Efficient Basic Event Orderings for Binary Decision Diagrams

J.D. Andrews • Loughborough University, Loughborough

L.M. Bartlett • Loughborough University, Loughborough

Key Words: Fault Tree Analysis, Binary Decision Diagrams

SUMMARY & CONCLUSIONS

Over the last five years significant advances have been made in methodologies to analyse the fault tree diagram. The most successful of these developments has been the Binary Decision Diagram (BDD) approach. The Binary Decision Diagram approach has been shown to improve both the efficiency of determining the minimal cut sets of the fault tree and also the accuracy of the calculation procedure used to determine the top event parameters. The BDD technique provides a potential alternative to the traditional approaches based on Kinetic Tree Theory.

To utilise the Binary Decision Diagram approach the fault tree structure is first converted to the BDD format. This conversion can be accomplished efficiently but requires the basic events in the fault tree to be placed in an ordering. A poor ordering can result in a Binary Decision Diagram which is not an efficient representation of the fault tree logic structure. The advantages to be gained by utilising the BDD technique rely on the efficiency of the ordering scheme. Alternative ordering schemes have been investigated and no one scheme is appropriate for every tree structure. Research to date has not found any rule based means of determining the best way of ordering basic events for a given fault tree structure.

The work presented in this paper takes a machine learning approach based on Genetic Algorithms to select the most appropriate ordering scheme. Features which describe a fault tree structure have been identified and these provide the inputs to the machine learning algorithm. A set of possible ordering schemes has been selected based on previous heuristic work. The objective of the work detailed in the paper is to predict the most efficient of the possible ordering alternatives from parameters which describe a fault tree structure.

1. INTRODUCTION

Since the development of Kinetic Tree Theory by Vesely in the early 1970's[1] this technique has provided the basis for the methods used to analyse fault trees. Over the past five years an alternative technique know as the Binary Decision Diagram (BDD) method has been developed[2-7]. The BDD method has the advantages that it is more accurate than the conventional approaches and that it is also more efficient. In calculating the system or top event parameters it does not need to first evaluate all the minimal cut sets, nor does it require the use of approximations, the exact calculations can be performed.

To take advantage of these features the fault tree constructed to represent causes of the system failure mode must first be converted to a BDD. This can be accomplished very efficiently but requires that the basic events in the fault tree are placed in an order. A good ordering of the basic events can result in a very efficient analysis, a poor ordering can lead to problems.

Several research papers have been published which investigate different ordering strategies for the fault trees[8-9]. As yet no rule-based approach to identifying an ordering scheme which yields an efficient ordering for each tree has been produced. It is the inability to obtain an efficient ordering of the fault tree basic events which has to date prevented a commercially available code being produced which is based on this method.

This paper presents a means of producing an ordering which is based on a classifier system where the rules are evolved using a genetic algorithm.

2. BINARY DECISION DIAGRAMS

A BDD is a directed acyclic graph, Figure 1. All paths through the BDD terminate in one of two states: either a '1' state, which corresponds to top event occurrence, or a '0' state which corresponds to top event non-occurrence. A BDD is composed of terminal and non-terminal vertices, which are connected by branches. Terminal vertices have the value 0 or 1 and the non-terminal vertices correspond to the basic events in the fault tree. Each non-terminal vertex has a 0 branch, which represents the basic event non-occurrence and a 1 branch which represents basic event occurrence.

0-7803-4362-X/98/$10.00 © 1998 IEEE

Every path through the BDD starts from the root vertex, and proceeds down through the diagram to a terminal vertex. Paths which terminate at a 1 vertex yield the cut sets. The events contained in each cut set are determined by the nodes whose 1 branch is included on the path to a terminal 1 vertex.

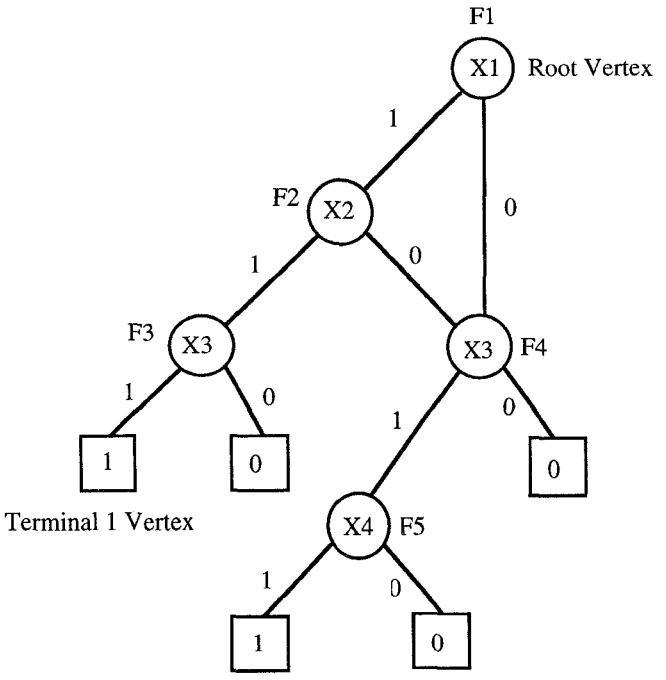

Figure 1: BDD for the fault tree

3. BASIC EVENT ORDERING SCHEMES

The order in which the basic events are considered will affect the size of the BDD constructed from the fault tree. An inefficient ordering scheme will make the BDD very large and produce many paths through the structure which correspond to non-minimal cut sets. Alternative ordering schemes will produce BDD's of different sizes, the smaller the BDD the more optimal the diagram. The objective would be to produce an ordering scheme which achieves the 'best' BDD. It has been shown[9] that there is no one scheme which will always guarantee the 'best' BDD formation for all fault trees and the most appropriate scheme must be selected depending on the characteristics of the fault tree under analysis.

4. PATTERN RECOGNITION APPROACHES

The objective of this paper is to use some pattern recognition technique which can identify the appropriate scheme for ordering the fault tree variables. There are many alternatives which include classifier systems[10], ID3[11], neural networks[12], Bayesian methods and Fuzzy Logic [13]. There is no way to determine which of these would be the most effective to apply to the ordering problem. As such the first one, the classifier system, has been selected and investigated in this paper.

5. CLASSIFIER MODEL

A classifier system is a machine learning system that learns syntactically simple string rules (called classifiers) to guide its performance[10]. A classifier system, depicted schematically in Figure 2, consists of three main components:

1) Rule and message system;
2) Apportionment of credit system;
3) Rule/message generation system.

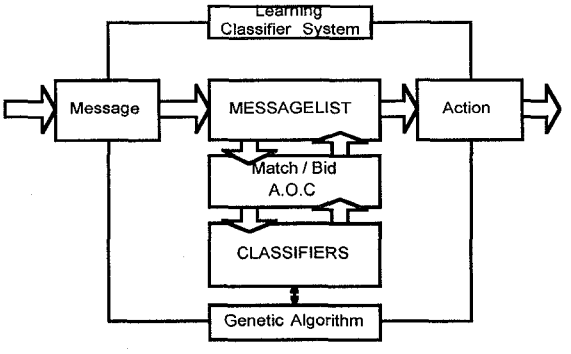

Figure 2: A Classifier System.

Input messages are posted to the message list where they are checked for matches within the classifier store. In the case of more than one match the winning classifier is decided by a bidding process in the apportionment of credit system. Once a winning classifier has been found, this then posts its message to the message list, and the process is repeated until an eventual winner is found. The final classifier selected by the process then produces an output.

5.1 *The Rule and Message System*

The rule and message system forms the computational backbone of the machine learner[14].

The two informational units in a rule and message system are the messages and the classifiers. A **message** within the classifier system is simply a finite length string over some finite alphabet, in terms of a binary alphabet, we have:

$$\text{message} \Rightarrow \{0,1\}^n$$

Simply, the message is defined as a concatenation of 0's and 1's of length n. Messages are the means of information exchange in a classifier system. A **classifier** is a production rule with the syntax:

$$\text{classifier} \Rightarrow \text{condition} : \text{message}$$

The condition is a simple pattern recognition device where a wild card character (#) is added to the underlying alphabet.

$$\text{condition} \Rightarrow \{0,1,\#\}^n$$

Thus, a condition matches a message if at every position a '0' in the condition matches a '0' in the message, a '1' matches a '1', and a '#' matches either.

Both the condition and the message parts of a classifier contain coded characteristics of the problem and coded outputs of the problem, for example:

if condition $\Rightarrow$ {011011010011}
then {0110110} $\Rightarrow$ the coded characteristics which describe problem
and {10011) $\Rightarrow$ the action or output of the problem.

Once a classifier matches a previous message from the message list, that classifier becomes a candidate to post its message to the message list on the next step of the genetic algorithm. Whether the candidate classifier posts its message is determined by the outcome of the bidding process determined in the apportionment of credit system.

5.2 *Apportionment of Credit Algorithm*

In a rule learning system, the relative value of different rules must be generated. To facilitate this type of learning, classifiers coexist in an information based service economy - **the apportionment of credit (A.O.C) system**. The A.O.C system contains two main components: an auction; and a clearinghouse. When classifiers are matched they do not directly post their message. Instead, having its condition matched qualifies a classifier to participate in an auction. To participate in the auction, each classifier maintains a record of its net worth, called its strength. A competition is held among classifiers where the right to answer relevant messages goes to the highest bidder. In this way rules that are highly fit are given precedence over other rules.

The auction permits appropriate classifiers to be selected to post their messages. Once a classifier is selected for activation, it must clear its payment through the clearinghouse, paying its bid to other classifiers for matching messages rendered. A matched and activated classifier sends its bid, to those classifiers responsible for sending the messages that matched the bidding classifiers condition. The bid payment is divided in some manner among the matching classifiers. This division of payoff among contributing classifiers helps to ensure the formation of an appropriate sized subpopulation of rules [10].

To illustrate the workings of the A.O.C system we consider four classifiers, see table 1. Assuming initial strength values of 200 for all four classifiers, we post the initial input (environmental) message 0111, where {01} represents the characteristics of the problem and the {11} represents the action/output of the problem. We assume a bid coefficient of 0.1 and take the bid as the product of this bid coefficient, and strength. In the initial step (t=0), classifier 1 matches the input and bids 20 units, since it is the only match it sends its message during the next time step. Classifier 1 pays its bid to the party responsible for its activation; in this case, the input (environment) strength is increased by 20 units as the environmental message was responsible for activating classifier 1. In subsequent time steps, activated classifiers make their payment to previously active classifiers. Finally, at time step 5, a reward comes into the system and is paid to the last active classifier, classifier 4 where message ouput corresponds to the input message (i.e. the last two digits are the same).

The A.O.C system provides a clean procedure for evaluating rules and deciding among competing rules. Yet we still require new possibly better rules to be injected into the system. This is achieved by the rule/message generation system, in this case a simple Genetic Algorithm, where new rules are created by the tripartite process - reproduction, crossover, and mutation.

6. GENERATION OF CLASSIFIERS AND MESSAGES

For the purpose of this application the message list of the classifier system is a list of binary coded strings which represent 6 chosen characteristics of a fault tree structure, and the best basic event ordering scheme is output. For the initial training process, classifiers and their messages are generated randomly. A selected proportion of these initial rules are then renewed at selected instances using a simple genetic algorithm.

6.1 *Genetic Algorithms*

Genetic algorithms (G.A's) are search algorithms based on the mechanics of natural selection and natural genetics. They originated from the studies of Holland and colleagues at the University of Michigan[15]. A genetic algorithm is composed of three operators: Reproduction, Crossover and Mutation. Reproduction is a process in which individual strings are copied according to their fitness values. Copying strings in this manner means that strings with a higher value have a higher probability of contributing one or more offspring in the next generation.

The reproduction operator is implemented by creating a biased roulette wheel, where each current string in the population has a roulette wheel slot sized in proportion to its fitness. The effect of roulette wheel parent selection is to return a randomly selected parent. This parental selection technique has the advantage that it directly promotes reproduction of the fittest population members by biasing each members chances of selection in accordance with its evaluation [16].

After the reproduction population is chosen, simple crossover may proceed in two steps. First, members of the newly reproduced strings in the mating pool are paired at random. Second, each pair of strings undergoes crossover as follows: an integer position k along the string is selected at random between 1 and the string length less one [1,t-1]. Two new strings are created by swapping all characters

Table 1: Apportionment of Credit System

---------- t = 0 ---------- ---------- t = 1 ---------- ---------- t = 2 ----------

	Classifier	Strength	Message	M	Bid	Strength	Message	M	Bid	Strength	Message	M	Bid
1	01##:0000	200		e	20	180	0000			220			
2	00#0:1100	200				200		1	20	180	1100*		
3	11##:1000	200				200				200		2	20
4	##00:0011	200				200		1	20	180		2	18
	Environment	0	0111			20				20			

---------- t = 3 ---------- ---------- t = 4 ---------- ----- t = 5 ------

	Classifier	Strength	Message	M	Bid	Strength	Message	M	Bid	Strength	Payoff
1	01##:0000	220				220				220	
2	00#0:1100	218				208				208	
3	11##:1000	180	1000			196				196	
4	##00:0011	162		3	16	156	0011			206	50
	Environment	20				20					

*when two bids are equal, tie broken by random selection.

between positions k+1 and t inclusively. For example, consider strings A1 and A2:

A1 = 0 1 1 0 | 1

A2 = 1 1 0 0 | 0

Suppose in choosing a random number between 1 and 4, we obtain a k=4 (as indicated by the separator symbol |). The resulting crossover yields two new strings:

A1' = 0 1 1 0 0

A2' = 1 1 0 0 1

Mutation, the final G.A operator, is the occasional (with small probability) random alteration of the value of a string position. It is an insurance policy against premature loss of important notions.

7. FAULT TREE CHARACTERISTIC PARAMETERS

In producing a learning system to relate a specific tree structure to the best ordering scheme for the basic events to produce the most efficient BDD, it is essential to code characteristics defining the fault tree structure in some form. Several characteristics can be used to represent the significant features of the trees. Figure 3 indicates a simple fault tree structure with common elements labelled.

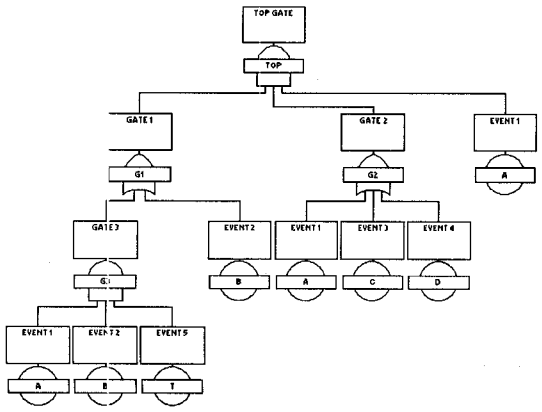

Figure 3: A simple fault tree with labelled elements

Table 2 shows a list of possible characteristics which could be used to define a tree structure, with example numerical values relating to the fault tree illustrated in figure 3. By examining the combinations of these factors a fault tree structure can be described.

	Characteristic	Eg. from fig.3
1	Total number of gates	4
2	Total number of AND gates	2
3	Total number of OR gates	2
4	Total number of basic events	8
5	Number of different basic events	4
6	Number of repeated events	2 (A,B)
7	Largest number of single repeated event	3 (A)
8	Number of levels in tree	3
9	Lowest level of repeated event (nr. top event)	1 (A/B)
10	Number of subtrees from top event	3 (G1,G2, A)
11	Highest number of levels in subtree	2 (G1)
12	Number of different repeated events in largest subtree	1 (B)
13	Highest number of single repeated events in subtree	2 (B)
14	Number of subtrees which contain repeated events within them	1

Table 2: Potential Fault Tree Characteristics.

The list in Table 2 is not exhaustive, and it is clear that to take account of all possible characteristics would be extremely slow computationally and require a great deal of data. Hence a select few need to be extracted which represent the most significant tree features with relevance to the BDD construction. In this study six characteristics were chosen as an initial starting point to investigate the potential of the classifier system.

The classes selected relate to the number of gates and the basic events. The number of gates relate to characteristics 1 - 3 in table 2. To reduce the number of parameters to a single coding it is wise to consider the percentage number of one of the gates, namely characteristic 1 - percentage of AND gates in tree. To incorporate classes 4 - 6 (in table 2) into two parameters, the percentage of different events that were repeated and the percentage of the total events that were repeated were considered, becoming characteristics 2 and 3 respectively.

To reduce the tree to a standard form all trees were converted to an alternating AND/OR gate structure. The start gate now determines the pattern of the gates throughout the whole tree. For this reason the gate type of the top event was chosen as characteristic number 4. To examine the size of the tree structure, two characteristics to consider were, the number of outputs from the top gate (characteristic number 10 in table 2) and the number of levels of the tree (characteristic 8 in table 2). These were chosen as characteristics 5 and 6 respectively. These characteristics were then converted to a binary representation.

This small group of six characteristics have been selected as a starting point to investigate the worth of the classifier system for the task described.

8. POTENTIAL ORDERING SCHEMES

As previously mentioned the ordering placed on the basic events of a fault tree will determine the size of the resulting BDD[17-19], and hence the number of cut sets. It is beneficial to achieve an ordering which is optimal in terms of the resulting size of the BDD.

It is clear that there is no limit to the number of methods of variable ordering available. In previous research[20] 6 different ordering schemes which lead to a reduction on the size of the resulting BDD for different tree structures, have been identified. These are:

1) Top-down, left-right approach,
2) New Top-down, left-right approach,
3) Depth first approach,
4) New Depth first approach,
5) Priority depth first approach,
6) New Priority depth first approach,

An example fault tree shown in figure 4 has been used to illustrate the differences between the variable ordering schemes. The results are given in table 3.

Ordering scheme number 1 is the most commonly used, and is produced by listing the variables in a top-down , left-right manner from the original fault tree structure. The new top-down, left-right approach is very similar. However at each gate the basic events inputs are listed with repeated events first. If the gate has more than one repeated event as an input then the most repeated event is placed first, if they occur the same number of times then the events are taken in gate list order to break the tie. Also if an event has been ordered due to its occurrence higher up the tree then it is ignored for the ordering as an input to the gate (as in the original top-down approach).

Ordering scheme 3 for investigating the basic event ordering involved breaking the whole tree structure into smaller trees (subtrees) and looking at the optimal ordering of these subtrees. The depth first ordering scheme gives each subtree a top-down, left-right ordering, working from the first gate inputs of the top event. The new approach to this ordering (scheme 4) is the same except that repeated events are considered first (as in the new approach number 2).

The final two ordering schemes take the depth first approach one step further. Experience shows that basic events which lie higher up the tree have greater influence, therefore in the final two schemes, subtrees with only basic event inputs take preference. This is priority depth first ordering. The new scheme (scheme 6) takes into account the repeated events as in the previous new approach.

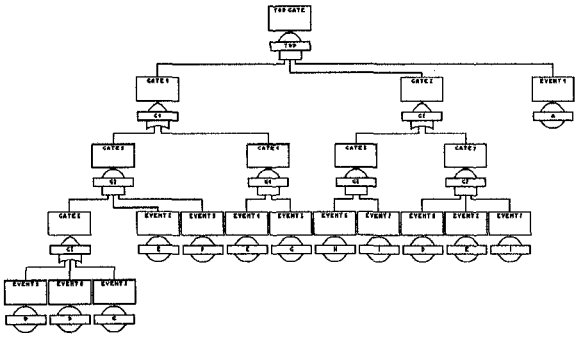

Figure 4: An example fault tree to illustrate the different ordering permutations

No	Scheme	Variable Ordering
1	Top-down, left-right	A<E<F<C<G<H<I<D<B
2	New Top-down, left-right	A<E<I<F<C<G<H<D<B
3	Depth-first	E<F<C<G<B<D<H<I<A
4	New Depth-first	E<F<C<G<B<D<I<H<A
5	Priority Depth-first	C<G<E<F<B<D<H<I<A
6	New Priority Depth-first	C<G<E<F<B<D<I<H<A

Table 3: Order schemes

This short list of schemes have been adopted to investigate the potential of the classifier approach.

9. INPUT/OUTPUT VARIABLE CODING

Each classifier message and condition comprises of 25 bits, which can be broken down to two main sections, namely: the characteristics coding and the scheme coding. The characteristics coding comprises 19 bits and is broken down to:

4 bits for percentage of and gates;
5 bits for percentage of different events repeated;
5 bits for percentage of total events repeated;
1 bit for top gate type;
2 bits for number of outputs from top gate;
2 bits for number of levels of tree.

The scheme coding comprises of 6 bits of 0's and 1's where a 1 represents the best scheme(s) and a 0 otherwise.

10. GENERATION OF TRAINING/TEST DATA

The training data set of fault tree structures came from industry and also by random production using a computer program. Each tree structure was analysed for the chosen characteristics, and these characteristics were converted to the appropriate binary representation.

Each tree was analysed prior to training for the best ordering scheme for the most efficient BDD representation. The best scheme was identified by the minimum number of nodes in the BDD structure before minimization (removal of redundant nodes). (Other ways of selecting the best, such as the least number of non-numerical cut sets are possible.)

To evaluate the performance of the learning classifier system a test set of data was produced with different tree structures and known best ordering schemes. The schemes for prediction purposes are set to wildcard characters. The performance is evaluated by comparing the number of correct scheme outputs predicted.

11. CLASSIFICATION SYSTEM APPROACH

Results to date have shown that testing each scheme separately produces improved accuracy. This involves training a classification scheme for each of the different ordering options.

To run the prediction program a tree is tested against each scheme classification system, starting at scheme 1, and producing an output of scheme 1 or not. Trees which produce a negative result are then further processed against the other ordering options until one scheme is chosen.

CONCLUSIONS

1. Initial work using classifiers has indicated that this method could be trained to predict the best of alternative ordering schemes to use on a fault tree structure yield an efficient BDD representation.
2. It I anticipated that the predictions made using the classifiers can be further improved by generating more training data.
3. As expected, the initial results indicate that the small group of factors usual in this preliminary study to define the fault tree structure do not adequately represent the complexity of the problem and other characteristics now need to be considered to develop the true potential of the method.

REFERENCES

1. Vesely W.E., "A Time Dependent Methodology for Fault Tree Evaluation", Nuclear Eng and des., 13, 1970, pp337-360.

2. Bryant R.E., "New Algorithms for Fault Tree Analysis", Reliability Engineering and System Safety, vol 40,1993, pp203-211.

3. Sinnamon R.M. and Andrews J.D., " New Approaches to Evaluating Fault Trees", Proceedings of ESREL 95 conference, June 1995.

4. Sinnamon R.M. and Andrews J.D., "Fault Tree Analysis and Binary Decision Diagrams", Proceedings RAMS 96, Las Vegas, Jan 96.

5. Rauzy A., "A Brief Introduction to Binary Decision Diagrams", European Journal of Automation", Vol 30, No 8, 1996.

6. Sinnamon R.M. and Andrews J.D., "Quantitative Fault Tree Analysis using Binary Decision Diagrams", European Journal of Automation", Vol 30, No 8, 1996.

7. Dugan J.B. and Doyle S.A., "Incorporating Imperfect Coverage into Binary Decision Diagrams", European Journal of Automation", Vol 30, No 8, 1996.

8. Boussiou M., "An Ordering Heuristic for building Binary Decision Diagrams from Fault Trees", Proceedings RAMS 96, Las Vegas, Jan 96.

9. Sinnamon R.M., "Fault Trees and Binary Decision Diagrams", PhD Thesis, 1997.

10. Goldberg D.F., "Genetic Algorithms in Optimization and Machine Learning", Addison Wesely 1997.

11. Quinlan J.R., "Discovering rules from large collections of examples: a case study", Expert Systems in the Macro Electronic Age (D. Michie Ed), Edinburgh University Press.

12. Bishop C.M. "Neural Networks for Pattern Recognition", Clarendon, 1995.

13. Eberhart, Simpson and Dobbins, "Computational Intelligence PC Tools", AP, 1996.

14. R. Davis, J. King, "An overview of Production Systems." In E.W. Elcock and D. Michie (Eds.), Machine Intelligence 8 (pp300-332), 1976. New York: Wiley.

15. K.A. DeJong, "Genetic Algorithms: A 10 year perspective," Proceedings of an International Conference on Genetic Algorithms and Their Applications, 1985, pp169-177.

16. L. Davis, Handbook of Genetic Algorithms,

17. R.E. Bryant, "Graph-Based algorithms for Boolean function," IEEE Trans. Computers, vol C-35, 1986, No. 8, pp667-691.

18. S.B. Akers, "Binary decision diagrams," IEEE Trans. Computers, vol C-27, 1978, No. 6, pp509-561.

19. A. Rauzy, "New algorithms for fault tree analysis," Reliability Engineering and System Safety, vol 40, 1993, pp203-211.

20. R.M. Sinnamon, J.D. Andrew, "Improved Efficiency in Qualitative Fault tree analysis", Advances in Reliability Technology Symposium, Manchester, 1996.

BIOGRAPHY

John D Andrews
Department of Mathematical Sciences
Loughborough University
Loughborough, LE11 3TU, U.K.
E-mail: J.D.Andrews@lboro.ac.uk

Dr Andrews is a Senior Lecturer in the Department of Mathematical Sciences at Loughborough University. He joined this department in 1989 having previously gained nine years industrial research experience with British Gas and two years lecturing experience in the Mechanical Engineering Department at the University of Central England.

His current research interests concern the assessment of the safety and risk of potentially hazardous industrial activities. This research has been heavily supported by industrial funding. Over the last five years grants have been secured from the Health and Safety Executive, Mobil North Sea, Bechtel , Shell UK. Exploration , London Underground, Lloyds Register and British Gas. Currently Dr Andrews has a team of five research students,. Numerous journal/conference publications have been produced along with a jointly authored book "Risk and Reliability Assessment".

Lisa M Bartlett
Department of Mathematical Sciences
Loughborough University
Loughborough, LE11 3TU, U.K.
E-mail: L.M.Bartlett@lboro.ac.uk

Lisa Bartlett is a research student in the Department of Mathematical Sciences at Loughborough University. She gained a first class honours degree from Loughborough University in 1997. She is conducting research into Fault Tree Analysis methods which focuses on the Binary Decision Diagram approach.