Investigating Availability Computation with Exponential Failure Rates and Fixed-Time Repair Delays

SILAS DE GRAAF, University of Twente, The Netherlands

Fault tree analysis is a widely used method for assessing system risks and reliability. In repairable fault trees, components are typically modeled with exponentially distributed failure and repair rates. However, a novel approach is to consider the repair time of these components to be fixed. Researchers at the University of Twente have developed a theorem for computing the time-specific availability of such components. Currently, the applicability of the theorem to real-life scenarios is unexplored, and there is a lack of practical implementations. Moreover, the theorem's current state lacks exploration of potential extensions. An existing case study was identified to demonstrate the practical applicability of the theorem. Furthermore, a Python tool was developed which utilizes the new approach. The study also explored extensions to the basic theorem. These include the calculation of other properties for components with an exponentially distributed failure rate and fixed time delay, as well as the availability computation for components with more complex failure behaviour. Through experiments, insights were gained into the change in component availability over time. The most significant finding is that accurately describing failure behaviour and considering fixed repair times improves our ability to estimate component availability effectively. Ultimately, the study serves as a first expedition into the possibilities presented by the novel approach. Multiple avenues for future research into the topic are identified.

Additional Key Words and Phrases: Fault Tree Analysis, Repairable Fault Trees, Fixed Repair Time, Availability, Phase-Type Failure

1 INTRODUCTION

Fault tree analysis is a commonly used technique in safety and reliability engineering to assess the durability of safety-critical systems. The technique is based on the assumption that the connection between individual components and events of a system can be expressed as a combination of logic gates. See Figure 1 for an example. The top level node represents a critical event such as system failure or any other undesired outcome. The leaf nodes represent basic events such as a component failure or a human error. Qualitative analysis of a fault tree diagram makes it possible to identify critical parts of a system by constructing the minimum combinations of basic events that cause the critical event, also called minimal cut sets. When it is possible to model the behaviour of individual components, it becomes feasible to conduct a quantitative analysis. Quantitative analysis enables engineers to go beyond qualitative assessments and obtain numerical values that provide a deeper understanding of the system's reliability and availability. Examples of fields that make use of fault tree analysis include, but are not limited to, aerospace engineering [7], social sciences [10], nuclear engineering [9] and aviation engineering [20, 13].

TScIT 39, July 7, 2023, Enschede, The Netherlands

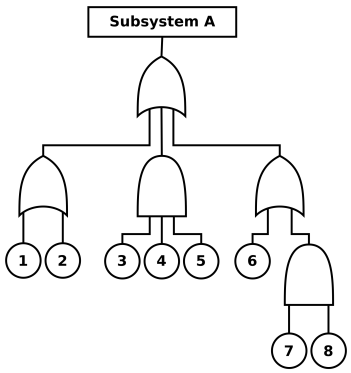

Fig. 1. Example of a fault tree diagram [11].

Currently, a prevalent method of modeling individual components is by means of continuous-time Markov chains (CTMC). Each state in the chain represents a specific condition of the component, which coincides with one of the two possible values a component can take in the fault tree: 'working' or 'broken'. The transitions between states occur probabilistically over continuous time, governed by exponentially distributed variables. An enhancement to basic fault tree models is the idea of repairable fault trees. In basic fault tree models, states which represent the component being broken are absorbing, that is, they never return to a working state. In repairable fault trees, components can be replaced or repaired. The time that it takes to perform such an operation can be expressed in a CTMC with a transition back to a working state.

A novel approach is to consider the required time for a repair as a fixed value. An advantage of this approach is that the required time for a repair of a component is often much more predictable than its failure. Repair processes are typically planned, and the necessary resources, such as spare parts and maintenance personnel, are allocated in advance. As a result, in some cases, the repair time for a component can be estimated with a reasonable level of accuracy. By using a fixed repair delay instead of a repair rate, the model provides a more realistic representation of the system. This implies more realistic values of properties computed from the model as well. Models which contain this combination of failure rates with fixed repair delays will henceforth be referred to as CTMC with fixed delay (CTFD). These CTFDs have thus far not been considered in the literature. A theorem to compute the availability at a given time for a CTFD with two states has been developed by Dr. E.M. Hahn and Dr. M.A. Lopuhaä-Zwakenberg from the FMT group at the University of Twente.

Definition 1.1. The availability of a system at time t is equal to the probability that the system is in a working state at t.

An illustration of such a CTFD system is depicted in Figure 2. In this example, the state S1 represents the working state, while S2

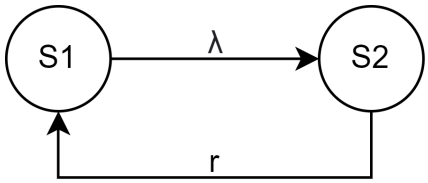

Fig. 2. Basic 2-state CTFD with exponential failure rate λ and fixed repair time r.

represents the broken state. Definition 1.1 makes it clear that we are targeting time-specific properties of fault tree models as opposed to long-term values. Further mentions of availability should be considered to refer to the time-specific availability. When this is not the case, this is made explicitly clear. The aforementioned theorem is presented below.

Theorem 1.2. *Let component C be an stochastically independent basic event of a fault tree diagram with a failure rate λ and a fixed repair delay r. Then the availability $P_C(t)$ can be calculated using the following formula:*

$$P_C(t) = \sum_{i=0}^{\lfloor \frac{t}{r} \rfloor} \frac{(\lambda(t-ir))^i}{i!} e^{-\lambda(t-ir)}.$$

The intuition behind Theorem 1.2 is that due to the fixed repair time there is a limit on the amount of failures that can occur within a timeframe. The availability is equal to the probability of being in a working state at the end of this timeframe. We can split up this probability into a finite number of cases, where each case represents a number of failures. Hence, there exists a case that covers the maximum number of failures, no failures at all and any amount in between. In each case, we account for being unable to fail during a repair. Therefore we subtract the total repair time from the amount of time available within the timeframe. We can then, for each case, compute the probability that the number of failures occur within the remaining time.

The formal proof for Theorem 1.2 is currently under development and will be released in a later paper. Even so, the practical applicability of the theorem to real-life scenarios remains unexplored. Furthermore, there is a notable absence of any practical implementation of the theorem, hindering the ability to conduct experiments and lacking a reference for other researchers to create their own implementation or build upon. Finally, the current state of the theorem remains limited, with no research undertaken to explore potential extensions. This research aims to address these three key problems by investigating the following research questions:

(1) Which case studies exist for which the usage of the approach is appropriate?
(2) How can Theorem 1.2 be implemented for use in fault tree analysis?
(3) How does the availability behaviour of a component described by the 2-state CTFD model differ from other models, and what specific patterns or characteristics does it exhibit?
(4) How can Theorem 1.2 be extended to support the computation of more complex properties?
(5) How can Theorem 1.2 be extended to support more complex CTFD models?

If a case study can be found for which it is reasonable to assume the usage of fixed repair delays, it serves as substantial validation for the practical applicability of Theorem 1.2. To find suitable case studies, we will perform a literature review. Section 2 will encompass both the methodology employed and the outcomes of this review. A Python tool which utilizes Theorem 1.2 to conduct fault tree analysis will be developed to facilitate experimentation and serve as a foundation upon which the theorem can be expanded. Architectural choices as well as implementation details will be provided in Section 3. Some experiments will be run with the tool to analyze the change in availability for individual components as well as complex systems. Next, this behaviour will be compared to that of other models, including the traditional approach of modeling a component according to an exponentially distributed failure and repair rate. The results and interpretations of these experiments are presented in Section 5. To find possible extensions to the computed properties, existing literature in the field of fault tree analysis will be explored. Any interesting properties that are found are then considered to determine their computability within our 2-state CTFD model. Extensions to the 2-state CTFD model will be considered as well. The results of this can be found in Section 4. In Section 6, the research findings will be summarized and analyzed. This will include an assessment of the case study's validation of the fixed repair delay assumption, the insights obtained through the tool's implementation and application, and potential extensions of the theorem. Finally, new avenues for future research and refinement of Theorem 1.2 will be proposed.

2 PRACTICAL APPLICABILITY

In order for a theorem to gain widespread acceptance within the research community, it is crucial to establish the practical relevance of the theorem. This section aims to identify relevant case studies from existing literature, where it is reasonable to assume that the repair or replacement of components can be accomplished within a fixed time. To locate such case studies, we made use of FFORT, a benchmark suite developed by the University of Twente, which includes a variety of fault tree diagrams and their original sources [15, 12]. We employed a selection procedure to make sure that the selected case studies adhered to three conditions.

2.1 Case study selection procedure

Firstly, the fault tree diagram corresponding to the case study should contain only basic fault tree diagram gates. These include AND, OR and K-of-M (voting) gates. AND gates in a fault tree diagram fail if and only if all children have failed. OR gates in a fault tree diagram fail if and only if one or more children have failed. K-of-M gates, also referred to as voting gates, fail when K out of N children have failed. In dynamic fault tree analysis, more complex gates exist, such as priority AND gates, sequence enforcers or functional dependency gates. These gates have in common that they enforce some conditional behaviour from their children, such as a certain order in which they must fail. This violates the precondition in Theorem 1.2,

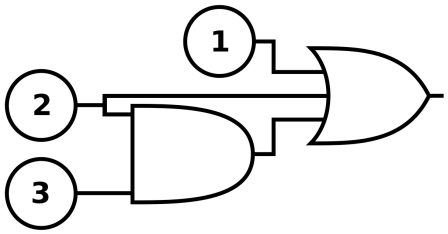

Fig. 3. Example of shared basic events.

Component	MTBF[h]	MTTR[min]	MTTR (modified)
CPU board	$1.35 * 10^5$	10	$10 * 10^6$
Bus	$2.25 * 10^5$	15	$15 * 10^6$
FPGA	$3.33 * 10^8$	15	$15 * 10^6$
Power supply	$5.5 * 10^4$	10	$10 * 10^6$
GSM-R card	$1.752 * 10^5$	10	$10 * 10^6$
WAR card	$4 * 10^5$	10	$10 * 10^6$

Table 1. Mean failure and repair times of components in Figure 4 [6].

which states that the theorem only holds for stochastically independent basic events. Due to the elaborate search functions of FFORT, it was an easy task to filter out the case studies that did not meet this condition.

The second condition was that each node in the fault tree diagram of the case study could only be the child of at most one parent. An example of a violation of this kind is given in Figure 3, where the basic event with label '2' has two parents and is thus 'shared' by its parents. The reason for this condition is once more to prevent dependencies between events. Since a fault tree diagram has one top level node of which all other nodes are descendants, when a node has multiple parents, those parents eventually converge. In most cases, this implies that they are not independent from each other, as both are dependent on the node with multiple parents. Although a dependency of this sort does not violate the precondition in Theorem 1.2, as all basic events are still stochastically independent from each other, it complicates the calculation of properties of the overall system. Budde and Stoelinga [2] proposed algorithms to effectively propagate properties through trees with shared subtrees. However, as this complexity does not contribute to the research objectives, it falls outside the scope of this study. A simple look over the visualized fault tree diagrams was enough to filter out any case studies which did not ensure the independence of all events.

The final step in the procedure was to read through all remaining case studies and find out if the assumption of fixed repair delay is reasonable to make. When this seemed to be the case, additional sources were sought that substantiated the assumption. Ultimately, the selection procedure culminated in the selection of a case study about a repairable critical computer system, namely the Radio Block Center (RBC).

2.2 Case study: Radio Block Center

The growing demand for performance, reliability, and safety in railways has led to the widespread use of computer-based railway control systems. Automatic train protection systems play a crucial role in supervising train speed by by ensuring compliance with speed profiles. They do so by calculating braking curves and initiating the braking process if necessary. The European Railway Traffic Management System / European Train Control System (ERTMS/ETCS) serves as the standard for modern European railway signaling and control systems [16]. ERTMS/ETCS encompasses three levels of increasing complexity and performance, which can be implemented individually or in combination. Pilot projects across Europe have been developed to test different levels of ERTMS/ETCS. For example, in Italy, ERTMS/ETCS level 2 is implemented in high-speed railways. This level relies on a continuous radio signaling system using GSM-Railway (GSM-R) for communication between the onboard system and the ground system. At level 2, the RBC plays a vital role in ensuring safe train spacing by managing information from the onboard subsystem and the Interlocking subsystem. The RBC computes and transmits movement authorities to trains via GSM-R based on the received positioning and track status information. The failure of an RBC is critical, as it could lead to a complete signaling system failure.

To allow for the testing of different repair policies on the availability of the RBC, a fault tree diagram was composed by researchers of the University of Napoli [6]. This diagram can be found in Figure 4. The dotted lines in the figure can be ignored, as they represent an 'off-line repair', which is a repair policy that is not relevant to this study. By examining data sheets of commercial devices and empirical observations, they also found reference values for the Mean Time Between Failure (MTBF) and the Mean Time To Repair (MTTR) for each individual component. These values can be found in Table 1. The modified values are for use in a later experiment and can be ignored for now.

2.3 Analysis

The RBC is a critical subsystem within the ERTMS/ETCS. The importance of the RBC in guaranteeing safe train spacing and managing movement authorities necessitates timely repairs in the event of failures or malfunctions. As such, it is subject to strict maintenance and repair protocols to ensure the continuous and reliable operation of the signaling system [3]. Due to this, repair time is mostly dependent on the accessibility of spare parts [5]. With well-established supply chains and inventory management systems, railway authorities can ensure the availability of spare parts needed for repairs. This allows for a streamlined repair process, affirming the assumption that the time required to replace faulty components is fixed. Obviously, unforeseen circumstances or exceptional cases may occur, resulting in deviations from a fixed repair delay. Nonetheless, the industry's focus on maintaining high levels of reliability and minimizing downtime makes it reasonable to assume fixed repair times. As the case study matches the three conditions of the selection procedure, it serves as a valid example to show the practical applicability of Theorem 1.2.

3 THEOREM IMPLEMENTATION

We developed a tool with the purpose of providing an example for future researchers and facilitating experimentation. The tool utilizes

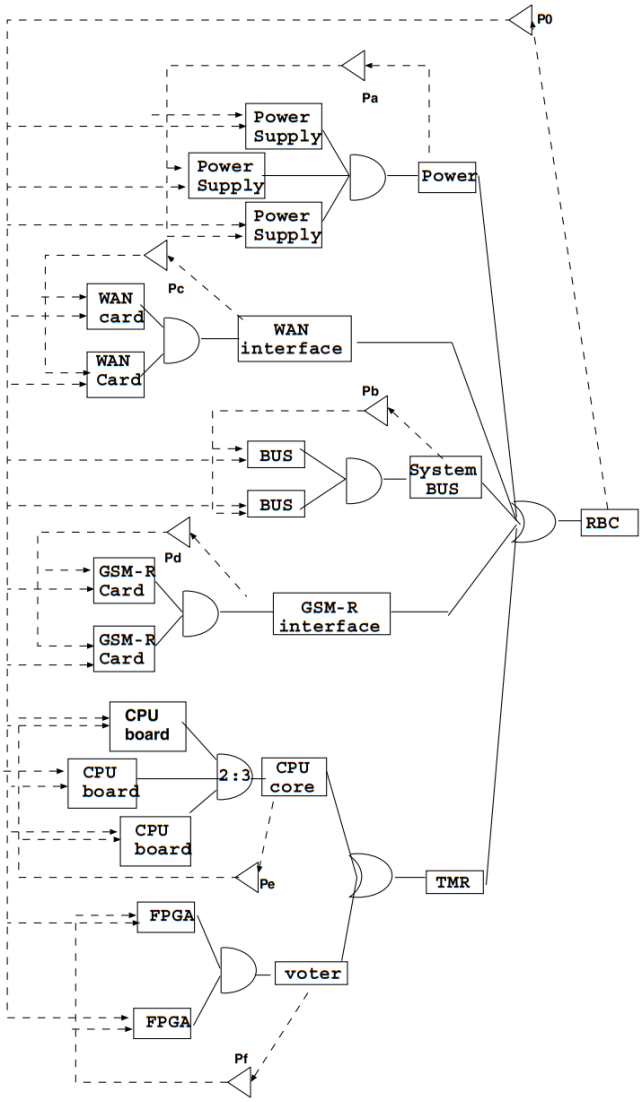

Fig. 4. RBC fault tree diagram [6].

Theorem 1.2 to compute the availability at a given time for nodes in a fault tree diagram whose basic events can be described by an exponentially distributed failure rate with a fixed repair delay, as depicted in Figure 2. Since the foremost consideration in building the tool was not to create the most efficient implementation, but to lay a foundation that could be easily understood and adapted by others, a high-level programming language was chosen for development. Python was selected for this purpose, as it has simple syntax with many well-maintained libraries that can be used for complex calculations. Furthermore, it was decided that the project would be containerized into a Docker image, so it could be run on any operating system without issues.

```
toplevel RBC;
RBC or Power WANinterface SystemBUS GSM-Rinterface TMR;
Power and PowerSupply1 PowerSupply2 PowerSupply3;
WANinterface and WANcard1 WANcard2;
SystemBUS and BUS1 BUS2;
GSM-Rinterface and GSM-RCard1 GSM-RCard2;
TMR or CPUcore voter;
CPUcore 2of3 CPUboard1 CPUboard2 CPUboard3;
voter and FPGA1 FPGA2;
BUS1 fixed failure=4.4444e-6 repair=0.25;
BUS2 fixed failure=4.4444e-6 repair=0.25;
FPGA1 fixed failure=3.003e-9 repair=0.25;
FPGA2 fixed failure=3.003e-9 repair=0.25;
PowerSupply1 fixed failure=1.8182e-5 repair=0.1667;
PowerSupply2 fixed failure=1.8182e-5 repair=0.1667;
PowerSupply3 fixed failure=1.8182e-5 repair=0.1667;
WANcard1 fixed failure=2.5e-6 repair=0.1667;
WANcard2 fixed failure=2.5e-6 repair=0.1667;
GSM-RCard1 fixed failure=5.7078e-6 repair=0.1667;
GSM-RCard2 fixed failure=5.7078e-6 repair=0.1667;
CPUboard1 fixed failure=7.4074e-6 repair=0.1667;
CPUboard2 fixed failure=7.4074e-6 repair=0.1667;
CPUboard3 fixed failure=7.4074e-6 repair=0.1667;
```

Listing 1. Input format example.

3.1 Input format

Functionally, the tool was build for two main purposes. Firstly, the calculation of the availability of a component in a fault tree diagram, at specific points in time. Secondly, the visualization of the behaviour of that same property over time in a graph. Both depend on the input of a fault tree diagram. For the input format of this diagram, a modified version of the Galileo textual input format was chosen [17]. The format was developed by researchers of the University of Virginia, to enable the analysis of dynamic fault trees. As explained before, dynamic fault trees can contain more complex gates which are not considered in this study. Therefore, the modified input format only allows for the inclusion of AND, OR and K-of-M gates. Moreover, all properties of basic events except for 'lambda' have been removed from the format. The remaining property describes the failure rate of a component and has been renamed to 'failure' to enhance readability. We introduced a new keyword positioned directly after the name of an individual component, which describes the model type the component adheres to. A component following the basic model from Figure 2 is characterized by the keyword 'fixed'. Finally, a new property called 'repair' has been introduced, which represents the fixed repair time of such a component. For an example of how the fault tree diagram from the case study, depicted in Figure 4 with values from Table 1, can be described by this input format, see Listing 1. The failure rate of each component is equal to the inverse of MTBF – MTTR [14]. The MTTR is already a fixed repair time, therefore it only needs to be converted from minutes to hours so both the failure rate and repair time are of the same time unit.

3.2 Availability computation

The availability of an individaul component can be calculated using Theorem 1.2. However, in order to compute the availability of the top level node, it is also necessary to propagate the availability through the intermediate logic gates. Since only systems with independent events are considered, the availability of gate nodes can be calculated

with a bottom-up approach [14]. For an AND gate, the availability is equal to the complement of the probability that all its children have failed. This approach is formalized in Theorem 3.1.

Theorem 3.1. *Let gate G be an AND gate with independent children $b_1..b_m$ and let $p_i(t)$ be the availability of child b_i. Then the availability $P_G(t)$ can be calculated using the following formula:*

$$P_G(t) = 1 - \prod_{i=1}^{m}(1 - p_i(t)).$$

The computation for an OR gate is even simpler. An OR gate is only available as long as all its children are working, since it fails as soon as one of its children does. The equation that follows can be found in Theorem 3.2.

Theorem 3.2. *Let gate G be an OR gate with independent children $b_1..b_m$ and let $p_i(t)$ be the availability of child b_i. Then the availability $P_G(t)$ can be calculated using the following formula:*

$$P_G(t) = \prod_{i=1}^{m} p_i(t).$$

The steps to calculate the availability of a K-of-M gate become a little more involved. The gate is available as long as any combination of less than K of its children have failed. Therefore, the availability of the gate can be calculated by summing up all probabilities of less than K children failing. This can result in very large expressions when K is large. To simplify the process, we can use polynomial multiplication. Firstly, a polynomial of degree one is assigned to each child. The coefficient of the first term is equal to the availability of that node, while the second coefficient takes the value of the failure probability. Then, when we compute the product of all polynomials, the coefficient of term i, is equal to the probability of i children failing. We formalize this in Theorem 3.3.

Theorem 3.3. *Let gate G be an K-of-M gate with independent children $b_1..b_M$ and let $p_i(t)$ be the availability of child b_i. Then let:*

$$\begin{aligned} f(x) &= \prod_{i=1}^{M} p_i(t) + (1 - p_i(t))x \\ &= \sum_{i=0}^{M} a_i x^i. \end{aligned}$$

Now define:

$$S = \{a_0, ..., a_{K-1}\}.$$

Then the availability $P_G(t)$ can be calculated using the following formula:

$$P_G(t) = \sum_{x \in S} x.$$

3.3 Operational details

The tool takes a path as a command-line argument. This argument refers to the file that contains a fault tree diagram expressed in the input format described above. The content of the file is then parsed and transformed into an in-memory representation of the fault tree diagram. To guarantee the submission of a valid fault tree diagram, multiple checks are made throughout this process. Helpful error messages assist the user in locating the problem when such a check fails. Besides a path to the input file, various other parameters can be included. A required argument specifies whether the tool should calculate and output specific values or visualize its results in a graph. Depending on which of these functions was chosen, the other arguments dictate the time points at which the values are computed or step size, destination path for the output and time interval over which the graph is plotted. Optionally, the name of a specific node can be supplied for which these calculations will be made. If not included, this value will default to the critical event. Alternatively, the tool is equipped with an interactive mode which can be enabled by setting a flag. This means that only the input file and function of the tool need to be specified up front. After the checks over the input file have succeeded, the command line utility will repeatedly prompt the user for the missing arguments. Each time, the output is printed or saved to an output file, after which the user can supply different arguments. Since the fault tree diagram does not have to be reloaded every time, this allows for an easy way to experiment with different parameters. Furthermore, in this mode, every result is cached to prevent redundant calculations, resulting in improved tool performance and speed.

4 EXTENSION EXPLORATION

The utilization of a fixed repair time extends beyond the computation of time-specific availability. To illuminate additional possibilities, we explore two potential directions for extensions to Theorem 1.2. Firstly, the computation of other properties over the 2-state CTFD model described in Figure 2. Secondly, the calculation of the availability for more complex CTFD models.

4.1 Property extension

Ruiters and Stoelinga [14] identify five relevant properties for conducting quantitative analysis of continuous-time fault trees. Among these is the availability, which we have adequately explored. We define the remaining properties and introduce equations for computing them over the 2-state CTFD model. Moreover, we present potential approaches derived from background literature for estimating their value in more complex systems.

4.1.1 Reliability.

Definition 4.1. The reliability of a system up to time t is equal to the probability that the system does not fail within time t.

For a single component, the computation of this property is trivial. Referring to Figure 2, we want to compute the probability that the component stays in S1. We simplify Theorem 1.2 by removing the summation and filling in $i = 0$ to achieve this.

Theorem 4.2. *Let component C be an independent basic event of a fault tree diagram with a failure rate λ and a fixed repair delay r. Then the reliability $P_C(t)$ can be calculated using the following formula:*

$$P_C(t) = \lambda t e^{-\lambda t}.$$

Since the component should remain operational for the entire duration, the repair time does not play a role. Computing the reliability of other nodes in a fault tree is more complex. Take for example an AND gate. One would have to compute the probability that all

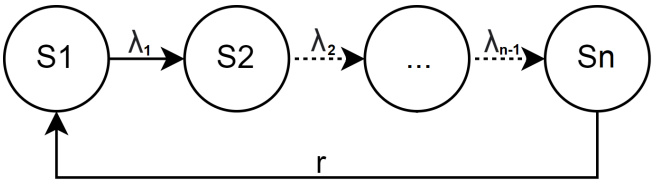

Fig. 5. CTFD with phase-type failure behaviour. $\lambda_1, \lambda_2, ..., \lambda_{n-1}$ represent exponentially distributed failure rates and r denotes the fixed repair time.

its children were in a failed state at the same time within time t. Durga Rao et al. [4] have shown that Monte Carlo simulations can achieve reasonable approximations for the reliability of a system with repairable components.

4.1.2 *Mean Time To Failure.*

Definition 4.3. The Mean Time To Failure is defined as the expected time it takes for a system to go from operational to failing.

A distinction can be made between the MTTF and the Mean Time To First Failure (MTTFF), which is the expected time it takes for a system to fail after it first becomes operational. For a single component with failure rate λ, both values are equal to $1/\lambda$. Amari and Akers [1] have shown that the MTTF and MTTFF of more complex systems can be approximated accurately and efficiently using the Vesely failure rate [18].

4.1.3 *Mean Time Between Failure.*

Definition 4.4. The Mean Time Between Failure (MTBF) is defined as the mean time between two consecutive failures.

The MTBF can be expressed as MTBF = MTTF + MTTR, where MTTR is the Mean Time To Repair. For individual components with failure rate λ and repair time r, the MTBF can therefore be calculated as follows: MTBF $= 1/\lambda + r$. The MTTR of complex systems can once again be approximated using the Vesley failure rate approach by Amari and Akers [1].

4.1.4 *Expected Number of Failures.*

Definition 4.5. The Expected Number of Failures (ENF) is defined as the expected number of failures of a system within time t.

The ENF can be calculated directly from the MTBF in the following way: ENF $= \frac{t}{\text{MTBF}}$.

4.2 Base model extension

There are many ways in which the 2-state CTFD model can be extended. For this first exploration, the focus lay on making it possible to describe more complex failure behaviours while retaining the assumption of a fixed repair delay. Specifically, a phase-type failure behaviour that could be split up into multiple states, where each transition to the following state is exponentially distributed. The final state ultimately represents the failure of the component, which can then be restored to the first working state with a fixed repair time. Such a CTFD model is depicted in Figure 5.

To compute the availability at time t of such a component, we take an approach similar to Theorem 1.2. Because of the fixed repair time r, there is a limit on the maximum number of failures that can have occurred in t, namely $\lfloor \frac{t}{r} \rfloor$. We can construct a finite CTMC with each state representing one of the working states in Figure 5 after a specific number of failures. STORM [8] is a probabilistic model checker, capable of taking such a CTMC and computing the probability of being in a specific state at a certain time. By calculating all probabilities of being in a working state after at most $\lfloor \frac{t}{r} \rfloor$ failures, and summing them up, the availability of the component at time t can be computed accurately. This approach was implemented in the tool by incorporating the Stormpy [19] library.

Component	Failure rate (λ)	Repair time (r)
C1	0.01	1.0
C2	0.1	1.0
C3	1.0	1.0
C4	10.0	1.0
C5	100.0	1.0

Table 2. Failure rates and repair times for individual components.

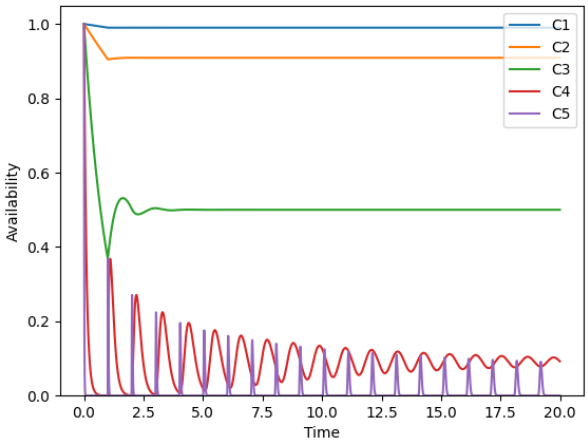

Fig. 6. Availability change over time for individual components that follow the basic 2-state CTFD model. The values for the failure rate and repair time of these components can be found in Table 2.

5 EXPERIMENTS

With a working tool at our disposal, some experiments can be run to gain insights into the behaviour of components which work in accordance with the model presented in Figure 2.

5.1 Individual components

Firstly, we will compare the change in availability over time for components with the same fixed repair times but variable failure rates. Through this comparative analysis, we can discern the impact of the relationship between these two properties on component availability. Specific values assigned to each component in the experiment are provided in Table 2. Inputting these values in the tool yields the graph depicted in Figure 6. Upon analyzing the results, several notable observations emerge.

We see that the availability of components does not exhibit a consistent pattern across the board. Nevertheless, certain similarities

```
# python main.py -i input/input.txt calc
Name of node to compute availability for (leave empty
     for top level node):

Compute availability at this/these time(s):
0 1 10 100 1000 10000
Availability is:
    *   1.0 at t=0.0
    *   0.9999999999976432 at t=1.0
    *   0.9999999999976432 at t=10.0
    *   0.9999999999976432 at t=100.0
    *   0.9999999999976432 at t=1000.0
    *   0.9999999999976432 at t=10000.0
```

Listing 2. Tool output of availability computation of RBC system displayed in Figure 4 at specific points in time. Original values from Table 1 are used for the individual components.

can be discerned. Primarily, the availability of all components initiates at a value of 1. Subsequently, it gradually declines in accordance with their respective failure rates until reaching the 1.0 time unit mark. This aligns logically with the fact that the availability ceases to drop once the repair time elapses, as repairing the component restores its functionality.

Another noteworthy similarity lies in the tendency of the availability to converge towards a specific value. Components with lower failure rates achieve this convergence shortly after surpassing the first repair time. Conversely, components characterized by higher failure rates display a wave-like pattern in their availability, gradually converging to the long-term value over time. The value at which the value ultimately stabilizes is called the long-term availability.

Definition 5.1. The long-term availability of a system is equal to the probability that the system is in a working state as time t approaches infinity.

Components with infrequent failure events relative to their repair time experience swifter convergence, as the repairs occur less frequently compared to their overall operational duration. Conversely, components with higher failure rates encounter more frequent failures, leading to a longer duration before their availability converges to the long-term value.

5.2 Complex systems

To see how this behaviour translates into a more complex system, we examine the case study on RBCs from Section 2.2. However, we know that RBCs are expected to be highly reliable systems, since their failure could cause massive disruptions to railroad traffic and even lead to large safety hazards. The values in Table 1 indeed show that the repair time of components is negligible relative to their failure rate, which predicts a high and constant availability. Calculating the availability of the system at a selection of specific moments in time, presented in Listing 2, quickly confirms this expectation. Although expected, it does not make for an interesting visualization. Therefore, we modified the original repair times to make them less negligible in comparison to the failure rates. These amended values can be also found in Table 1. The corresponding graph is pictured in Figure 7.

Noteworthy similarities can be observed when comparing the graph with Figure 6. Once again, we witness a decline in availability until reaching a specific point in time. This point aligns with

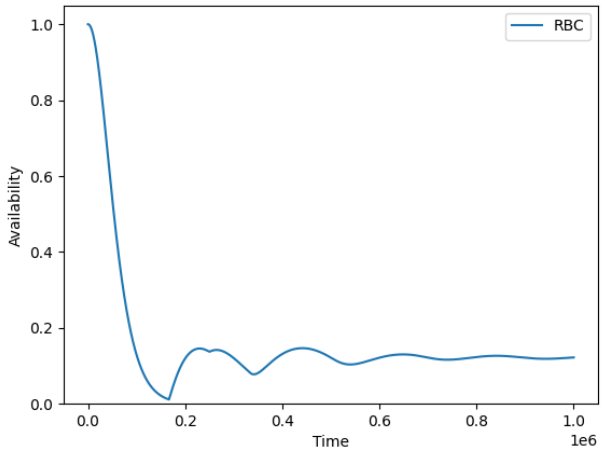

Fig. 7. Availability change over time for RBC system displayed in Figure 4. Modified values in Table 1 are used for the individual components.

the shortest repair time in Table 1. Additionally, a semblance of a wave-like pattern emerges, albeit with some deviations attributed to variations in component failure rates. Eventually, the availability converges to a specific value, mirroring the behaviour of individual components. It is important to note that the values presented are not realistic for this case study. Nonetheless, the visualization serves as an illustrative example, showcasing how the availability of less reliable systems could fluctuate over time, particularly in scenarios where the expected time to fail and the repair time of components are closely aligned.

5.3 Behaviour comparison

Having described the behaviour of components that adhere to the 2-state CTFD model, we can now draw a comparison to the behaviour observed in components governed by a different model. Specifically, we will examine models that consider repair rates instead of fixed repair times, as well as models featuring more intricate failure behaviour, as illustrated in Figure 5. Both these model types were added to the tool by making use of the Stormpy [19] library. To allow for an accurate comparison, the expected failure and repair times are kept the same for each model. The change in availability over time for these components can be found in Figure 8.

Analyzing the results reveals a distinct disparity when considering a fixed repair time versus a repair rate. It becomes evident that both C1 and C3 exhibit fluctuating availability, whereas C2 initially declines to the long-term availability and remains stable thereafter. Eventually, all components converge to this value, but it is worth noting that C3 takes the longest time to reach convergence and experiences larger fluctuations compared to C1. From this, we can conclude that describing a more complex failure behaviour when applicable has significance on the ability to predict the availability of a component. Moreover, the results show that considering a fixed repair time can have substantial impact on the accuracy of time-specific availability predictions as well.

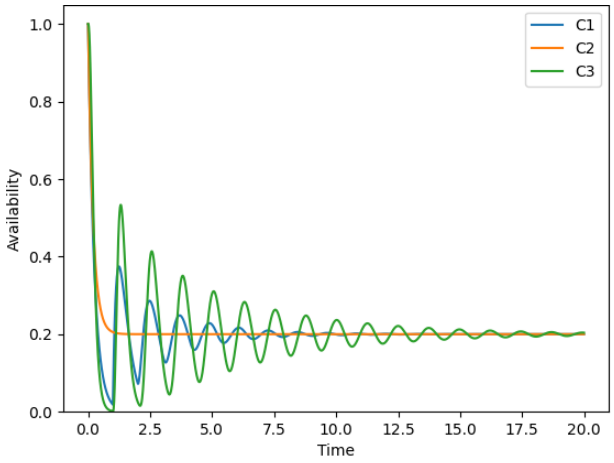

Fig. 8. Availability change over time for individual components following distinct models. C1 follows the basic 2-state CTFD model from Figure 2 with $\lambda = 4.0$ and $r = 1.0$. C2 Follows the same model as C1 but with a repair rate instead of a fixed repair time where $\lambda = 4.0$ and $r = 1.0$. C3 follows a 4-state variant of the model from Figure 5 with $\lambda_1 = 8.0$, $\lambda_2 = 16.0$, $\lambda_3 = 16.0$ and $r = 1.0$.

6 CONCLUSION

This research explored the practical applicability of a novel approach to compute the availability of components that can be described by an exponentially distributed failure time and a fixed repair delay. Furthermore, we sought to realize a foundation upon which future research into the topic could be made by creating a tool for experimentation and identifying several potential directions for expansions to the theorem. Finally, by using the tool to conduct experiments, insights were gained into the usefulness of the theorem.

To show the practical applicability of Theorem 1.2, an existing case study from the literature was analyzed, namely the Radio Block Center [6]. The failure rate of its individual components are stochastically independent. Additionally, supporting sources reinforce the assumption of a fixed repair delay. Still, the process is dependent on the availability of spare parts which could mean a significant deviation from the fixed repair time when spare parts are unavailable. However, assuming an exponentially distributed repair time does not offer a more accurate depiction of the situation. Considering the large consequences of a system failure, it is also reasonable to assume that spare parts are kept in adequate supply under normal conditions. Through this case study, we have effectively substantiated the practical relevance and applicability of Theorem 1.2 in real-world scenarios.

A tool was developed with the intention of serving as a fundamental framework upon which other researchers can build. The tool was specifically designed for experimentation purposes. Fault tree diagrams can be inputted according to a modified version of the Galileo textual input format [17]. Then, by utilizing Theorems 1.2, 3.1, 3.2 and 3.3, the availability of any node in the tree can be computed.

Two potential directions for the extension of Theorem 1.2 were explored. Firstly, the computation of other relevant properties for the quantitative analysis of continuous-time fault trees. The study presented formulas for calculating these properties for individual components, along with potential approaches for approximating them in more intricate systems. Secondly, we considered how to calculate the availability of CTFD models with a phase-type failure behaviour and a fixed repair time. This similarity to the 2-state CTFD model enables us to adopt a comparable approach to the one employed in developing Theorem 1.2. By making use of STORM [8], we are able to accurately calculate the availability of these types of models as well.

Through experimental analysis, we found that the availability of components characterized by the CTFD model in Figure 2 changes over time until it converges to the long-term availability. The speed of this convergence is contingent upon the relationship between the failure rate and the repair time of the component. Components with a low failure rate relative to their repair time converge faster then components with the opposite characteristic. The same behaviour is displayed by more complex systems. From these findings, it can be concluded that the analysis of time-specific availability holds greater significance for systems with lower reliability, as the convergence towards the long-term value occurs at a slower pace.

It was shown that this behaviour differs significantly from that of the traditional approach, where the repair time is considered to be exponentially distributed. In the traditional approach, availability quickly converges to its long-term value, rendering time-specific analysis useless. Another comparison was made with components with a fixed repair delay that follow a phase-type failure behaviour instead of a single failure rate. It was found that these components display more fluctuation in availability and take longer to converge to the long-term value. Consequently, we conclude that by accurately describing failure behaviour and employing appropriate repair time considerations, we can improve our ability to effectively estimate the time-specific availability of systems.

6.1 Future work

This novel topic offers numerous unexplored research directions worthy of investigation. An interesting approach could be to formalize the impact of the relationship between the failure rate and fixed repair time on the availability behaviour. With the foundation build, the road is now open for other researchers to extend the tool with additional features. A good start could be to implement the properties described in Section 4.1 in the tool, which would expand its capabilities to analyze complex system behaviours. Finally, more complex CTFD models can be investigated. For example, models with multiple states from which a repair can take place.

ACKNOWLEDGEMENTS

Thanks to Dr. E.M. Hahn for his feedback and continued supervision over the duration of this research project. Also, thanks to Dr. M. Volk for his help in getting started with the Stormpy library. Finally, thanks to Dr. M.A. Lopuhaä-Zwakenberg for his help in considering extensions to the basic theorem.

REFERENCES

[1] S.V. Amari and J.B. Akers. "Reliability analysis of large fault trees using the Vesely failure rate". In: *Proceedings of the Annual Reliability and Maintainability Symposium* (2004), pp. 391–396. ISSN: 0149144X. DOI: 10.1109/RAMS.2004.1285481.

[2] C. Budde and M. Stoelinga. *Efficient Algorithms for Quantitative Attack Tree Analysis*. May 2021.

[3] C. Darmenia. *Alstom ETCS Trackside Maintenance Manual*. 1st ed. Sydney: Sydney Trains, Mar. 2019.

[4] K. Durga Rao et al. "Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment". In: *Reliability Engineering & System Safety* 94.4 (Apr. 2009), pp. 872–883. ISSN: 0951-8320. DOI: 10.1016/J.RESS.2008.09.007.

[5] F. Flammini et al. "A Multiformalism Modular Approach to ERTMS/ETCS Failure Modelling". In: *International Journal of Reliability, Quality and Safety Engineering* 21 (June 2014), p. 1450001. DOI: 10.1142/S0218539314500016.

[6] F. Flammini et al. "Using repairable fault trees for the evaluation of design choices for critical repairable systems". In: *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)*. 2005, pp. 163–172. DOI: 10.1109/HASE.2005.26.

[7] B.E. Goldberg et al. *System engineering toolbox for design-oriented engineers*. Tech. rep. Alabama: National Aeronautics and Space Administration, Dec. 1994, pp. 35–3. URL: https://ntrs.nasa.gov/citations/19950012517.

[8] C. Hensel et al. "The probabilistic model checker Storm". In: *International Journal on Software Tools for Technology Transfer* 24.4 (2022), pp. 589–610. ISSN: 1433-2787. DOI: 10.1007/s10009-021-00633-z. URL: https://doi.org/10.1007/s10009-021-00633-z.

[9] H.G. Kang et al. "An overview of risk quantification issues for digitalized nuclear power plants using a static fault tree". In: *Nuclear Engineering and Technology* 41.6 (2009), pp. 849–858.

[10] P. Lacey. "An Application of Fault Tree Analysis to the Identification and Management of Risks in Government Funded Human Service Delivery". In: *2nd International Conference on Public Policy and Social Sciences*. Ed. by K. Singh and B. Singh. Kuching, Nov. 2011.

[11] Offnfopt. *File:fault tree.svg*. Oct. 2016. URL: https://commons.wikimedia.org/wiki/File:Fault_tree.svg.

[12] M. Peppelman. *FFORT Collection*. 2019.

[13] H. Ren, X. Chen, and Y. Chen. "Chapter 6 - Fault Tree Analysis for Composite Structural Damage". In: *Reliability Based Aircraft Maintenance Optimization and Applications*. Academic Press, 2017, pp. 115–131. ISBN: 978-0-12-812668-4. DOI: https://doi.org/10.1016/B978-0-12-812668-4.00006-X. URL: https://www.sciencedirect.com/science/article/pii/B978012812668400006X.

[14] E. Ruijters and M. Stoelinga. "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools". In: *Computer Science Review* 15-16 (2015), pp. 29–62. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2015.03.001. URL: https://www.sciencedirect.com/science/article/pii/S1574013715000027.

[15] E. Ruijters et al. "Ffort: A Benchmark Suite for Fault Tree Analysis". In: June 2019, pp. 878–885. DOI: 10.3850/978-981-11-2724-3{_}0641-cd.

[16] B. Stamm, E. Lepailleur, and A. Hougardy. *Responsibilities and rules for the assignment of values to ETCS variables*. Ed. by D. Degravre. 2.1.0. European Railway Agency, Nov. 2010.

[17] K. Sullivan and J.B. Dugan. *Galileo User's Manual & Design Overview*. 2.11-Alpha. University of Virginia, 1998.

[18] W.E. Vesely. "A time-dependent methodology for fault tree evaluation". In: *Nuclear Engineering and Design* 13.2 (1970), pp. 337–360. ISSN: 0029-5493. DOI: https://doi.org/10.1016/0029-5493(70)90167-6. URL: https://www.sciencedirect.com/science/article/pii/0029549370901676.

[19] M. Volk et al. *moves-rwth/stormpy: v1.8.0*. June 2023. DOI: 10.5281/zenodo.8025504. URL: https://doi.org/10.5281/zenodo.8025504.

[20] P. Wang. "Chapter 5 - Preliminary System Safety Assessment". In: *Civil Aircraft Electrical Power System Safety Assessment*. Ed. by P. Wang. Butterworth-Heinemann, 2017, pp. 101–156. ISBN: 978-0-08-100721-1. DOI: https://doi.org/10.1016/B978-0-08-100721-1.00005-4. URL: https://www.sciencedirect.com/science/article/pii/B9780081007211000054.