DEDUCTIVE CAUSE-CONSEQUENCE ANALYSIS (DCCA)

*Lehrstuhl für Softwaretechnik und Programmiersprachen,
Universität Augsburg, D-86135 Augsburg
{ortmeier, reif, schellhorn}@informatik.uni-augsburg.de*

Frank Ortmeier, Wolfgang Reif and Gerhard Schellhorn

Abstract: In this paper we present a new form of formal safety analysis: deductive cause-consequence analysis (DCCA). Deductive Cause-Consequence Analysis is a way to use formal methods for safety analysis. It substitutes error-prone informal reasoning by mathematical proofs. DCCA allows to rigorously prove whether a failure on component level is the cause for system failure or not. DCCA is a formal generalization of the two most common safety analysis techniques: failure modes and effects analysis (FMEA) and fault tree analysis (FTA).
We apply the method to a real world case study: the height control in the Elbe-tunnel in Hamburg. This shows how formal safety analysis with DCCA helps identifying design flaws and weaknesses in a real-world industrial system. *Copyright ©2005 IFAC*

Keywords: formal methods, safety critical system, safety analysis, failure modes and effects analysis, fault tree analysis, dependability

1. INTRODUCTION

The central question of safety analysis is to determine what components of a safety-critical system must fail to allow the system to cause damage. Most safety analysis techniques rely only on informal reasoning techniques which depend heavily on the skill and knowledge of the safety engineer. Some of these techniques have been formalized.

In this paper we present a new safety analysis technique: Deductive Cause-Consequence Analysis (DCCA). This technique is a generalization of well-known safety analysis methods like FMEA [McDermott et al. (1996)][IEC (1998)], FMECA [ECSS (2001)]and FTA [Vesley et al. (2002)]. The logical framework of DCCA may be used to rigorously verify the results of informal safety analysis techniques. It is also strictly more expressive (in terms of what can by analyzed) than traditional FMEA. We show, that the results of DCCA have the same semantics as those of formal FTA [Schellhorn et al. (2002)]. Because of this DCCA may be used to verify fault trees without formalizing the inner nodes of the tree.

In Sect. 2 the semantics of DCCA is presented. A comparison to FMEA and FTA is done in Sect. 3. We illustrate the technique on a real world case study: the height control system of the Elbe-tunnel in Hamburg (see Sect. 4). In Sect. 5 some related approaches are discussed and Sect. 6 summarizes the results and concludes the paper.

2. DCCA

In this section we describe the formal semantics of DCCA. The formalization is done with Computational Tree Logic(CTL) [Emerson (1990)]. We use finite automata as system models. The use of CTL and finite automata allows to use powerful model checkers like SMV [McMillan (1990)] to verify the proof obligations.

In the following we assume that a list of hazards on system level and a list of possible basic component failures modes is given. Both data may be collected by other safety analysis techniques like failure-sensitive specification [Ortmeier and Reif (2004a)] or HazOp [Kletz (1986)]. We assume that system hazards H and primary failure δ are described by predicate logic formula. This is true for most practical problems. We call the set of all failure predicates Δ.

Now, we define a temporal logic property which says, whether a certain combination of failures may lead to the hazard or not. We call this property *criticality* of a set of failure modes.

Definition 1. critical set / minimal critical set
For a system SYS and a set of failure modes Δ a subset of component failures $\Gamma \subseteq \Delta$ is called critical for a system hazard, which is described by a predicate logic formula H if

$$SYS \models \mathbf{E}(\overline{\lambda}\ \mathbf{until}\ H) \ \ where\ \ \overline{\lambda} := \bigwedge_{\delta \in (\Delta \setminus \Gamma)} \neg\, \delta$$

We call Γ a minimal critical set if Γ is critical and no proper subset of Γ is critical.

Here, $\mathbf{E}(\varphi\ \mathbf{until}\ \psi)$ denotes the existential CTL-UNTIL-operator. It means there exists a path in the model, such that φ holds until the property ψ holds. The property *critical set* translates into natural language as follows: there exists a path such that the system hazard occurs without the previous occurrence of any failures except those which are in the critical set. In other words this means, it is possible that the systems fails, if only the component failures in the critical set occur. Intuitively, criticality is not sufficient to define a cause-consequence relationship. It is possible, that a critical set comprises failure modes, which have nothing to do with the hazard.

Therefore, the notion *minimal critical set* also requires, that no proper subset of it is critical. So minimal critical sets really describe what we would expect for a cause-consequence relationship in safety analysis to hold: firstly the causes may - but not necessarily - lead to the consequence and secondly the causes are necessary to allow the consequence to happen. So the goal of DCCA is to find minimal critical sets of failure modes. Testing all sets by brute force would require an effort exponential in the number of failure modes.

However, DCCA may be used to formally verify the results of informal safety analysis techniques. This reduces the effort of DCCA a lot, because the informal techniques often yield good "initial guesses" for solutions. Note, that the property critical is monotone with respect to set inclusion i.e. $\forall \Gamma_1, \Gamma_2 \subseteq \Delta : \Gamma_1 \subseteq \Gamma_2 \Rightarrow (\Gamma_1$ is critical set $\Rightarrow$ Γ_2 is critical set). This helps to reduce proof efforts a lot.

3. COMPARISON TO OTHER SAFETY ANALYSIS METHODS

We can now identify different cases according to the number of elements in the set of failure modes which is analyzed and relate them to other existing safety analysis techniques.

$|\Gamma| = 0$
If the empty set of failure modes is examined, then the proof obligation of minimal criticality corresponds to the verification of functional incorrectness. Minimality is of course satisfied (the empty set does not have real subsets). The property of criticality states, that there "exists a path where no component fails but eventually the hazard occurs" (in CTL: $EF\ H$). This is the negation of the standard property of functional correctness "on all paths where no component fails, the hazard will globally not occur" (in CTL: $AG\ \neg H$). In other words, if the empty set can be proven to be a critical set, then the system has design errors and is functionally incorrect.

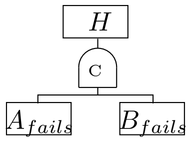
$|\Gamma| = 1$
The analysis of single failure modes corresponds to traditional FMEA. Traditional FMEA analyzes the effects of a component failure mode on the total system in an informal manner. If the failure modes appears to be safety critical than this cause-consequence relationship is noted as one row of a (FMEA) spreadsheet. If a singleton set is minimal critical for a hazard H, then a correct FMEA must list the hazard H as effect of the analyzed failure mode. Note, that functional correctness is a pre-condition for formal FMEA. If the system is not functionally correct, then there will be no singleton sets of failure modes which are minimal critical.

$|\Gamma| > 1$
This is a true improvement to FMEA. Combinations of component failure modes are traditionally only examined by FTA. FTA analyzes top-down the reasons of system failure. Cause and consequence are linked by certain *gates*. The *gates* of a fault tree state if all causes (AND-gate c) or any of the causes (OR-gate c) are necessary to allow the consequence to occur. Iteration builds a tree like structure where the root is the system hazard and the leaves are component failure.

The result of FTA is a set of so called *minimal cut sets*. These sets may be generated automatically from the structure of the tree. Each *minimal cut set* describes a set of failure modes, which together may make the hazard happen. This corresponds to the definition of *minimal critical sets* obtained by

DCCA. So FTA may be seen as a special case of DCCA. An introduction to FTA may be found in [Vesley et al. (2002)].

FTA has been enhanced with formal semantics. Formal FTA allows to decide whether failure modes have been forgotten or not. The idea is here to assign a temporal logic formula to each gate. If this formula is proven correct for the system, then the gate is *complete*. This means no causes have been forgotten. An example is given in figure 1.



A $((A_{fails} \wedge B_{fails})\ P$ H)

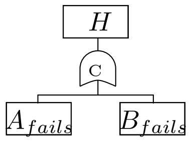
Fig. 1. Fault tree AND-gate and formalization

The figure shows a synchronous cause-consequence AND-gate. The semantics is that both reasons – component A fails and component B fails – must occur simultaneously , before the hazard may occur. Here, $A(\varphi\ P\ \psi)$ denotes the derived CTL-operator *PRECEDES*, which is defined as $\neg E(\neg\varphi\ until\ (\psi \wedge \neg\varphi))$. Informally *PRECEDES* means, that whenever ψ holds, φ must have happened before. Altogether formal FTA distinguishes 7 different types of gates, which reflect temporal ordering, environment constraints and synchronous vs. asynchronous dependencies between cause and consequences. A detailed description of formal FTA may be found in [Thums and Schellhorn (2003)]. One of the main results of FTA is the minimal cut set theorem. This theorem states, that for a complete fault tree the prevention of only one failure mode of every minimal cut set, assures that the system hazard will never occur. A fault tree is called complete, if all its gates have been proven to be complete.

DCCA may be used to verify the completeness of a fault tree analysis as well. To apply DCCA to FTA we must first introduce the notion of a complete DCCA. We call a DCCA complete if all minimal critical sets have been identified. If a DCCA has been shown to be complete, then it is proven, that the minimal critical sets of the DCCA have the same meaning as the minimal cut sets of a fault tree done with formal FTA. In particular the following theorem holds:

Theorem 1. Minimal critical sets
For a complete DCCA prevention of one element of every minimal critical set will prevent the hazard H from occurring.

This is the same property that holds for a formal fault tree analysis with the semantics of [Thums (2004)]. However there is a difference as DCCA is more precise. Formal FTA may yield weaker cut sets than DCCA. As a simple example take a system with only one minimal cut set $\{A, B\}$ (for e.g. A_{fails} and B_{fails} are two redundant units). An intuitively correct fault tree is shown in Fig. 1. But formal FTA may miss the opportunity to to find this cut set. Instead a fault tree consisting of a single OR-gate can be proven complete (see Fig. 2) and thus formal FTA yields two singleton minimal cut sets $\{A_{fails}\}$ and $\{B_{fails}\}$. This may



A $((A_{fails} \vee B_{fails})\ P$ H)

Fig. 2. Incorrect fault tree

make a safe system appear unsafe. The reason is that the formula of Fig. 2 implies the on of Fig. 1. The underlying problem is that formal FTA relies on linear time logics. DCCA is defined in a branching time logic. This allows to distinguish both cases and yields more precise results than formal FTA. A second advantage is that DCCA does not require inner nodes to be formalized. This is a big advantage in practical applications. Inner nodes are often very hard to formalize. For example an inner node of the fault tree of the example of Sect. 4 is “Timer is started too early”. Since “too early” refers to the past, this is not directly expressible in CTL. This problem was experienced in the case study “Elbe-tunnel” a lot and was one of the reasons which triggered the development of DCCA.

A problem of showing completeness of DCCA is of course the exponential growth of the number of proof obligations. However, only big minimal critical sets will result in a lot of proof effort. In many real applications minimal critical sets are rather small. In addition, informal safety analysis helps to find candidates for minimal cut sets in advance. FTA is one possibility, FMEA is another. This reduces the combinatorial effort of checking all possible sets of failure modes a lot. Finally, monotony of the property *critical* may be exploited; if e.g. a singleton set is minimal critical, then other minimal critical sets will not contain this element.

4. APPLICATION

As an example for the application of DCCA we present an analysis of the height control system of the Elbe-tunnel in Hamburg. For formal modeling we used finite automata. The proofs were done using the SMV model checker [McMillan (1990)].

The Elbe-tunnel is a road tunnel which connects the harbor with the city of Hamburg. The old tunnel consisted of three tubes with two lanes each. This tunnel has been enhanced in late 2002 with a new fourth tube. The tunnel comprises a very complex control system which contains traffic engineering aspects like dynamic route control, locking of tunnel tubes, etc. We will consider only a small part of the whole system, the height control.

The new tube has been built larger than the old tubes. This allows overhigh vehicles carrying goods from the harbor to use the tunnel to reach the city. The height control system must assure, that such vehicles may only enter the correct tube.

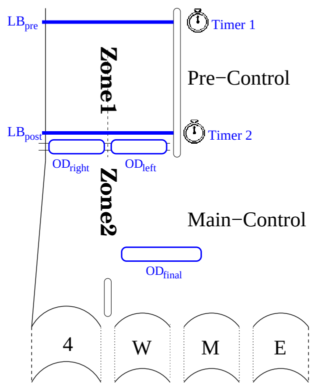

LB_{pre}
Timer 1
Zone1
Pre−Control
LB_{post}
Timer 2
OD_{right}
OD_{left}
Zone2
Main−Control
OD_{final}
4
W
M
E

Fig. 3. Layout of the northern tunnel entrance

In the following, we will distinguish between *high vehicles* (HVs), which may drive through all tubes and *overhigh vehicles* (OHVs), which can only drive through the new, fourth tube. Figure 3 sketches the layout of the tunnel. The fourth tube may be cruised from north to south and the east-tube from south to north only. We focus our analysis on the northern entrance, because OHVs may only drive from north to south. The driving direction on each of the four lanes of the mid- and west-tube can be switched, depending on the traffic situation. Flexible barriers, signals and road fires guide drivers to the tubes, which are open in their direction.

The system uses two different types of sensors. Light barriers (LB) are scanning all lanes of one direction to detect, if an OHV passes. For technical reasons they cannot be installed in such a way, that they supervise only one lane. Therefore overhead detectors (OD) are necessary to detect, on which lane a HV passes. ODs can distinguish vehicles (e.g. cars) from high vehicles (e.g. buses, trucks), but not HVs from OHVs (but light barriers can!). If the height control detects an OHV heading towards a different than the fourth tube, then an emergency stop is signaled, locking the tunnel entrance.

The idea of the height control is, that the detection starts, if an OHV drives through the light barrier LB_{pre}. To prevent unnecessary alarms through faulty triggering of LB_{pre}, the detection will be switched off after expiration of a timer (30 minutes). Road traffic regulations require, that after LB_{pre} both HVs and OHVs have to drive on the right lane through tunnel 4. If nevertheless an OHV drives on the left lane towards the west-tube, detected trough the combination of LB_{post} and OD_{left}, an emergency stop is triggered. If the OHV drives on the right lane through LB_{post}, it is still possible for the driver to switch to the left lanes and drive to the west- or mid-tube. To detect this situation, the height control uses the OD_{final} detector. To minimize undesired alarms (remember, that normal HVs may also trigger the ODs), a second timer will switch off detection at OD_{final} after 30 minutes. For safe operation it is necessary, that after the location of OD_{final} it is impossible to switch lanes. Infrequently, more than one OHV drives on the route. Therefore the height control keeps track of several but at most three OHVs. A formal specification of this system using finite automata may be found in [Ortmeier et al. (2003)].

4.1 *Primary failure and hazards*

There are two different, interesting hazards for the Elbtunnel height control which we analyzed - the collision (H_{Col}) of an OHV with the tunnel entrance and the tripping of a false alarm (H_{FA}). There exists a variety of failure modes. Besides the obvious detector errors, another error comes into play. An OHV may need more than the upper bound of 30 minutes to travel through one of the zones. This may be caused by a traffic jam. This is not a failure in the traditional sense, but rather an unexpected bad influence from the environment. However, from a logical point of view it may be treated in the same way as a (hardware) component failure. Finally, the misbehavior of high vehicles must be taken into account.

The component failures may be divided into four different types: (i)false detection (FD) - The sensor *does* indicate a vehicle, although there is *none*. Possible for all sensors. (ii) miss detection (MD) - The sensor *does not* indicate a vehicle, although there is *one*. Only possible for OD-type sensors. (iii) overtime (OT) - Actual driving time of an OHV exceeds the runtime of a timer. Possible for zone 1 and zone 2. (iv) high vehicles (HV) - A high vehicle beneath an overhead detector is interpreted as an OHV. We write FD_{final} as abbreviation for false detection at overhead detector OD_{final} and analogously for all other sensors. Overtime failures can occur in zone 1 and zone 2. We write OT_1 resp. OT_2. Note, that the last item (HV) is not a failure in the traditional sense, as overhead detectors can not distinguish between high vehicles and OHVs, high vehicles at the location of the sensors are (incorrectly) interpreted as OHVs. For the control system this has the same effect as a FD of the sensor. Traffic regulations require high vehicle to drive on the right lane. Because of this we introduce HV-type error only for OD_{left} and OD_{final}. High vehicles at OD_{right} are of course modeled (and may trigger the detector), but are not a failure as they are part of the expected working environment of the system. So we get a set of hazards $H = \{H_{Col}, H_{FA}\}$ and a set of component failure modes $\Delta = \{FD_{right}, MD_{right}, FD_{left}, MD_{left}, HV_{left}, FD_{final}, MD_{final}, HV_{final}, OT_1, OT_2, FD_{pre}, FD_{post}\}$.

For any kind of formal safety analysis failure modes must be integrated in the system model. The failures are integrated into the formal model by adding parallel automata and modifying transition conditions. The automata describe the occurrences or absence of the failure mode. Figure 4 show two such failure automata.

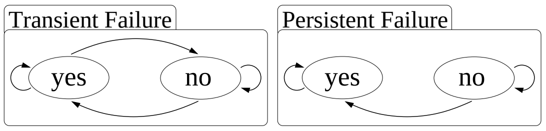


Fig. 4. Failure automata for transient and persistent failures

The left automaton models a transient failure which can indeterministically occur and disappear. The right one models a persistent failure. In the example false detection of the first light barrier FD$_{pre}$ is a transient failure (e.g. a passing bird) while false detection at OD$_{left}$ is normally a persistent failure (e.g. e.g. rainy weather). Similar automata may be constructed to reflect maintenance intervals, automatic repair, etc. The predicates δ describing the failure modes can then be expressed as *Failure automaton*$_x$ = *yes*.

In a second step the direct effect of this failure mode must be implemented into the formal system model. This is usually done by adding new transitions which can be taken only if the condition *Failure automaton*$_x$ = *yes* holds. The complete system is then the (synchronous) parallel composition of the old system and all failure automata

4.2 *DCCA*

When we began with the verification of DCCA proof obligations we experienced, that even the empty set a failure modes is critical for the hazard collision. As described in section 2 this corresponds to functional incorrectness. Indeed the presented system was functionally incorrect. The critical scenario may occur, if two overhigh vehicles pass the first light barrier simultaneously and travel a very different speeds. A more detailed description of this problem may be found in [Ortmeier et al. (2003)] where we describe formal verification of this system in detail.

To go on with the analysis, we marked the simultaneous passing of overhigh vehicles through the first light barrier as an error. This is justified, as overhigh vehicles are required to drive on the right anyway. We label this error OHV_{sim}. Now we could prove, that the empty set is not critical and thus, that the system is functionally correct. For singleton sets we could prove that only the failures MD_{final}, OT_1, OT_2, FD_{post} and OHV_{sim} are critical. All minimal critical sets with more than one element may not include these five failure modes. Therefore if there exist any such sets they must be composed only of elements of the other failure modes. Instead of analyzing in a naive way all possible combinations we try to rule them out all at once. We start with the set of all other possible failure modes. The proof shows that the set $\{FD_{right}$, MD_{right}, FD_{left}, MD_{left}, HV_{left}, FD_{final}, HV_{final}, $FD_{pre}\}$ is not critical. Because of the monotony of the critical set property no other (minimal) critical sets of failure modes exist for the hazard collision.

For the hazard false alarm the same analysis yielded a lot of critical singleton sets. We found the following sets to be minimal critical: $\{MD_{right}\}$, $\{FD_{left}\}$, $\{HV_{left}\}$, $\{FD_{final}\}$, $\{HV_{final}\}$ and $\{FD_{post}\}$. With a similar proof strategy as before we could show that these are the only minimal critical sets. This result is surprising. A short glance at the control system seems to show that redundancy has been introduced to avoid false alarms: the control has been split into three stages where each stage activates the next stage and deactivates it after some time. So one would expect that at least for the hazard false alarm multiple points of failure would be necessary (for example false detections of both light barriers within a certain time interval seem to be necessary for triggering of a false alarm). But this is not the case. The reason is hidden in the design of the system. A false detection of *both* light barriers can be "simulated" by a *correct* driving overhigh vehicle. Therefore all the redundancy does not help in improving the system. As a matter of fact this (previously undiscovered design flaw) also resulted in bad quantitative approximations. A quantitative analysis of the system which takes correct driving overhigh vehicles into account is presented in [Ortmeier and Reif (2004b)].

In conclusion the example shows that DCCA uncovered design flaws that might not be discovered with normal FMEA or FTA. The use of monotony of critical sets helps a lot in reducing the proof effort. In the example a complete DCCA done with naive methodology would have required 2^{13} different proofs. Monotony reduced this to 18 proofs which were proven automatically by SMV in a few minutes.

5. RELATED WORK

There exist some other methods of formally verifying dependencies between component failures and system failure modes. One such technique is formal FTA [Thums (2004)]. Formal FTA requires, that all inner nodes of a fault tree are formalized. This can be very time consuming and difficult (see the example in Sect. 4). A second problem with formal FTA is, that it relies on universal theorems. But, proof obligations for gates must be universal, since only universal properties can transitively lead to properties for the whole system. DCCA uses existential proof obligations. This allows to distinguish whether an (failure) event is a necessary or sufficient condition. Please note, that it is not the goal of DCCA to actually "find/discover" failure modes. Failure modes are assumed to be known. This is not a restriction in practical application, as failure modes must be *explicitly* integrated into the formal model. There also exist complementary methods – like failure-sensitive specification [Ortmeier and Reif (2004a)] – which allow to systematically find possible failure modes of components. Another related approach has been developed in the ESACS project [Bieber et al. (2002)]. Here again model checking and FTA is used as basis. The ESACS approach does not require inner nodes of the fault tree to be formalized. However, the approach still relies on linear temporal logics and thus universal theorems.

6. CONCLUSION

We presented a general formal safety analysis technique: DCCA. DCCA is a generalization of the most widely spread safety analysis techniques: FMEA and FTA. In the formal world, verification of functional correctness, formal FMEA and formal FTA may be found as special cases of DCCA. So DCCA may be used to verify different types of safety analysis techniques in a standardized way. The proof obligations of DCCA may be constructed automatically and the proofs can be done - for finite state systems - by model checking. DCCA formalization is strictly more precise than other formal formal safety analysis techniques like formal FTA. Theoretically, the effort for DCCA grows exponentially. But we have not found this case to happen in real world applications. The costs are more likely to grow linear (for non redundant systems) or polynomial by n (for systems with n-times redundancy), if monotony is used. We showed the application of DCCA to a real world case study: the Elbe-Tunnel in Hamburg. DCCA has shown a design error and has rigorously proven which sets of failure modes are critical for a hazard and which are not.

References

P. Bieber, C. Castel, and C. Seguin. Combination of fault tree analysis and model checking for safety assessment of complex systems. In *Dependable Computing EDCC-4: 4th European Dependable Computing Conference*, volume 2485 of *LNCS*, Springer-Verlag.

ECSS. Failure modes, effects and criticality analysis (FMECA). In European Cooperation for Space Standardization, editor, *Space Product Assurance*. ESA Publications, 2001.

E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

IEC. *IEC 61508-7 - Functional safety of electrical/electronic/programmable electronic safety systems - Part 7: Overview of techniques and measures*. International Electrotechnical Commission, 1998.

T. A. Kletz. Hazop and HAZAN notes on the identification and assessment of hazards. Technical report, The Institution of Chemical Engineers, Rugby, England, 1986.

Robin E. McDermott, Raymond J. Mikulak, and Michael R. Beauregard. *The Basics of FMEA*. Quality Resources, 1996.

K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1990.

F. Ortmeier and W. Reif. Failure-sensitive specification: A formal method for finding failure modes. Technical Report 3, Institut für Informatik, Universität Augsburg, 2004a.

F. Ortmeier and W. Reif. Safety optimization: A combination of fault tree analysis and optimization techniques. Technical Report 5, Institut für Informatik, Universität Augsburg, 2004b.

F. Ortmeier, W. Reif, G. Schellhorn, A. Thums, B. Hering, and H. Trappschuh. Safety analysis of the height control system for the Elbtunnel. *Reliability Engineering and System Safety*, 81 (3):259–268, 2003.

G. Schellhorn, A. Thums, and W. Reif. Formal fault tree semantics. In *Proceedings of The Sixth World Conference on Integrated Design & Process Technology*, Pasadena, CA, 2002.

A. Thums. *Formale Fehlerbaumanalyse*. PhD thesis, Universität Augsburg, Augsburg, Germany, 2004. (in German).

A. Thums and G. Schellhorn. Model checking FTA. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *FME 2003: Formal Methods*, LNCS 2805, pages 739–757. Springer-Verlag, 2003.

Dr. W. Vesley, Dr. Joanne Dugan, J. Fragole, J. Minarik II, and J. Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, Washington DC 20546, August 2002.