DFTCalc:
A Tool for Efficient Fault Tree Analysis

Florian Arnold[1], Axel Belinfante[1], Freark Van der Berg[1],
Dennis Guck[1], Mariëlle Stoelinga[1]

[1] Department of Computer Science, University of Twente, The Netherlands
{f.arnold,d.guck,a.f.e.belinfante,m.i.a.stoelinga}@utwente.nl
f.i.vanderberg@student.utwente.nl

Abstract. Effective risk management is a key to ensure that our nuclear power plants, medical equipment, and power grids are dependable; and it is often required by law. Fault Tree Analysis (FTA) is a widely used methodology here, computing important dependability measures like system reliability. This paper presents DFTCalc, a powerful tool for FTA, providing (1) efficient fault tree modelling via compact representations; (2) effective analysis, allowing a wide range of dependability properties to be analysed (3) efficient analysis, via state-of-the-art stochastic techniques; and (4) a flexible and extensible framework, where gates can easily be changed or added. Technically, DFTCalc is realised via stochastic model checking, an innovative technique offering a wide plethora of powerful analysis techniques, including aggressive compression techniques to keep the underlying state space small.

1 Introduction

Risk analysis is a key feature in reliability engineering: in order to design and build medical devices, smart grids, and internet shops that meet the required dependability standards, we need to assess at design time how dependable these systems are, and take appropriate measures if they are not dependable enough.

Fault Trees. Fault tree analysis (FTA) [19] is a graphical technique that is often used in industry. Fault trees (FTs) model how component failures lead to system failures: the leaves of a FT are basic events (BEs) that represent component failures; the other nodes express how failures propagate through the system via AND and OR gates. Discrete time FTs equip each BE with a probability p, representing the probability that the component fails within a certain discrete time interval. We consider continuous FTs. Here, each BE is equipped with a probability distribution f showing how the failure behaviour evolves over time, i.e. $F(t)$ represents the probability that the BE is still running at time point t. The root of the tree, called the top-level event, represents a system failure. FTA typically computes for a given FT the *system reliability,* i.e. the probability that the system has not failed within a given mission time T, the *mean time to failure* (MTTF), i.e. the expected time of a failure to occur, and the *availability*, i.e. the time that the system is up in the long run.

Dynamic Fault Trees (DFTs) extend standard (or static) fault trees with a number of intuitive gates. These gates facilitate the modelling of often recurring concepts in reliability engineering: spare management, functional dependencies, and order-dependent behaviour.

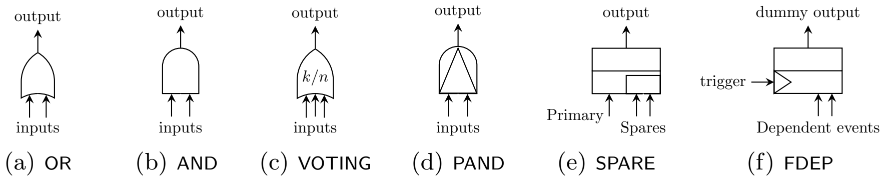
DFTCalc. DFTCalc is a powerful tool for modelling and analysis of DFTs. It can efficiently model DFTs and provides means to compute various dependability metrics, given BEs whose failure probabilities are given by exponential and phase type distributions. The major innovation of DFTCalc is the deployment of stochastic model checking (SMC) techniques [4]: SMC is an innovative technique to systematically explore the state space of a stochastic system. SMC provides a wide plethora of powerful analysis techniques, with fully-fledged tool support. By deploying SMC, DFTCalc can handle DFTs with BEs that are statistically dependent; in fact, the FDEP gate has specifically been designed to model interdependent events. Repairs, however, have not yet been included.

The main problem in time-dependent reliability analysis is its complexity: The state space of models of real systems can grow arbitrarily large [15] and, thus, highly efficient techniques are required to yield results in a feasible time. Furthermore, an accurate modelling of all dependencies in these inherently complex systems requires an ever growing diversity of new gates. DFTCalc constitutes an architectural framework that addresses both challenges.

Related work. A wide range of FTA methods exists: Classically, one obtains the minimal cut sets in the FT [5]. This enables to order components based on their structural importance. Further, with additional information one can compute the system reliability. A popular technique is to exploit Bayesian networks, which are useful both in discrete time [9] and in continuous time [8]. Our approach focuses on continuous timed systems, with currently no maintenance. Therefore, we will translate DFTs into continuous time Markov chains (CTMCs) and use state of the art techniques as described in [2,3]. This allows us to compute reliability measures by use of efficient techniques for transient analysis of CTMCs.

A wide number of commercial and academic tools for static fault tree analysis are available. Some are merely drawing tools, while others provide probabilistic analysis, like the popular FaultTree+ package from Isograph [14]. Dynamic FTA is supported by tools like Windchill [18], NASA's Galileo/ASSAP software [11], and the simulation tool DFTSim [10]. A first implementation of DFT analysis using I/O-IMCs was realized in Coral [7], the predecessor of DFTCalc.

Organisation of the paper. Section 2 presents DFTCalc's modelling and analysis capabilities and Section 3 the architecture and internal structure. In Section 4 we provide experimental results and Section 5 concludes the paper. Due to space constraints, we refer to [1] for more details of our main results and case studies.



(a) OR (b) AND (c) VOTING (d) PAND (e) SPARE (f) FDEP

Fig. 1. Dynamic fault tree gates.

2 DFTCalc: modelling and analysis

DFT modelling. Dynamic fault trees (DFTs) model the failure propagation in complex systems. The leaves of a DFT are labeled with *basic events* and the non-leaves with *gates*. The root is called *top-level event*.

Basic events. A basic event (BE) represents the failure behaviour of a basic system component, and can be in three different modes: *dormant*, *active* and *failed*. The component is in dormant mode, if it is not in use. In this mode, the failure rate of a BE is decreased by a *dormancy factor* $\alpha \in [0,1]$. In case $\alpha = 0$ the BE cannot fail (cold BE) and in case $\alpha = 1$ the failure rate is the same as in active mode (warm BE). The component is in active mode, when it is in use. If the component breaks down, it is in failed mode.

Gates. A gate expresses how component failures induce a system failure. Gates consist of one or more inputs, and one output. Fig. 1 depicts the DFT gates.

(a) The OR gate fails when at least one input fails.
(b) The AND gate fails when all of its inputs fail.
(c) The VOTING gate fails when at least k out of n inputs fail.
(d) The PAND gate fails when all of its inputs fail from left to right.
(e) The SPARE gate consists of a *primary* input and one or more *spare* inputs. At system start, the primary is active and the spares are in dormant mode. When the primary input fails, one of the spare inputs is activated and replaces the primary. If no more spares are available, the SPARE gate fails. Note that a spare component can be shared among several spare gates.
(f) The FDEP (functional dependency) gate consists of one *trigger* event and several *dependent events*. When the trigger event occurs, all dependent events fail. The FDEP has a "dummy" output, which is represented by a dotted line and ignored in calculations.

Example 1. Fig. 2 depicts a DFT representing a cardiac assist system (CAS) [9] consisting of three subsystems: the CPU, the motor and pump units. If either one of these subsystems fails, then the entire CAS fails, as modelled by the top level OR gate. The CPU unit consists of a primary (P) and a backup (B) CPU, as indicated by the SPARE gate. The primary and backup CPU are subject to a common cause failure, modelled by the CPU FDEP gate: if either the crossbar switch (CS) or the system supervisor (SS) fails, the primary and backup CPU

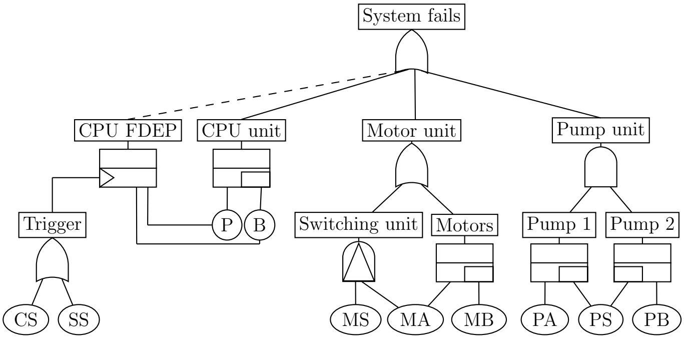


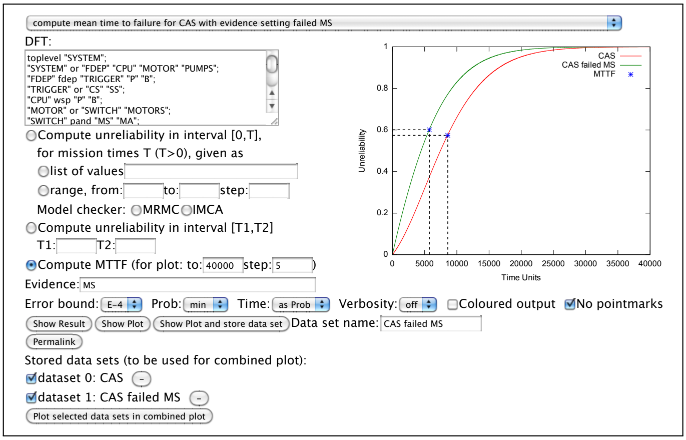
Fig. 2. The cardiac assist system DFT.

become unavailable. The motor unit consists of a primary (MA) and a backup (MB) motor. If the primary fails, the motor switching component (MS) will turn on the backup motor. Because of the PAND gate the failure of the switching component can then be ignored. Finally, the pump unit consists of two pumps (PA and PB), which share a common cold spare (PS).

DFT analysis. DFTCalc can compute a number of different reliability metrics, namely all metrics that can be expressed as reachability properties in the logic CSL. This includes properties such as: (1) *Timed-Reliability*: the probability that the system fails until a given time point T or in a given interval $[T, T']$; (2) *Mean time to failure*: the expected time to a system failure; (3) *Reliability*: the probability that the system fails in the long-run. In case of non-determinism, we calculate the minimum and maximum values for the above metrics. Each of these properties can either be evaluated from the initial state (i.e. the system is fully functional), or by *setting evidence* (i.e certain components have failed already).

DFTCalc fruitfully exploits the technique of *compositional aggregation,* see Fig. 6. Whereas traditional FTA methods translate a DFT into a large and monolithic CTMC, we do this in a stepwise fashion: First, DFTCalc translates each element (i.e., gate or BE) into an input-output interactive Markov chain (I/O-IMC), implementing the methodology from [6,7]. Then, we obtain the underlying CTMC by composing all I/O-IMCs. We compose these I/O-IMCs one-by-one, and employ aggressive state space compression technique in each step, to keep the state space minimal. This compositional approach has four major advantages:

- *Increased modelling power.* Compared to earlier DFT tools, DFTCalc's input language is more powerful and imposes fewer syntactic restrictions: DFTCalc allows any DFT to be a spare component or a trigger, and not only a BE, as in [16]. This is a big advantage in practice, since spare components and triggers are often complete subsystems.
- *Increased analytical power.* SMC enables DFTCalc to analyse a wide range of dependability metrics, namely those expressed in a large subset of the logic



compute mean time to failure for CAS with evidence setting failed MS

DFT:

```
toplevel "SYSTEM";
"SYSTEM" or "FDEP" "CPU" "MOTOR" "PUMPS";
"FDEP" fdep "TRIGGER" "P" "B";
"TRIGGER" or "CS" "SS";
"CPU" wsp "P" "B";
"MOTOR" or "SWITCH" "MOTORS";
"SWITCH" pand "MS" "MA";
```

○ Compute unreliability in interval [0,T], for mission times T (T>0), given as

○ list of values

○ range, from: to: step:

Model checker: ○ MRMC ○ IMCA

○ Compute unreliability in interval [T1,T2]

T1: T2:

● Compute MTTF (for plot: to: 40000 step: 5)

Evidence: MS

Error bound: E-4 Prob: min Time: as Prob Verbosity: off □ Coloured output ☑ No pointmarks

Data set name: CAS failed MS

Stored data sets (to be used for combined plot):

☑ dataset 0: CAS

☑ dataset 1: CAS failed MS

Fig. 3. DFTCalc web-tool interface.

CSL. Also, as argued in [6], certain DFTs give rise to non-determinism. If so, the I/O-IMC leads to a continuous time Markov decision process (CTMDP).

- *Efficiency.* The compositional aggregation technique leads to significant speed ups of several orders of magnitude.
- *Flexibility.* The compositional aggregation approach makes the framework very extendable. In order to change the behaviour of a gate or even add new gate types, we only need to provide the underlying I/O-IMC model.

Web Interface. DFTCalc can be used by downloading a stand-alone version, and via a web interface. Both are accessible at `http://fmt.cs.utwente.nl/tools/dftcalc/`. DFTCalc is open source, but requires a license for CADP, which is free for academic institutions. The web interface extends the downloadable version with a GUI as well as the plot function and is shown in Fig. 3. It allows the user to (1) input DFT models via a text screen, the topmost box in Fig. 3; (2) select the dependability metrics. This can be (a) the reliability for one or more mission times x, or (b) the probability on a system failure during an interval $[T1, T2]$, or (c) the mean time to failure; (3) set various options: which model checker to use; the error bound, the level of verbosity, and whether to color output. The results can be given either by numbers, via the button *show result*, or as a plot, via the button *plot result*. The input and configuration of the web interface can be saved via the button *permalink*.

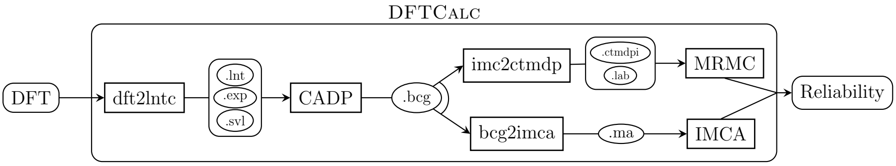
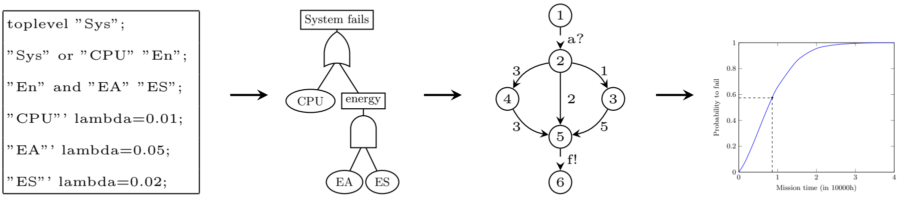


Fig. 4. The DFTCalc tool-chain.



```
toplevel "Sys";
"Sys" or "CPU" "En";
"En" and "EA" "ES";
"CPU"' lambda=0.01;
"EA"' lambda=0.05;
"ES"' lambda=0.02;
```

Fig. 5. Graphical overview of the processing steps in DFTCalc.

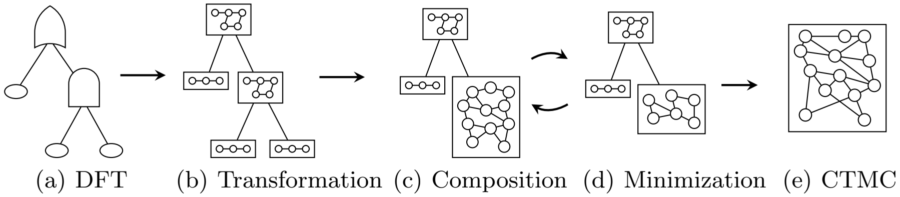


Fig. 6. Graphical overview of the compositional aggregation of DFT models.

3 DFTCalc's internal structure

Architecture. DFTCalc combines dedicated code and state-of-the-art model checkers. The architecture is displayed in Fig. 4 and the processing steps in Fig. 5: First, `dft2lntc` translates a DFT in Galileo format into `.lnt` format, a process calculus enriched with data that is input to CADP. Technically, this step transforms each DFT element into an I/O-IMC representing the element's behaviour. Additionally, a `.exp` file is generated that defines the interaction between components. The clear distinction between local component and global system information together with the compositional semantics of I/O-IMCs makes DFTCalc highly flexible: New components can be added or existing components adapted by specifying their behaviour in `.lnt` format and adding them to the tool's library. In the next step, the CADP tool set [12] uses the compositional aggregation method to generate the state space of the system, which is a I/O-IMC representation of the whole DFT. The output of CADP is a `.bcg` file. This format is translated either into a `.ctmdpi` file, which is input to the Markov Reward Model Checker MRMC [15], or into an `.ma` file, which is the input of the Interactive Markov Chain Analyzer IMCA [13]. Finally, the requested dependability metrics are computed.

Compositional aggregation. Compositional aggregation of I/O-IMCs lies at the heart of DFTCalc. As depicted in Fig. 6, after transforming each DFT element into an I/O-IMC, we iteratively compose the obtained I/O-IMCs: We take two I/O-IMCs, compose them, hide all action labels that are no longer needed for synchronisation, and then minimise the composition via bisimulation minimisation. This process continues until a single I/O-IMC remains. The order of the aggregation process heavily influences the number of states in the obtained I/O-IMC, and is determined by a smart heuristic. Compositional aggregation yields reductions up to several orders of magnitude [7].

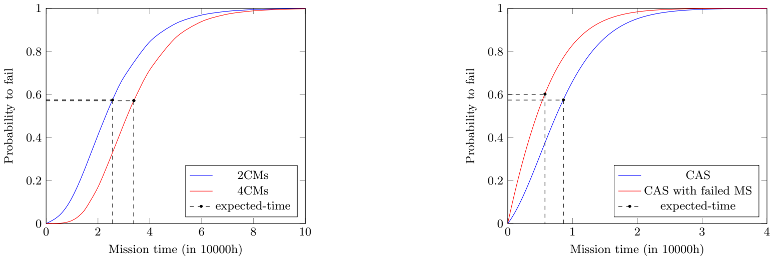
4 Case Studies

We show the applicability of DFTCalc by three case studies: a multiprocessor computing system (MCS) [17,7] which consists of two computing modules (CMs), a bus, a power supply and a spare memory module; the cardiac assist system (CAS) [9,8] from Fig. 2; and a fault-tolerant parallel processor (FTPP) [7] of a redundant computer system consisting of four groups of n processors. The MCS and CAS models were originally developed for discrete time models [17,9], but were analyzed, as we do, for continuous time models in [7,8].

All our experiments were conducted on a single core of a 2.7 GHz Intel Core2Duo processor with 2GB RAM running on Linux. Fig. 7 presents the increasing failure probability over time as well as the expected failure time. Table 1 shows the scalability. We compare *Coral* and DFTCalc: Since DFTCalc is up to three times faster than Coral it also outperforms earlier tools like Galileo [7].

5 Conclusion

We have presented an efficient tool chain which allows to model and analyse DFTs with a number of prominent dependability metrics. The flexible architecture of DFTCalc exploits state-of-the-art techniques to compose, compress and analyse DFTs, and is easily extendable. We have conducted several case studies demonstrating DFTCalc's high performance in the analysis of DFTs.



(a) Failure probability of the MCS over time. (b) Failure probability of the CAS over time.

Fig. 7. Reliability plots for the case studies.

Model	Tool	Time (s)	P(fail)	States	Transitions	Speedup
MCS 2CMs, t=10000	Coral	131.492	0.998963	18	55	1
	DFTCalc	55.395	0.998963	18	55	2.37371
MCS 4CMs, t=10000	Coral	339.752	0.997927	151	992	1
	DFTCalc	201.461	0.997927	151	992	1.68644
CAS, t=10000	Coral	135.155	0.0460314	16	50	1
	DFTCalc	51.267	0.0460314	16	50	2.64794
FTPP-4 , t=1	Coral	491.114	0.0192186	142	923	1
	DFTCalc	234.905	0.0192186	72	386	2.09069
FTPP-5, t=1	Coral	730.761	0.0030616	2167	27438	1
	DFTCalc	603.630	0.0030616	400	3369	1.21061

Table 1. Results of the case studies.

As future work, we aim to include cost structures and repairable basic events. Moreover, we will use DFTCalc's flexible architecture to implement additional gates to broaden DFTCalc to other formalisms like attack trees.

Acknowledgements. This research has been partially funded by the NWO under the project ArRangeer (12238), and by the DFG/NWO bilateral project ROCKS (DN 63-257) and by the EU FP7 under the project TREsPASS (318003).

References

1. F. Arnold, A. Belinfante, F. Van der Berg, D. Guck, and M. Stoelinga. Dftcalc: a tool for efficient fault tree analysis (extended version). Technical Report TR-CTIT-13-13, CTIT, University of Twente, Enschede, June 2013.
2. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE TSE*, 29(6):524–541, 2003.
3. C. Baier, H. Hermanns, J.-P. Katoen, and B. R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science*, 345(1):2 – 26, 2005.
4. C. Baier and J.-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.
5. R. E. Barlow and F. Proschan. *Statistical theory of reliability and life testing: probability models.* Holt, Rinehart and Winston, 1975.
6. H. Boudali, P. Crouzen, and M. Stoelinga. Dynamic fault tree analysis using Input/Output interactive Markov chains. In *DSN*, pages 708–717, 2007.
7. H. Boudali, P. Crouzen, and M. Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE TDSC*, 7:128–143, 2010.
8. H. Boudali and J. Dugan. A continuous-time bayesian network reliability modeling and analysis framework. *IEEE transactions on reliability*, 55(1):86–97, 2006.
9. H. Boudali and J. B. Dugan. A Bayesian network reliability modeling and analysis framework. *IEEE Transactions on Reliability*, 55:86–97, 2005.
10. H. Boudali, A. P. Nijmeijer, and M. Stoelinga. DFTSim: A simulation tool for extended dynamic fault trees. In *ANSS 2009*, page 31, 2009.
11. D. Coppit and K. Sullivan. Galileo: a tool built from mass-market applications. In *International Conference on Software Engineering*, pages 750–753, 2000.
12. H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: A toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, pages 1–19, 2012.
13. D. Guck, T. Han, J.-P. Katoen, and M. Neuhausser. Quantitative timed analysis of interactive Markov chains. In *NFM*, volume 7226 of *LNCS*, pages 8–23, 2012.
14. Isograph. Fault Tree +. `www.isograph-software.com/2011/software/`.
15. J.-P. Katoen, I. Zapreev, E. M. Hahn, H. Hermanns, and D. Jansen. The ins and outs of the probabilistic model checker MRMC. *Perf. Eval.*, 68(2):90–104, 2011.
16. R. Manian, J. Bechta Dugan, D. Coppit, and K. Sullivan. Combining various solution techniques for dynamic fault tree analysis of computer systems. In *Proc. IEEE Int. High-Assurance Systems Engineering Symposium*, pages 21–28, 1998.
17. S. Montani, L. Portinale, A. Bobbio, M. Varesio, and D. Codetta-Raiteri. A tool for automatically translating dynamic fault trees into dynamic Bayesian networks. In *RAMS*, pages 434–441, 2006.
18. PTC. Windchill FTA. `http://www.ptc.com/product/relex/fault-tree`.
19. W. E. Veseley, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook, NUREG-0492. Technical report, NASA, 1981.