTs, events with either zero probability or failures with zero cost can be omitted.

5.3. Scalability

Manual creation of such a DFT model for multiple components and monitors with multiple faults quickly becomes cumbersome. However, under the assumption of independence, this model could be scaled by replication of the DFTs to the desired number of components, monitors, and structures and their associated faults. The inclusion of emergent behaviors and combinatory effects to model multi-fault scenarios will lead to an exponential growth of model size.
While such scaling by replication can be trivially automated, enabling the creation of big models containing numerous

faults, the scalability of DFT simulation tools in use has to be considered.

6. Implementation and Results

In the following subsections, we discuss the results obtained using our example application considering single and multiple faults.

6.1. Single fault model

We assume a minimalistic, exemplary system consisting of a single component and its corresponding monitor. There are two possible types of component faults Q and X, the latter being detectable by the monitoring device with given $Recall$, with the monitor exhibiting a given $Fall\text{-}out$ rate for fault X. Based on our proposed single fault scenario DFTs from Section 5-A, we implement this model using *Storm Dft*[31][1]. For the sake of simplicity, we implemented all BEs as Poisson point processes on a real line representing time with a constant rate λ.

While it would be possible to model the abstract events TP_X, FP_X, and FN_X using probabilistic dependency gates in theory, doing so would require the inclusion of XOR, NEG, or POR gates, possibly resulting in non-coherent DFTs [26]. Further, not all DFT software packages, including Storm, support the required gates.

We therefore calculate the rates λ_{TP_X} and λ_{TP_X} based on component failure rate $\lambda_{ComponentFailureX}$ and monitor detection performance as follows:

$$\lambda_{TP_X} = Recall_X * \lambda_{ComponentFailureX}$$

$$\lambda_{FN_X} = (1 - Recall_X) * \lambda_{ComponentFailureX}$$

We consider our system failed when any of the DFTs $E1$ to $E5$ failed and modeled this combination using OR gates. Table III shows the default parameters used in our implementation. Listing 1 shows the *Storm Dft* source code of our model.

```
toplevel "top";
"top" or "failure_e1" "failure_e2" "failure_e3"
    "failure_e4" "failure_e5";

"failure_e1" or "false_negative_x" "
    mitigation_failed";
"mitigation_failed" csp "true_positive_x" "
    correct_mitigation_x";

"failure_e2" csp "false_positive_x" "
    over_mitigation_x";

"failure_e5" or "monitor_compromise_structure" "
    component_compromise_structure";

"m" mutex "true_positive_x" "false_negative_x" "
    false_positive_x";

"failure_e3" lambda=1.000e-08 dorm=0;

"failure_e4" lambda=1.000e-09 dorm=0;

"true_positive_x" lambda=9.900e-07 dorm=0;
"false_negative_x" lambda=1.000e-08 dorm=0;
"false_positive_x" lambda=1.000e-05 dorm=0;

"over_mitigation_x" lambda=1.260e-06 dorm=0;
"correct_mitigation_x" lambda=1.000e-06 dorm=0;

"monitor_compromise_structure" lambda=1.000e-09
    dorm=0;
"component_compromise_structure" lambda=1.000e
    -08 dorm=0;
```

Listing 1. Storm DFT of our single fault model.

[1]https://www.stormchecker.org/about.html

TABLE III
Default parameters of our single fault model

Parameter	Value
$\lambda_{ComponentFailure_X}$	10^{-6}
$Recall_X$	0.99
$\lambda_{FallOut_X}$	10^{-5}
$\lambda_{OverMitigation_X}$	$10^{-5.9}$
$\lambda_{CorrectMitigation_X}$	10^{-6}
$\lambda_{ComponentFailure_Q}$	10^{-8}
$\lambda_{MonitorCompromisingFunction}$	10^{-9}
$\lambda_{MonitorCompromisingStructure}$	10^{-9}
$\lambda_{ComponentCompromisingStructure}$	10^{-8}

We performed sensibility analysis using this *Storm* implementation of our single fault model. We visualize the effects of the diagnostic monitors' performance and mitigation performance on overall system reliability in terms of Mean Time To Failure (MTTF).

Figure 7 shows the systems MTTF in dependence of the monitors $Recall$ for various $Fall - out$ rates. Trivially, larger $Recall$ results in a higher probability of detecting and mitigating a failure X, therefore increasing the MTTF of the system. However, Figure 7 also shows that the $Fall - out$ rate has a significant and more profound impact on the systems MTTF, as high $Fall - out$ rates can result in lower overall reliability than an unmonitored system.

This results from the combination of $Fall - out$ rates and risks from $Overmitigation$. Figure 8 shows the systems MTTF in dependence on $Fall - out$ rate for various $Overmitigation$ failure rates. If $Overmitigation$ does not come at an additional cost, therefore $\lambda_{OverMitigation_X}$ being the same as $\lambda_{ComponentFailure_X}$ and $\lambda_{CorrectMitigation_X}$, a high $Fall - out$ rate results in no benefit in terms of system MTTF over an unmonitored system. At lower $Fall - out$ rates, the system can benefit in terms of increased MTTF. However, if $\lambda_{OverMitigation_X}$ comes at a cost in terms of an increased failure rate, the addition of the diagnostic monitor and mitigation actions can make the system less reliable than in its original state.

In both Figures 7 and 8 we can observe the MTTF converging towards a limit. In order to investigate the limiting factors, Figure 9 shows the systems' MTTFs' dependence on the components fault X rate for various monitor-caused structural and functional fault rates. We can observe that given a low enough rate of component fault X, the systems MTTF can not

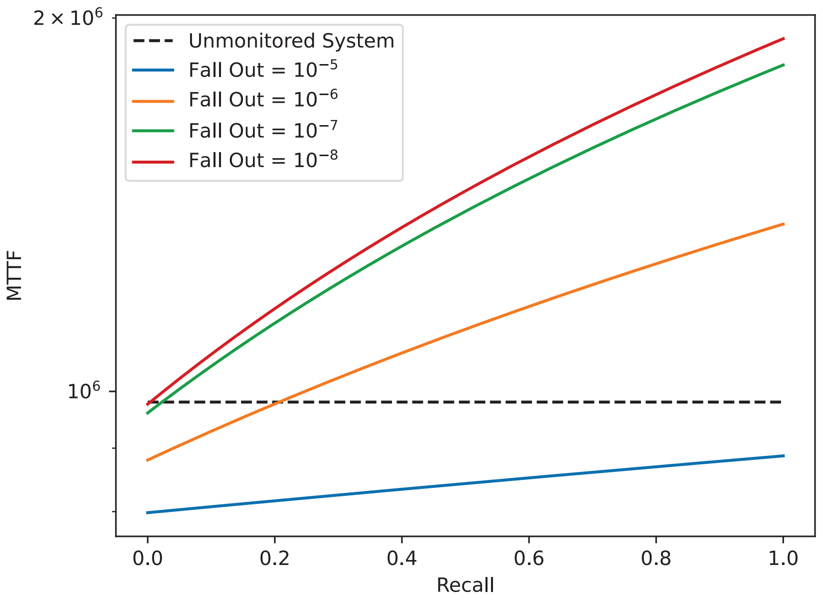


Figure 7. Recall

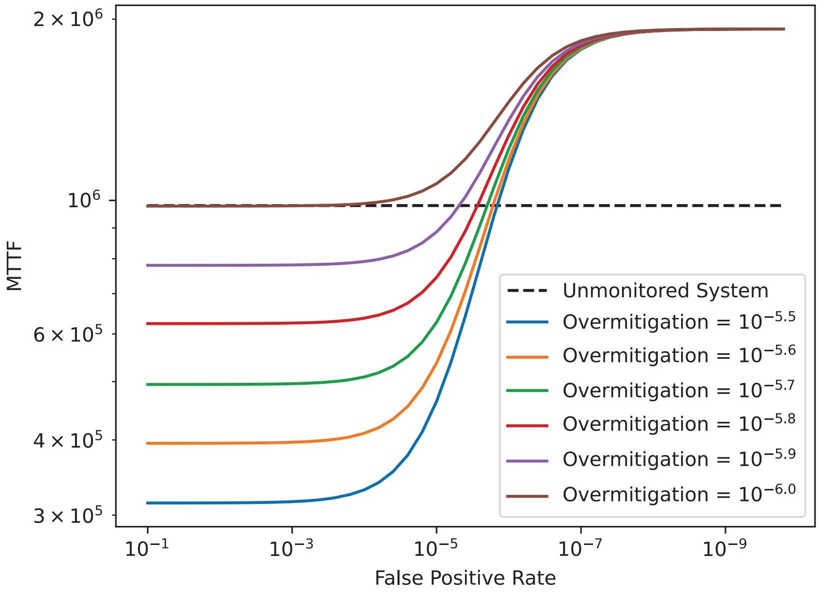


Figure 8. False Positive Rate

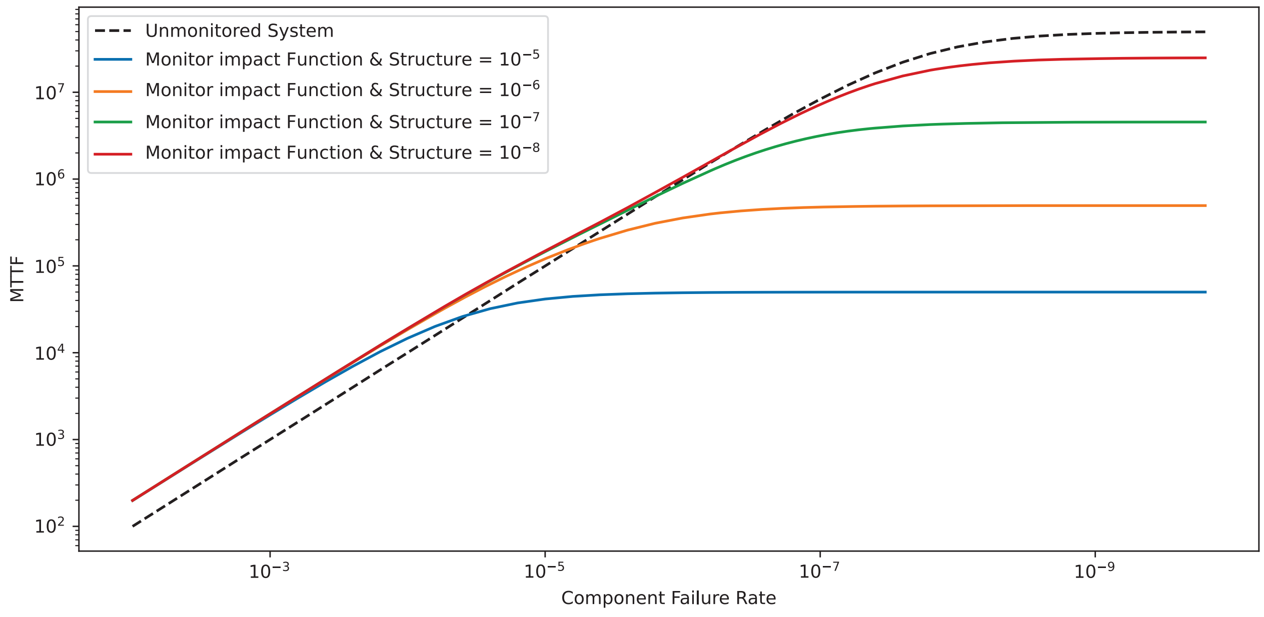


Figure 9. Component Failure Rate

TABLE IV
ADDITIONAL DEFAULT PARAMETERS FOR MULTI FAULT MODEL

Parameter	Value
$\lambda_{CompleteMitigation_{XY}}$	$10^{-5.9}$
$\lambda_{CompleteOvermitigation_{XY}}$	10^{-5}
$\lambda_{CompleteUndermitigation_{XY}}$	10^{-1}
$\lambda_{WrongMitigation_{X}}$	10^{-4}
$\lambda_{PartialOvermitigation_{X}}$	$10^{-5.9}$
$\lambda_{PartialUndermitigation_{X}}$	$10^{-5.9}$
$\lambda_{WrongMitigation_{Y}}$	10^{-4}
$\lambda_{PartialOvermitigation_{Y}}$	$10^{-5.9}$
$\lambda_{PartialUndermitigation_{Y}}$	$10^{-5.9}$

benefit any more from the addition of the diagnostic monitor as it is hampered by the additional complexity reflected in additional structural and functional fault rates.

6.2. Multi fault model

Based on our single fault model, we create a multi-fault model comprising two different diagnosable faults X and Y. Rates for both faults, including their corresponding monitor and mitigation performance, are parameterized the same way as the single fault model listed in Table III.
Additional parameters used to represent combinatory and emergent behavior are shown in Table IV.
Some states of this model are not reachable with the given rates. For example, full and partial under mitigations are due to a single unmitigated fault, as the model is already in a failed state. However, we include them to create a model that conforms with our model definition in Section 5-B, to enable reusability of our implementation, and to enable investigations into combinatory effects in separation.
We use our *Storm* implementation of our multi-fault model to demonstrate its suitability for sensitivity analysis. As an example, Figure 10 shows the effect of $\lambda_{CompleteOvermitigation_{XY}}$ on MTTF for various $Fall-out$ rates, with and without combinatory effects.

7. LIMITATIONS AND THREATS TO VALIDITY

Order of events: In a real-world system, the order in which faults and states occur can lead to different outcomes and different risks associated with these outcomes. While DFTs offer the capability to model such behavior, it is out of scope for this work and we exclude such effects in our models.
Stickyness: States and events in DFTs are considered sticky, meaning that a path that is taken cannot be reverted. For example, once our model is in the false positive state, the true positive state is not reachable anymore, while in reality, the component could still fail and render the overmitigation into the correct mitigation. This is a limitation of FTs in general. While non-coherent DFTs [26] could be used under certain circumstances, more complex approaches, for example, Markov Chains, are required to model such behavior.
Representation of real-world events: Modeling our monitor/-component states as independent Poisson point distributions that are mutually excluded does not capture their actual dependence. The calculation of λ_{TP_X} and λ_{FP_X} used in our sensitivity analysis are simplifications that only roughly estimate the real process.

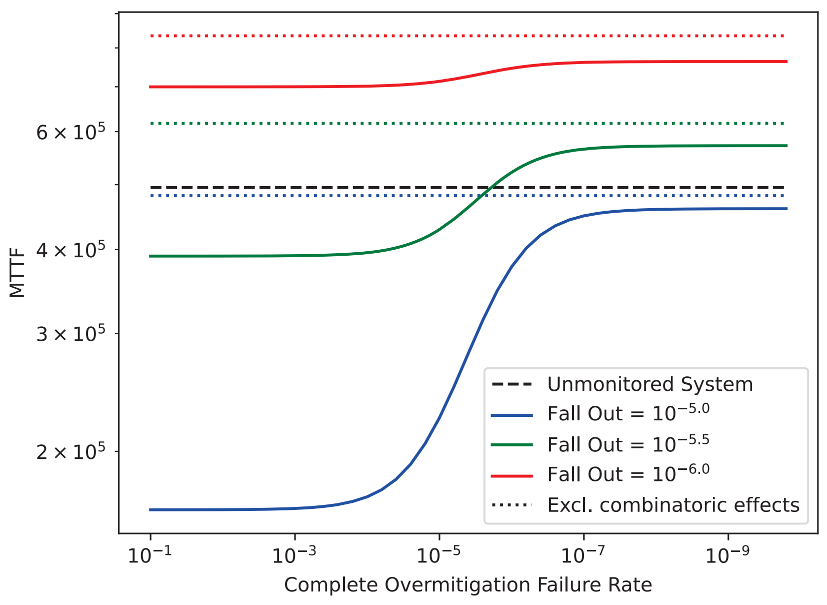


Figure 10. Overmitigation failure rate

Assumption of fault independence: Our proposed multi-fault modeling approach assumes independence of component failures and monitors diagnosis and detection.

8. Conclusions

In this paper we proposed an abstract DFT model of a technical system containing a fault detection/diagnosis mechanism. We used this model to explore the impacts of a fault diagnosis based risk reduction measure on system residual risk and reliability.

We performed residual risk and sensibility analysis of a small toy example comprised of a single component and its corresponding fault diagnosis monitor and mitigation actions. We investigated fault diagnosis performance, mitigation effectiveness, and additional risks introduced by the diagnostic system and mitigation actions.

We discuss how high $Fall\text{-}out$ rates in combination with the negative effects of mitigation actions can undermine a risk reduction measure, despite good coverage in the form of high $Recall$ and mitigation effectiveness. Further, we discuss the limiting factors for achievable risk reduction, e.g. structural risks, and highlight new risks that are introduced by the risk reduction measure.

Our DFT model was implemented in Storm DFT and uses only well-defined and common DFT gates. We deliberately designed our model to avoid DFT constructs, techniques, and gates that are known to be implementation-dependent. Our DFT models are intended to be easily translatable to other DFT simulation languages aside from Storm DFT.

We showed how our model could be used for rudimentary risk and reliability analysis. We hope that our model is of use for other researchers in determining the performance requirements of a fault diagnosis monitor and its corresponding mitigation actions, or for evaluation of existing systems. Further, our model provides a catalog of risks that can form the basis for a more detailed risk and reliability analysis of such technical systems.

In future work, we will apply our approach to a real-world system in a case study to compare our approach to a conventional coverage-based residual risk analysis approach. Further, we will evaluate the suitability of various DFT modeling software frameworks for our purpose. We will investigate the effectiveness of modularization in increasing scalability, and perform a performance study on bigger systems.

Acknowledgment

ArchitectECA2030 receives funding within the Electronic Components and Systems For European Leadership Joint Undertaking (ESCEL JU) in collaboration with the European Union's Horizon2020 Framework Programme and National Authorities, under grant agreement number 877539. All ArchitectECA2030 related communication reflects only the author's view and the Agency and the Commission are not responsible for any use that may be made of the information it contains. The work was partially funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) under the program "ICT of the Future" project 877587.

References

[1] T. Aven and J. Wiley, *Foundations of Risk Analysis A Knowledge and Decision-Oriented Perspective*. Wiley, 2012. [Online]. Available: www.wileyeurope.com

[2] R. Reiter, “A theory of diagnosis from first principles,” *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987.

[3] J. de Kleer and B. C. Williams, “Diagnosing multiple faults,” *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[4] D. Kaufmann, I. Nica, and F. Wotawa, “Intelligent agents diagnostics - enhancing cyber-physical systems with self-diagnostic capabilities,” *Adv. Intell. Syst.*, vol. 3, no. 5, p. 2000218, 2021.

[5] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, “A review of process fault detection and diagnosis: Part i: Quantitative model-based methods,” *Computers & Chemical Engineering*, vol. 27, pp. 293–311, 3 2003.

[6] Z. Gao, C. Cecati, and S. X. Ding, “A survey of fault diagnosis and fault-tolerant techniques-part i: Fault diagnosis with model-based and signal-based approaches,” *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 3757–3767, 6 2015.

[7] F. Wotawa and H. Lewitschnig, “Monitoring hierarchical systems for safety assurance,” in *IDC*, ser. Studies in Computational Intelligence, vol. 1026. Springer, 2021, pp. 331–340.

[8] K. Tidriri, N. Chatti, S. Verron, and T. Tiplica, “Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges,” *Annual Reviews in Control*, vol. 42, pp. 63–81, 1 2016.

[9] Y. Zhang and J. Jiang, “Bibliographical review on reconfigurable fault-tolerant control systems,” *Annual Reviews in Control*, vol. 32, pp. 229–252, 12 2008.

[10] S. Poll, A. Patterson-Hine, J. Camisa, D. Nishikawa, L. Spirkovska, D. Garcia, D. Hall, C. Neukom, A. Sweet, S. Yentus, C. Lee, J. Ossenfort, O. J. Mengshoel, I. Roychoudhury, M. Daigle, G. Biswas, X. Koutsoukos, and R. Lutz, “Evaluation, selection, and application of model-based diagnosis tools and approaches,” *Collection of Technical Papers - 2007 AIAA InfoTech at Aerospace Conference*, vol. 3, pp. 2287–2312, 2007. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2007-2941

[11] S. Narasimhan and G. Biswas, “Model-based diagnosis of hybrid systems,” *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 37, pp. 348–361, 5 2007.

[12] R. Isermann, “Model-based fault-detection and diagnosis – status and applications,” *Annual Reviews in Control*, vol. 29, pp. 71–85, 1 2005.

[13] W. G. Bouricius, W. C. Carter, and P. R. Schneider, “Reliability modeling techniques for self-repairing computer systems,” in *Proceedings of the 1969 24th National Conference, ACM 1969*. Association for Computing Machinery, Inc, 8 1969, pp. 295–309.

[14] J. B. Dugan and K. S. Trivedi, “Coverage modeling for dependability analysis of faulttolerant systems,” *IEEE Transactions on Computers*, vol. 38, pp. 775–787, 1989.

[15] J. B. Dugan, S. J. Bavuso, and M. Boyd, “Fault trees and sequence dependencies,” in *Proceedings of the Annual Reliability and Maintainability Symposium*. Publ by IEEE, 1990, pp. 286–293.

[16] S. A. Doyle and J. B. Dugan, “Fault trees and imperfect coverage. a combinatorial approach,” in *Proceedings of the Annual Reliability and Maintainability Symposium*. Publ by IEEE, 1993, pp. 214–219.

[17] ——, “Dependability assessment using binary decision diagrams (bdds),” in *Proceedings - Annual International Conference on Fault-Tolerant Computing*. IEEE, 1995, pp. 249–258.

[18] S. V. Amari and J. B. Dugan, “Optimal reliability of systems subject to imperfect fault-coverage,” *IEEE Transactions on Reliability*, vol. 48, pp. 275–284, 9 1999.

[19] M. Ghadhab, S. Junges, J. P. Katoen, M. Kuntz, and M. Volk, “Safety analysis for vehicle guidance systems with dynamic fault trees,” *Reliability Engineering and System Safety*, vol. 186, pp. 37–50, 6 2019.

[20] H. Habibi, I. Howard, and S. Simani, “Reliability improvement of wind turbine power generation using model-based fault detection and fault tolerant control: A review,” *Renewable Energy*, vol. 135, pp. 877–896, 5 2019.

[21] E. Ruijters and M. Stoelinga, “Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools,” *Computer Science Review*, vol. 15, pp. 29–62, 2 2015.

[22] J. B. Dugan, S. J. Bavuso, and M. A. Boyd, “Dynamic fault-tree models for fault-tolerant computer systems,” *IEEE Transactions on Reliability*, vol. 41, pp. 363–377, 1992.

[23] M. Walker and Y. Papadopoulos, “Qualitative temporal analysis: Towards a full implementation of the fault tree handbook,” *Control Engineering Practice*, vol. 17, pp. 1115–1125, 10 2009.

[24] S. Montani, L. Portinale, A. Bobbio, and D. Codetta-Raiteri, “Radyban: A tool for reliability analysis of dynamic fault trees through conversion into dynamic bayesian networks,” *Reliability Engineering and System Safety*, vol. 93, pp. 922–932, 7 2008.

[25] L. Portinale, D. C. Raiteri, and S. Montani, “Supporting reliability engineers in exploiting the power of dynamic bayesian networks,” *International Journal of Approximate Reasoning*, vol. 51, pp. 179–195, 1 2010.

[26] S. Junges, D. Guck, J. P. Katoen, and M. Stoelinga, “Uncovering dynamic fault trees,” in *Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016*. Institute of Electrical and Electronics Engineers Inc., 9 2016, pp. 299–310.

[27] E. Edifor, M. Walker, and N. Gordon, "Quantification of priority-or gates in temporal fault trees," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7612 LNCS, 2012, pp. 99–110.

[28] J. Ni, W. Tang, and Y. Xing, "A simple algebra for fault tree analysis of static and dynamic systems," *IEEE Transactions on Reliability*, vol. 62, pp. 846–861, 12 2013.

[29] S. Montani, L. Portinale, A. Bobbio, S. Montani, L. Portinale, and A. Bobbio, "Dynamic bayesian networks for modeling advanced fault tree features in dependability analysis," ResearchGate, Tech. Rep., 2014. [Online]. Available: https://www.researchgate.net/publication/229458364

[30] K. D. Rao, V. Gopika, V. V. S. Rao, H. S. Kushwaha, A. K. Verma, and A. Srividya, "Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment," *Reliability Engineering and System Safety*, vol. 94, pp. 872–883, 4 2009.

[31] C. Dehnert, S. Junges, J. P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10427 LNCS. Springer Verlag, 2017, pp. 592–600.