Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023)
Edited by Mário P. Brito, Terje Aven, Piero Baraldi, Marko Čepin and Enrico Zio
©2023 ESREL2023 Organizers. *Published by* Research Publishing, Singapore.
doi: 10.3850/978-981-18-8071-1_P407-cd

SAFEST: the static and dynamic fault tree analysis tool

Matthias Volk
University of Twente, The Netherlands. E-mail: m.volk@utwente.nl

Muzammil Ibne Irshad
DGB Technologies, US

Joost-Pieter Katoen
RWTH Aachen University, Germany and University of Twente, The Netherlands

Falak Sher
DGB Technologies, US

Mariëlle Stoelinga
University of Twente, The Netherlands and Radboud University Nijmegen, The Netherlands

Ahmad Zafar
DGB Technologies, US

We present SAFEST, the Static And dynamic Fault trEe analySis Tool. While standard (or static) fault trees (SFT) appeal as a relatively simple tool, they are limited in their modeling capabilities. Dynamic fault trees (DFT) extend SFT by support for faithfully modeling spare management, order-dependent failures and functional dependencies. While various analysis approaches for DFTs exist in the literature, tool support is scarce.
During the last years, we developed SAFEST, a modern, state-of-the-art tool for modeling and analysing (SFTs and) DFTs. SAFEST's web-based interface offers a drag-and-drop editor for creating fault trees. A step-by-step simulator visualizes how failures – given by the user – affect the state of DFT elements.
SAFEST employs a plethora of analysis approaches. SFTs are best analyzed using binary decision diagrams (BDD) and SAFEST performs comparable to existing SFT tools. DFTs are analyzed via state-based techniques by translation into a Markov model. SAFEST can analyze these models via probabilistic model checking, yielding exact results efficiently. It also provides an approximation approach that builds only the most "important" parts of the DFT's behaviour. This enables the analysis of gigantic DFTs at the expense of exactness of results. This approximation provides upper and lower bounds on the reliability measure of interest. The precision can be tuned according to the user's needs.
The modeling and analysis capabilities of DFTs as well as the performance of our tool has been demonstrated in several practical and industrial case studies. DFTs with up to several hundreds of elements have been successfully analyzed with SAFEST.

Keywords: Fault tree analysis, Dynamic fault trees, Model checking, Formal methods, Risk management, Tool

1. Introduction

Probabilistic risk assessment (PRA) is a vital task in ensuring the security of mission- and safety-critical fault-tolerant/fail-operational systems. Its importance increased due to the tighter restrictions set by international standards – such as ISO 26262 for autonomous driving – and the increasing prevalence of AI/ML components in crucial systems. This motivates considering PRA methods throughout all system design and development phases, eventually leading to a full integration of PRA with *model-based systems engineering (MBSE)* methodology.

Fault trees Stamatelatos et al. (2002); Ruijters and Stoelinga (2015) are widely used in industry to perform PRA of complex systems. Their use is required for instance by the Federal Aviation Authority (FAA), the Nuclear Regulatory Commission (NRC) and space agencies such as NASA and ESA. While standard (or static) fault trees (SFT) appeal as a relatively simple tool, they are limited in their modeling capabilities. To express more complex dependability scenarios, Dugan's *dynamic fault trees (DFT)* Dugan et al. (1990) have been developed which extend SFTs by support for modeling spare management, order-dependent failures and (probabilistic) functional dependencies e.g., common cause failures (CCF). While various analysis approaches for DFTs have been developed Ruijters and Stoelinga (2015) – e.g., via Markov models, Bayesian networks, Petri nets or Monte Carlo simulation – tool support for DFTs is scarce.

The SAFEST tool We developed SAFEST, a modern, state-of-the-art tool for modeling and analyzing DFTs. The web-based interface of SAFEST provides a graphical editor to create fault trees as well as simplify and simulate them. The tool also allows to automatically extract DFTs from SysML 2.0 models annotated with safety information, e.g., redundancy, functional dependency, etc., and therefore integrates PRA with MBSE. SAFEST provides several efficient analysis techniques for DFTs based on probabilistic model checking, a dedicated analysis for SFTs based on binary decision diagrams (BDDs), as well as a hybrid technique combining both of them.

The modeling capabilities of DFTs as well as the performance of our tool has been demonstrated in several practical and industrial case studies. Examples include DFT models for vehicle guidance systems in the automotive domain Ghadhab et al. (2019), and analyzing infrastructure failures in railway station areas Weik et al. (2022). DFTs with up to several hundred elements have been successfully analyzed with SAFEST.

SAFEST and its documentation is publicly available at www.safest.dgbtek.com.

Outline Sect. 2 briefly introduces (dynamic) fault trees. Sect. 3 presents the major features of SAFEST for modeling with DFTs; Sect. 4 presents its analysis approaches. Sect. 5 presents industrial case studies which have been performed using SAFEST. Sect. 6 compares SAFEST to various existing fault tree analysis tools. Sect. 7 concludes.

2. Dynamic fault trees

Fault trees (FT) Ruijters and Stoelinga (2015) model how component failures propagate through a system and yield a system failure. FTs are directed acyclic graphs. Leaves (without children) correspond to *basic events (BE)* which fail according to an associated failure distribution. Inner nodes correspond to *gates* which propagate the failure. *Static fault trees (SFT)* have gates of type AND, OR and VOT_k which fail if all, at least one or at least k children have failed, respectively.

Dynamic fault trees (DFT) Dugan et al. (1990) extend SFTs by dynamic gates modeling spare management, ordered failures and functional dependencies. Fig. 1 depicts an example DFT modeling the rotor failures in a floating offshore wind turbine and is based on Zhang et al. (2016). The top-level element *RotorSystemFailure* is represented by an OR-gate and occurs when for example BE *BladeFailure* occurs. The gates in orange color are the three main dynamic gates which we will introduce by example. For a detailed presentation of the DFT semantics, we refer to Junges et al. (2018).

SPARE The SPARE-gate *PowerFailure* models spare management. First, the primary child *Power1Failure* is actively used and the spare component *Power2Failure* is in standby, i.e., it has a reduced failure probability. If *Power1Failure* fails, the spare *Power2Failure* starts to be activly used. If it also fails, the SPARE-gate fails.

Priority-AND Switching from one power source to another can also fail. This is modeled by the *Priority-AND (*PAND*) gate SwitchFailure*. A PAND only fails if its children fail in order from left-to-right. For instance, first *PowerSwitch* fails and then *Power1Failure*. At this time point, the power source cannot be switched, because the switching mechanism already failed before. In contrast, if first *Power1Failure* fails, the switching can still be performed.

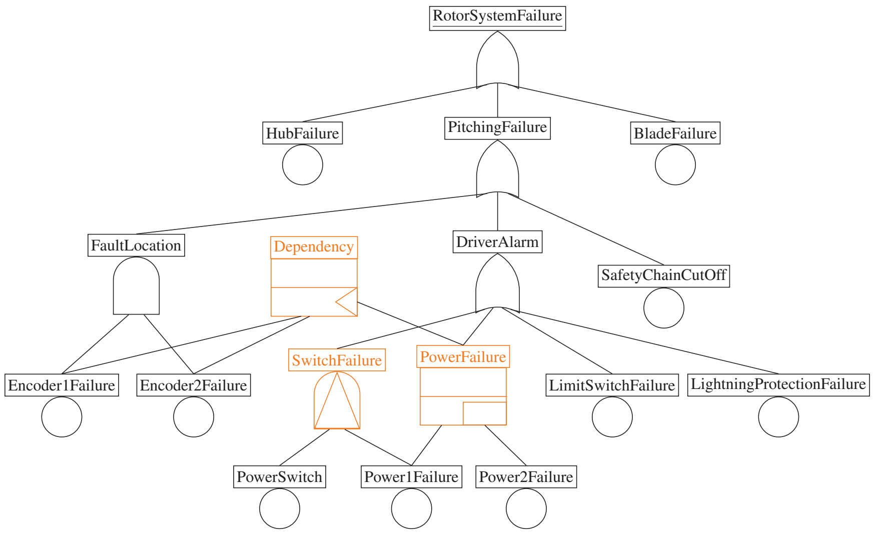


Figure 1. DFT modeling a rotor in an offshore wind turbine

Functional dependency The *functional dependency* (FDEP) models dependencies by forwarding a failure from one component to others. FDEP *Dependency* models the dependency of the encoders on the power source. If *PowerFailure* occurs, both encoders cannot function anymore, because the power source is unavailable.

3. Modeling with Safest

Graphical editor The drag-and-drop fault tree editor in Safest's web-based interface allows for the graphical construction and updating of fault trees. Fig. 2 shows the DFT from Fig. 1 in Safest's editor. The editor allows for hierarchical modeling and reuse of sub-trees. Moreover, fault trees can also be presented in tabular form or the Galileo format. The latter can also be used as input format when importing existing DFT. Note that Safest supports static gates (AND, OR, VOT) as well as all dynamic gates described in the literature (FDEP, PAND, SEQ, SPARE).

SysML import Safest can automatically produce DFTs from SysML 2.0 models[a] that have been annotated with safety data, such as functional dependencies, redundancies, etc. This enables reliability study of systems to take place concurrently with their design and development, assisting with the exploration of design space to determine the optimal design in relation to various metrics. Fig. 3 shows an example SysML model (right) and its corresponding DFT (left).

[a]`https://www.omgsysml.org/SysML-2.htm`

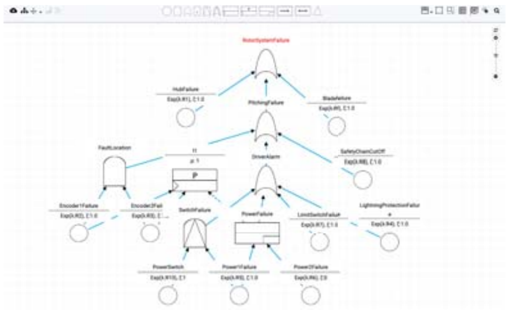
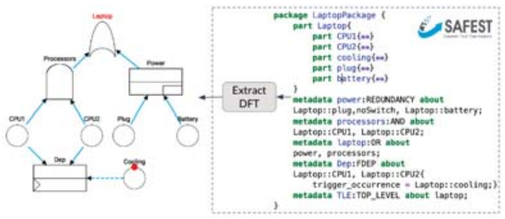
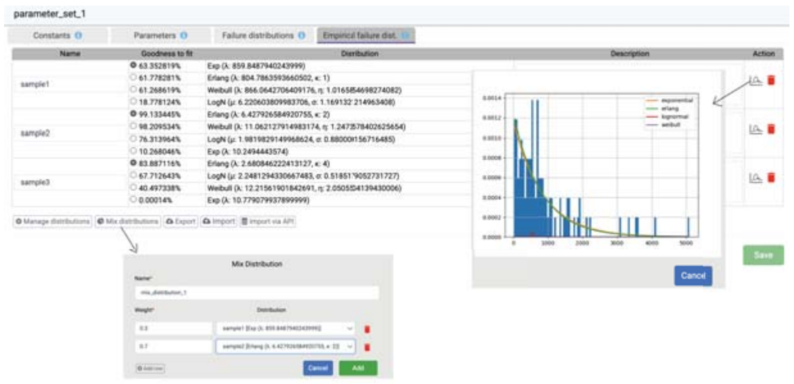


Figure 2. Graphical editor for fault trees



```
package LaptopPackage {
    part Laptop{
        part CPU1{**}
        part CPU2{**}
        part cooling{**}
        part plug{**}
        part battery{**}
    }
    metadata power:REDUNDANCY about
    Laptop::plug,noSwitch, Laptop::battery;
    metadata processors:AND about
    Laptop::CPU1, Laptop::CPU2;
    metadata laptop:OR about
    power, processors;
    metadata Dep:FDEP about
    Laptop::CPU1, Laptop::CPU2{
        trigger_occurrence = Laptop::cooling;}
    metadata TLE:TOP_LEVEL about laptop;
}
```

Figure 3. Translation from a SysML model to a DFT



Name	Goodness to fit	Distribution
sample1	● 63.352819%	Exp (λ: 859.8487940243999)
	○ 61.778281%	Erlang (λ: 804.7863593660502, κ: 1)
	○ 61.268619%	Weibull (λ: 866.0642706409176, η: 1.016584698274082)
	○ 18.778124%	LogN (μ: 6.220603809983706, σ: 1.169132'214963408)
sample2	● 99.133445%	Erlang (λ: 6.427926584920755, κ: 2)
	○ 98.209534%	Weibull (λ: 11.062127914983174, η: 1.2473578402625654)
	○ 76.313964%	LogN (μ: 1.9819829149968624, σ: 0.880001156716485)
	○ 10.268046%	Exp (λ: 10.2494443574)
sample3	● 83.887116%	Erlang (λ: 2.680846222413127, κ: 4)
	○ 67.712643%	LogN (μ: 2.2481294330667483, σ: 0.5185179052731727)
	○ 40.497338%	Weibull (λ: 12.21561901842691, η: 2.0505504139430006)
	○ 0.00014%	Exp (λ: 10.779079937899999)

Weight*	Distribution
0.3	sample1 [Exp (λ: 859.8487940243999)]
0.7	sample2 [Erlang (λ: 6.427926584920755, κ: 2)]

Figure 4. Empirical distributions

Model parameters SAFEST allows to generate different variants of DFTs by changing the values of model parameters, e.g., failure distributions of BEs. The model parameters can be constants, real-valued expressions, and probability distributions. The model parameters allow easy comparison of different model variants by simply changing the parameter values during the analysis step.

Failure distributions SAFEST supports exponential, Erlang, Weibull and Log-normal distributions for SFTs, and exponential and Erlang distributions for DFTs. Failure distributions can be (1) provided explicitly, (2) can be automatically generated from given failure data of components via statistical methods, or (3) can be composed as a mixture distributions, i.e., the weighted average of other distributions. Fig. 4 shows how distributions are generated from existing data (right) or as a mixture distribution (bottom).

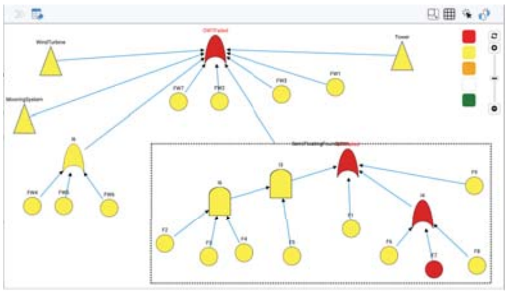


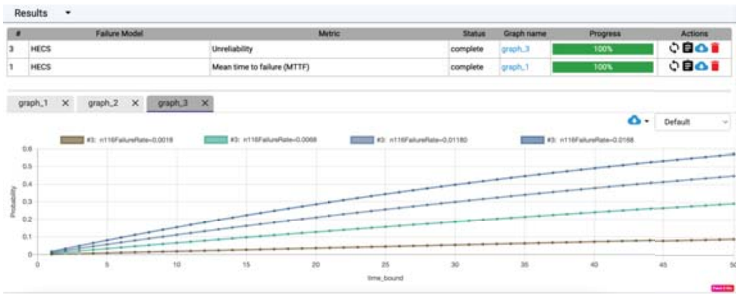
Figure 5. Interactive simulator

Interactive simulator To improve understanding of the fault tree model, SAFEST offers an interactive DFT simulator. The step-by-step simulator allows users to select which BE should fail next. It then shows how the failures affect the state of all DFT elements. An example is depicted in Fig. 5 where the failure of BE *F7* (marked in red color) propagates through the system and yields a failure of *OWTFailed*. Yellow color indicates irrelevant DFT elements whose (future) failure does not have an impact anymore. The interactive simulator allows users to get a better understanding of their model and helps detect modeling errors early on.

DFT simplification SAFEST can automatically simplify the structure of a DFT while still preserving its behavior. The simplification is based on Junges et al. (2017) and transforms the DFT such that it is better suited for efficient analysis.

4. Fault tree analysis

SAFEST supports different approaches to analyze fault trees. Given a fault tree and a metric of interest, the corresponding results are calculated in a fully automated manner. Fig. 6 exemplary shows the failure probability over time for different system variants modeled as DFTs. The analysis in SAFEST is performed in the backend by the open-source STORM-DFT library of the STORM model checker Hensel et al. (2022). Depending on the given fault tree, metric to check, and user needs, the best suited analysis approach can be selected.


Results

#	Failure Model	Metric	Status	Graph name	Progress	Actions
3	HECS	Unreliability	complete	graph_3	100%	
1	HECS	Mean time to failure (MTTF)	complete	graph_1	100%	

graph_1
graph_2
graph_3
Default
#3: n116FailureRate=0.0018
#3: n116FailureRate=0.0068
#3: n116FailureRate=0.01180
#3: n116FailureRate=0.0168
Probability
time_bound

Figure 6. Analysis results for DFT variants

Metrics SAFEST supports analysis with regard to all significant quantitative dependability metrics, including system reliability, mean-time-to-failure (MTTF), and component criticality based on importance. Experienced user also have the freedom to specify their own unique measures of interest, up to the point of expressing intricate measures in mathematical logics.

Static fault trees Classical (static) fault trees are best analyzed using *binary decision diagrams (BDD)*. The approach supports BE with arbitrary probability distributions – going far beyond exponential failure distributions. SAFEST allows to obtain the *minimal cut sets (MCS)* of an SFT. The unreliability can be directly computed on the BDD, yielding precise values. Moreover, the unreliability over time – for hundreds of time points – can efficiently be computed via dedicated methods Basgöze et al. (2022). Evaluations have shown that our BDD-based analysis performs comparable to existing academic tools for SFT analysis Basgöze et al. (2022).

Dynamic fault trees DFTs are analyzed via state-based techniques by translation into a Markov model Boudali et al. (2010); Volk et al. (2018). Our automated translation yields small models by exploiting – among others – irrelevant failures and symmetries in the DFT. These optimization are crucial to mitigate the state space explosion problem. The generated Markov model is analyzed with the state-of-the-art probabilistic model checker STORM Hensel et al. (2022). Probabilistic model checking allows to efficiently calculate a broad range of metrics – far beyond simple reliability calculations – including mean-time-to-failure, conditional probabilities and performance under degradation. Most importantly, model checking yields exact results – which is crucial when analyzing safety-critical system. Comparison to existing tools such as DFTCALC Arnold et al. (2013) and the original GALILEO tool Sullivan et al. (1999) has shown that our tool significantly outperforms its competitors – up to orders of magnitude Volk et al. (2018).

Modular analysis SAFEST allows modular analysis of fault trees based on the modularization technique of Gulati and Dugan (1997). Instead of analyzing the complete system at once, different parts of a fault tree – called *modules* – which represent individual subsystems are analyzed independently. The analysis of each module is performed via the technique best suited for them, e.g., SFTs are analyzed via BDDs while DFTs are translated into Markov models.

Approximation via partial state space Lastly, the tool provides an approximation approach that builds only the most "important" parts of the DFT's behavior Volk et al. (2018). By only building parts of the state space, this approach requires significantly less computational resources than building the full state space. This approximation provides an upper and lower bound on the exact measure of interest. The approach therefore still provides correctness guarantees – in contrast to other approaches such as Monte Carlo simulation which only provide confidence intervals. The pre-

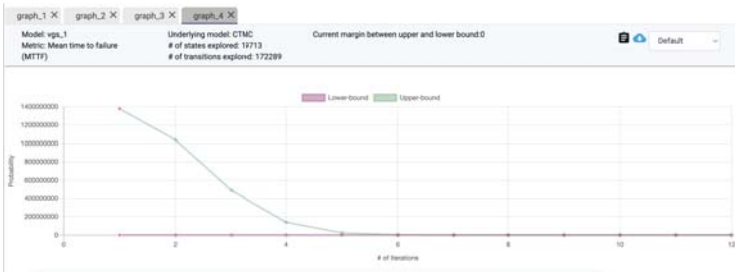

Model: vgs_1
Metric: Mean time to failure (MTTF)
Underlying model: CTMC
# of states explored: 19713
# of transitions explored: 172289
Current margin between upper and lower bound:0

Figure 7. Approximative analysis

cision of our approximation can be increased by exploring larger parts of the state space. SAFEST allows the users to interactively tune this precision according to their needs. Fig. 7 shows the approximation approach. In the first iterations, the difference between upper and lower bounds is large, but by exploring more parts of the state space, the bounds quickly become tight. For instance, the fifth iteration only builds half of the total state space while already providing tight bounds.

5. Case studies

We showed the modeling capabilities of DFTs as well as the performance of our tool in several practical and industrial case studies. We refer to the FFORT benchmark suite[b] for an extensive list of DFT models from the literature.

In Ghadhab et al. (2019), we built DFT models for a vehicle guidance system in the automotive domain. We modeled different safety concepts and different partitioning of functions on hardware. Our DFT models allowed us to compare the different designs and find the best one. The DFTs consisted of up to 300 elements and are, to the best of our knowledge, the largest real ones in the literature. Nevertheless, the DFTs could be evaluated within minutes.

In Khan et al. (2021), we modeled fire sprinkler systems in malls using DFTs and validated metrics beyond standard reliability, e.g., the likelihood of failing without previous degradation and the worst-case reliability obtained after degradation.

In Weik et al. (2022), we considered train routing options in railway station areas w.r.t. the available infrastructure elements. The DFTs were automatically generated from infrastructure and routing data. Our analysis showed, for example, the impact of switch failures on train routes and also identified the most critical infrastructure elements.

In a recent industrial case study, we automatically derived DFTs from SysML 2.0 models of electric devices after annotating them with safety data. In model-based systems engineering, this opened the door for model-based safety analysis.

6. Related tools

A number of tools for fault tree analysis exists, both academic and commercial. We refer to Ruijters and Stoelinga (2015), Baklouti et al. (2017) and Aslansefat et al. (2020) for overviews of such tools. The vast majority of these tools however only support SFTs. Only a few tools support the analysis of DFTs, most of them academic tools. These tools only implement dedicated analysis algorithms, but do not provide a complete framework for modeling and analyzing DFTs in all their aspects. SAFEST in contrast provides such an integrated framework for DFT analysis as well as for modeling DFTs in a hierarchical manner.

We provide a comparison of SAFEST with commonly used tools for fault tree analysis in Tab. 1. For each tool, we indicate which operating systems are supported, which type of fault tree is supported, which analysis techniques are available – via minimal cut sets, binary decision diagrams, model checking of Markov models, and Monte Carlo simulation – and which steps of the workflow are supported by a graphical user interface. We gathered the data for each tool from publicly available information on the respective websites.

[b]https://dftbenchmarks.utwente.nl/ffort.php

Tool	OS			FT type		Analysis technique				GUI		Notes
	Windows	macOS	Linux	SFT	DFT	MCS	BDD	Markov model	MC simulation	FT editor	Visualize results	
SAFEST	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	analysis via STORM tool
DFTCALC	✓	✓	✓	✓	✓			✓			✓	academic
HiP-HOPS	✓			✓	*	✓				✓	✓	
Isograph FaultTree+	✓			✓	*	✓		✓		✓	✓	BE as Markov model
RAM Commander's FTA	✓			✓		✓	✓			✓	✓	part of RAM Commander
RiskSpectrum PSA	✓			✓	*	✓	✓			✓	✓	part of RiskSpectrum
Saphire	✓			✓		✓				✓	✓	BE as Markov model
Windchill FTA	✓			✓	✓	✓		✓		✓	✓	part of Windchill
XFTA	✓	✓	✓	✓		✓	✓		✓			from AltaRica Association

From Tab. 1, we can see that most tools are only available on Windows. Apart from SAFEST, only the academic tools DFTCALC and XFTA are available on all platforms. As stated before, most tools only support SFTs. DFTs are only fully supported by SAFEST, DFTCALC and WINDCHILL FTA. Other tools such as FAULTTREE+, HIP-HOPS and RISKSPECTRUM support only some dynamic gates, e.g., PANDs. As most tools only support SFTs, their main analysis technique is based on determining MCS. Both DFTCALC and SAFEST support analysis via model checking of Markov models. ISOGRAPH FAULT TREE+ and WINDCHILL FTA also support Markov models. SAFEST supports the broadest range of analysis techniques, combining efficient approaches for both static and dynamic fault trees. All commercial tools provide a graphical user interface shipped with a fault tree editor and visualization of analysis results. Both academic tools DFTCALC and XFTA lack in this regard.

In summary, the tools most closely related to SAFEST are DFTCALC and WINDCHILL FTA as both also support DFTs. DFTCALC – as an academic tool – is fully focused on DFTs and no explicit support for SFTs exists. It only provides a rudimentary GUI without any FT editor and supports only a small set of DFT analysis capabilities. WINDCHILL FTA from PTC is most likely the closest competitor to SAFEST. As part of the WINDCHILL tool ecosystem it supports a broad range of input formats, metrics and GUI features.

Besides DFTs, there exist other extensions of SFTs such as *Boolean-logic Driven Markov decision Processes (BDMP)* Bouissou and Bon (2003), *State/Event Fault Trees (SEFT)* Kaiser et al. (2007) and *Pandora temporal fault trees* Walker and Papadopoulos (2009). We refer to Ruijters and Stoelinga (2015) for an overview of related models and corresponding tool support.

7. Conclusion

We presented the SAFEST tool for modeling and analyzing static and dynamic fault trees. SAFEST provides an easy-to-use interface for modeling, adapting, simplifying and interactively simulating DFT. Fault tree analysis is performed through dedicated, powerful approaches based on binary decision diagrams (for SFT) and probabilistic model checking (for DFT). The latter allows to efficiently calculate a broad range of metrics going far beyond simple unreliability. An approximation approach building only the most "important" parts of the Markov model allows to fine-tune the trade-off between computational resources and precision of the result. SAFEST and its DFT analysis have been successfully applied in practical and industrial case studies, allowing to handle DFTs with several hundreds elements. SAFEST is available at www.safest.dgbtek.com.

Future work In the future, we aim to further improve the usability of the user interface, based on feedback by industrial practitioners. We also plan to provide better integration with existing workflows, e.g., by supporting import and export from common file formats used by other tools.

Acknowledgement

This work has been partially funded by the NWO grant NWA.1160.18.238 (PrimaVera), by the ERC Consolidator Grant 864075 (*CAESAR*) and by EU Horizon 2020 project MISSION, number 101008233.

References

Arnold, F., A. Belinfante, F. I. van der Berg, D. Guck, and M. Stoelinga (2013). DFTCalc: A tool for efficient fault tree analysis. In *SAFECOMP*, Volume 8153 of *Lecture Notes in Computer Science*, pp. 293–301. Springer.

Aslansefat, K., S. Kabir, Y. Gheraibia, and Y. Papadopoulos (2020). Dynamic fault tree analysis: State-of-the-art in modeling, analysis, and tools. In *Reliability Management and Engineering* (1 ed.)., pp. 73–112. CRC Press.

Baklouti, A., N. Nguyen, J.-Y. Choley, F. Mhenni, and A. Mlika (2017). Free and open source fault tree analysis tools survey. In *SysCon*, pp. 1–8. IEEE.

Basgöze, D., M. Volk, J.-P. Katoen, S. Khan, and M. Stoelinga (2022). BDDs strike back - efficient analysis of static and dynamic fault trees. In *NFM*, Volume 13260 of *Lecture Notes in Computer Science*, pp. 713–732. Springer.

Boudali, H., P. Crouzen, and M. Stoelinga (2010). A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE Trans. Dependable Secur. Comput. 7*(2), 128–143.

Bouissou, M. and J. Bon (2003). A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. *Reliab. Eng. Syst. Saf. 82*(2), 149–163.

Dugan, J. B., S. J. Bavuso, and M. A. Boyd (1990). Fault trees and sequence dependencies. In *RAMS*, pp. 286–293.

Ghadhab, M., S. Junges, J.-P. Katoen, M. Kuntz, and M. Volk (2019). Safety analysis for vehicle guidance systems with dynamic fault trees. *Reliab. Eng. Syst. Saf. 186*, 37–50.

Gulati, R. and J. B. Dugan (1997). A modular approach for analyzing static and dynamic fault trees. In *RAMS*, pp. 57–63.

Hensel, C., S. Junges, J.-P. Katoen, T. Quatmann, and M. Volk (2022). The probabilistic model checker storm. *Int. J. Softw. Tools Technol. Transf. 24*(4), 589–610.

Junges, S., D. Guck, J.-P. Katoen, A. Rensink, and M. Stoelinga (2017). Fault trees on a diet: automated reduction by graph rewriting. *Formal Aspects Comput. 29*(4), 651–703.

Junges, S., J.-P. Katoen, M. Stoelinga, and M. Volk (2018). One net fits all - A unifying semantics of dynamic fault trees using GSPNs. In *Petri Nets*, Volume 10877 of *Lecture Notes in Computer Science*, pp. 272–293. Springer.

Kaiser, B., C. Gramlich, and M. Förster (2007). State/event fault trees - A safety analysis model for software-controlled systems. *Reliab. Eng. Syst. Saf. 92*(11), 1521–1537.

Khan, S., J.-P. Katoen, M. Volk, M. A. Zafar, and F. Sher (2021). Modelling and analysis of fire sprinklers by verifying dynamic fault trees. In *LADC*, pp. 1–10. IEEE.

Ruijters, E. and M. Stoelinga (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Comput. Sci. Rev. 15*, 29–62.

Stamatelatos, M., W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback (2002). Fault tree handbook with aerospace applications.

Sullivan, K. J., J. B. Dugan, and D. Coppit (1999). The Galileo fault tree analysis tool. In *FTCS*, pp. 232–235. IEEE Computer Society.

Volk, M., S. Junges, and J.-P. Katoen (2018). Fast dynamic fault tree analysis by model checking techniques. *IEEE Trans. Ind. Informatics 14*(1), 370–379.

Walker, M. and Y. Papadopoulos (2009). Qualitative temporal analysis: Towards a full implementation of the fault tree handbook. *Control Engineering Practice 17*(10), 1115–1125.

Weik, N., M. Volk, J.-P. Katoen, and N. Nießen (2022). DFT modeling approach for operational risk assessment of railway infrastructure. *Int. J. Softw. Tools Technol. Transf. 24*(3), 331–350.

Zhang, X., L. Sun, H. Sun, Q. Guo, and X. Bai (2016). Floating offshore wind turbine reliability analysis based on system grading and dynamic FTA. *Journal of Wind Engineering and Industrial Aerodynamics 154*, 21–33.