# Root Cause Analysis using Artificial Intelligence

Asha Chigurupati, Google X

Noah Lassar, Google X

Key Words: Root Cause Analysis, Complex Systems, Machine Learning, Weibull

## *SUMMARY & CONCLUSIONS*

A complex hardware system experiences several different types of failure modes when deployed in its field of use. Although each failure mode merits its very own root cause analysis, experience has shown us that a majority of root cause analyses require us to answer a very similar set of questions. These recurring questions often use very similar type of field data to be answered. For example, usage parameters such as hours of operation, total power cycles, vibration fatigue, etc. are used almost always while trying to figure out the failure mechanism. By taking advantage of this, we can automate the process of finding the most likely failure mechanism for a given failure mode.

In the current work, we start by computing what we believe are parameters that are most relevant across all different types of failure modes. Once these parameters are computed, we will build a Bayesian Network to model the cause-effect relationship between the degradation parameters (cause) and failure modes(effect) that occur on the field. Bayesian Network is a probabilistic graphical model which concisely describes the relationship between many random variables and their conditional independence via an acyclic directed graph. Once the topology of the Bayesian network is laid out by a domain expert, the conditional probabilities can be learnt from existing data, thereby completely describing the system using the notion of joint probabilities. Two real life field issues are used as examples to demonstrate the accuracy of the network once it is modelled and built.

This paper demonstrates that accurately modelling the hardware system as a Bayesian Network substantially accelerates the process of root cause analysis.

## *1. INTRODUCTION*

Artificial Intelligence, by definition, is a machine that exhibits rationality that is typically associated with human intelligence, and interacts with its environment via thought processes and reasoning which lead to a successful outcome. AI is relevant to any intellectual task as the sole purpose of the algorithm is to replace processes of reasoning which are otherwise performed by a human. In the current work, we build an algorithm which performs root cause analysis for the user.

Automating discovery of failure patterns in failure data sets becomes very important when

- Failure Modes and Failure Mechanisms are innumerable, making it impossible for engineers to manually keep track of patterns
- Failure data is precious and difficult to obtain, compelling engineers to mine any knowledge that the failure patterns might offer
- Timelines to get to the root cause get progressively tighter

Research today on Bayesian Networks shows its success in applications ranging from Machine diagnosis, Robotics, Vision and Planning to Medical Diagnosis. In this paper, we describe our work on modeling a Bayesian Network in order to use it as a tool to diagnose hardware failures on the field.

Although rootcause analysis can never be fully automated, there are several advantages to constructing a *diagnostic Hybrid Bayesian Network* that captures the cause-symptom relationship within the hardware system. Some of these advantages are:

- Ability to take into account a very large number of causes, and their combinations, while assessing the root cause of a symptom
- Concise representation of the system leading to better visualization, insight and understanding
- Readily extendable as new information is available
- Better accuracy and reliability of results

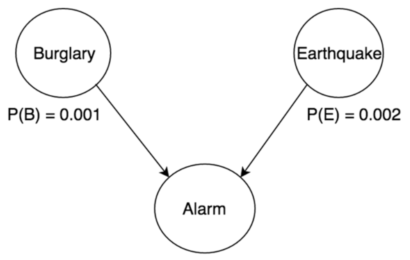
### *1.1 Bayesian Networks: A Primer*

A Bayesian Network is a concise syntactical representation of a joint probabilistic distribution. A network describing relationship between *n* random variables consists of *n* nodes (where each node corresponds to a random variable), and directed arrows connecting one node to another. If an arrow goes from X to Y, X is called the Parent node and Y is its child node. The intuitive meaning of an arrow is that X has a direct effect on Y, which suggests that causes should be parents of effects [1]. Additionally, each node has a conditional probability distribution that quantifies the effects of its parents on the node $\mathbf{P}(X_i|Parents(X_i))$.

Clearly there is a qualitative and quantitative aspect to constructing a Bayesian network.

Encoding the probabilistic influences between domain's random variables as a directed graph by an expert captures the underlying domain knowledge. Borrowing from the classic example, knowing that a burglary causes an alarm to go off will help the expert to direct the arrow from Burglary to Alarm. This is the qualitative aspect of building a Bayesian network.

The quantitative aspect of the problem is the task of assigning the conditional probabilities to nodes, given their parent nodes, that is, **P**$(X_i|Parents(X_i))$. Nodes, or domain random variables, can be deterministic, discrete random variables or continuous random variables. Our problem has all three types of variables, making it a hybrid Bayesian Network. Deterministic nodes do not require any probability modeling, as their value is specified exactly by the values of their parents without any uncertainty. Discrete random variables use a CPT (Conditional Probability Table) to capture the relationship between the parent nodes and child nodes (note that both are discrete random variables). Continuous random variables often use a standard probability distribution function with a finite set of parameters. For continuous variables with continuous parents, Linear Gaussian is commonly used. For continuous variables with discrete parents, Conditional Gaussian is a popular example of the type of probability distributions used. In our work, linearly varying Weibull and Lognormal distribution are a few among the many distributions that have been used. Additionally, we use *Conditional Variants* of these distributions as most of the continuous nodes have both discrete and continuous parent variables.



| P(Alarm) | B | E |
|---|---|---|
| 0.95 | 1 | 1 |
| 0.94 | 1 | 0 |
| 0.29 | 0 | 1 |
| 0.001 | 0 | 0 |

*Figure 1. Network*

*QUERYING THE NETWORK*

A Bayesian Network, once constructed, completely describes the joint probability distribution of the system of random variables. A typical query asks for the posterior probability distribution for a set of *query variables (X),* given a set of evidence variables E=e, i.e., *P(X/e )*. Evidence variables are usually symptoms which are noticed and hence known. Query variables are variables which we would like to learn more about. In the example shown in Figure 1, a typical probabilistic inference query would be: what is the probability that there was burglary given that the alarm rang, or, *P(B/a)*?

There are many probabilistic inference methods which can be used to obtain the required posterior probabilities. Broadly, they are classified as exact inference methods and approximate inference methods. Gibbs sampling, which is a Markov chain Monte Carlo method, is a very popular algorithm for querying Bayesian Networks. In this algorithm, Gibbs Sampling was used for querying posterior probabilities.

The magnitude of P(X/e) is an indicator of the influence of X on producing the evidence E=e. This ease in finding the cause (X) of a given symptom (E=e) is the reason why Bayesian Networks are really popular in the field of Medical Diagnosis.

## 2. *VARIABLES AND NETWORK TOPOLOGY*

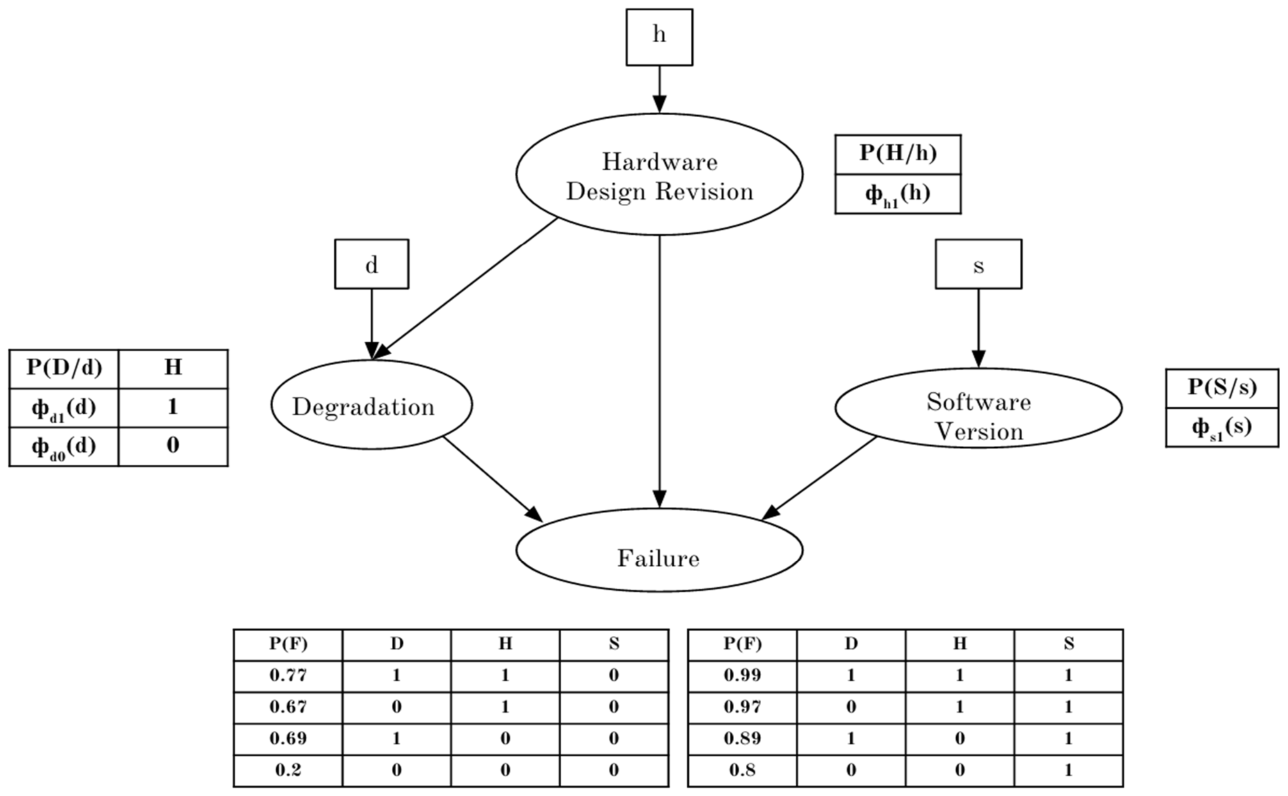
Bayesian Networks make use of *"conditional independence"* to come up with compact and concise structures which can describe really complex interactions between a large number of variables. Many real world scenarios have limited probabilistic dependencies between variables, making it a sparse system. This saves computational cost and increases interpretability. Networks can be constructed by hand or can be learned from data. This Bayesian network was constructed by adhering to the following guidelines:

- Begin the network by adding Root Causes first
- Root Causes are followed by variables that they influence, with an arrow pointed from Root Cause to the variables that they affect
- Repeat this process until you reach the failure modes, which do not have any direct influence on other variables

The Bayesian Network we create has 15 nodes - 5 of them are deterministic, another 5 are nodes with no parents (initial nodes), while the other 5 are child nodes. Each child node has either 2 or 3 parent nodes. Unfortunately, the confidentiality of the project discourages the authors from publishing the exact network topology. Therefore, we will discuss the results obtained from 3 parent nodes and two child nodes.

Figure 2 shows the topology of the abridged network, what each node represents and the influence of variables over each other. A few features to highlight about this network are:

- There is an inherent assumption that all the possible causes are listed. In situations where listing all the causes is not possible, a "*leak node*" can be used to capture miscellaneous root causes.
- The node 'Hardware Design Revision' influences longevity of a part, and is hence the parent of the node 'Degradation'
- The node 'Software Release', or 'Firmware Release', does not influence the other two nodes, but can be responsible for failures, thus becoming the parent of the node 'failure'
    - The node Failure is 1 for any failure mode seen on the field, and is agnostic to the exact failure mode of the system. Therefore, the Bayes Net can be used for any failure mode.
    - The CPT of each node was calculated using field data



| P(H/h) |
|---|
| $\phi_{h1}(h)$ |

| P(D/d) | H |
|---|---|
| $\phi_{d1}(d)$ | 1 |
| $\phi_{d0}(d)$ | 0 |

| P(S/s) |
|---|
| $\phi_{s1}(s)$ |

| P(F) | D | H | S |
|---|---|---|---|
| 0.77 | 1 | 1 | 0 |
| 0.67 | 0 | 1 | 0 |
| 0.69 | 1 | 0 | 0 |
| 0.2 | 0 | 0 | 0 |

| P(F) | D | H | S |
|---|---|---|---|
| 0.99 | 1 | 1 | 1 |
| 0.97 | 0 | 1 | 1 |
| 0.89 | 1 | 0 | 1 |
| 0.8 | 0 | 0 | 1 |

*Figure 2. Bayesian Network*

In the next section, we talk about the probabilities that each node takes given its parents, and how these probabilities were calculated.

## 3. *CONDITIONAL PROBABILITIES*

Assigning probabilities to each variable is a very important step towards obtaining an accurate model. This step is handled differently with different types of nodes, and this is discussed in detail below:

Standard probabilistic models which capture the relationship between hours of operation and probability of failure, such as 2 parameter Weibull, Lognormal Distribution are used to model nodes.

### *3.1 Deterministic Variables*

These nodes take in raw data from the field and convert it into features that can be used by the remaining nodes. Input data from the field is used to derive these deterministic variables, which will be referred to as "features" throughout the paper.

### *3.2 Discrete Variables*

In our model, we have 4 discrete variables:

- Is the problem due to a software change? 1/0
- Is the problem due to wear out? 1/0
- Is the problem due to a hardware design change? 1/0
- Is there a failure? 1/0

### *3.3 Continuous Variables*

The continuous variables in our model are

- h: Largest fraction of hardware failures occurring on a single hardware design rev
- s: Largest fraction of hardware failures occurring on a single software release
- d: Degradation undergone by the part. This could be hours of operation, vibration fatigue, temperature fatigue and so on.

These variables have discrete child nodes, and hence also act as soft threshold functions for the child nodes, defining their probabilities.

The CPTs are obtained by using samples from the field data. For "*Failure*" node, the probability that the system fails given that it is a software failure is obtained by dividing the number of failures on a given software release by the total number of modules running on a given faulty software release. The very same method can be used for hardware design revision.

Probability Distribution Functions $\phi_{d1}$ (d) and $\phi_{d0}$ (d) are "*conditional Weibull distributions*" which give 1 when damage value d is large and 0 otherwise. The shape and slope parameter of the Weibull distribution depend on the value of the random variable H. Hence, we have two Weibull distributions for H=1 ($\phi_{d1}$ (d)) and H=0 ($\phi_{d0}$ (d)). The name conditional Weibull is derived from the fact that the beta and eta parameters are conditioned on another (parent) variable, H. A Weibull distribution is used because it is very adept at capturing the failure rate behavior of a system.

$\phi_{s1}$ (s) and $\phi_{h1}$ (h) are logit distributions which nicely convert continuous variables, s and h, to discrete variables, S and H, by smoothly transitioning through a threshold. An alternative to the logit distribution is the Probit distribution. These two are shown in Figure 3.

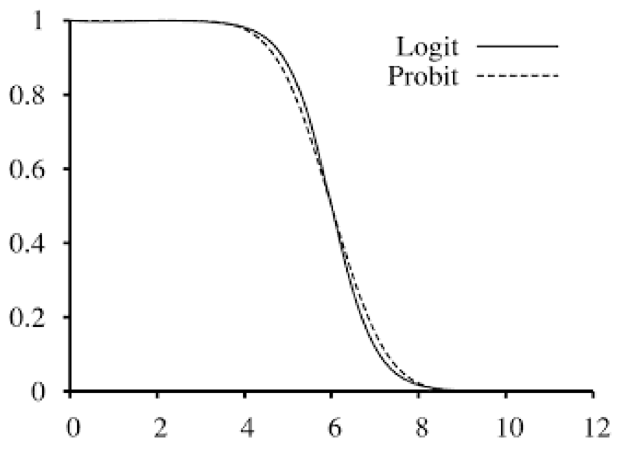


*Figure 3 Reliability Function*

Although we did not explicitly draw a *leak node* in our network, we have made it available implicitly by declaring that the probability of failure given none of the root causes are true is 0.2. This means that there is a 20% chance that other miscellaneous causes could potentially lead to a failure.

## 4. QUERYING AND INFERENCE

Constructing a network topology and assigning conditional probabilities completes the Bayesian Network, making it ready for us to query anything from it.

Some useful queries include:

- Given a failure, what is the probability that it was caused due to Wear Out/Degradation.
- Given a failure, what is the probability that it was caused due to a change in software release?
- Given a failure, what is the probability that it was because of a change in hardware release?
- Given a failure, what is the probability that the root cause is none of the above.

Comparing these probability values gives us insight into which node is the mostly likely one to cause the failure.

## 5. RESULTS

### 5.1 Input Data

Field data acts as the input to defining the probability values of nodes. Each record consists of the following fields:

- Did this module fail?
- Time in operation at the instant of failure
- Rainflow Counting of temperature fatigue at the instant of failure
- Rainflow Counting of vibration fatigue at the instant of failure
- Software Version
- Hardware Design Rev

Many records of data containing these fields constitute the input data used to build the Bayes net. Two cases are evaluated using this method, the process and results are described below.

### 5.2 Example 1

A failure was found on the field which was occurring across many modules. Field data for that module was imported into the Bayes Net and the standard prior probabilities were computed:

| | |
|---|---|
| P(Degradation/F=1) | 0.4 |
| P(Hardware Rev/F=1) | 0.01 |
| P(Software Version/F=1) | 0.6 |

This table tells us that although there is a seeming correlation with degradation, the most likely cause of the issue is the software revision.

### 5.3 Example 2

Another failure found on the field had many failures recurring within a small batch of hardware modules. The prior probabilities of this failure are:

| | |
|---|---|
| P(Degradation/F=1) | 0.1 |
| P(Hardware Rev/F=1) | 0.1 |
| P(Software Version/F=1) | 0.1 |
| P(~D, ~H,~S / F=1 ) | 0.4 |

This distribution indicates that the root cause of the failure is not captured by the three root causes specified by the Bayes Net.

## 6. APPLICATIONS AND IMPACT

Bayesian Networks have very interesting and widespread applications. When the complexity of the network is increased to include all the possible root causes, we obtain many unique advantages. Some of these are:

1. Avoid Red Herrings: Since a Bayesian Network has insight into the system as a whole, it is not susceptible to letting seemingly plausible root causes distract us from the real root cause. For example, one may assume that an infant mortality phenomena is causing field failures, however the real issue could be that the newer hardware parts have a new software on them which has a calibration issue. In such tricky cases, Bayes Nets gives you a full picture of all the underlying causes, so that you can be assured that you are not missing any relevant information.
2. Breaks a Tie: When there are multiple possible reasons of failure, the Bayes Net helps you decide which one might be more likely, by comparing P(variable/failure) of all the variables(root causes).
3. Ensures Completeness: If the probability of failure is zero when all the root causes are absent, that means that we have exhaustively listed all the root causes leaving no scope for a leak.
4. A network with more nodes and capabilities can handle more nuanced queries. For example, queries can be futuristic in nature. Will the failure be rectified upon replacing the part? Will the failure be rectified by revising the software? Is the failure catastrophic?

Another key outcome of building a Bayesian Network is that it introduces *Conditional Weibull and Lognormal* Distributions. Here, the distribution parameters such as the shape parameter, slope parameter and mean depend upon features such as hardware design revision, fatigue undergone by the part, weather etc. This gives a much better estimates of warranty costs and reliability as they take into account many more variables than just *'time to failure*

## *ACKNOWLEDGEMENTS*

The authors would like to thank Vaibhav Nangia and Masoud Rabiei for their contributions and help.

## *REFERENCES*

1. Norvig P, Russel S, "Artificial Intelligence: A Modern Approach", 1994, pp. 505 - 520 Prentice Hall
2. Tobon-Mejia D.A, Medjaher K., Zerhouni N. (2012) "CNC machine tool's wear diagnostic and prognostic by using dynamic Bayesian networks" Mechanical Systems and Signal Processing, Vol. 28 pp. 167 - 182
3. W Weibull "A statistical representation of fatigue failures in solids", 1949, Elander.

## *BIOGRAPHIES*

Asha Chigurupati
1600 Amphitheatre Pkwy
Google Inc.
Mountain View, CA, 94043, USA

e-mail: ashac@google.com

Asha is a hardware reliability engineer at Google X. She works on developing data driven prognostic and diagnostic models for hardware systems. Before joining X, Asha was a graduate student at Stanford University where she received Master's in Mechanical Engineer with a focus on Mechatronics, Machine Learning and Fluid Dynamics. She is passionate about applying mathematics to solve tricky problems.

Noah Lassar
1600 Amphitheatre Pkwy
Google Inc.
Mountain View, CA, 94043, USA

e-mail: nlassar@google.com

Noah Lassar is the Reliability Manager at Google X, where he manages a team of 10 reliability engineers. His work spans across the fields of product reliability, component reliability and hardware data analytics. Noah also acts as an advisor to Alta Motors, developer of high performance electric motorcycles. Prior to joining Google[x], Noah worked as the Manager of Reliability at Tesla Motors, where he developed and executed the reliability program for Tesla's model S and the Toyota Rav4 EV. Noah received his Master's degree in Mechanical Engineering from Stanford University in 2004. When Noah is not working on reliability challenges, he enjoys exploring the world with his wife and two children.