

# Automated Fault Tree Generation: Bridging Reliability with Text Mining

Saikat Mukherjee, PhD, Siemens Corporate Research

Amit Chakraborty, PhD, Siemens Corporate Research

Key Words: fault tree synthesis, text mining, natural language processing

## *SUMMARY & CONCLUSIONS*

Proper preventive maintenance of complex systems, such as those used for power generation and medical diagnosis is dependent on the availability of their up-to-date reliability models. These models are constructed from historical maintenance and fault information of the equipments. Due to the complex nature of these machines, constructing these models involve significant manual effort which limits the widespread use of reliability-centric maintenance schemes. In this paper, we describe a process for automating the construction of fault trees, a class of non-state space reliability models, by analyzing maintenance data available as free-form text. It uses a combination of linguistic analysis and domain knowledge to identify the nature of the failure from short plain text descriptions of equipment faults. This information is used to automatically enrich and evolve existing fault trees for better reliability estimation.

## *1 INTRODUCTION*

Modern industries make extensive use of complex machines. These machines serve a variety of purposes ranging from routine task automation to accomplishing sophisticated functions. Some companies are consumers of such complex equipments and then there are the manufacturers. For instance, in the energy sector, companies such as Mitsubishi, GE and Siemens manufacture complex turbines which in turn are used by the electric utilities. Similarly, in the medical industry, manufacturing companies produce CT and MR scanners which are bought by healthcare providers.

These machines, such as turbines for generating power or scanners for medical diagnosis, have some fundamental characteristics in common:

- They are expensive to build and often require significant investment from the manufacturers.
- As a direct consequence of the above, they are also expensive to buy and operate. For instance, only customers such as power utilities or healthcare providers who have the requisite capital can afford to purchase them.
- They often have long life spans. For example, in the power generation industry, gas and steam turbines often function for more than 25 years.

- Doing repair work on such machines by completely shutting them down is prohibitively expensive since it grinds entire operations, which are often critical to the business, to a halt. For instance, in the power industry it is very expensive to completely shut down a turbine and perform maintenance work on it since this could trigger a complete outage.
- Often the problems faced by these machines have similarities to each other within a class. For instance, there will be similarities between the host of turbine related problems faced by different power utilities.
- Repairing the problems usually involves sophisticated expertise from service organizations.

Equipment failures can be very expensive for the owners since they not only involve operational loss of the machine but also costly repairs.

The unique demands of these types of equipment have resulted in certain responses from industry:

- Due to the large post-failure repair costs involved, *preventive maintenance* techniques are often used which help in reducing servicing costs and increase operational life. Preventive maintenance involves using reliability estimates, such as mean time to failure (MTTF) and system reliability/availability, for predicting failures and scheduling servicing before an actual breakdown occurs.
- In order to exploit similarities among problems and maintenance work across multiple customers, manufacturers started building and maintaining *knowledge repositories* within their service organizations. These repositories typically contain problem descriptions from customers and the engineering responses to them.

The essence behind preventive maintenance is to estimate reliability measures which serve as indicators of likely maintenance points. Thus, instead of servicing equipment when a failure occurs, preventive maintenance determines service points before explicit failures. This reduces the likelihood of a complete machine shutdown.

Reliability measures are computed using mathematical models, such as fault trees, Petri Nets, Markov models, of the systems and their failure events. There are two primary prerequisites for successfully applying these models to

preventive maintenance. First, these models have to be constructed from available data. Second, the parameters of these models have to be estimated. There has been significant work on parameter estimation from reliability data and numerous techniques have been developed. However, obstacles in *model construction* have hampered the easy deployment of preventive maintenance techniques.

Constructing these models often require significant domain expertise and manual effort. For instance, non-state space models such as fault trees can be used only after defining the hierarchical relationships between the root causes and the overall system failure in terms of intermediate failure events. Complex systems can have large fault trees and enumerating all the error possibilities hierarchically becomes a challenging task. State space modeling, using stochastic Petri Nets and Markov processes, involves even further domain expertise due to the possibility of complicated relationships between states. Furthermore, synchronizing any of these models with the changing dynamics of product life-cycles becomes dependent on the availability of domain experts. Hence, *automating* the construction of reliability models is an important enabler [1,3] which can lead to the widespread deployment of reliability-centric maintenance systems in industry.

To improve their efficiency, service organizations often maintain knowledge repositories of problems and solutions. These repositories contain information about the machine itself, its component hierarchy, as well as reports on problems and the final analyses of the problem resolution implemented by the service engineers. Typically, these reports exist as a mix of unstructured text (free text) as well as structured relational databases. These knowledge repositories within the service organizations are crucial for more automated model construction.

In this paper, we discuss a technique for automating the construction of reliability models by analyzing service data. While it is our ultimate objective to automate this process for a variety of reliability models and from a wide variety of service data, in this paper we have limited ourselves to a single class of reliability model and a particular kind of service data. In particular, we try to learn *fault tree reliability models from service data*.

The rest of the paper is organized as follows: in the following section, we describe the nature of the service data and provide a brief introduction to fault trees. In Section 3, we describe the analysis technique in details. In the last two sections, we discuss related work, preliminary conclusions, and future directions.

## 2.1 Fault Tree Modeling

Fault tree analysis (FTA) and Fault tree synthesis (FTS) evolved primarily within the aerospace and nuclear industries, and have been extensively used in fault diagnosis and systems safety analysis. Fault trees and reliability block diagrams (RBD) belong to the class of non-state space models.

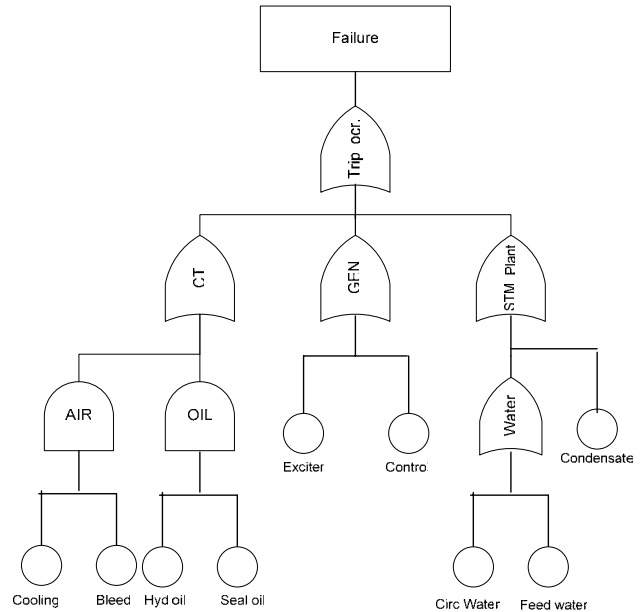


Figure 1: Fault Tree

Fault trees start with a failure event or hazard rate and trace its influence back until the basic influence factors (root causes) are reached. This resulting influence hierarchy is depicted as an upside-down tree with the failure event as the top-level event. In an event of system failure, a fault tree can predict what the most likely causes of the failure are by evaluating all combinations of basic events (e.g. component failures). When the system complexity increases, hierarchical decomposition is used to solve the system fault tree. Intuitively, a fault tree decomposes a problem hierarchically into sub-problems associating AND/OR relationships, among others, between a parent and its immediate children. An AND relationship between a node and its children indicates that all the children nodes together are responsible for its problem while an OR relationship signifies any one of the children could have caused the problem. For example, Figure 1 shows a fault tree with failure events in combustion turbines generators, and steam plants.

Analysis of a fault tree can generate valuable system insights such as reliability, availability and mean time to failure (MTTF). This in turn can be used for designing

efficient preventive maintenance schemes.

Fault tree synthesis involves construction of fault trees. Despite their simplicity, fault trees suffer from the problem of requiring significant manual effort in model building. Constructing fault trees require knowledge of the problems as well as their root-cause analysis. Typically, this information is obtained from domain experts and, hence, model building becomes critically dependent on their availability. Moreover, complex systems can have large fault trees and enumerating all the error possibilities hierarchically becomes a challenging task. Another issue to keep them updated through the product life-cycle. To obviate this, we propose a fault tree development scheme as shown in Figure 2. Inputs to the system are the “Bill of Materials” for the machine under consideration, service data as “input text” and certain “Domain Rules” that embody the available domain information. In addition, the WordNet Knowledgebase is also used as a reference input. In the following the process is discussed in further details.

### 2.2 Service Data

Figure 3 illustrates the nature of the service data which we are using for automated fault tree construction. Observe that the data is a combination of free text as well as structured and corrected information.

In order to limit the complexity of the analysis algorithm, our technique is focused on short text strings (usually containing around 5-10 words) rather than larger chunks of text such as paragraphs. Figure 3 shows a fragment of a typical maintenance data where the column *Cause* contains the plain text description of the root cause while the columns *Area*, *System*, and *System2* contain information on the system component where the failure occurred. Note that while the *Area*, *System*, and *System2* columns can be used to generate an initial fault tree, it would lack sufficient expressiveness without using the root causes.

Almost all service organizations maintain databases which are similar to the one shown in Figure 3. In some cases, the data might contain even more free text (unstructured) while in other situations it might be more structured with the root cause explicitly identified. Our analysis technique, described in the following section, makes use of service data where some part of the knowledge is not explicitly identified.

## 3 ANALYSIS TECHNIQUE

Our technique is focused on data similar to Figure 3 and reliability models of the type shown in Figure 1. It takes an existing fault tree, which could have been manually constructed, and extends it with new events identified by analyzing short plain text descriptions of service problems and solutions. In order to do this, the technique uses a combination

of linguistic analysis and domain knowledge to automatically identify the nature of failure from these descriptions of equipment faults. The different steps in our analysis are described in the following subsections.

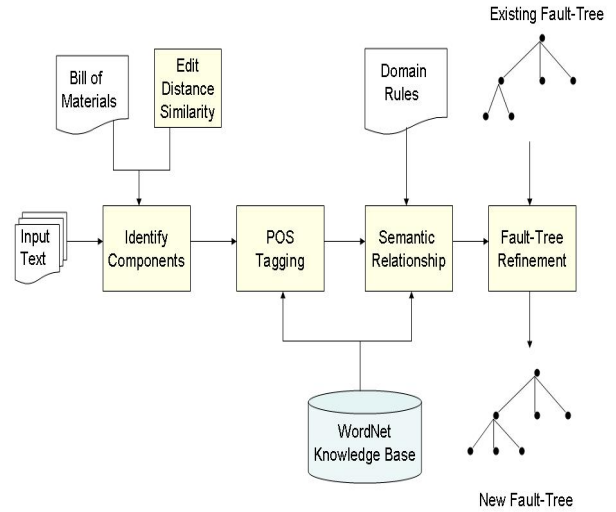


Figure 2: System Architecture

| Area        | SYSTEM        | SYSTEM 2        | CAUSE  |
|-------------|---------------|-----------------|--|
| CT          | Cooling air   |                 | RAC Electric motor showed short circuit and CT tripped manually.     |
| CT          | Cooling air   | Rotor Cooling   | Rotor Cooling air high temperature trip                              |
| CT          | Bleed Air     | Bleed           | Momentarily lost LP Bleed valve position indication                  |
| CT          | Hydraulic oil |                 | Pilot gas valve out of position                                      |
| CT          | Hydraulic oil |                 | failed servo valve on pilot gas control valve                        |
| Generator   | Controls      |                 | ST GEN TRIPPED ON REVERSE POWER                                      |
| Generator   | Controls      |                 | Rotor EFF protection caused by steam leak at 60LBA10AA416            |
| Generator   | Exciter       |                 | external protection fault  |
| Generator   | Controls      |                 | tripped due to instantaneous load increase to 40 MWM on start-up     |
| Steam_Plant | Feedwater     | Unknown         | Feed pump run out  |
| Steam_Plant | Feedwater     | Unknown         | Loss of LAC pumps  |
| Steam_Plant | Feedwater     | Unknown         | Loss of LAC pumps and low suction pressure                           |
| Steam_plant | Feedwater     | IP Bypass       | IP-Bypass flow protection due problems with the bypass control valve |
| Steam_Plant | Condensate    | Unknown         | Tripped due to Low Condensate Pump suction pressure                  |
| Steam_plant | Condensate    | IP attemperator | IP Bypass attemperation valve issues.                                |

Figure 3: Service Data for turbines, generators, and steam plants

### 3.1 Initial Fault Tree Construction

As mentioned before, an initial fault tree is constructed from the *Area*, *System* and *System2* columns of the service data. For instance, for generators, this initial fault tree could be of the nature:



An indentation in the above denotes sub-events. Thus, a

failure in a generator occurs due to failure either in the Control, Exciter or the Seal Oil section. A Seal Oil failure can either be explained by a failure in the Transformer or a failure in the Vibration section. This initial fault tree can be constructed either by reading the service database when such information is present or could be manually constructed by domain experts. Similarly, the initial fault tree for combustion turbines could be:

|  |
|--|
| <p><i>CT</i></p> <ul style="list-style-type: none"> <li><i>Blow off Air</i> <ul style="list-style-type: none"> <li><i>Blade Path</i></li> <li><i>Burner Temperature</i></li> <li><i>Wiring</i></li> </ul> </li> <li><i>Exhaust</i> <ul style="list-style-type: none"> <li><i>Emulsion Water</i></li> <li><i>Seal Water Pump</i></li> <li><i>Valve</i></li> </ul> </li> </ul> |
|--|

### 3.2 Component Identification

Our technique takes these initial fault trees and extends them with root cause events automatically extracted from the text of *Cause*. These events are binary relationships of the form  $\langle Event1, Event2 \rangle$  where *Event1* and *Event2* are two related events such that either *Event1* is due to failure in *Event2* or *Event2* is a kind of failure in *Event1* itself. Note that in the former case, both the events are equipment components while in the latter only *Event1* is a component.

As evident from the above discussion, the technique hinges on correct identification of machine components in the text description. There are two ways in which components are identified:

- *Bill of Materials Knowledge*: The Bill of Materials (BOM) of a machine is a hierarchical decomposition of its parts. This is an invaluable piece of knowledge for complex equipments. Since the BOM lists the different components, our first step is to identify their presence in the text description. However, a simplistic character matching scheme is often not sufficient due to slight differences in representation of the same component in the BOM and in a user’s description. Hence, we use an *edit-distance* based similarity detection algorithm to match descriptions to BOM components.
- *Linguistic Analysis*: Often, the BOM by itself is not enough to identify system components. Furthermore, when *Event2* is not even a component but a failure type, such as “trip”, then further analysis methods are needed. We use linguistic analysis in these situations. Research in natural language processing [6] over the last decade has made significant progress in understanding free text. The following two subsections, POS Tagging and Semantic Relationship, describes how we use natural language

processing techniques for fault tree construction.

### 3.3 Part of Speech (POS) Tagging

The “part of speech” of a word identifies its *sense* in a sentence. A sense of a word indicates whether it is used as a noun, or a verb, or an adjective, etc. In order to determine the sense, it is necessary to parse the sentence using a grammar for English. The correctness of these parsers depends to a large extent on the formulation of the sentence. However, a significantly large portion of our descriptions are malformed or incomplete sentences and, hence, parsing does not always succeed.

In these cases, we use WordNet [2] for POS tagging. WordNet is an online thesaurus of the English language. It provides semantic relationships, such as synonyms, hypernyms, hyponyms, etc., between different words as well as part-of-speech (POS) information such as noun, verb, adjective, etc. of the different usages of words. The POS indicates whether a word has been used as a system component or as a failure type of a component. In particular, words with a noun POS are potential components while those with a verb POS signify failure types. However, since WordNet is independent of any particular sentence, very often a word would be associated with multiple POS tags. To disambiguate this, we use the simple heuristic of associating a word with multiple POS tags to the tag with the maximum senses listed. For instance, if word *w* in WordNet has a noun POS tag, with 5 senses, and also a verb POS tag, with 2 senses, then *w* will be associated to the noun POS tag.

The heuristic described above follows a majority principle and, while successful over large data sets, can be improved further when the description is a valid sentence which can subsequently be parsed with an English grammar. The next module describes how this is done.

### 3.4 Semantic Relationship

This module performs two main functions:

- *Parsing*: We use the OpenNLP [7] toolkit for detecting and parsing sentences. When a sentence can be syntactically parsed, the *noun phrases* and the *verb phrases* in it are extracted. These phrases are combinations of nouns with adjectives, etc. or verbs compounded with parts of speech. They denote a block of homogeneous content. While a noun phrase is always found, if in addition a verb phrase is also found then a relationship of  $\langle Event1, Event2 \rangle$  where *Event2* is a failure type is extracted. If two noun phrases are found, then a relationship of  $\langle Event1, Event2 \rangle$  where *Event2* is a component is extracted.
- *Relationships*: Since the same root cause could be variously expressed in different descriptions, it is

necessary to establish semantic similarities between the extracted relationships. In our current system, we check for synonym relationships between words. For a pair  $\langle Event1, Event2 \rangle_a$  and  $\langle Event1, Event2 \rangle_b$  of relationships, we use WordNet to check for synonymy between  $Event1_a$  and  $Event1_b$  and  $Event2_a$  and  $Event2_b$ . This prevents repetition of similar events in the fault tree modeling.

### 3.5 Fault-tree Refinement

Finally, the relationships mined from the descriptions in the previous sections are used to extend an existing fault tree. This is done using each of the extracted  $\langle Event1, Event2 \rangle$  relationships. In the presence of the  $Event1$  in a current fault tree, the  $Event2$  clause is used to extend it. The presence is checked using an edit distance similarity measure to tide over slight variations. The  $Event2$  is added with OR semantics if  $Event1$  already has children events in the fault tree.

In the absence of  $Event1$  in the fault tree, knowledge from a Bill of Materials part hierarchy is used in adding  $Event1$  as an event. Specifically, the event in the fault tree which is closest to  $Event1$  in terms of the BOM is chosen as the parent of  $Event1$ . The  $Event2$  specifies further the problem nature for  $Event1$  in the extended fault tree.

We illustrate the process given the initial fault tree for Generator and the description “Broken fuse on volt transformer”. Assuming no match was found against the BOM, this description is parsed in the Semantic Relationship module according to:

(Noun-Phrase (Adjective Broken) (Noun fuse) (Particle on) (Noun volt) (Noun transformer))

Observe that even though it is not a fully formed sentence, the parser is still able to distil the essential information. The relationship extracted from this description is  $\langle \text{“Fuse”, “Fuse Transformer”} \rangle$ . Using edit distance matching,  $Fuse Transformer$  is mapped to the event Generator  $\rightarrow$  Seal Oil  $\rightarrow$  Transformer, and hence the new extension to the fault tree is the event “Fuse” as a sub-event of Transformer.

As another example, consider the description “Blade Path Spread”. Matching against the BOM identifies “Blade Path” as a component and “Spread” as an event with verb POS from WordNet. Thus, the relationship is  $\langle \text{“Blade Path”, “Spread”} \rangle$  and the event Blade Path in the fault tree for CT is made a parent of the event Spread.

## 4 RELATED WORK

Note that most reliability models are in many ways similar to finite state automata with the added capability of probabilities on transitions. Automated model construction has been a topic of research within computer science for a long time [4, 5]. In general, it is recognized that learning the minimal size automaton that accepts all positive examples is known to be NP-complete. Hence, efficient learning of succinct automata is a known hard problem. Consequently, different kinds of heuristics have been proposed for this problem.

While similar to standard automata synthesis in some respects, our problem is also different in many ways. We are interested in *analyzing textual service* data for model construction. In contrast, traditional automata synthesis attempts to learn from strings of elementary symbols. The difference in the nature of the modality involved makes the direct application of traditional work very difficult.

In the reliability engineering domain, the emphasis has been traditionally on better parameter estimation from data. As we have pointed out at the beginning of this paper, often such work does not result in widespread acceptance of reliability methods by the industry due to the bottleneck involved in model construction. Previous work [1, 3] realizes this problem and addresses this issue. However, the solutions proposed involve better system interfaces to help the domain expert construct fault tree models. In contrast, we are leveraging service data to propose a more automated technique. While the interaction with domain experts should never be ruled out, incorporating more automation leads to more scalable model building.

## 5 DISCUSSIONS

The previous sections describe a technique for automatically enriching and evolving existing fault trees by analyzing short textual descriptions of maintenance events. The novelty of our technique lies in applying concepts from text mining and natural language processing to the field of system reliability for more automated model construction.

We are in the process of implementing a prototype system based on the technique described here and evaluating it on real-world maintenance data. For the purposes of this experiment, our data set consists of approximately 1000 fault descriptions of steam turbines, gas turbines, and generators. Each of these descriptions is of the form shown in Figure 3. Preliminary evaluation confirms the validity of the technique. Specifically, we are evaluating: (a) the precision of the system – the correctness of the extensions to the fault tree proposed by the system, and (b) the recall of the system – of the total

number of extensions possible, how many were discovered by the technique. The technique has better recall than precision since a deliberate objective was not to compromise on yield but rather have a domain expert improve the precision. Full experimental evaluation will result in more detailed analysis of the technique.

In this paper, we address one of the core problems of reliability centric maintenance, namely the automatic development of reliability models from available service data. Our solution synergizes novel techniques from natural language processing and text mining with service knowledge bases. While, as of now, the technique is geared towards automatically enriching fault trees with information mined from short free text descriptions of root cause analysis, in the future we expect to extend our work in three major directions:

- *Fully automated complete fault tree construction* – Currently, we start with a manually constructed fault tree and extend it with events mined from free text. In future, we would like to extend it to fully automated model construction.
- *Mining more unstructured service data* – instead of being restricted to short text descriptions of service problems and solutions we would like to investigate more free-form text data containing paragraphs of problem descriptions from customers and corresponding solutions from engineers.
- *More expressive reliability models* – our current technique is tied to fault trees. However, state of the art reliability analysis often uses more sophisticated models such as reliability block diagrams and Markov chains, etc. We would like to investigate the feasibility of extending our ideas, if not the exact technique, to these other reliability models.

It is our belief that the fusion of technologies from different domains, as laid out in this paper, will address the core problem of automatically constructing reliability models which in-turn can facilitate the widespread deployment of reliability centric tools among industry practitioners.

#### REFERENCES

1. P. Gaspar, G. Szabo, “Automatic Fault-Tree Generation as a part of a Complex Development System” *International Scientific Conference Elektro*, 1999.
2. G. Miller, R. Beckwith, C. Fellbaum, D. Gros, “Wordnet: An On-line Lexical Database”, *Intenational Journal of Lexicography*, 3(4), 1990.
3. I. Renault, M. Piliere, N. Villatte, P. Mouttapa, “KB3: Computer Program for Automatic Generation of Fault Trees”, *Proceedings of Annual Reliability and Maintainability Symposium (RAMS)*, 1999.
4. J. Hopcroft, R. Motwani, J. Ullman, “Introduction to Automata Theory, Languages, and Computation”, Addison Wesley, 2000.

5. K. Murphy, “Passively Learning Finite Automata”, Technical Report, 1996.
6. D. Jurafsky, J. Martin, “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition”, Prentice Hall, 2000.
7. OpenNLP Free Software, <http://opennlp.sourceforge.net>

#### BIOGRAPHIES

Saikat Mukherjee, PhD  
Integrated Data Systems Department  
Siemens Corporate Research  
755 College Road East  
Princeton, NJ 08540 USA

e-mail: saikat.mukherjee@siemens.com

Saikat Mukherjee is a Research Scientist in the Integrated Data Systems Department at Siemens Corporate Research. His research interests are in text analytics, machine learning, and natural language processing. He is particularly interested in the applications of these techniques to different areas such as reliability based maintenance and problems in service data analysis. Within the area of reliability, he has been working with Siemens Westinghouse Power Corporation to bring some of the broad ideas discussed in this paper, such as the use of text analytics for model construction, into reality. He has a PhD in Computer Science from the State University of New York, Stony Brook, and a Bachelors from Indian Institute of Technology, Kharagpur.

Amit Chakraborty, PhD  
Integrated Data Systems Department  
Siemens Corporate Research  
755 College Road East  
Princeton, NJ 08540 USA

e-mail: amit.chakraborty@siemens.com

Amit Chakraborty is a Program Manager in the Integrated Data Systems Department at Siemens Corporate Research. His research interests include robust information fusion, resource constrained planning and scheduling, and reliability and availability analysis. He heads a program in Siemens Corporate Research which is looking at different problems in service optimization. As part of this focus, he is interested in applying reliability centric maintenance techniques within different Siemens operating companies such as Power Generation, Siemens Medical, and Siemens VDO Automotive. He received his PhD. from Yale university and Bachelors from Indian Institute of Technology, Kharagpur, in Electrical and Computer Engineering.