A Search Engine Risk Analysis

Reliability & Safety Final Project Report

Rufeng Ma

12/5/2020

Abstract

The software AIMS can analyze the risk of a system using event trees and fault trees. The system could be a mechanical system (e.g. a vehicle), a nuclear system (e.g. nuclear plant), or a safety alarm system (e.g. fire suppression system). AIMS is powerful enough to analyze a virtual system (e.g. email system and a website). In this report, I introduced how can we use AIMS to analyze the search engine system risks. From a safety point of view, I also used statistical methods to measure safety improvement. This report is a combination of theory and practice.

Introduction

Risk analysis is the process of assessing the likelihood of a system/event breakdown. Risk analysis is a key project management practice to ensure the least number of surprises occur while the project is underway. If we would like to assess the safety of a system. We have to quantify the crash probability of each part of this system. Finally, we will integrate all parts together.

As a virtual system, the search engine has 2 subsystems. The whole search engine can serve users normally if and only if those 2 subsystems both work. The main 2 subsystems are the frontend system and the backend system. The frontend system includes the design of the interface for users and the interaction with users. The backend system includes the crawler which in charge of collecting data and the database to save all data. The frontend will be stimulated by users. The backend is running all the time to collect up to date data.

Aims-PSA is a software developed to integrate the event trees and fault trees that are the major models of the level-1 internal PSA. Aims-PSA can generate the PSA model for each scope of a PSA automatically, by using the information and the level-1 PSA model [1].

Using Aims-PSA to analyze a search engine is a new type of risk analysis that never has been done before. The most challenging part of this project is to determine the basic event failure probability. In our case, we used the most related computer component failure probability to represent that virtual part failure probability.

Basic search engine components

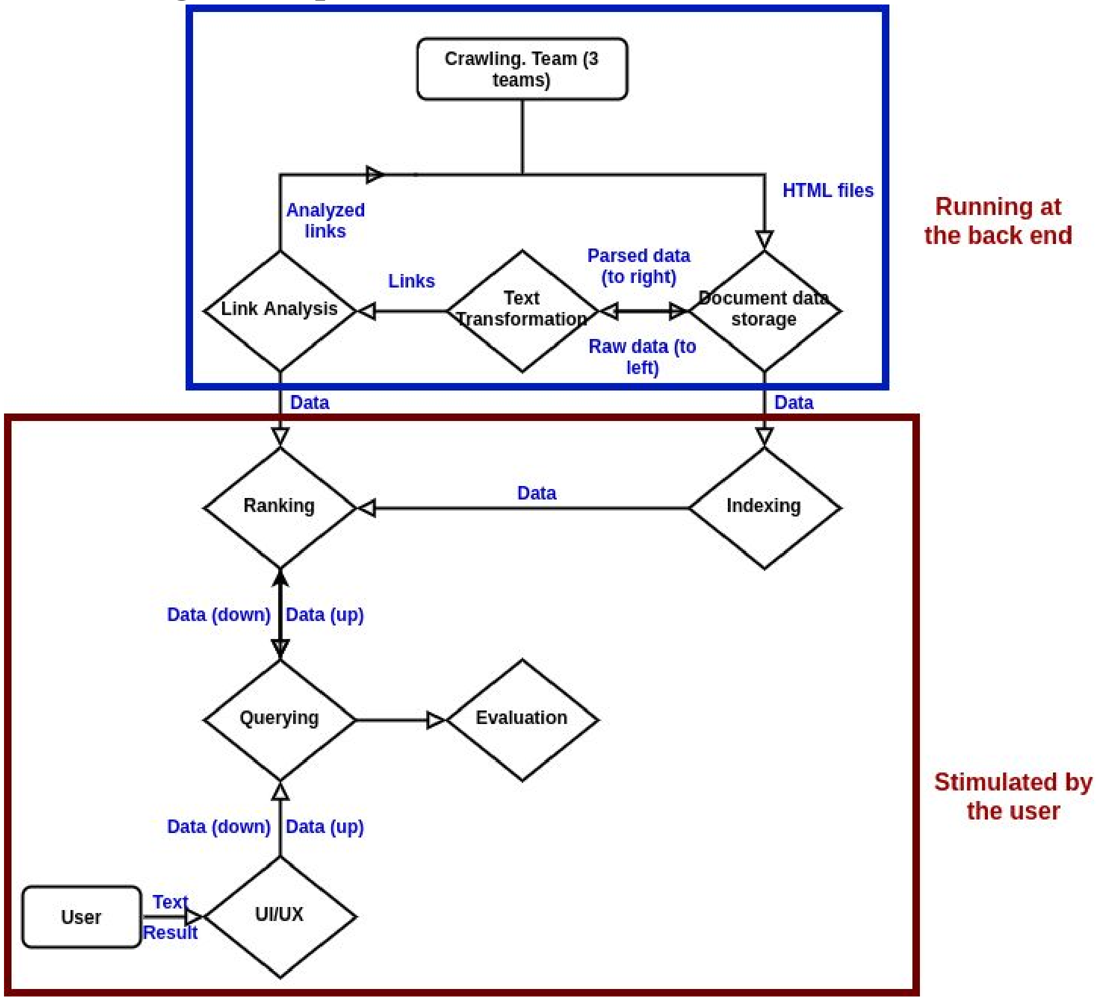


Figure 1, search engine components

In this part, I introduce the essential components that a search engine required.

<u>Web crawler</u>:

It is a computer program that gathers and categorizes information on the World Wide Web. A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the pages and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites (or web archiving), it copies and saves the information as it goes. [2].

Document data storage:

The database will save all documents for future usage. It will save both structured information and unstructured information. Unstructured information is the documents with a raw format. All the different pieces of information mixed together. If we would like to use them, we have to format the information then extract information from the document. But structured information can let us use information by calling the fixed names or tags.

Text transformation:

This part is doing the transformation from unstructured information to structured information. They also will communicate with other search engine components to add new items in their structured documents. Or can remove some useless information from the raw data. This component will pull raw data from DDS, then they push the formatted data back to DDS for another component pulling.

Link analysis:

This component is the core of the whole system. Because this part determined the component which will be downloaded by the crawling part. After analyzing the original URLs and their neighbor URLs. They will give a graph that only included the most related URLs.

Indexing:

Getting specific data related to the user's requests. They will index the information. After the indexing, the information is ready to be passed to the ranking team to get sorted before present to the user's interactive webpage.

Ranking:

The ranking determines the output sequence. When we search a word, the output usually determined which website we will click and browse. We usually only browse the 1st several links they provide on the 1st page. I rarely go to the next page. So here the ranking algorithm is critical for searching purposes, accuracy purposes, and commercial purposes.

Querying:

For processing the request from the user. We need someone to determine where the input from users to go. Which tag it will relate to for the output. The querying team will handle the decision making part.

Evaluation:

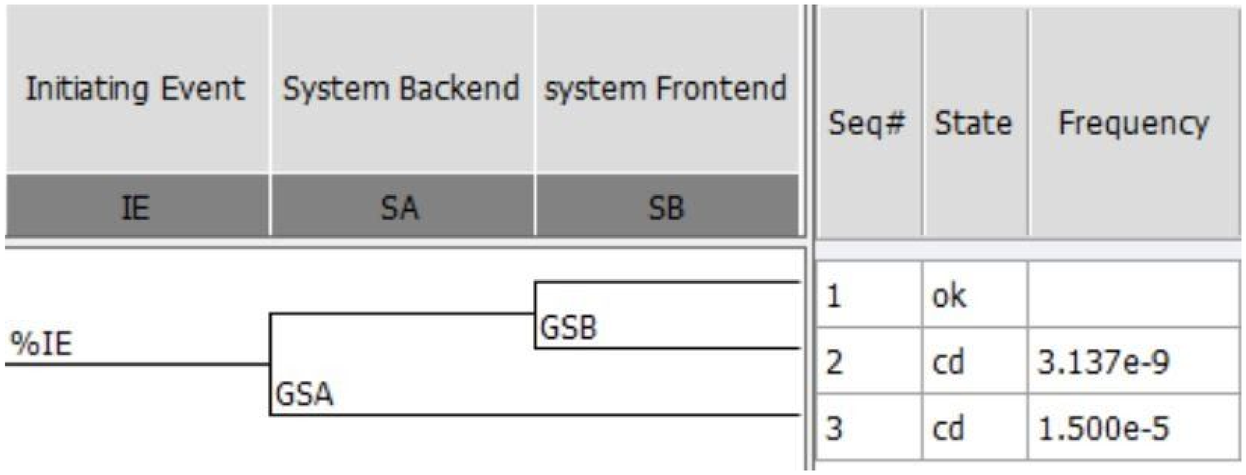
This is not that essential. But in this component, people will evaluate the accuracy of the querying. This will not necessarily happen during every stimulation. This will happen regularly to report the search engine accuracy.

UI/UX:

Here is the interface for users. For a search engine, the essential components on the interface are a dialog box and a button for execution.

Model build-up using Aims

For building the model in Aims, the first thing I will do is to divide the whole system into sub-systems. I decided to divide them into the frontend and backend systems. So the event tree looks like in Figure 2.



Seq#	State	Frequency
1	ok	
2	cd	3.137e-9
3	cd	1.500e-5

Figure 2, Event tree

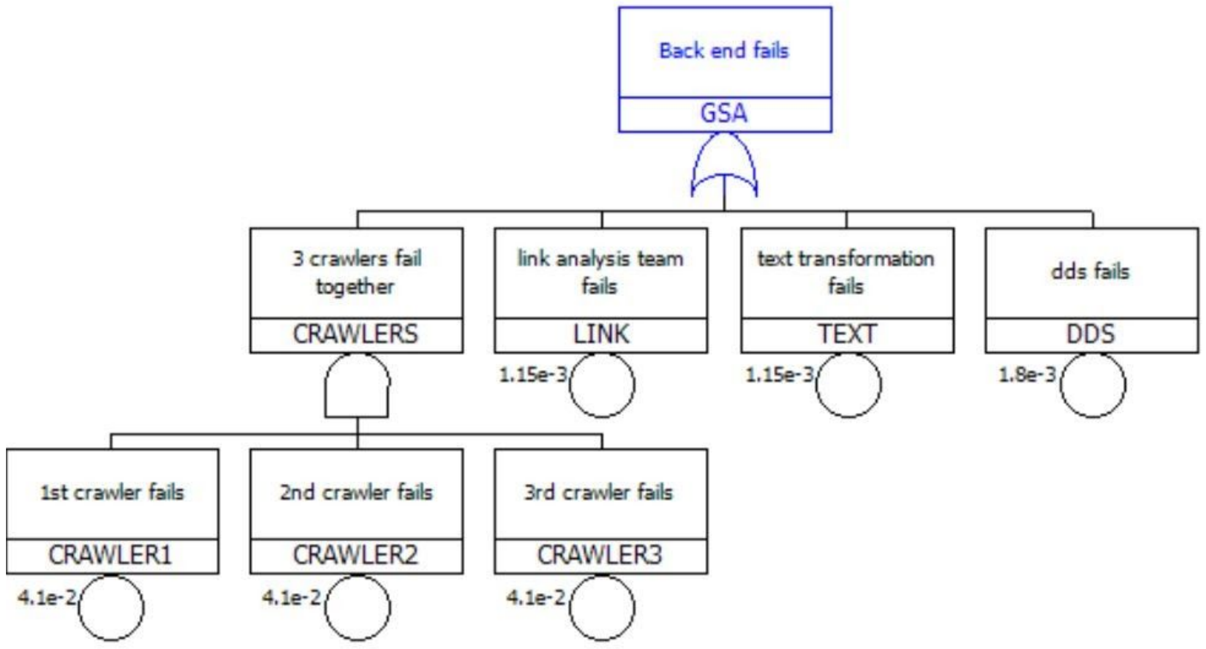


Figure 3, Fault tree of the backend.

For the backend, Figure 3, we choose to use an "OR" gate for those components in the backend part. But because the crawler part includes 3 crawling teams parallelly, this will be an "AND" gate. If we have 1 crawler keep running, the whole system will be fine. The only defect will be the data collecting speed will dramatically decrease. The probabilities here we used is from the table, Table 1.

For the frontend, Figure 4, we choose an "OR" gate for all components in the frontend. Further, the most vulnerable and complex part is the user end. Because users have their own divide (e.g. PC, Mac os, and Linux system). The operating system and the hardware on each user's device will also affect the result shows. They probably have an internet connection issue. They probably have a power outage. Even browsers could change the UI presenting speed and format. So we divide the user part into user, internet, power, and CPU, Figure 4. The probabilities here used are also in Table 1. Here I made a mistake, the "OR" gate was an "AND" gate in the presentation. But I correct it here.

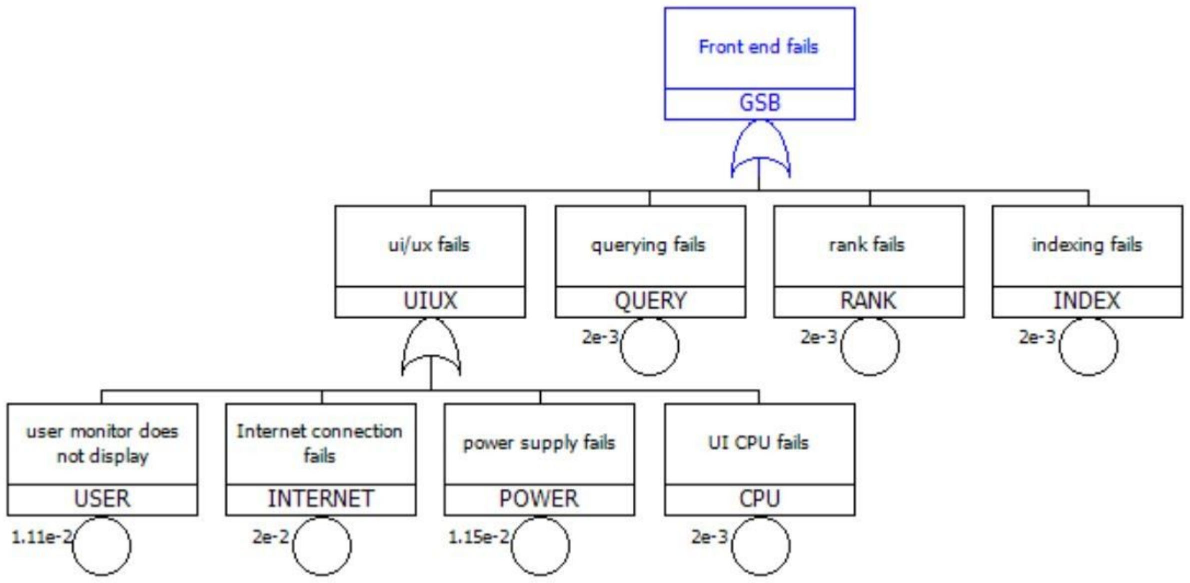


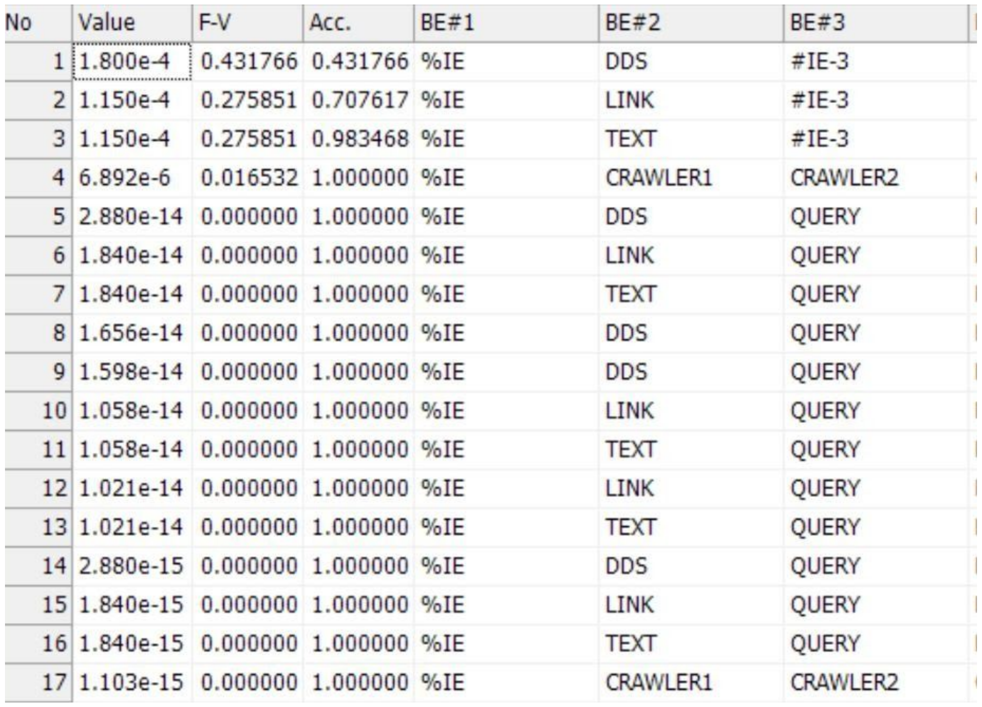
Figure 4, fault tree of the frontend.

Component name	Key hardware	Failure probability
Crawler1, 2, 3	RAM	4.1e-2
Link analysis	Power supply	1.15e-3
Text transformation	Power supply	1.15e-3
Data storage	Storage	1.8e-3
User	Power supply	1.15e-3
Internet connection	Internet connection	2e-2
Power supply	Power supply	1.15-3
UI/UX	CPU	2e-3

Table 1, component failure probabilities

Statistical analysis

From the cut set result in Aims, we found the components most likely cause the whole system failure are DDS, LINK, and TEXT, Table 2.



No	Value	F-V	Acc.	BE#1	BE#2	BE#3
1	1.800e-4	0.431766	0.431766	%IE	DDS	#IE-3
2	1.150e-4	0.275851	0.707617	%IE	LINK	#IE-3
3	1.150e-4	0.275851	0.983468	%IE	TEXT	#IE-3
4	6.892e-6	0.016532	1.000000	%IE	CRAWLER1	CRAWLER2
5	2.880e-14	0.000000	1.000000	%IE	DDS	QUERY
6	1.840e-14	0.000000	1.000000	%IE	LINK	QUERY
7	1.840e-14	0.000000	1.000000	%IE	TEXT	QUERY
8	1.656e-14	0.000000	1.000000	%IE	DDS	QUERY
9	1.598e-14	0.000000	1.000000	%IE	DDS	QUERY
10	1.058e-14	0.000000	1.000000	%IE	LINK	QUERY
11	1.058e-14	0.000000	1.000000	%IE	TEXT	QUERY
12	1.021e-14	0.000000	1.000000	%IE	LINK	QUERY
13	1.021e-14	0.000000	1.000000	%IE	TEXT	QUERY
14	2.880e-15	0.000000	1.000000	%IE	DDS	QUERY
15	1.840e-15	0.000000	1.000000	%IE	LINK	QUERY
16	1.840e-15	0.000000	1.000000	%IE	TEXT	QUERY
17	1.103e-15	0.000000	1.000000	%IE	CRAWLER1	CRAWLER2

Table 2, major components as a cause of system failure. Sorted by the probability.

Improve the search engine

I focused on improving DDS, LINK, and TEXT. The improved system will contain more parallel components. For example, the DDS team will not only have 1 data storage center. For giant commercial companies, they will have 2~3 databases working together. We use the same method to improve those other 2 components. See the improved backend fault tree in Figure 5. Finally, I found those 3 outstanding unsafe features disappeared, Figure 6.

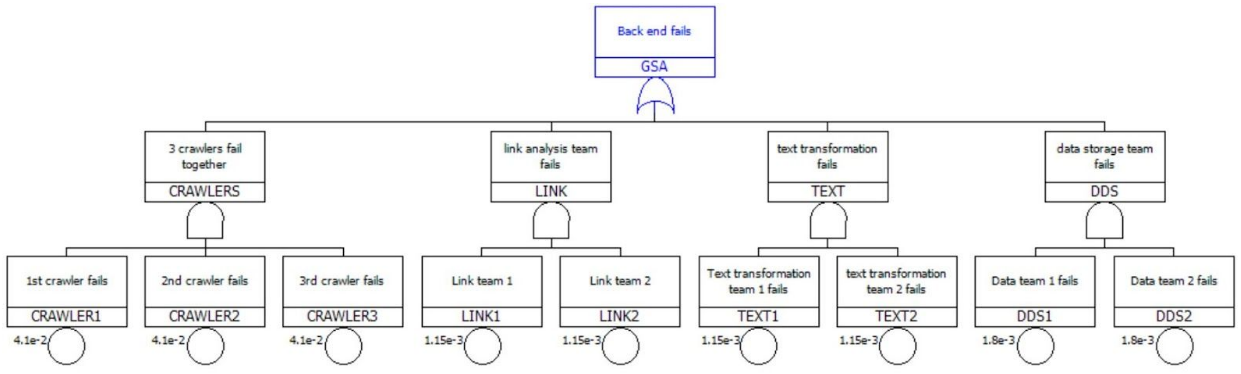
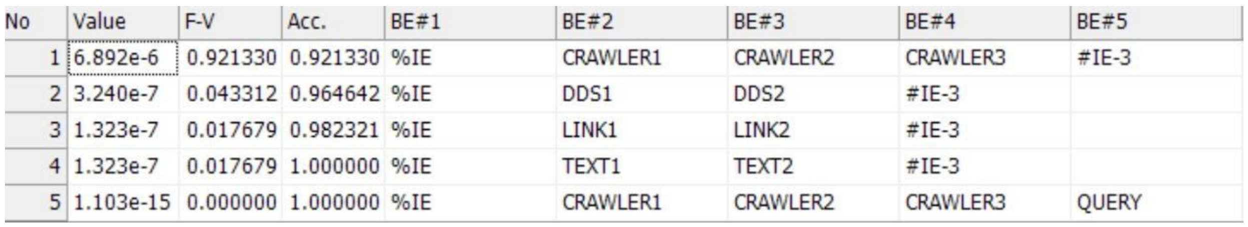


Figure 5, the improved backend fault tree.

Improved statistics:



No	Value	F-V	Acc.	BE#1	BE#2	BE#3	BE#4	BE#5
1	6.892e-6	0.921330	0.921330	%IE	CRAWLER1	CRAWLER2	CRAWLER3	#IE-3
2	3.240e-7	0.043312	0.964642	%IE	DDS1	DDS2	#IE-3	
3	1.323e-7	0.017679	0.982321	%IE	LINK1	LINK2	#IE-3	
4	1.323e-7	0.017679	1.000000	%IE	TEXT1	TEXT2	#IE-3	
5	1.103e-15	0.000000	1.000000	%IE	CRAWLER1	CRAWLER2	CRAWLER3	QUERY

Table 3, Improved major components as a cause of system failure. Sorted by the probability.

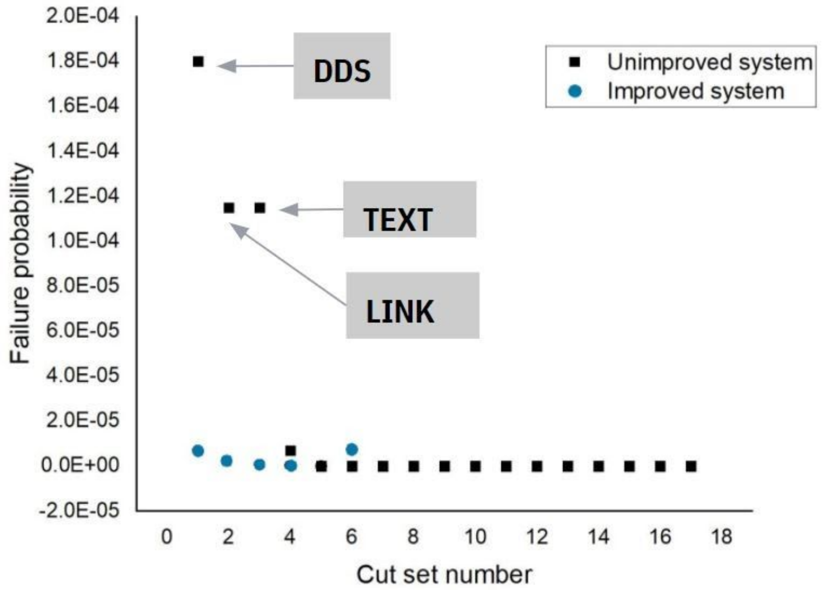


Figure 6, DDS, TEXT, and LINK are outstanding in the unimproved system (black), in the new improved system (blue), they are not that outstanding anymore.

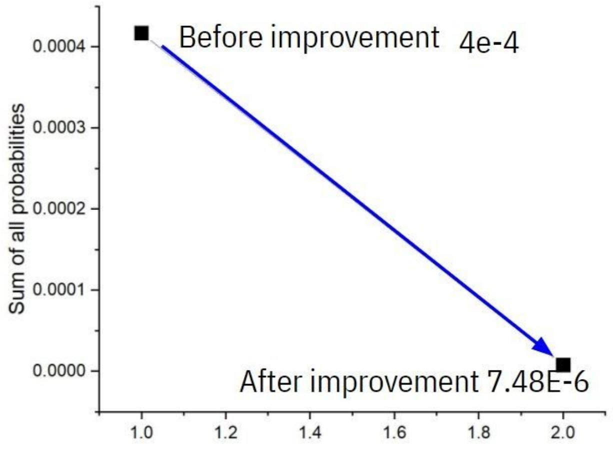


Figure 7, the sum of all failure probabilities in the cutset table. 1.0 in the x-axis means an unimproved system, 2.0 means an improved system.

It is obvious, after the improvement, the total failure probability (the sum of all possible failures) dropped sharply from 4e-4 to 7.48e-6.

Conclusion:

System failure analysis is essential for any critical system. If that system is not too critical, there is still no harm to calculate where can be improved. During the maintenance, we could focus on those most vulnerable parts. I learned a lot from this class. I could use the analysis method from this class to analyze any system. We also could quantify them using the Aims software. This is a useful class.

References:

[1] S.H. Han, H.-G. Lim, J.-E. Yang AIMS-PSA: a software for integrating various types of PSAs.

[2] Masanès, Julien (15 February 2007). Web Archiving. Springer. p. 1. ISBN 978-3-54046332-0. Retrieved 24 April 2014.