STATENS KÄRNKRAFTINSPEKTION
SWEDISH NUCLEAR POWER INSPECTORATE

[redacted]

Postadress - Postaddress	Telegram - Cable	Telex	Telefon - Telephone	Lokaler - Office
Box 27106	SWEDATC , Stockholm	119 51 SWEATOM S	08-63 55 60	Sehlstedtsgatan 11
102 52 STOCKHOLM		Telefax		
		08-61 90 86		

EDISH NUCLEAR POWER INSPECTORATE
stem and Reliability Analysis Branch

PSE01-81

A COMPENDIUM OF COMPUTER CODES IN FAULT TREE ANALYSIS

Bengt Lydell
March 1981

SKI - PSE - 01 - 81

N O T I C E

This report is a revised version of report RE01-79 published by Department of Energy Conversion, Chalmers University of Technology.

FOREWORD

In the past ten years principles and methods for a unified system reliability and safety analysis have been developed. Fault tree techniques serve as a central feature of unified system analysis, and there exists a specific discipline within system reliability concerned with the theoretical aspects of fault tree evaluation.

Ever since the fault tree concept was established, computer codes have been developed for qualitative and quantitative analyses. In particular the presentation of the kinetic tree theory and the PREP-KITT code package has influenced the present use of fault trees and the development of new computer codes.

This report is a compilation of some of the better known fault tree codes in use in system reliability. Numerous codes are available and new codes are continously being developed. The report is designed to address the specific characteristics of each code listed. A review of the theoretical aspects of fault tree evaluation is presented in an introductory chapter, the purpose of which is to give a framework for the validity of the different codes.

Please note that the code summaries are -- with a few exceptions -- based on the manuals as given in the references. Some of the codes have been revised at later dates. No pretentions as to the completeness of this compendium are given.

Stockholm in March 1981

Bengt Lydell

TABLE OF CONTENTS

ABBREVIATIONS AND NOTATION

ALLCUTS	determination of all cut sets
ANSI	American National Standards Institute
BAM	Boolean Algebra Minimization
BICS	Boolean Indicated Cut Sets
CAT	Computed Automated Tree
COMCAN	Common Cause Analysis
DIN	Deutsche Industrienorme
DRAFT	Drawing of Fault Trees
ELRAFT	Effecient Logic Reduction Analysis of Fault Trees
FATRAM	Falt Tree Reduction Algorithm
FAUNET	Fault Trees and Network
IEEE	Institute of Electrical and Electronics Engineers
KITT	Kinetic Tree Theory
MOCUS	Method for Obtaining Cut Sets
NEA	Nuclear Energy Agency
PL-MOD	PL/1 code for the Modularization of fault trees
PREP	Preprocessor
RAP	Reliability Analysis Package
RAS	Reliability Analysis Ssystem
SALP-MP	System Analysis by List Processing - Multi Phase
SETS	Set Equation Transformation System
UCLA	University of California, Los Angeles

t	time
x_i	binary indicator variable
λ	failure rate
μ	repair rate
$\phi(x)$	structure function

τ	mean-time-to-repair
$\bar{\tau}$	median mean-time-to-repair

1. FAULT TREE ANALYSIS -- A TOOL IN RELIABILITY AND SAFETY ENGINEERING

The general principles of fault tree analysis are well established. There now exist standards where certain procedures of the techniques involved are documented [1,2]. Research and development efforts during the past two decades have produced the tools -- i.e. computer codes -- to effectively perform reliability analyses of complex systems. However, the validity of fault tree analyses has received comparatively little attention.

The present application of fault tree techniques are mainly related to four areas of system analysis: (1) Qualitative analyses -- identification of potential component failures leading to the failure of a system, (2) quantitative analyses -- calculation of system failure probability (system unavailability), (3) identification of dependencies in a system, and (4) generic failure analysis.

It is to be noted that fault tree analyses in many cases serve as input to specific reliability evaluations such as those concerned with periodic testing and repair strategies. A number of reliability codes are designed to be compatible with fault tree codes related to the four areas mentioned above.

1.1 Evolution of the Fault Tree technique

During the 1950s the foundation of the statistical reliability theory was established. Of principal

interest at that time was the analysis of multi-component reliability. The papers of Birnbaum, Esary and Saunders [3], and Esary and Proschan [4] introduced the concepts of coherent structures, paths and cuts. These concepts made it possible to clarify the relationships between component reliability and system reliability in a mathematically stringent way.

However, it was the introduction of the fault tree technique in the early 1960s that provided the practical utilization of the coherent structure theory [5]. The concept of fault tree analysis was originated by H.A. Watson of Bell Telephone Laboratories in 1961. The System Safety Symposium in Seattle in 1965 initiated a wide-spread interest in using the fault tree technique as a qualitative and quantitative tool for the evaluation of system reliability characteristics. A classical paper from this era is that of Haasl [6].

Originally the quantitative fault tree evaluations were performed with simulation techniques. In 1970 Vesely [7] presented the foundation of the analytical fault tree evaluation -- kinetic tree theory. Later this theory has been improved [8] to include models for different repair and inspection policies.

In the years following the presentation of the kinetic tree theory, efforts were directed to the development of effecient computer codes for fault tree evaluation. There now exist numerous computer codes for different aspects of fault tree analysis. The fault tree technique has evolved into a state of being an integral part of system analysis.

In general, fault trees have proved to be effective in identifying potential failure events in complex systems. However, there are a number of aspects of the technique that make its validity very sensitive to the way a system analysis is performed.

It is a strength -- and in the same time a possible weakness -- that the quality of a fault tree model is controlled only by the skill and thoroughness of the analyst. A fault tree of any system is a graphical representation of the analyst's knowledge of a specific system. Errors of over-sight and omission can, however, severly distort an analysis.

A quite recent trend in the fault tree technique is the development of systematic, computerized methods of fault tree synthesis [9,10]. The object of this work is to find a procedure for simplifying the often very tedious and time consuming construction phase of a fault tree analysis. The analyst can hereby direct his efforts to the understanding of the system to be evaluated, and thus reducing possibilities of over-sight and omission. The automatic fault tree construction is often designated as generic failure analysis.

The literature on the various aspects of fault tree analysis is extensive. A number of bibliographies covering certain areas of the subject have been published [11,12], and the availability of computer codes is in general unrestricted. The Argonne Code Center in the U.S.A. and the NEA Data Bank in France respectively prepare and publish program abstracts and bibliographies of computer programs.

1.2 Basic Concepts of Fault Tree Evaluation

Basically, fault tree analysis is a systematic approach of representing a structure (system) in terms of its potential failure initiators (primary events). The fault tree theory on which the reviewed computer codes are based is restricted to systems with binary components, i.e. the components can have two states only.[1] Because of this restriction a fault tree can be described as a logic diagram showing the various failure events as they combine through a set of Boolean gate operators leading to the system failure (top event).

The technical systems under consideration in fault tree analysis are defined as coherent structures. In the coherent structure theory the state of a set of n components is indicated by the vector $\mathbf{x} = (x_1, x_2, \ldots\ldots, x_n)$, denoted as a vector of basic event outcomes. A binary structure function $\phi(\mathbf{x})$ is used to determine the state of the system. We have

$$x_i = \begin{cases} 1 \text{ when basic event } i \text{ occurs} \\ 0 \text{ when basic event } i \text{ does not occur} \end{cases}$$

$$\phi(\mathbf{x}) = \begin{cases} 1 \text{ when top event occurs} \\ 0 \text{ when top event does not occur} \end{cases}$$

[1] A generalized fault tree theory has been developed by Caldarola [13]. This theory is not discussed further.

A structure (system) is defined as coherent when [3]

a. $\Phi(\mathbf{x}) \geq \Phi(\mathbf{y})$ whenever $\mathbf{x} \geq \mathbf{y}$
b. $\Phi(1) = 1$, where $1 = (1,1,\ldots\ldots, 1)$
c. $\Phi(0) = 0$, where $0 = (0,0,\ldots\ldots, 0)$

Consequently, a structure is coherent if its structure function is increasing (condition "a") and each component is relevant (condition "b" and "c").

In a fault tree analysis the failure characteristics of any system are described by logic gate operators. The AND and OR gates are the basic logic operators. The structure function representing an AND gate is given by

$$\Phi_{AND}(\mathbf{x}) = x_1 \cdot x_2 \cdot \ldots\ldots \cdot x_n \equiv \prod_{i=1}^{n} x_i \qquad (1)$$

while an OR gate is represented by

$$\begin{aligned} \Phi_{OR}(\mathbf{x}) &= 1 - (1-x_1)(1-x_2)\ldots\ldots(1-x_n) \\ &\equiv \coprod_{i=1}^{n} x_i \end{aligned} \qquad (2)$$

Equation (1) & (2) represent Boolean structure functions that are coherent. Due to a number of reasons

fault tree construction usually is limited to the application of AND and OR logic gate operators. However, in certain situations it is necessary to describe the logical behaviour of a system in a more detailed way. This can be done by using the NOT logic gate operator. A single event x_i operated by a NOT gate is per definition represented by

$$\phi_{NOT}(x_i) = 1 - x_i \qquad (3)$$

Contrary to the AND and OR gates, a NOT gate is not represented by a coherent Boolean structure function. We have

$$\phi_{NOT}(0) = 1 \quad > \quad \phi_{NOT}(1) = 0 \qquad (4)$$

Fault trees with AND, OR and NOT gates are defined as complete fault trees [14], and complete fault trees are incoherent. Processing of such trees is tedious and requires large computer memories. Kumamoto and Henley [15] have presented an approach for the effecient evaluation of incoherent fault trees. Many of the computer codes reviewed in this report are restricted to handling coherent fault trees only.

An <u>exact</u> quantitative fault tree analysis requires the determination of the minimal cut set represen-

tation of a system. Per definition a cut set is a group of primary events whose occurence will cause the top event (system failure) to occur. A cut set is minimal if it cannot be further reduced and still remains a cut set.

A minimal cut set determination is equal to a qualitative fault tree evaluation. Usually qualitative fault tree evaluations -- fault tree reductions -- are performed by application of the laws of Boolean algebra. For complex tree structures these procedures can be very time consuming. The quantitative analysis of binary fault trees does not require the minimal cut set determination strictly, however. In the case of multinary fault tree structures minimal cut set determination is a requirement for quantification.

1.3 Generic Fault Trees

The concept 'generic fault trees' refers to an approach that reduces the time required to construct detailed system fault trees significantly. In this approach generic logic models are developed for commonly occurring components and systems. The logic models are stored on a computer, and when a need arises specific, overall system fault trees can be constructed.

Any fault tree analysis always involves at least three steps: (1) System definition, (2) construction of a fault tree, and (3) qualitative fault tree evalu-

ation. Until recent years research and development efforts mainly have been directed towards the last step. During the early 1970s general principles for computerized fault tree construction were presented. Fussell [16] and Powers and Tompkins [17] respectively defined two approaches for the automatic generation of fault trees.

There now exist several computer codes for automatic fault tree construction. The experience gained from the application of these codes indicates that their effectiveness very much depends on the way the component failure models (generic logic models) are represented. The different failure models in use are

- Equation models, [9]
- Mini fault tree models, [16]
- Digraph models, [17]
- Input-output models, [18]

The resulting algorithms have been tested on a number of academic sample problems. Further development is needed before the algorithms are properly validated and a widespread use of available computer codes is attempted.

1.4 Limitations in the Fault Tree Evaluation Techniques

In the practical utilization of fault tree evaluation techniques an awareness of the basic assumptions and the limitations is of prime importance. The general assumptions are (1) component independence, (2) no component failures exist at time t = 0, and (3) that the minimal cut sets of a system adequately represent the system.

Whenever a fault tree is drawn, the reliability analyst has to decide on which granularity (level of detail) to be used. A very fine fault tree granularity does not always imply a high degree of accuracy. Furthermore, if the level of detail of a tree is too fine, computer times can be excessive. Hereby making computerized fault tree analysis uneconomical.

There are a number of ways to reduce computer times. In many cases -- but not always! -- the importance of a minimal cut set is a decreasing function of its order n.[1] Therefore, a commonly employed procedure of reducing computer times is to neglect minimal cut sets having order greater than a fixed value $\bar{n}$ -- logical cut off.

A more effecient computer time reduction is achieved with the probabilistic cut off. Here, minimal cut sets having unavailabilities less than a chosen cut off value are discarded. This approach is sometimes being referred to as a "built in sensitivity analysis" [19].

---

[1] The order of a minimal cut set is defined as the number of primary events in a minimal cut set.

The use of the logical cut off and the probabilistic cut off criteria respectively can be more advantageous if, at the same time, it is possible to estimate the importance of the discarded minimal cut sets -- residual error estimation. Estimation of residual error helps the analyst to judge the choice made for logical cut off and probabilistic cut off.

If an analysis is to be extended beyond the manual construction of a fault tree, the analyst has to proceed with great care. Implementation of a computer code can be very time consuming. Besides, there is not always an assurance that a specific code can handle any system. Even if a computer code is at hand the preparation of necessary input data can be deterrent. The choice between manual and automatic fault tree evaluation is not unequivocal!

In summary, the limitations of fault tree analysis fall into two categories: (1) Limitations occurring during implementation, and (2) limitations in theory. Computer programs for qualitative and quantitative evaluations cannot always handle the large trees encountered in practice due to computer storage requirements. Theoretical limitations are related to the independent component assumption, and the binary assumption. Fussell [20] and Rowsome [21] respectively give overviews on limitations in fault tree analysis.

1.5 References

1. IEEE Guide for General Principles of Reliability Analysis of Nuclear Power Generating Station Protection Systems, IEEE Std 352-1975 (ANSI N41.4-1976)

2. Fault tree analysis; method and symbols, DIN 25424, 1977
(in German)

3. Z.W. Birnbaum, J.D. Esary, S.C. Saunders, "Multi-Component Systems and Structures and Their Reliability", Technometrics, Vol 3, 1961, pp 55-77

4. J.D. Esary, F. Proschan, "Coherent Structures of Non-Identical Components", Technometrics, Vol 5, 1963, pp 191-209

5. S.C. Saunders, "Birnbaum's Contribution to Reliability Theory", Proceedings of the Conference on Reliability and Fault Tree Analysis, University of California, Berkeley, September 3-7, 1974 (SIAM 1975)

6. D.F. Haasl, "Advanced Concepts in Fault Tree Analysis", Proc. Systems Safety Symp., Seattle, June 8-9, 1965

7. W.E. Vesely, "A Time Dependent Methodology for Fault Tree Evaluation", Nucl. Eng. Des., Vol 13, pp 337-367

8. L. Caldarola, "Unavailability and Failure Intensity of Components", Nucl. Eng. Des., Vol 44, 1977, pp 147-162

9. F.P. Lees, P.K. Andow, C.P. Murphy, "The Propagation of Faults in Process Plants: A Review of the Basic Event/Fault Information", Reliability Engineering, Vol 1, 1980, pp 149-163

10. P.K. Andow, "Difficulties in Fault-Tree Synthesis for Process Plant", IEEE Trans. Rel., Vol R-29, 1980, pp 2-9

11. J.B. Fussell, G.J. Powers, R.G. Bennetts, "Fault Trees - A State of the Art Discussion", IEEE Trans. Rel., Vol R-23, 1974, pp 51-55

12. B.S. Dhillon, C. Singh, "Bibliography of Literature on Fault Trees", Microelectron. Reliab., Vol 17, 1978, pp 501-503

13. L. Caldarola, Generalized Fault Tree Analysis Combined with State Analysis, KfK 2530, Kernforschungszentrum Karlsruhe, 1980

14. A. Amendola, et al, Analysis of Complete Logical Structures in System Reliability Assessment, EUR 6886 EN, CEC-JRC, Ispra, 1980

15. H. Kumamoto, E.J. Henley, Top Down Algorithm for Obtaining Prime Implicant Sets of Non-Coherent Fault Trees", IEEE Trans. Rel., Vol R-27, 1978, pp 242-249

16. J.B. Fussell, Synthetic Tree Model: A Formal Methodology for Fault Tree Construction, ANCR-1098, Aerojet Nuclear Company, 1973

17. G.J. Powers, F.C. Tompkins, "Fault Tree Synthesis for Chemical Processes", AIChE Journal, Vol 20, 1974. pp 376-387

18. S.L. Salem, J.S. Wu, G. Apostolakis, "Decision Table Development and Application to the Construction of Fault Trees", Nuclear Technology, Vol 42, 1979, pp 51-64

19. A.G. Colombo, G. Volta, "Sensitivity Analysis in Systems Reliability Evaluation", Commission of the European Communities, Joint Research Establishment, Ispra Establishment, Annual Report 1973, EUR-5260e, pp 124-131

20. J.B. Fussell, A Review of Fault Tree Analysis With Emphasis on Limitations, Aerojet Nuclear Company, 1975

21. F.H. Rowsome, III, "How Finely Should Faults be Resolved in Fault Tree Analysis?", Proceedings of American Nuclear Society/Canadian Nuclear Association Joint Meeting, Toronto, June 18, 1976

2. CODE SUMMARIES -- QUALITATIVE FAULT TREE EVALUATION†

†Some of the codes reviewed in this chapter also appear in chapters 3 & 4; i.e. the respective codes also include quantitative evaluation or dependent failure identification options.

2.1 ALLCUTS

The ALLCUTS code package was developed at the Atlantic Richfield Hanford Company in the U.S.A. in the early 1970s for the safety assessment of nuclear fuel cycle operations. The code development was specifically directed towards problems related to evaluation of very large fault trees with high-order minimal cut sets.

2.1.1 Purpose

ALLCUTS is a qualitative evaluation code using already constructed fault trees. The code is capable of finding up to 10-event cut sets from a fault tree with 175 bottom events and 425 logic gates [1, p 4]. The ALLCUTS algorithm is similar to the top down algorithm of the MOCUS code.

In order to make detailed, and in the same time computer time conserving system analyses a so called risk index cut off is employed. The risk index is defined as [1, p 1] "the logarithm of the product of the cut set probability, multiplied by the quantity and type of material released and transported".

In practice, a logical cut off is used to discard cut sets of a certain order from further analysis. A probabilistic cut off is applied on the cut sets surviving the first cut off. The cut sets surviving the second cut off are finally used for calculating the risk index. The result of these procedures is an ord-

ered list of the important cut sets. All the discarded cut sets are retained for a final estimation of total risk.

The desire to handle high-order minimal cut sets is related to the fact that low probability minimal cut sets -- multi event minimal cut sets -- can lead to severe consequences. High probability minimal cut sets do not necessarily imply a high level of risk. A review of various aspects on fault tree analysis in the safety assessment of nuclear fuel cycle operations is presented in [2].

2.1.2 Program Description

ALLCUTS is a Fortran code capable of handling coherent fault trees. The INHIBIT gate -- which is a special case of the AND gate -- can be handled as well.[1] A top-down algorithm is employed in the fault tree evaluation.

The fault tree expansion -- determination of minimal cut sets -- is initiated by defining a starting cut set (usually the top event) and the tree is then systematically searched for bottom events (primary events). At each level of the tree the developing set is analysed and if any event in the developing set is a bottom event, no further development is

---

[1]The INHIBIT logic gate is used in cases where an output is caused by a single input but some qualifying condition must be satisfied before the input can cause the output. Typcally, the INHIBIT logic gate is used in the analyses of chemical process systems.

possible and the event label (specification of the event) is carried with the set until it is fully expanded. Logical gates are treated by the algorithm in the following way

- AND gate; the gate is replaced by the gates and/or bottom events which are input to the AND gate.
- OR gate; the inputs to the gate are used to create cut sets (n inputs make n cut sets). Each cut set have the OR gate of the original set replaced by its first input.
- INHIBIT gate; the gate is replaced by its input event label.

ALLCUTS produces outputs in the form of tables containing bottom event importances and importance factors as well as lists of all the cut sets containing a predefined bottom event. A quantitative evaluation option is included in the code.

2.1.3 References

1. W.J. Van Slyke, D.E. Griffing, ALLCUTS, A Fast, Comprehensive Fault Tree Evaluation Code, ARH-ST-112, Atlantic Richfield Hanford Company, 1975

2. T.H. Smith, et al, A Risk-Based Fault Tree Analysis Method for Identification, Preliminary Evaluation, and Screening of Potential Accidental Release Sequences in Nuclear Fuel Cycle Operations, BNWL-1959, Battelle Pacific Northwest Laboratories, 1976

2.2 ELRAFT

The ELRAFT code was developed at the Westinghouse Electric Corporation and a first paper presenting the code was published in 1971 [1]. It is not known to what extent the code has been used in system reliability analyses.

2.2.1 Purpose

ELRAFT is a qualitative code for the identification of minimal cut sets in coherent fault trees. The code is restricted to finding minimal cut sets containing up to 6 basic events, i.e. $1 \leq n \leq 6$ [2].

2.2.2 Program Description

ELRAFT is a Fortran code using a bottom-up algorithm based on the unique factorization property of natural numbers. Every natural number greater than one can be expressed as a product of powers of prime factors -- a natural number is called prime if it is greater than one and has no factors except itself and one. The expression characterizing a certain natural number is unique, except from the order in which the factors are written.[1]

---

[1] We have for example that $60 = 2^2 \cdot 3 \cdot 5 = 3 \cdot 2^2 \cdot 5 = 5 \cdot 2^2 \cdot 3$, etc

In the ELRAFT algorithm, every basic event is assigned a unique prime number. The fault tree is examined from the bottom-up. The cut sets for the gate events on successively higher levels in the tree are determined as the product of the numbers associated with each of the input events. Note that any fault tree in essence consists of three levels: (1) Top event with its OR gate, (2) cut sets with AND logic gates, and (3) basic events.

Set containment is indicated whenever one number is a factor of another. A cut set indicated by 30 is logically contained in the cut set indicated by 15. Consequently, the cut set indicated by the larger number can be eliminated. Repeated use of this principle finally produces the minimal cut sets for the top event.

In the code the tree top is identified as event number 1. The first event of the next level is identified as 12, etc. The number of the digits in the identification number is equal to the level in the tree to which the event belongs. Within one numbering sequence, both the number of levels and the number of inputs to a gate are limited to nine [2].

A quantitative evaluation option is included in the code. If quantification is required in addition to the minimal cut set identification, probabilities of basic event outcomes have to be assigned to the input cards describing the events.

2.2.3 References

1. S.N. Semanders, "ELRAFT. A Computer Program for the Efficient Logic Reduction Analysis of Fault Trees", IEEE Transactions of Nuclear Science, Vol NS-18, 1971, pp 481-487

2. R.B. Worrell, G.R. Burdick, "Qualitative Analysis in Reliability & Safety Studies", IEEE Trans. Rel., Vol R-25, 1976, pp 164-170

2.3 FATRAM

The development of the FATRAM algorithm [1] was performed at the Idaho National Engineering Laboratory and the algorithm is included in the Reliability Analysis System (RAS) code package [2]. Requirements for effective use of computer core memory initiated the FATRAM development.

2.3.1 Purpose

FATRAM is basically similar to the MOCUS algorithm (see section 2.5). All algorithms for minimal cut set determination have advantages and disadvantages related to the computer implementation. If complex trees are to be analysed external storage devices sometimes must be used. FATRAM is characterized by its effecient use of a main computer memory.

2.3.2 Program Description

FATRAM is included in the RAS package, which can handle coherent fault trees. RAS is an integrated package of computer programs for qualitative and quantitative evaluations as well as dependent failure identification. The algorithms/codes included in RAS are MOCUS, FATRAM, POCUS, KITT-1, and COMCAN.

The steps of the FATRAM top-down algorithm are [1]

- Evaluation begins with the top event. If the gate under the top event label is an AND gate, all inputs are listed as one set; if it is an OR gate, the inputs are listed as separate sets.

- Repeate until all OR gates with gate inputs and all AND gates are resolved. OR gates with only primary event inputs are not resolved at this time.

- Remove any supersets -- cut sets that are not minimal -- that still exist.

- Process any repeated basic events remaining in the unresolved OR gates. Each repeated event is processed as
  a. The repeated event replaces all unresolved gates of which it is an input to form new sets.
  b. These new sets are added to the collection.
  c. This event is removed as an input from the appropriate gates.
  d. Supersets are removed.
  e. Resolve the remaining OR gates. All sets are minimal cut sets.

RAS is controlled by a number of keywords that activate the different algorithms and provide input information to the algorithms. The output of RAS is of three types: (1) printed, (2) punched cards, or (3) a file on a storage device.

2.3.3 References

1. D.M. Rasmuson, N.H. Marshall, "FATRAM - A Core Efficient Cut Set Algorithm", IEEE Trans. Rel., Vol R-27, 1978, pp 250-253

2. D.M. Rasmuson, N.H. Marshall, G.R. Burdick, User's Guide for the Reliability Analysis System (RAS), TREE-1168, EG & G Idaho Inc., 1977

2.4 FAUNET

The FAUNET package [1] was primarily designed for implementation on mini computers. The code development was performed at the Danish Research Establishment Risø in the mid 1970s. At a later date FAUNET has formed a basis for the development of the Reliability Analysis Package (RAP) [2].

2.4.1 Purpose

FAUNET includes a qualitative and a quantitative evaluation module for the analysis of coherent fault trees. The FAUNET algorithm is based on the Fussell-Vesely algorithm for obtaining minimal cut sets [3].[1] A special modularization technique is included to reduce computer time and computer storage requirements.

2.4.2 Program Description

FAUNET is a Fortran code and the qualitative part of the package fall into two main categories: (1) TREE, and (2) CUT. The programs under the TREE label perform various error checks on the input data and provide graphical representations from the input data

[1]The Fussell-Vesely algorithm is often denoted as the Boolean Indicated Cut Sets (BICS) algorithm.

-- fault tree structures are drawn on a line printer. TEECH is the error checking program in the TREE package. It lists the number of OR and AND gates and the number of basic events as well as produces a list of basic events and gates together with the times they occur in the tree. It checks the tree to see if it is connected and to see it there are multiple specifications of a gate. Finally it calculates the number of BICS and the maximum length of these for each gate. The graphical programs consist of a number of modules that structure the gates and events in an orderly way and draw the trees on the line printer.

The CUT package consists of seven programs that make the top-down assessment of the minimal cut sets. CUTUP is the modularization part of CUT and performs a bottom-up analysis by identifying pairs of basic events always occurring in the same types of gate. Such a pair is replaced by a "complex event". If two complex events always appear as a pair in the same type of gate, they are combined into a new complex event. This process continues until no more pairs are found.

2.4.3 References

1. O. Platz, J.V. Jensen, FAUNET: A Program Package for Evaluation of Fault Trees and Network, Risø Report No 348, Research Establishment Risø, 1976

2. E.R. Corran, H.H. Witt, "Reliability Analysis Techniques for the Design Engineer", Proceedings of the 6th Advances in Reliability Technology Symposium, University of Bradford, 9-11 April, 1980

3. J.B. Fussell, W.E. Vesely, "A New Methodology for Obtaining Cut Sets for Fault Trees", Transactions of the American Nuclear Society, Vol 15, 1972, pp 262-263

2.5 MOCUS

The MOCUS user's manual was published in 1974 [1]. One aim with the code development was to replace PREP (see section 2.7) in the PREP-KITT code package with a more efficient algorithm for minimal cut set determination.

2.5.1 Purpose

MOCUS is a program that locates minimal cut sets in coherent fault trees. The MOCUS algorithm is described in [1,2,3]. The cut sets obtained by the algorithm are called Boolean Indicated Cut Sets (BICS). If there is no replication of primary events then the BICS are precisely the minimal cut sets

2.5.2 Program Description

The program is a Fortran code. An already constructed fault tree is required and a description of the tree forms the input. The fault tree description is similar to the input needed for TREBIL -- a program included in the PREP package. Only OR and AND gates can be handled. Other types of logical gates must be described in terms of OR and AND gates.

The MOCUS output always contains (a) an input data edit, (b) the number of minimal cut sets, and (c) punched or printed minimal sets for at least one gate [1, p 18]. The output options are

- Printed minimal cut sets for up to 20 specified logic gates;
- Punched failure data and minimal cut sets for up to 20 specified logic gates.

MOCUS is compatible with the quantitative fault tree evaluation code KITT (see chapter 3). It is possible to define a MOCUS output that contains a cross reference table that correlates a component failure integer name as supplied to KITT to each component alphanumeric name as supplied by the MOCUS user.

2.5.3 References

1. J.B. Fussell, E.B. Henry, N.H. Marshall, MOCUS - A Computer Program to Obtain Minimal Sets from Fault Trees, ANCR-1156, Aerojet Nuclear Company, 1974

2. J.B. Fussell, W.E. Vesely, "A New Methodology for Obtaining Cut Sets for Fault Trees", Transactions of the American Nuclear Society, Vol 15, 1972, pp 262-263

3. R.E. Barlow, H.E. Lambert, "Introduction to Fault Tree Analysis", Proceedings of the Conference on Reliability and Fault Tree Analysis, University of California, Berkeley, September 3-7, 1974, pp 7-35
(SIAM 1975)

2.6 PL-MOD

The program PL-MOD has been developed at the Massachusetts Institute of Technology -- a user's manual was published in 1977 [1]. In comparison with the other codes reviewed PL-MOD is unique in that it is not based on the minimal cut set theorem. Fault trees are described by means of modular decomposition [2] which circumvents the often very tedious minimal cut set determination.

2.6.1 Purpose

The qualitative part of PL-MOD performs a modularization of fault trees -- there is no generation of minimal cut sets! A fault tree module is a group of components acting as a super-component. It is sufficient to know the state of the module in order to determine the overall state of a system. Modular composition is derived directly from fault tree diagrams.

Modularization of fault trees having no replicated events or gates is straightforward. Every intermediate gate for such a fault tree will be the top event for a tree submodule. When replicated events and gates occur i a fault tree, the modularization becomes a more involved procedure. The modularization algorithm uses a Boolean vector form to characterize the modular composition of a fault tree.

2.6.2 Program Description

PL-MOD is a PL-1 code for the evaluation of coherent fault trees. The PL-MOD algorithm is described in [2, pp 545-547]. A number of PL-1 language features are employed in a certain order for the fault tree modularization.

PL-MOD generates a complete Boolean vector representation for the modular minimal cut sets of a fault tree. This representation forms a basis for quantification.

2.6.3 References

1. J. Olmos, M. Modares, L. Wolf, User's Manual for PL-MOD and PL-MODT, Department of Nuclear Engineering, Massachusetts Institute of Technology, 1977

2. J. Olmos, L. Wolf, "A Modular Representation and Analysis of Fault Trees", Nucl. Eng. Des., Vol 48, 1978, pp 531-561

2.7 PREP

PREP -- short for "preprocessor" -- is the qualitative code in the well-known PREP-KITT code package. A user's manual was published in 1970 [1]. PREP-KITT was one of the first fault tree evaluation code available to reliability analysts and consequently it has served as a basis for later code developments.

2.7.1 Purpose

PREP is designed for the qualitative eveluation of coherent fault trees. Minimal cut sets are found either by deterministic testing or by Monte Carlo simulation. According to the code manual fault trees with up to 2000 primary events and up to 2000 logical gates can be handled.

2.7.2 Program Description

PREP is a Fortran code which consists of the programs TREBIL and MINSET. TREBIL produces the logical equivalent of the fault tree in the form of a subroutine -- TREE -- and serves as the input processor for MINSET. The MINSET program determines the minimal cut sets.

Only AND and OR gates can be accepted by TREBIL. Other types of logical gates must be described in terms of AND and OR gates. The fault tree logic building algo-

rithm requires that each logical gate and each component of the tree be given an alphanumeric name. A set of cards is used as input to TREBIL. One card -- containing the name of the gate, the type of the gate, the number and names of the other gates that are attached to it -- is read in for each logical gate in the fault tree.

The MINSET program determines the minimal cut sets by either deterministic testing or Monte Carlo simulation. COMBO is the deterministic testing algorithm for finding minimal cut sets. The Monte Carlo simulation procedure for finding minimal cut sets is performed by the subroutine FATE.[1]

In the deterministic testing procedure primary events are imagined to have occurred first one at a time, then two at a time, etc. For each combination the logic of the fault tree is tested to determine whether or not that combination causes the top event. Combinations that cause the top event represent cut sets. A minimization of cut sets is performed using an element by element comparison to other cut sets. The testing procedure is repeated until minimal cut sets of order n are obtained. The order n is restricted to ten.

In the Monte Carlo simulation procedure the number of minimal cut sets will be found in accordance with their probabilistic importance. The time of failure is computed for each component from the exponential distribution based on an input time called "mission length" (or "mixing parameter").

---

[1]Strictly speaking, when FATE is used PREP is a quantitative code.

The component with the smallest time of failure is first failed and TREE is called. If the system is in a non-failed state, then the component with the next smallest time of failure is failed, and TREE is tested again. This process is repeated until the system fails. When failure occurs, this set of components is reduced to the minimal cut set.

2.7.3 Reference

1. W.E. Vesely, R.A. Narum, PREP and KITT: Computer Codes for the Automatic Evaluation of a Fault Tree, IN-1349, Idaho Nuclear Corporation, 1970

2.8 SALP-MP

The SALP-MP code is an improved version of the original SALP code. SALP-MP has been developed at the Joint Research Centre at Ispra in Italy. The User's manual was published in 1980 [1]. There are certain similarities between SALP-MP and the PATREC code [2,3] developed at Commissariat à l'Energie Atomique in France.

2.8.1 Purpose

The qualitative part of the code package searches for minimal cut sets by a direct manipulation of an already constructed fault tree. A bottom-up algorithm is employed, and a single phase logical analysis as well as a multi-phase analysis can be performed.

A multi-phase system is a system whose structure function changes at fixed times. The structure function changes as a consequence of the different tasks the system is requested to perform in subsequent mission intervals. In the analysis of a multi-phase system as many fault trees as the number of phases are required.

The list processing technique [4] is used for the internal computer representation of fault trees. This technique makes it possible to store and easily handle complex data structures. A preorder or end-

order traverse [1, p 12] determine the minimal cut sets.

2.8.2 Program Description

SALP-MP is a PL/1 code for the qualitative and quantitative evaluation of coherent fault trees. The input description of a fault tree differs depending on whether single phase or multi-phase structures are considered. The code consists of two programs; SALP-MP1 and SALP-MP2. When a single phase structure is to be analysed SALP-MP1 is used. Multi-phase structures are analysed by means of SALP-MP2.

The user specifies the maximum order of the minimal cut sets to be determined (logical cut off). A probabilistic cut off level is also included in the data input. Estimation of residual error is included in the code.

A fault tree can be described using a free format. Each input card describes a gate and its descendants (gate inputs). It is not necessary to respect any certain order in the description of the gates. The computer output consists of a list of minimal cut sets ordered from high to low probability.

2.8.3 References

1. M. Astolfi, et al, SALP-MP (Multi Phase). A Computer Program for Fault Tree Analysis of Complex Systems and Phased Missions. Description and How-to-Use, P.E.R. 389, Joint Research Centre, Ispra, 1980

2. B.V. Koen, A. Carnino, "Reliability Calculations with the List Processing Technique", IEEE Trans. Rel., Vol R-23, 1974, pp 43-50

3. A. Blin, et al, "A Computer Code for Fault Tree Calculation: PATREC", Proceedings of the American Nuclear Society Meeting Probabilistic Analysis of Nuclear Reactor Safety, Newport Beach, May 8-10, 1978, paper no. XIII.6

4. D.E. Knuth, The Art of Computer Programming, Vol 1. Fundamental Algorithms, Addison-Weseley, 1968

2.9 SETS

The SETS code has been developed at the Sandia National Laboratories in the U.S.A. A first user's manual was published in 1973. The basic principles of the code have been improved over the years since its inception and the latest version of the user's manual appeared in 1980 [1]. Due to the flexibility of the SETS code, it has found a wide field of application.

2.9.1 Purpose

SETS is basically a code for the qualitative evaluation of complete fault trees. There are however two procedures available for certain quantitative evaluations. SETS manipulates Boolean equations. By applying Boolean identities on the set equations representing a fault tree, the equations can be transformed in a way which allows the minimal cut set determination.

In addition to the standard AND and OR gates, the code can contain INHIBIT, PRIORITY AND, and EXCLUSIVE OR gates. The INHIBIT gate represents a situation where the output occurs if the single input occurs in presence of an enabling condition. The PRIORITY AND gate is used to represent an output event that occurs if all input events occur in a specific sequence. Finally, the EXCLUSIVE OR gate is used to represent cases where an output event occurs if exactly one of the input events occurs.

An option exists for the specification of logical combinations that cannot be readily expressed with any of the gates above. The analyst may define "SPECIAL" gates to represent m-out-of-n logics.

2.9.2 Program Description

SETS is a Fortran-Extended code. The input to the code consists of the fault tree input and the SETS user program. The fault tree input is written in a free format language and is directly prepared from the graphical representation of the fault tree.

The SETS user program is an algorithm for reading the input representation of a fault tree, establish a Boolean equation for each intermediate gate as a function of its input events, and then make these equations available for further processing. The Boolean equations are transformed in a way which allows the minimal cut sets to be obtained. Three steps are necessary for finding minimal cut sets for a particular intermediate gate [1, p 17]

1. Generate all of the intermediate gate equations defined by the fault tree;
2. Generate an equation for the selected intermediate gate as a function of only primary events by a repeated substitution process using the intermediate gate equations generated in step 1;
3. Reduce the equation resulting from step 2 by applying the Boolean absorption identities $P \cdot P = p$ and $P + P \cdot Q = P$.

The result of step 3 is an equation in disjunctive normal form equal to a listing of the minimal cut sets.

2. .3 Reference

1. R.B. Worrell, D.W. Stack, A SETS User's Manual for the Fault Tree Analyst, SAND77-2051, Sandia National Laboratories, 1980

3. CODE SUMMARIES -- QUANTITATIVE FAULT TREE EVALUATION

3.1 BOUNDS

The computer code BOUNDS has been develcped at UCLA. A user's manual was published in 1976 [1]. The code is not to be characterized as a conventional quantitative fault tree evaluation code, but rather as an option available to the analyst requireing a more complete quantification of top event probabilities. BOUNDS is compatible with some of the qualitative fault tree codes.

3.1.1 Purpose

BOUNDS is specifically designed for the calculation of the mean and variance of top event probabilities. From the mean and variance of a top event, the confidence bounds are determined. The methods employed are analytical.

The analytical approach taken in BOUNDS is based on the method of moments [1,2,3]. The first two moments -- mean and variance -- are calculated for each primary event. A Taylor series expansion of the top event equation -- in terms of the primary event probabilities -- produces the mean and variance of the top event. Finally, the confidence bounds are calculated either by using empirical distribution functions [4] or standard inequalities.[1]

[1]There exist standard inequalities such as Cantelli's and Tchebycheff's that for any distribution give the relationships between the mean and the variance.

3.1.2 Program Description

BOUNDS is a Fortran code which can handle 1000 primary events and up to 500 minimal cut sets [1, p 63]. The order n of the minimal cut sets is limited to 5 ($n < 5$). A qualitative fault tree evaluation must preceed a BOUNDS computer run.

The required input data consists of system information (number of components, number of minimal cut sets) and primary event failure data. The failure data must consist of median failure rates, error factors of the respective failure rates, median mean-time-to-repair ($\bar{\tau}$), and error factors of $\bar{\tau}$.

The output consists of two lists with primary event and minimal cut set means and variances, as well as a system unavailability summary with the confidence bounds. An estimation of system failure distribution is also included in the output.

3.1.3 References

1. Y.T. Lee, G. Apostolakis, Probability Intervals for the Top Event Unavailability of Fault Trees, UCLA-ENG-7663, Energy and Kinetics Dept., University of California, Los Angeles, 1976

2. G. Apostolakis, Y.T. Lee, "Methods for the Estimation of Confidence Bounds for the Top-Event Unavailability of Fault Trees", Nucl. Eng. Des., Vol 41, 1977, pp 411-419

3. M. Mazumdar, J.A. Marshall, S.C. Chay, "Propagation of Uncertainties in Problems of Structural Reliability", Nucl. Eng. Des., Vol 50, 1978, pp 163-167

4. G.J. Hahn, S.S. Shapiro, Statistical Models in Engineering, John Wiley & Sons, Inc., 1967, pp 195-224

3.2 IMPORTANCE

In the same sense as the BOUNDS code IMPORTANCE is an option available to the analyst requireing more detailed quantifications of fault tree unavailabilities. The IMPORTANCE code manual was published in 1975 [1].

3.2.1 Purpose

In the design of reliable complex systems a common approach is to concentrate the resources on the subsets of components that are most critical to the system reliability. One aim with system reliability analyses therefore often is to calculate measures of probabilistic importance [2].

Different measures of component and minimal cut set importance have been developed. Some measures are based on a knowledge of the system structure only -- a series component generally plays a more important role than the same component in parallel -- other measures are based on component reliability and system structure.

In IMPORTANCE the following seven measures of component importance can be computed: (1) Birnbaum's measure, (2) criticality importance, (3) upgrading importance, (4) Vesely-Fussell importance, (5) Barlow-Proschan importance, (6) steady-state Barlow-Proschan importance, and (7) contributory sequential importance.

Two measures of minimal cut set importance can be computed: (1) Barlow-Proschan cut set importance, and (2) Vesely-Fussell cut set importance.

3.2.2 Program Description

IMPORTANCE is a Fortran code. It requires as input the minimal cut sets of a fault tree, and primary event characteristics -- failure rates and fault duration times. The failure and repair distributions are assumed to be exponential, and furthermore, all measures of importance are calculated assuming statistical independence of the primary events. A listing of the computer code is found in [1].

Four options are available in the code. The first option computes measures of importance as a function of time. The second option computes measures of importance as a function of the probability of the top event. In the third option the measure of importance is computed as a function of the top event probability and the failure rates are expressed proportionally. Finally, the fourth option computes the measures of importance as a function of a reference time unit.

The type of importance measures computed and the type of code option to be used is defined by the data input. The output consists of a series of tables listing the measures of importance in descending order as a function of the data input.

3.2.3 References

1. H.E. Lambert, Fault Trees for Decision Making in System Analysis, UCRL-51829, Lawrence Livermore Laboratory, 1975

2. D.A. Butler, "An Importance Ranking for System Components Based Upon Cuts", Operations Research, Vol 25, 1977, pp 874-879

3.3 KITT

The KITT code was developed at the Idaho Nuclear Corporation and the user's manual was published in 1970 [1]. The code has influenced the development of many fault tree codes.

3.3.1 Purpose

The KITT computer code is an application of the kinetic tree theory developed by Vesely [2]. Two versions -- KITT 1 and KITT 2 -- are available, designed to obtain quantitative reliability characteristics of primary events, minimal cut sets and top events. The PREP code (see chapter 2) produces the necessary input to KITT.

The primary events are restricted to having constant failure rates, and with regard to repair of the primary events, constant repair rates are used. KITT 1 is a so called single phase code, meaning that the failure rate and type of repair must remain the same for all time. KITT 2 is a multi phase code. Here, failure rates and type of repair must be constant in one time phase, but can change in an arbitrary manner from phase to phase.

3.3.2 Program Description

KITT is a Fortran code. For each primary event, minimal cut set, and for the top event the following

reliability characteristics are obtained at time points specified by the user:

- Probability of being in a failed state at time t
  (differential characteristic)
- Expected number of failures per unit time t
  (differential characteristic)
- The probability of suffering a failure per unit time t, given it is functioning at time t
  (differential characteristic)
- The expected number of failures during the time interval from 0 to t
  (integral characteristic)
- The probability of suffering one or more failures in the time interval from 0 to t
  (integral characteristic)

The input to KITT 1, besides the information obtained from PREP, consists of the failure rates and repair data for the primary events. Exact, time-dependent reliability information is determined for each component of the fault tree and for each minimal cut set. The system reliability information is obtained by upper bound approximations (denoted as bracketing in the code manual.

The bracketing principle determines the upper and lower bounds of the probability of the top event by succesively taking into account intersections of an increasing number of events. A good approximation of

the top event probability is obtained by this procedure when the primary event probabilities are much less than one. The number of terms in the top event equation used in the bracketing procedure are determined by the user.

KITT 2 obtains the same reliability information as KITT 1, and a large portion of the input to KITT 2 is identical to that of KITT 1. In KITT 2 multiphase descriptions of components can be evaluated. The components may have different reliability properties during different time intervals (phases). Up to 50 phases per component are allowed in KITT 2.

3.3.3 References

1. W.E. Vesely, R.A. Narum, PREP and KITT: Computer Codes for the Automatic Evaluation of a Fault Tree, IN-1349, Idaho Nuclear Corporation, 1970

2, W.E. Vesely, "A Time-Dependent Methodology for Fault Tree Evaluation", Nucl. Eng. Des., Vol 13, 1970, pp 337-367

3.4 LENC

The LENC code was developed at Asea-Atom in Sweden [1], and a user's manual was published in 1975 [2]. The code is based on KITT.

3.4.1 Purpose

The LENC code is based on the kinetic tree theory and is specifically designed to calculate mean-unavailabilities on system levels. In comparison with the KITT code the following characteristics are unique to LENC: (1) Four types of components with time dependent structure functions can be handled, (2) evaluation of the impact of periodic testing can be performed, and (3) an importance ranking of the minimal cut sets by means of fault duration times is within the capability of the code. The component types are

- Type 1; constant repair rate
- Type 2; constant repair time
- Type 3; not repairable immediately, time between inspections = T
- Type 4; the probability of failure is constant

In comparison with KITT computer times have been reduced, as well as computer core memory require-

ments [1]. A sample problem analysed with LENC is presented in [3].

3.4.2 Program Description

LENC is coded in Fortran and consists of two subroutines, DIM and PAGE. In DIM the reliability calculations are performed. PAGE completes the edited output with table heads.

A qualitative fault tree evaluation must preceed a LENC run. Information about the minimal cut sets must be included in the data input, together with the reliability data and the number of time steps, i.e. the time points at which the calculation of the failure characteristics are made. The intervals between inspections must be multiples of a time step.

Failure probabilities and failure rates are calculated on the primary event level, minimal cut set level, and top event level. Fault duration times are calculated for each time interval. Finally, the mean-unavailability of the system is calculated as the ratio of the fault duration time to the total time period that is analysed.

3.4.3 References

1. L. Carlsson, Development of Computer Codes for System Availability Analysis, TR KUA 73-350, Asea-Atom, 1973
(in Swedish)

2. Y. Larsson, LENC - A Computer Program for the Analysis of Mean-Unavailability of Complex Systems. User's Manual, REA 75-81, Asea-Atom, 1975
(in Swedish)

3. O. Johansson, H. Tuxen-Meyer, Fault Tree Analysis with the LENC Code, K5-79/30, Studsvik Energiteknik, 1979
(in Swedish)

3.5 SALP-MP

The quantitative part of SALP-MP is integrated with the qualitative part described in Chapter 2. Quantification is performed on the significant minimal cut set (the non-significant minimal cut sets are discarded by means of logical cut off and probabilistic cut off).

3.5.1 Purpose

The following quantitative analyses are performed in SALP-MP: (1) Calculation of unavailabilities and unreliabilities of each significant minimal cut set, (2) calculation of upper bounds of the top event unavailability and unreliability, and (3) determination of the importance for the primary events. Only exponential distributions for the time to failure and time to repair can be accepted.

Cold standby structures with perfect switching can be handled. The allowable type of maintenance is on line maintenance where repair is started immediately after failure. System components are assumed "as good as new" after repair.

3.5.2 Program Description

SALP-MP is a PL/1 code and a qualitative evaluation of the fault tree must preceed quantification (see

Chapter 2. section 8). The input data consists of the λ-and μ-values for each primary event. The maximum number of primary events and logical gates are 9999 and 20.000 respectively [1, p 66].

For multi-phase systems the different phases are analysed in a sequential way starting with the first phase. For each phase both the tree and primary event data are read. Then the minimal cut sets are determined and stored on a specific file.

It is to be noted that the failure mode of an event cannot change from one phase to another. Furthermore, events cannot have an inspection time, and an event repairable in a given phase must be repairable in all the successive phases in which it appears.

3.5.3 Reference

1. M. Astolfi, et al, SALP-MP (Multi Phase). A Computer Program for Fault Tree Analysis of Complex Systems and Phased Missions. Description and How-to-Use, P.E.R. 389, Joint Research Centre, Ispra, 1980

3.6 SUPERPOCUS

SUPERPOCUS was developed at the University of Tennessee and a user's manual [1] appeared in 1977. The code is based on approximations of the kinetic tree theory [2].

3.6.1 Purpose

Being based on approximations of the kinetic tree theory the code yields highly bounded primary event, minimal cut set, and top event failure information. The system reliability characteristics are obtained under the following assumptions:

- The primary event failures and repairs are statistically independent;
- The failure distribution of each primary event is exponential and the failure rate λ is known;
- The primary event repair distribution is exponential and the mean-time-to-repair τ is known.

Since in many practical situations there is a lack of detailed failure data and the assumptions are therefore not considered to be overly restrictive. An importance ranking of primary events and minimal cut sets can be performed.

3.6.2 Program Description

SUPERPOCUS is a Fortran code that is designed to be compatible with the MOCUS and PREP codes. A system containing up to 400 primary events, and 1000 cut sets with up to 10 primary events per set can be handled. Results of a computer run always consists of an input data edit, and time-dependent reliability characteristics for the top event. A maximum of 25 time steps can be handled.

The input to SUPERPOCUS consists of groups containing control parameters, primary event failure rates and repair times, the time coordinates used for the analysis, and minimal cut sets (obtained from a qualitative fault tree evaluation). Optional output available to the analyst includes reliability characteristics for the primary events and the minimal cut sets. A further option is the importance ranking.

3.6.3 References

1. J.B. Fussell, D.M. Rasmuson, D.P. Wagner, SUPERPOCUS. A Computer Program for Calculating System Probabilistic Reliability and Safety Characteristics, NERS-77-01, Nuclear Engineering Department, College of Engineering, The University of Tennessee, 1977

2. J.B. Fussell, "How to Hand-Calculate System Reliability and Safety Characteristics", IEEE Trans. Rel., Vol R-24, 1975, pp 169-174

3.7 WAM-BAM

The WAM-BAM code package has been developed by the Science Applications Incorporated [1]. A user's manual was published in 1976 [2], and the code package has been implemented at the Studsvik Laboratories in Sweden at a later date [3].

3.7.1 Purpose

WAM-BAM is designed for fault tree unavailability calculations. It uses programs in the WAM code family as preprocessor. WAM-CUT determines the minimal cut sets from complete fault trees. By incorporating the NOT logic gate operator possibilities exist for the analysis of dependent failures [1].

BAM uses Boolean algebra minimization techniques to find the resultant logic expressions from an input tree and then calculates the associated point unavailability. It is possible to calculate the means and variances of the top event and by application of the Tchebycheff inequality a 95% confidence bound can be found.

3.7.2 Program Description

BAM accepts the fault tree logic input to include any of the 16 operations of two binary variables

[1, p 9]. The preprocessor develops an alpha-numeric description of the fault tree, which is converted to a numerical form and used as input to BAM. WAM-BAM will accept fault trees with up to 1500 primary events and 1500 two-input gates [3].

3.7.3 References

1. E.T. Rumble, et al, Generalized Fault Tree Analysis for Reactor Safety, EPRI-217-2-2, Electric Power Research Institute, 1975

2. F.L. Leverenz, H. Kirch, User's Guide for the WAM-BAM Computer Code, EPRI 217-2-5, Electric Power Research Institute, 1976

3. J.-P. Bento, K. Pörn, Fault Tree Analysis. Implementation of the WAM-Codes, K2-79/169, Studsvik Energiteknik, 1979

4. CODE SUMMARIES -- DEPENDENT FAILURE ANALYSIS

4.1 BACFIRE

The BACFIRE code was developed at the University of Tennessee and a user's manual appeared in 1977 [1]. BACFIRE is similar to COMCAN. Reviews of qualitative dependent failure analysis are presented in [2,3].

4.1.1 Purpose

BACFIRE is designed to find common cause candidates -- defined as minimal cut sets with common potential causes of failures. The code is compatible with several of the codes for qualitative and quantitative fault tree evaluation. Only coherent fault trees can be handled. The program produces lists of the qualitative failure characteristics of the common cause candidates.

4.1.2 Program Description

BACFIRE is a Fortran code. A minimal cut set will be identified as a common cause candidate by either of two criteria:

- If all the primary events in a minimal cut set are associated by a certain condition -- such as calibration, test and maintenance, energy flow -- which may alone increase the probability of multiple component malfunction;

- If all the primary events in a minimal cut set are susceptible to the same secondary cause -- defined as a component malfunction for which the component itself is not held responsible -- and are located in the same domain for that cause of secondary failure.

Necessary input information for the BACFIRE code consists of five data sets:

- Event definition;
- Event location references;
- Cause of secondary failure library;
- Component manufacturer;
- Minimal cut sets

BACFIRE searches the qualitative failure characteristics of the primary events contained in the minimal cut sets to find those characteristiscs common to all primary events by either of the two criteria defined above. The minimal cut set that is defined as a common cause candidate is then listed. The common failure characteristic is also listed with the common cause candidate. BACFIRE successively determines all other common failure characteristics af the minimal cut sets.

4.1.3 References

1. C.L. Cate, J.B. Fussell, BACFIRE - A Computer Program for Common Cause Failure Analysis, NERS-77-02, Nuclear Engineering Department, College of Engineering, The University of Tennessee, 1977

2. J.R. Wilson, et al, "Techniques for Qualitative Analysis of Common Cause Failures", included in A Collection of Methods for Reliability and Safety Engineering, ANCR-1273, Aerojet Nuclear Company, 1976

3. D.M. Rasmuson, G.R. Burdick, J.R. Wilson, Common Cause Failure Analysis Techniques: A Review and Comparative Evaluation, TREE-1349, EG & G Idaho, Inc., 1979

4.2 COMCAN

COMCAN was the first code for qualitative dependent failure analysis available to the reliability analyst. The code development was performed by the Aerojet Nuclear Company, and a user's manual [1] appeared in 1976.

4.2.1 Purpose

COMCAN is designed to qualitatively determine the susceptibilities to common cause failures for each primary event in a minimal cut set and to locate common links between components.[1] The main objective of a COMCAN analysis is to indicate weak points in a system and to suggest corrective action. COMCAN therefore, in certain situations, is a useful complement to computer codes for qualitative fault tree analysis.

4.2.2 Program Description

The COMCAN code is written in Fortran and its format is compatible with the input format used with computer codes for qualitative and quantitative fault tree analysis such as PREP, MOCUS, and KITT. The computerized analysis is concerned with locating

---

[1]Please note that a common cause failure is a dependent failure which can be identified by means of fault tree techniques.

common cause candidates (defined in Section 4.1.1), prime common cause candidates (all comonents in the minimal cut set share a common location), and identifying significant common cause events. A significant common cause event is a secondary cause that is common to all the primary events in one or more minimal cut sets.

The key to obtaining the common cause candidates and prime common cause candidates is the set of input cards containing all the variables that can cause dependencies. These inputs are

- Primary event description input. Involves a system code, component identification and failure mode code;
- Location input. Involves the physical location of the components in the primary events;
- Manufacturer input;
- Generic cause susceptibility input. Involves information about common links that result in dependence between components.
- Domain definition. The input on the domain definition cards relates regions within a structure, usually a single building, to causes which may present a hazard to components located within those regions.
- Minimal cut set input. Describes the minimal cut sets generated by some other program.
- Cause ranking cards. The cause ranking cards permit an importance ranking of the causes.
- Generic cause table cards.
- Option cards.

The outputs of the computer executions are in the form of lists containing information about dependencies due to various common causes. A number of printer options are available and the user determines the printout.

4.2.3 Reference

1. G.R. Burdick, N.H. Marshall, J R. Wilson, COMCAN - A Computer Program for Common Cause Analysis, ANCR-1314, Aerojet Nuclear Company, 1976

4.3 COMCAN II-A

COMCAN II-A is an improved version of the COMCAN code. The user's manual was published in 1979. The code development was performed at the Idaho National Engineering Laboratory.

4.3.1 Purpose

COMCAN II-A is designed for the identification of potential common causes for the failure of fault tree minimal cut sets. Like its predecessor, COMCAN, it locates common cause candidates and identifies associated significant common cause events.[1]

A major drawback of COMCAN is that since it uses fault tree minimal cut sets as inputs, complex trees cannot be analysed very effectively. Usually the inputs consists of minimal cut sets of order 3 or less. In dependent failure analysis it is however necessary to evaluate high order minimal cut sets.

To circumvent the COMCAN dependence upon minimal cut sets obtained by other codes, the FATRAM algorithm (see Chapter 2, Section 3) has been included as a program module of COMCAN II-A. Consequently, COMCAN II-A is an integrated qualitative and dependent failure evaluation code.

---

[1] A significant common cause event is a secondary event that is common to all primary events in one or more minimal cut sets.

4.3.2 Program Description

COMCAN II-A is coded in Fortran. The input phase of a COMCAN II-A execution consists of fault tree definition and information about the potential common causes to be analysed. Optional inputs include (1) the manufacturer of each component with an event appearing in the fault tree, (2) a barrier map desctibing common locations for various possible common cause events, (3) the location of each component with an event in the input, and (4) numbers in the range 0 to 9 indicating the relative ranking of each primary event's susceptibility to the common cause (importance ranking).

The output from COMCAN II-A is a list of minimal cut sets with events that share a common location, common link, or common manufacturer, or that represent similar components. The program can handle 2000 primary events and 700 logic gates, and the amount of initiator data is stated as 1000 initiators (physical cases or generic causes) for about 100 rooms [1, p 30,

4.3.3 Reference

1. D.M. Rasmuson, et al, COMCAN II-A - A Computer Program for Automated Common Cause Failure Analysis, TREE-1361, EG & G Idaho, Inc., 1979

4.4 SETS

A discussion on how to use the SETS code (see Chapter 2, Section 9) in dependent failure analysis is presented in [1]. Applications of the SETS code in dependent failure analysis are represented by the vital area identification [2] and the evaluation of systems interaction in nuclear power plants [3].

4.4.1 Purpose

The common cause failure analysis option of SETS is used to achieve a description of how cause events -- or common causes -- are related to primary events, and how minimal cut sets and top events are affected by cause events. Implementation of common cause analysis using SETS is based on a transformation of variables using the Boolean distributive law, idempotent law, and the law of absorption.

Before the transformation of variables, a Boolean equation must be defined for every primary event in the fault tree. The equation for a primary event specifies the special conditions and secondary causes that are applicable to the primary event. The equation also indicates the physical location of the primary event.

A special condition is defined as [1] a characteristic which closely links some of the primary events in the fault tree. A secondary cause is defined as [1] an event which may contribute to the occurrence of some

of the primary events in the fault tree.

4.4.2 Program Description

SETS is a Fortran-Extended code. The primary event equations are contained in special input blocks. A SETS user program is used to determine the common cause candidates. Some special importance ranking routines for the selection of the common cause candidates with the highest probability of occurrence is included in the code.

4.4.3 References

1. R.B. Worrell, D.W. Stack, Common-Cause Analysis Using SETS, SAND77-1832, Sandia Laboratories, 1977

2. G.B. Varnado, N.R. Ortiz, Fault Tree Analysis for Vital Area Identification, SAND79-0946, Sandia National Laboratories, 1979

3. G.J. Boyd, et al, Final Report - Phase 1 Systems Interaction Methodology Applications Program, SAND80-0384, Sandia National Laboratories, 1980

5. CODE SUMMARIES -- GENERIC FAILURE ANALYSIS

5.1 CAT

The CAT code, developed at UCLA, is a general computer implemented approach to the modelling of complex systems. The CAT algorithm is based on input-output models. A user's manual appeared in 1978 [1]. Critical reviews of the CAT algorithm have been published at later dates [2,3].

5.1.1 Purpose

The purpose with the CAT code is to let a computer construct fault trees. The drawing of fault trees is however not included. Decision tables for component and system behaviour form the basis of the CAT algorithm. Given a proper system description a fault tree for a certain top event will be constructed.

Automatic fault tree construction requires the availability of two basic types of information: (1) Component operating and failure modes , and (2) description of how the various components of the system are interconnected. Decision tables are used to formalize the description of the functioning and failure characteristics of the components.

5.1.2 Program Description

CAT is coded in Fortran and the input consists of the following information:

- Program control data. Consists of program dimensions used to define the sizes of the component library nad system configuration;

- Decision table models. These contain the basic information for the component types, number of inputs, internal failure mechanisms, outputs, and the decision table itself;

- System configuration. Coupling of components, numbers are assigned to the nodes at which output of a component is connected to the inputs of one or more succeding components;

- Top event definition. Top event described in terms of system states at specific nodes;

- Boundary conditions. System or component states which have been predefined as existing or not existing boundary conditions within the system. A boundary condition is defined initially and continues to exist throughout the fault tree.

- Failure and repair data. This group is optional.

The output of the code consists of two parts. First is the printed output of all the input data, and then the fault tree itself in coded form. The preparation of input data can be very time consuming in certain cases [3]. Consequently, the field of application of the CAT code is limited.

5.1.3 References

1. G.E. Apostolakis, S.L. Salem, J.S. Wu, CAT: A Computer Code for the Automated Construction of Fault Trees, NP-705, Electric Power Research Institute, 1978

2. G. Squellati, Critical Review of the CAT Algorithms for Automated Fault Tree Construction, P.E.R. 392/80, Joint Research Centre, Ispra, 1980

3. U. Berg, P. Hellström, B. Lydell, Fault Tree Synthesis Using the CAT Algorithm, PSE02-81, Swedish Nuclear Power Inspectorate, 1981

5.2 DRAFT

An early approach to the automatic fault tree construction was presented by Fussell in 1973 [1]. Fussell's approach is called synthetic tree model (STM), and the DRAFT code algorithm -- based on the synthetic tree model -- is based on mini fault tree models. It is not known if any further code developments have been performed.

5.2.1 Purpose

The DRAFT code is limited to the automatic fault tree construction of electrical systems. It does not draw the fault trees! The code output consists of the fault tree in a coded form, which the analyst has to translate into a tree.

In DRAFT a fault tree is constructed from small segments, called component failure transfer functions. The transfer functions (component models) are obtained by conventional FMEA. The differnet component models are stored, and the system schematic diagram provides the input for piecing together the different component models.

5.2.2 Program Description

The DRAFT input consists of the system schematic and the system boundary conditions -- which define

the operating conditions of the system. Component library data must exist in coded form. In its original form the code would construct coherent fault trees with 100 logic gates "in typically less than seven second" [1, p 77].

5.2.3 Reference

1. J.B. Fussell, Synthetic Tree Model. A Formal Methodology for Fault Tree Construction, ANCR-1098, Aerojet Nuclear Company, 1973

INDEX