Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

DATA-DRIVEN FAULT TREE MODELING FOR RELIABILITY ASSESSMENT OF CYBER-PHYSICAL SYSTEMS

Sanja Lazarova-Molnar
Parisa Niloofar
Gabor Kevin Barta

Mærsk Mc-Kinney Møller Institute
University of Southern Denmark
Campusvej 55
Odense, 5230, DENMARK

ABSTRACT

Reliability of a system is a measure of the likelihood that the system will perform as expected for a predefined time period. Fault tree analysis, as a popular method for analyzing reliability of systems, has gained a widespread use in many areas, such as the automotive and aviation industry. Fault trees of systems are usually designed by domain experts, based on their knowledge. Since systems change their behaviors during their lifetimes, knowledge-driven fault trees might not accurately reflect true behaviors of systems. Thus, deriving fault trees from data would be a better alternative, especially for non-safety-critical systems, where the amounts of data on faults can be significant. We present an approach and a tool for Data-Driven Fault Tree Analysis (DDFTA) that extract fault trees from time series data of a system, and uses simulation to analyze the extracted fault trees to estimate reliability measures of systems.

1 INTRODUCTION

Reliability of a system is a measure of the likelihood that the system will perform as expected for a predefined time period. Reliability, however, can be a difficult metric to measure, and engineers have struggled with it throughout history. Most of the tragic accidents that have happened in the field of engineering were due to overestimating reliability of a system (Yuge et al. 2016). Fault tree analysis is a prominent method in analyzing safety and reliability of systems (Ruijters and Stoelinga 2015). It has been adopted as a standard reliability model by many industries such as automotive, aviation, and nuclear power (Lambert 2004; Griffor 2016; Haasl et al. 1981).

Fault tree analysis investigates which components influence failure of a system through modeling probabilistic causal chains of events that end in a global system failure. These causal relationships, which shape the graphical structure of fault trees, are usually constructed by domain experts, based on their experience and the information they have gathered on the topic (Berikov 2004). Therefore these fault trees are mostly knowledge-driven and not data-driven. Fault trees of complex systems with huge number of components are more prone to uncertainties and faults, which cannot be always accurately captured through experts experience.

Nowadays, using the power of computers and new technologies, it has become easier to gather, store and process large amounts of data. Hence, researchers are able to use data-driven methods to extract information about the status of a system under study. Extracting a model from data is known as data-driven modeling (Lazarova-Molnar and Li 2019). Data-driven modeling approaches are gaining attraction in many areas for their ability to analyze data from system in order to derive behaviours of systems (Solomatine

et al. 2008; Kim et al. 2017). Consequently, using data sets and data streams, stemming from fault-related behaviours of a system, could be a fundamental step in deriving fault trees of systems.

The goal of this paper is to introduce a method for Data-Driven Fault Tree Analysis (DDFTA), and a tool that implements it. The tool extracts and analyzes fault trees from time series data of a system to estimate the system's reliability measures. Deriving reliability models from data can be more insightful than utilizing solely expert knowledge, as it captures the true behavior of a system, after it has been put into use. The data-driven models can, however, very well be supplemented by expert knowledge for validation and approval of extracted models.

2 FAULT TREE ANALYSIS: BACKGROUND AND RELATED WORK

Fault tree analysis (FTA) works by modeling, both logically and graphically, the paths to failure for a system or component (Volkanovski et al. 2009). Thus, a fault tree is a directed acyclic graph (DAG) whose leaves are the basic events (typically basic faults), and the root represents the top event, which is typically a system failure. The gates in a fault tree represent the propagation of failure through the tree (Ruijters and Stoelinga 2015).

Repairable fault trees consider both faults and repairs in a system. Hence, for each basic event that is typically associated with a fault, there is a probability distribution that describes the fault's occurrences, and potentially a probability distribution of the times to repair. Correspondingly, the relevant fault-related time series data of a system consists of a sequence of status change times for each basic event and the system state.

There are two essential analysis techniques for fault trees, qualitative analysis and quantitative analysis (Geymayr and Ebecken 1995). Qualitative analysis considers the structure of the fault tree, while the quantitative analysis computes failure probabilities, reliability, etc. of the fault tree. The first step towards computing reliability of a system is to extract the structure of the system's underlying fault tree. Once the structure of the fault tree is discovered, using the probability distribution functions of the basic events, we can calculate the reliability of the system, the likelihood of a top event occurrence, as well as those of the basic events that have caused the occurrence of the top event. Lee et al. wrote a survey paper explaining different methods of fault tree analysis (Lee et al. 1985). They mention techniques of automated fault tree construction such as using Synthetic Tree Model (STM), Computer Automated Tree (CAT), which uses decision tables, and automatic fault-tree synthesis from the reliability graphs.

Until recently, the only source to build structures of fault trees was experts' knowledge. Considering the complexity of nowadays systems, adaptability of these knowledge-based fault trees can be questionable (Linard et al. 2019). Furthermore, systems often evolve during their lifetime, exhibiting behaviors that are, even, unpredictable during design time. Thus, there have been some notable efforts that attempt to utilize data, detailed as follows.

As one of the earlier attempts, Mukherjee and Chakraborty (2007) describe a technique to automatically generate fault trees using historical maintenance data in text form. Their technique relies on domain knowledge and linguistic analysis. The above-mentioned techniques also use data to construct a fault tree, however they do not use time series data, but rather other types of data, such as text and reliability graphs. More recently, Nauta et al. (2018) introduced LIFT to learn the structure of static fault trees from untimed data bases with Boolean event variables. The relationships between events in a fault tree are causal, so they applied Mantel-Haenszel statistical test to infer the most likely causal relationships among the events. Linard et al. (2019) applied an evolutionary algorithm to learn fault trees from untimed Boolean basic event variables. Instead of the independence test in the LIFT algorithm, they used a score based algorithm to extract fault trees. Also their data bases did not need any information about intermediate events. Before the work of Nauta et al. (2018) and Linard et al. (2019), observational data were used to generate FTs with the IFT algorithm (Nolan et al. 1994) based on standard decision-tree statistical learning. Our method is different from those we discovered in the literature since we consider time series data of a system's fault-related events, and provide a complete solution to calculate reliability measures from observational

data. A challenge in working with time series data is that in case of a system with rare events (as faults typically are), we need to monitor the system for a long time, and collect a lot of observational data, in order to make sure that we have information about almost all factors leading to the system failure. This challenge can be overcome through collaborative data analytics (Lazarova-Molnar et al. 2018; Lazarova-Molnar et al. 2019).

3 DATA-DRIVEN RELIABILITY ANALYSIS

In this section, we describe the methods and techniques that we developed to enable the data-driven reliability modeling and analysis. The process workflow of the proposed approach is shown in Figure 1, where we can see that our approach depends on the availability of time series data on faults' occurrences and the time it takes for them to be repaired. Furthermore, Algorithm 1, describes the pseudocode of our method.

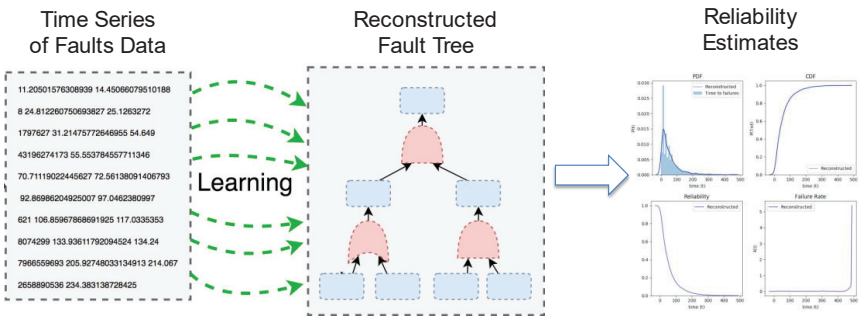

Figure 1: Process workflow for the data-driven reliability analysis.

The DDFTA (Algorithm 1) comprises of two main steps: 1) structure learning part which is also known as qualitative analysis and 2) quantitative analysis based on the output of the first step. In the structure learning step, we extract the minimal cut sets from the time series data set, and then use Boolean algebra to build a fault tree that aims to be mathematically identical to the true fault tree of the system. Although there are other methods (Nauta et al. 2018; Linard et al. 2019) to learn the structure of a fault tree, for building non-dynamic repairable fault trees, Boolean algebra is still an efficient method. Comparing the performance of different algorithms in modelling more complex systems can be proposed for future research. In the second step, the distribution functions of the basic events are approximated from the time series data. These reliability and repair distribution functions of components associated with basic events, along with the fault tree structure, are inputs to the proxel-based simulation, which is then used to calculate system's reliability measures, in form of transient complete solutions. The proxel-based simulation is well-known for its ability to cope with stiff models, as fault models are typically (Lazarova-Molnar and Horton 2003; Lazarova-Molnar 2005). Namely, the proxel-based simulation has proved more accurate and less sensitive to rare events than Monte Carlo, and also efficient once the improvement of extrapolation of solutions has been implemented.

Quantitative reliability analysis can be performed both analytically and through simulation. The analytically computed reliability distribution shows only the probability that the system is operating correctly before the first failure. It does not consider if the components were to be repaired. Therefore, this analytically computed reliability is different from the simulation-based computed reliability of the system. An example result of analytically computed reliability assessment for the top event in a fault tree is shown in Figure 2, where we have the PDF (probability density function), CDF (cumulative distribution function), Reliability and Failure Rate for the top event. In contrast to the analytically computed reliability, the simulation-based computed reliability shows the reliability of the top event when failures and repairs are taken into account.

Algorithm 1 Data-Driven Fault Tree Analysis: DDFTA

```
1: procedure STRUCTURE(timeseries dataset)
2:     binary dataset ← Binary(timeseries dataset)          ▷ Converts timeseries into binary data set
3:     MC ← MCS(binary dataset)                             ▷ Extract the minimal cut sets
4:     FTBooleanEquation ← BooleanAlgebra(MC)   ▷ Shows the fault tree in terms of boolean equation by simplifying the MCS using Boolean algebra
5:     FaultTree ← Graph(FTBooleanEquation)     ▷ returns the graphical structure of the Boolean equation
6:     return FaultTree
7: procedure MCS(binary dataset)
8:     data ← binary dataset
9:     t ← number of rows(data)
10:    CutSet ← 0
11:    for i ← 1 to t do                                    ▷ Identifying the cut sets
12:        if data[i, TopEvent]=1 then
13:            BESet ← indices of Basic events with state 1 in the ith row
14:        CutSet ← CutSet ∪ BESet                          ▷ Set of all cut sets
15:    CutSetUnique ← Remove duplicate sets(CutSet)
16:    c ← number of sets(CutSetUnique)
17:    for i ← 1 to c do                                    ▷ Refining the cut sets into MCS
18:        if ∃ C ∈ CutSetUnique s.t. C ⊂ Ci then
19:            remove Ci from CutSetUnique
20:    return CutSetUnique
21: procedure RELIABILITY(FaultTree, timeseries dataset)    ▷ Calculates the reliability measures given a fault tree structure
22:    N ← Number of Basic events
23:    for k ← 1 to N do
24:        RelDist_k ← DistributionFitt(time-to-failure for BasicEvent_k)
25:        MainDist_k ← DistributionFitt(time-to-repair for BasicEvent_k)
26:        MTTF_k ← average(time-to-failure for BasicEvent_k)
27:        MTTR_k ← average(time-to-repair for BasicEvent_k)
28:        OpeAva_k ← Operational availability(BasicEvent_k)
29:        InsAva_k ← Proxel(FaultTree, BasicEvent_k)
30:    RelDist_Top ← DistributionFitt(time-to-failure for TopEvent)
31:    MainDist_Top ← DistributionFitt(time-to-repair for TopEvent)
32:    MTTF_Top ← average(time-to-failure for TopEvent)
33:    MTTR_Top ← average(time-to-repair for TopEvent)
34:    OpeAva_Top ← Operational availability(TopEvent)
35:    InsAva_System ← Proxel(FaultTree, TopEvent)
36:    return RelDist, MainDist, MTTF, MTTR, OpeAva, InsAva
```

DDFTA, described by Algorithm 1, consists of three functions: *STRUCTURE* (lines 1-6) returns the skeleton of the fault tree, *MCS* (lines 7-20) which extracts the minimal cut sets is a sub function here, *RELIABILITY* function (lines 21-36) applies the output of the *STRUCTURE* function and returns the reliability measures of the system.

To illustrate DDFTA, let us consider the fault tree configuration shown in Figure 3 as the ground truth. We use abbreviations BE*n* to denote Basic Event *n*. Time series are generated from this fault tree with the reliability and maintainability distributions shown in Table 1. An example of the generated time series data is shown in Figure 4.

Table 1: Reliability and maintainability distribution functions of the basic events in Figure 3.

Distribution	BE1	BE2	BE3	BE4	BE5
Reliability	N(4 ,1)	Exp(0.333)	N(4 ,1)	Exp(0.333)	LogN(4 ,1)
Maintainability	Exp(0.5)	N(4 ,1)	LogN(4 ,1)	Weibull(5,2)	Exp(0.5)

The first elements in the time series of each event show the time of failures for the component associated with that event. For example, the basic event BE1 first occurs at time 4.5911. The second element illustrates the repair time for the component modeled by that event. It means that the component modeled by basic event BE1 is back in operation at time 5.0549. Again, the third element represents a failure and the fourth element, the corresponding repair time, and so on.

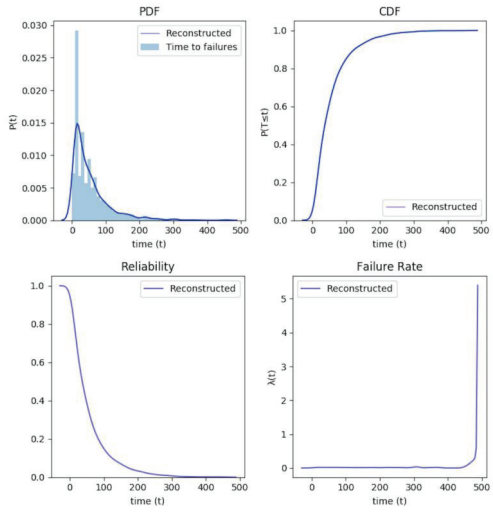

Figure 2: Example result of an analytically calculated reliability of a top event.

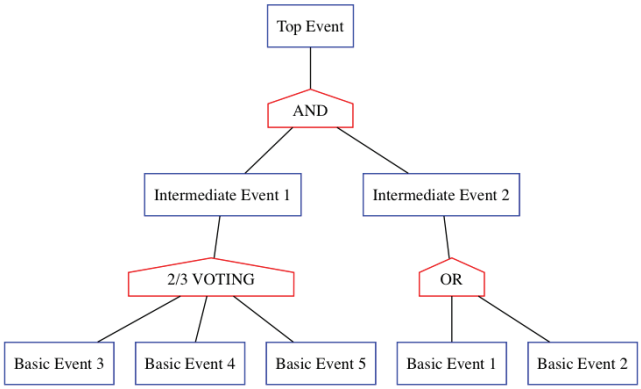

Figure 3: Ground truth fault tree.

The structure of the fault tree is learnt from the time series data, through discovering the combinations of basic events that lead to system failure (details are discussed in section 3.1). Once the structure is learnt, the probability distributions and reliability metrics can then be estimated from the data and the reconstructed fault tree. An example solution for the above-described scenario is shown in Figure 5. In the plot, availability changes over time and if we compare it with the operational availability, the instantaneous availability is firstly oscillating, but it also approaches the operational availability.

3.1 Qualitative Analysis: Learning the Structure of a Fault tree

In our approach, the fault tree structure is determined solely based on time series data, using basics of qualitative analysis of fault trees. Specifically, cut sets and minimal cut sets (MCS) are used in determining the structure (Lee et al. 1985). A cut set in a fault tree is a set of basic events whose (simultaneous) occurrence ensures that the top event occurs. A minimal cut set is a cut set which has no subset that is a cut set. This, essentially, means the minimal cut set cannot be reduced to a smaller set which will cause a system failure. In Table 2, cut sets and minimal cut sets of the fault tree of Figure 3 are listed. Minimal cut sets, extracted from the time series data, provide information about what basic events have caused a system failure, and from this we can deduce the gate connections between the basic events.

```
Top Event : [3.361903105203138, 4.1152506529116755, 8.43170959056411, 8.891932106379965]
└── AND
    ├── Intermediate Event 1 : [3.361903105203138, 4.220513102203445, 8.43170959056411, 8.891932106379965]
    │   └── 2/3 VOTING
    │       ├── Basic Event 3 : [3.361903105203138, 4.220513102203445, 5.755724149456178, 8.891932106379965]
    │       ├── Basic Event 4 : [1.6238068160608952, 5.356367682992914, 8.43170959056411, 14.025100465736877]
    │       └── Basic Event 5 : [12.502588706108872, 13.954767453010827, 28.40473670037755, 28.522610613059907]
    └── Intermediate Event 2 : [0.6981863673738924, 4.1152506529116755, 4.591100866343563, 5.054954198424569]
        └── OR
            ├── Basic Event 1 : [4.591100866343563, 5.054954198424569, 8.287684109731572, 8.773946295576877]
            └── Basic Event 2 : [0.6981863673738924, 4.1152506529116755, 7.676586464829812, 11.580304958793104]
```

Figure 4: An example of the generated time series data from the ground truth fault tree of Figure 3.

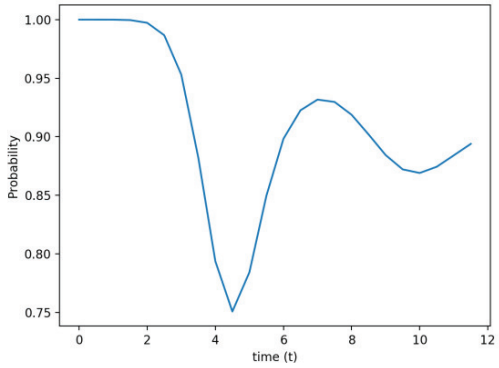

Figure 5: System availability, calculated using the proxel-based simulation method for the fault tree of Figure 3.

In order to obtain the minimal cut sets, we need to identify the cut sets of the fault tree by examining the time series of the top event (Lines 11-15 in Algorithm 1). At each time-of-failure of the top event we determine which combination of the basic events (cut sets) have caused the failure, or in other words, which basic components are in a failed/faulty state. Then, we refine the cut sets into minimal cut sets (Lines 17-19). In the following, we illustrate the fault tree structure learning algorithm using time series data and the techniques from Boolean algebra.

Let us consider the time series data in Figure 4. The original time series data can be converted into a data set with binary entries for basic events and the top event (see Table 3). Each event is given a state of 0 or 1, representing if the component associated with the event is operational or not, respectively. For example, we observe that the component associated with basic event BE3 is in a faulty state from time 3.3619 to 4.2205, when it begins operating till its broken again at time 5.7557 to 8.8919.

Table 2: Sets of cut sets and minimal cut sets for the fault tree of Figure 3.

Cut sets	Minimal cut sets
{BE3, BE4, BE1}	{BE3, BE4, BE1}
{BE3, BE4, BE2}	{BE3, BE4, BE2}
{BE3, BE4, BE1, BE2}	
{BE3, BE5, BE1}	{BE3, BE5, BE1}
{BE3, BE5, BE2}	{BE3, BE5, BE2}
{BE3, BE5, BE1, BE2}	
{BE4, BE5, BE1}	{BE4, BE5, BE1}
{BE4, BE5, BE2}	{BE4, BE5, BE2}
{BE4, BE5, BE1, BE2}	
{BE3, BE4, BE5, BE1, BE2}	

Table 3: Time series of Figure 4 converted into a binary data set.

Time	BE1	BE2	BE3	BE4	BE5	TopEvent
0.6982	0	1	0	0	0	0
1.6238	0	1	0	1	0	0
3.3619	0	1	1	1	0	1
4.1153	0	0	1	1	0	0
4.2205	0	0	0	1	0	0
4.5911	1	0	0	1	0	0
5.0550	0	0	0	1	0	0
5.7557	0	0	1	0	0	0
7.7566	0	1	1	0	0	0
8.2877	1	1	1	0	0	0
8.4317	1	1	1	1	0	1
8.7740	0	1	1	1	0	1
8.8919	0	1	0	1	0	0

Considering the system failure times in Table 3, or the rows in which the top event is in a failure state (has the label 1), we extract the cut sets. At times 3.3619 and 8.7740, components associated with basic events BE2, BE3 and BE4 are all failed at the same time, and also the system fails. Obviously there are more cut sets, like $\{BE1, BE2, BE3, BE4\}$ at time 8.4317, leading to system failure, that are not shown in our illustrative data set example here. If the time series data set is a representative and large enough sample of the true system under study, then the minimal cut sets represented in the first column of Table 2 can be extracted from the data.

Now we use laws of Boolean algebra (see Table 4) on the extracted minimal cut sets, to represent the reconstructed fault tree as a Boolean equation. Symbol "+" is used to represent the logical OR operator and the symbol "." is used to represent the logical AND operator (One can refer to Haasl et al. (1981) for a more comprehensive rule tabulation and discussion of Boolean algebra).

Table 4: Some selected rules of Boolean algebra.

Rule	Mathematical form
Commutative rule	A.B=B.A
	A+B=B+A
Associative rule	A.(B.C)=(A.B).C
	A+(B+C)=(A+B)+C
Distributive rule	A.(B+C)=A.B+A.C
	A+(B.C)=(A+B).(A+C)
Idempotent rule	A.A=A
	A+A=A
Absorption rule	A.(A+B)=A
	A+A.B=A

Table 5 illustrates the steps in building a fault tree in terms of Boolean equations from the minimal cut sets. Step 1, begins by putting an OR gate between the minimal cut sets. In step 2 the distributive law is applied.

Table 5: Reconstructing a fault tree based on minimal cut sets of Table 2.

Step	Boolean representation
1	T=(BE1.BE3.BE4)+(BE2.BE3.BE4)+(BE1.BE3.BE5)
	+(BE2.BE3.BE5)+(BE1.BE4.BE5)+(BE2.BE4.BE5)
2	T=BE1.(BE3.BE4+BE3.BE5+BE4.BE5)
	+BE2.(BE3.BE4+BE3.BE5+BE4.BE5)
3	T=(BE1+BE2).(BE3.BE4+BE3.BE5+BE4.BE5)
4	T=IE1.IE2

Step 3, indicates that there is an OR gate between basic events BE1 and BE2. Also $(BE3.BE4 + BE3.BE5 + BE4.BE5)$ means that there is a 2/3 voting gate between basic events BE3, BE4 and BE5. Finally if take $(BE1 + BE2)$ as intermediate event IE1, and $(BE3.BE4 + BE3.BE5 + BE4.BE5)$ as intermediate event IE2, these intermediate events are connected by an AND gate. So the fault tree in step 4, which is constructed from the minimal cut sets of Table 2, is identical to the ground truth fault tree in Figure 3.

There is not one *correct* fault tree for a problem but many correct forms which are equivalent to one another. The rules of Boolean algebra can thus be applied to restructure the tree to a simpler, equivalent form for ease of understanding or for simplifying the evaluation of the tree (Haasl et al. 1981).

3.2 Quantitative Analysis: Computing the Reliability of the System

Once the structure of the fault tree is extracted from data, we use it to obtain reliability metrics such as mean time to failure (MTTF), mean time to repair (MTTR), and operational availability of the system. The first step to the quantitative analysis is to estimate reliability and maintainability probability distribution functions of the basic events based on the time series data.

The average of the times-to-failure yields the MTTF and, correspondingly, the average of the times-to-repair data samples yields the MTTR. Furthermore, to also consider non-exponential events, probability distribution fitting techniques, featuring parametric or regression methods (Oosterbaan 2019), are used to determine the reliability and maintainability probability distribution for each basic event from the sample data sets of times-to-failure and times-to-repair, respectively. Since reliability is complementary to probability of failure, reliability of each basic event can also be estimated. Reliability of the whole system is then obtained using probability algebra. For example, for two basic events BE1 and BE2, if they are connected by an AND gate then: $R(t) = R_1(t).R_2(t)$, and if they are connected with an OR gate: $R(t) = 1-(1-R_1(t)).(1-R_2(t))$.

The proxel-based simulation is used to determine the instantaneous availability of basic components and the entire system. After reconstruction of the fault tree, the reliability and maintainability distribution of each basic component are used to calculate the availability with respect to time. We then propagate the availability of each individual component through the fault tree to calculate the availability of the system. The availability is the probability that the component or system is operational.

4 SIMULATION EXPERIMENT

In this section we illustrate our method by a simulation study of a repairable fault tree. The original repairable fault tree of Figure 6(a) with eight basic events and five gates (three AND, an OR and a 2/3 voting gates) is used to generate 10000 failure and repair times from the reliability distribution function of Exp(0.1) and maintainability distribution function of Exp(0.5) for all basic events. All the minimal cut sets were extracted from the binary version of the time series data set and the final reconstructed fault tree is shown in Figure 6(b). Using this reconstructed fault tree and the methods of section 3.2, we estimate the reliability and maintainability distributions and obtain the reliability measures of the top event as well as those of the basic events. Figure 7 illustrates the results for the basic event BE1 and Figure 8 shows the top event's results.

5 SUMMARY AND OUTLOOK

Fault trees and other reliability models enable describing fault behaviors of systems and evaluating how reliable they are. Reliability is one of the most important underlying quality measures for a system's performance. Accurate prediction of reliability of an operational system would help schedule maintenance and replace unreliable components in a timely manner. We developed a method to construct a fault tree from time series data from faults of an arbitrary system. A screenshot of the interface of the developed tool is shown in Figure 9.

To test our approach, we constructed a framework to design fault trees that could generate time series data representing time-of-failures and time-of-repairs for both basic components and the entire system.

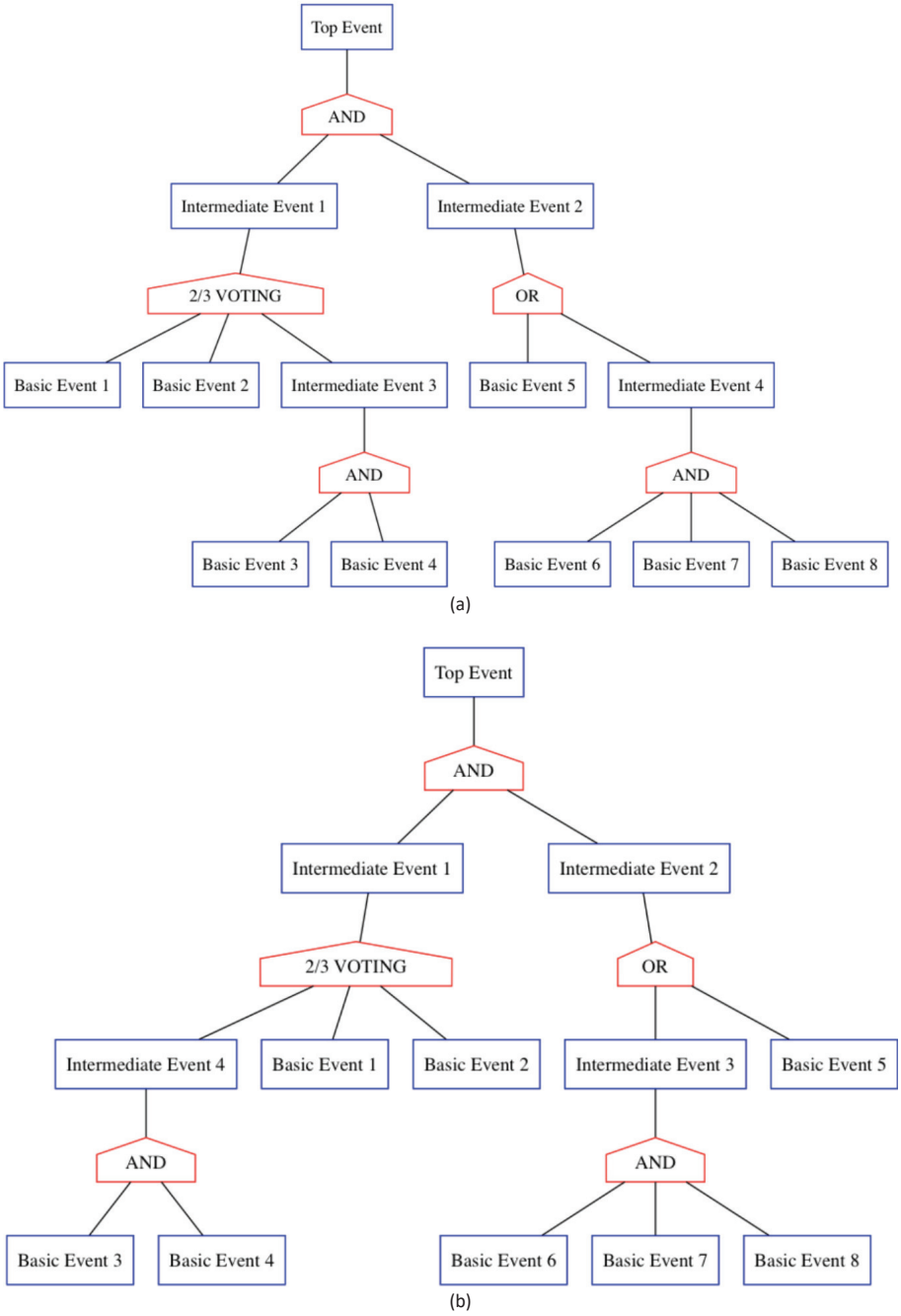

Figure 6: (a) The original fault tree, (b) Reconstructed fault tree from a time series data fabricated from the original fault tree.

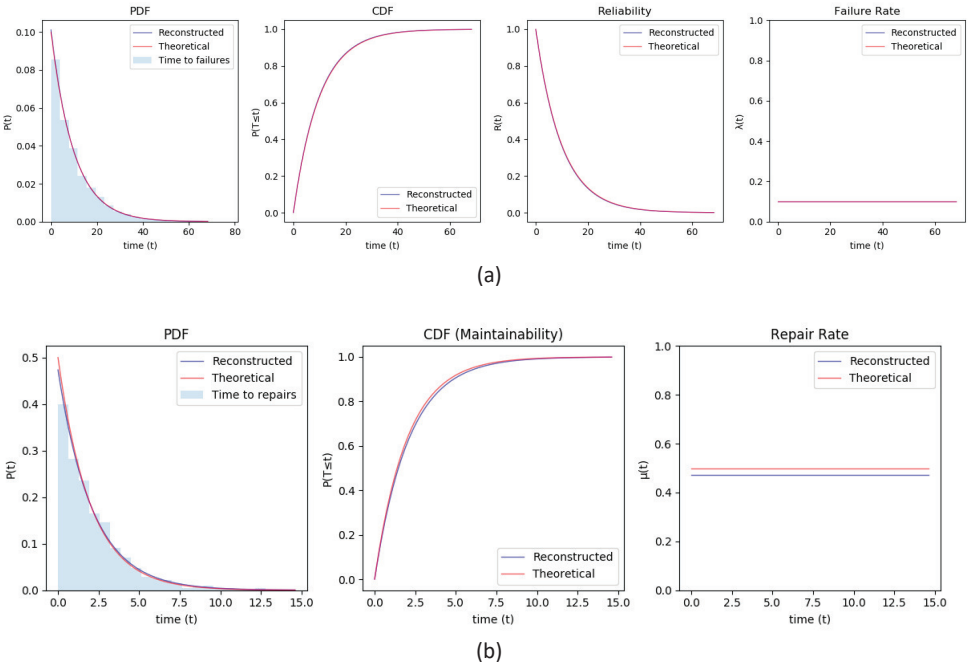

Figure 7: Results of the reliability analysis for the basic event BE1 of the fault tree in Figure 6. Reconstructed lines are estimated from time series data and the theoretical lines are the true distributions that the time series data are generated from (a) Reliability distribution analysis, (b) Maintainability distribution analysis.

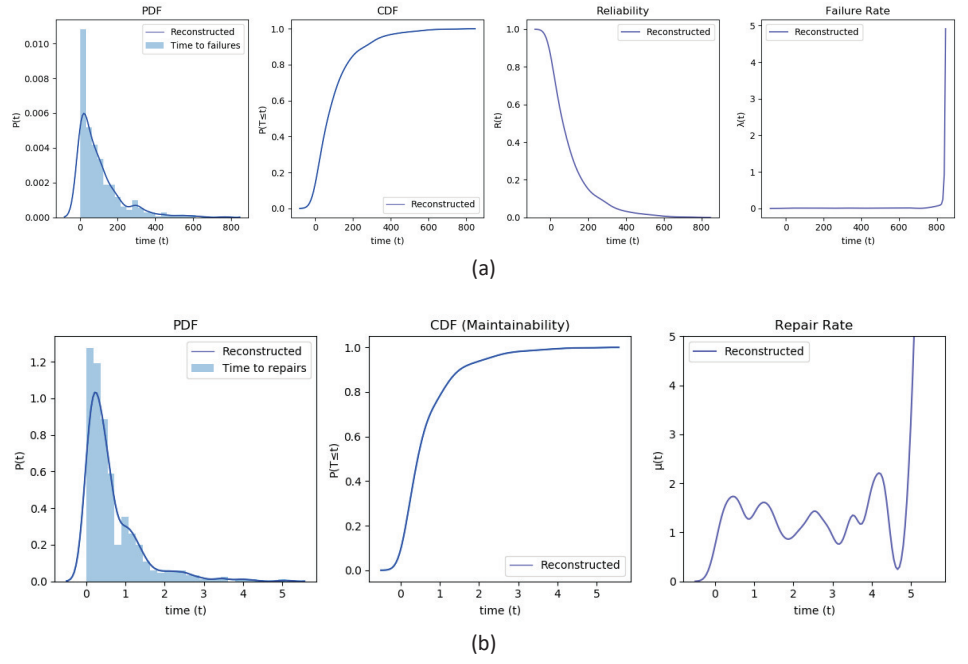

Figure 8: Results of the reliability analysis for the top event of the fault tree in Figure 6 (a) Reliability distribution analysis, (b) Maintainability distribution analysis.

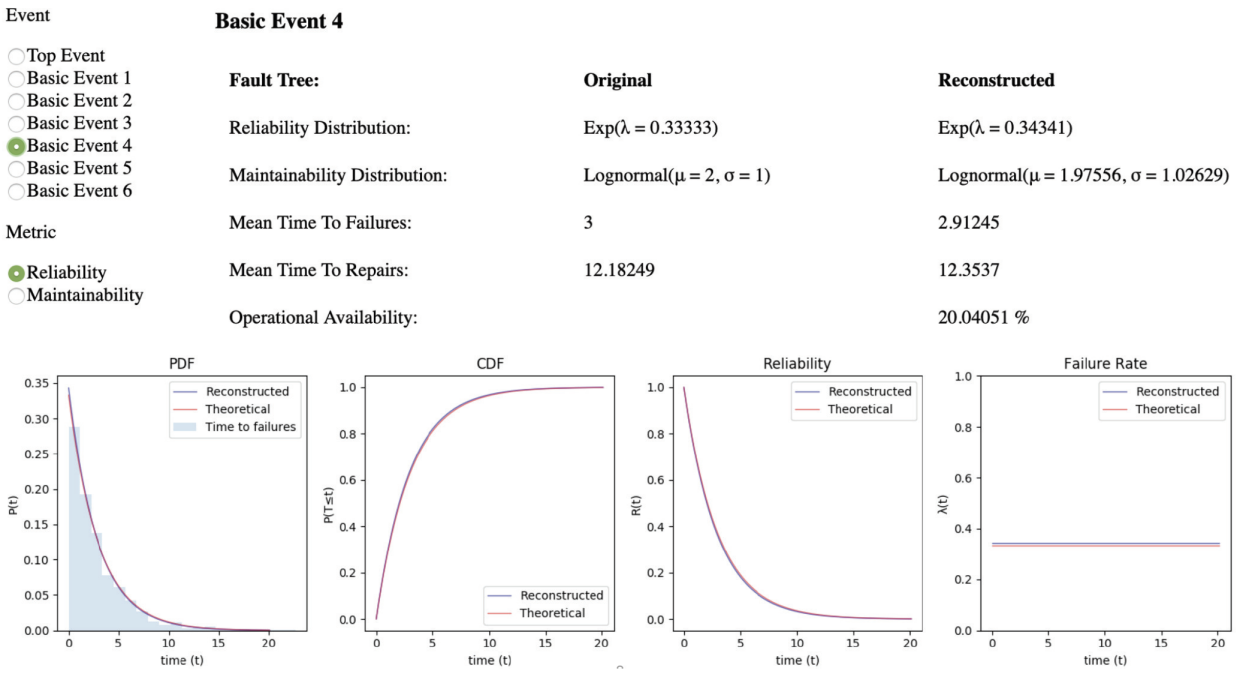

Figure 9: Part of the Graphical User Interface of the developed tool.

Using the time series data we conducted qualitative and quantitative analysis techniques to reconstruct the fault tree. We evaluated the quality of the reconstruction by comparing the original fault tree with the reconstructed fault tree. We compared both structural aspects of the fault trees and the reliability and maintainability distributions of the individual components. Based on these, we determine reliability and maintainability of the system. Using the proxel-based method, we also determine the instantaneous availability of the components and the system. Our algorithm is efficient in terms of computation time in constructing the structure of the fault tree. However the proxel-based simulation in the quantitative analysis, is more computationally expensive in proportion to the number of basic events. The entire fault tree reconstruction process is a data-driven modeling technique that models the behavior of a system with a fault tree. The algorithm constructs a white-box model, since the fault tree allows us to understand how the individual basic events cause the top event to occur. By using this method, the fault-related behavior of a system can be determined by analyzing time series data. The proposed method can work very well for non-safety-critical systems, where faults are more common occurrence and do not have associated catastrophic consequences. A large portion of manufacturing systems belongs to this category, and therefore, we see a great potential for its application within the Industry 4.0 development (Lazarova-Molnar and Mohamed 2019). This also forms one of the focuses for our future work.

REFERENCES

Berikov, V. 2004. "Fault Tree Construction on the Basis of Multivariate Time Series Analysis". In *Proceedings of the* 8^{th} *Russian-Korean International Symposium on Science and Technology*. June 26^{th}- July 3^{rd}, Tomsk, Russia, 103–106.

Geymayr, J. A. B., and N. F. F. Ebecken. 1995. "Fault-tree Analysis: A Knowledge-engineering Approach". *IEEE Transactions on Reliability* 44(1):37–45.

Griffor, E. 2016. *Handbook of System Safety and Security: Cyber Risk and Risk Management, Cyber Security, Threat Analysis, Functional Safety, Software Systems, and Cyber Physical Systems*. 1st ed. Massachusetts: Syngress Publishing.

Haasl, D. F., N. H. Roberts, W. E. Vesely, and F. F. Goldberg. 1981. "Fault Tree Handbook". Technical Report No. 82-003645, U.S. Nuclear Regulatory Commission, Washington.

Kim, B. S., B. G. Kang, S. H. Choi, and T. G. Kim. 2017. "Data Modeling Versus Simulation Modeling in the Big Data Era: Case Study of a Greenhouse Control System". *Simulation* 93(7):579–594.

Lambert, H. E. 2004. "Use of Fault Tree Analysis for Automotive Reliability and Safety Analysis". *SAE transactions* 113:690–696.

Lazarova-Molnar, S. 2005. *The Proxel-Based Method: Formalisation, Analysis and Applications*. Ph. D. thesis, Otto-von-Guericke University, Magdeburg.

Lazarova-Molnar, S., and G. Horton. 2003. "Proxel-based Simulation of Stochastic Petri Nets Containing Immediate Transitions". In *On-Site Proceedings of the Satellite Workshop of ICALP*. June 30th- July 4th, Eindhoven, The Netherlands.

Lazarova-Molnar, S., and X. Li. 2019. "Deriving Simulation Models from Data: Steps of Simulation Studies Revisited". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 2771–2782. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lazarova-Molnar, S., and N. Mohamed. 2019. "Reliability Assessment in the Context of Industry 4.0: Data as a Game Changer". *Procedia Computer Science* 151:691–698.

Lazarova-Molnar, S., N. Mohamed, and J. Al-Jaroodi. 2018. "Collaborative Data Analytics for Industry 4.0: Challenges, Opportunities and Models". In *6th International Conference on Enterprise Systems*. October 1st-2nd, Limassol, Cyprus, 100–107.

Lazarova-Molnar, S., N. Mohamed, and J. Al-Jaroodi. 2019. "Data Analytics Framework for Industry 4.0: Enabling Collaboration for Added Benefits". *IET Collaborative Intelligent Manufacturing* 1(4):117–125.

Lee, W. S., D. L. Grosh, F. A. Tillman, and C. H. Lie. 1985. "Fault Tree Analysis, Methods, and Applications: A Review". *IEEE Transactions on Reliability* R-34:194–203.

Linard, A., D. Bucur, and M. Stoelinga. 2019. "Fault Trees from Data: Efficient Learning with an Evolutionary Algorithm". In *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*, edited by N. Guan, J. Katoen, and J. Sun, 19–37. Shanghai: Springer.

Mukherjee, S., and A. Chakraborty. 2007. "Automated Fault Tree Generation: Bridging Reliability with Text Mining". In *2007 Annual Reliability and Maintainability Symposium*. January 22nd-25th, Orlando, USA, 83 - 88: Institute of Electrical and Electronics Engineers, Inc.

Nauta, M., D. Bucur, and M. Stoelinga. 2018. "LIFT: Learning Fault Trees from Observational Data". In *Quantitative Evaluation of Systems: 15th International Conference*, edited by A. McIver and A. Horvath. Beijing, China: Springer.

Nolan, P., M. Madden, and P. Muldoon. 1994. "Diagnosis Using Fault Trees Induced from Simulated Incipient Fault Case Data". In *Second International Conference on Intelligent Systems Engineering*. September 5th-9th, Hamburg-Harburg, Germany, 304 - 309.

Oosterbaan, R. 2019. "Software for Generalized and Composite Probability Distributions". *International Journal of Mathematical and Computational Methods* 4:1–9.

Ruijters, E., and M. Stoelinga. 2015. "Fault Tree Analysis: A Survey of the State-of-the-art in Modeling, Analysis and Tools". *Computer Science Review* 15-16:29–62.

Solomatine, D., L. See, and R. Abrahart. 2008. "Data-Driven Modelling: Concepts, Approaches and Experiences". In *Practical Hydroinformatics*, edited by R. Abrahart, L. See, and D. Solomatine, Volume 68, 17–30. New York: Springer.

Volkanovski, A., M. Čepin, and B. Mavko. 2009. "Application of the Fault Tree Analysis for Assessment of Power System Reliability". *Reliability Engineering & System Safety* 94(6):1116–1127.

Yuge, T., M. Maruyama, and S. Yanagi. 2016. "Reliability of a k-out-of-n System with Common-cause Failures Using Multivariate Exponential Distribution". *Procedia Computer Science* 96:968–976.

AUTHOR BIOGRAPHIES

SANJA LAZAROVA-MOLNAR is an associate professor with the Faculty of Engineering at the University of Southern Denmark. Her current research interests include modeling and simulation of stochastic systems, reliability modeling and data analytics for decision support. Sanja Lazarova-Molnar obtained her PhD in Computer Science from the University in Magdeburg, Germany in 2005, specializing in the area of Modeling and Simulation. Her email address is slmo@mmmi.sdu.dk.

PARISA NILOOFAR is a Postdoc researcher with the Faculty of Engineering at the University of Southern Denmark. Her current research interests include Bayesian networks, simulation and modelling, data science and reliability modeling. Parisa Niloofar obtained her PhD in Statistics, in 2013, specializing in the area of Graphical modeling. Her email address is parni@mmmi.sdu.dk.

GABOR KEVIN BARTA is a M. Sc. in Software Engineering graduate from the University of Southern Denmark. His research interests included Artificial Intelligence and Modeling Simulation. He is currently working as a Python developer at a financial services firm in Budapest, Hungary. His email address is gkevinba@gmail.com.