Repairable Systems Reliability Modelling

A REPORT

Submitted in partial fulfilment of the

Requirements for the award of the degree

Of

Master of Technology

By

Soumya Ranjan Mishra

(Roll No: 213100072)

Under the guidance of

Prof. Makarand S Kulkarni

Department of Mechanical Engineering

Indian Institute of Technology, Bombay

Mumbai 400 076

Dissertation Approval

This dissertation entitled **Repairable Systems Reliability Modelling** by **Soumya Ranjan Mishra**, Roll No. **213100072**, is approved for the degree of **Master of Technology**.

.....	.....
Examiner	Examiner
.....	.....
Supervisor	Chairman

Date: .....

Place: .....

DECLARATION

I declare that this written submission represents my ideas in my own words and where others ideas have been included I have adequatelty referenced the original sources. I also declare that I have adhered to all prinicples of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea, fact or source in my submission. I understand that any violation of the above will cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

Date: 19 June 2023

Place: IIT Bombay, Mumbai

Name: Soumya Ranjan Mishra

Roll No: 213100072

Acknowledgment:

I would like to express my special thanks of gratitude to my guide Prof. Makarand S Kulkarni for providing me the opportunity to work on this project and providing me with useful directions and inputs whenever necessary. I would like to express my gratitude to Department of Mechanical Engineering IIT Bombay for providing me with this opportunity. I would like to thank my parents for their continuous support and motivation. I would like to thank my seniors, friends and all the people who have helped me directly or indirectly in this project.

Date: 19/06/2023 Soumya Ranjan Mishra

213100072

ABSTRACT

This project work was focused on the development of a reliability modeling framework for repairable systems. A simulator was created to conduct failure simulations and provide predictive insights based on generated failure data. A virtual system with 20 components was constructed, and the simulator was utilized to simulate failures within the system.Two primary predictions were made using the failure data. The first prediction aimed to estimate the time until the next failure event. Starting with a power law model, the predictive approach advanced to ARIMA and LSTM models, enabling increasingly accurate predictions of the time to the subsequent failure event.The second prediction focused on identifying the next component likely to fail. Initially, an n-gram model was employed, later replaced by an improved RNN approach. Through the transition from language modeling to a classification model, the accuracy and precision of identifying the next failing component were tried to improve though didn't succeed to achieve. To provide a valuable tool for analysts, an interactive UI-based simulator was developed. Users can create their own system, specifying the desired number of components and establishing dependencies among them. The simulator enables conducting failure simulations within these user-defined systems. Various visual plots and data, from individual component-level information to comprehensive system-level analysis, are presented to assist users in drawing meaningful conclusions from the simulated data. In conclusion, this project presents a comprehensive reliability modeling approach for repairable systems.

Contents

LIST OF FIGURES

1 Introduction

1.1 QUALITY:

Quality is an adjective grammatically and hence can be defined as the amount by which a product or substance can satisfy the requirements of a customer. The quality of a product is deliverables as per the product's design and working specifications. It is highly dependent on the manufacturing processes involved in building up the product, finishing done as well as the tolerances given to the product. Quality of a product is achieved through a quality assurance program where the product is subjected to multiple number of tests and experiments so that they satisfy the requirements and impress the customers. Quality assurance is a planned set of processes and procedures necessary to achieve high product quality and this program should be organized for every components before getting to the hands of the customer.

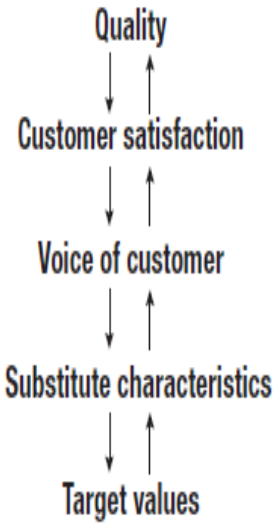

Figure 1-1 The relationship of quality, customer satisfaction and target values.

Quality Function deployment (QFD) is a way of forwarding the requirements of the customer into perfect design configurations and parameters, quality characteristics as well as the specifications achieved through the whole organization. It is present through different steps like marketing of the product, designing and manufacturing the product, purchasing the raw materials involved in building of the product. Every product would have a perfect ideal condition where the benchmark is set and that will be the target for the product and quality can be achieved if the target value is achieved that depicts its ideal condition. QFD is the methodology to develop the target values for substitute quality characteristics that satisfy the requirements of the customer.

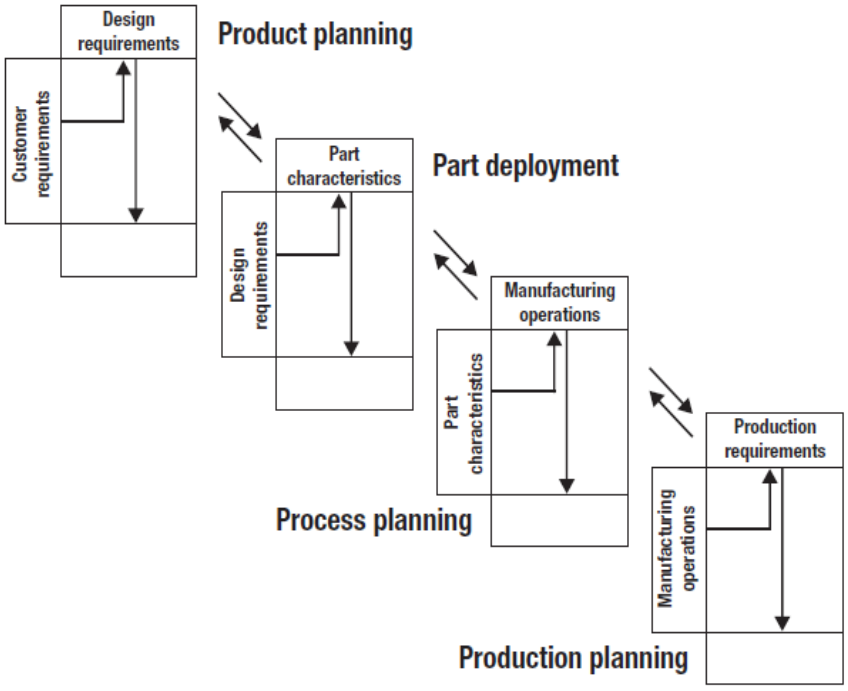

Figure 1-2- Illustration of the steps in QFD

1.2 Reliability:

Every component or thing present in the earth undergoes failure. A refrigerator can stop cooling the ingredients present inside it; an automobile engine starter can fail, a water heater can stop heating the water as per required temperature, a house roof may leak, the ceiling fan may fail to stop running. The load intended by the refrigerator may be greater than prescribed limit that may lead to heavy load on the compressor; the automobile battery may have experienced a wear out leading to failure; the coils of the water heater may have gotten corroded leading to unable the heat the water properly; the glues in the roof have lost the binding capacity leading to the leaking of the house roof; the motor of the fan has failed leading to stop running. The refrigerator has failed by over-burdening; the automobile starter was failed to over-burdening in the inaccurate environmental conditions at high temperatures; similarly water heater was due to lack of preventive maintenance; house roof was due to poor construction; and ceiling fan was due to an unknown mechanical failure. Some of these failures may have a serious economic impact, some may have impact on personal safety. Thus knowing out the reliability of a components is quite necessary and important.

Although there is no universal definition of reliability, but we can define as the probability that a component or system will perform a required function for a given period of time when used under stated operating conditions. It can be defined as the probability of non-failure over time. To determine reliability in a true sense, we need to define it specifically. The initial point is to define an unambiguous and observable description of the failure. This failure definition should

be relative to the function being performed by the system. Secondly the unit of time must be defined whether it's in terms of operating cycles or operating hours. Some of the cases the units are defined in terms of amount of distance travelled or number of jobs completed before failure. For production systems failures may be defined in terms of units or batches produced. Thirdly the system should be observed under normal performance. This may include the operating conditions like storage, maintenance and transportation; environmental conditions like temperature, humidity, vibration and altitude; design loads applied like pressure, current, voltage and weight.

1.2.1 The Ability To Perform As Intended

When a product is purchased, there is an expectation that it will perform as intended. The intention is usually stated by the manufacturer of the product in the form of product specifications, datasheets, and operations documents. For example, the product specifications for a cellular phone inform the user that the cell phone will be able to place a call so long as the user follows the instructions and uses the product within the stated specifications.1 If, for some reason, the cell phone cannot place a call when turned on, it is regarded as not having the ability to perform as intended, or as having "failed" to perform as intended. In some cases, a product might "work," but do so poorly enough to be considered unreliable. For example, the cell phone may be able to place a call, but if the cell phone speaker distorts the conversation and inhibits understandable communication, then the phone will be considered unreliable. Or consider the signal problems reported for Apple's iPhone 4 in 2010. The metal bands on the sides of the iPhone 4 also acted as antennas for the device. Some users reported diminished signal quality when gripping the phone in their hands and covering the black strip on the lower left side of the phone. The controversy caused Apple to issue free protective cases for the iPhone 4for a limited time to quell consumer complaints (Daniel Ionescu 2010).

1.2.2 For A Specified Time

When a product is purchased, it is expected that it will operate for a certain period of time. Generally, a manufacturer offers a warranty, which states the amount of time during which the product should not fail, and if it does fail, the customer is guaranteed a replacement. For a cell phone, the warranty period might be 6 months, but customer expectations might be 2 years or more. A manufacturer that only designs for the warranty can have many unhappy customers if the expectations are not met. For example, most customers expect their car to be able to operate at least 10 years with proper maintenance.

1.2.3 Life-Cycle Conditions

The reliability of a product depends on the conditions (environmental and usage loads) that are imposed on the product. These conditions arise throughout the life cycle of the product, including in manufacture, transport, storage, and operational use.3 If the conditions are severe

enough, they can cause an immediate failure. For example, if we drop or sit on a cell phone, we may break the display. In some cases, the conditions may only cause a weakening of the product, such as a loosening of a screw, the initiation of a crack, or an increase in electrical resistance. However, with subsequent conditions (loads), this may result in the product not functioning as intended. For example, the product falls apart due to a missing screw, causing a connection to separate; cracking results in the separation of joined parts; and a change in electrical resistance causes a switch to operate intermittently or a button to fail to send a signal.

1.2.4 Reliability As A Relative Measure

Reliability is a relative measure of the performance of a product. In particular, it is relative to the following:

■ Definition of function from the viewpoint of the customer
■ Definition of unsatisfactory performance or failure from the viewpoint of the customer
■ Definition of intended or specified life
■ Customer's operating and environmental conditions during the product life cycle.

1.3 Reliability Of Systems:

A system can be considered as a combination of various components and each of those components contribute to the systems performance so to find out the systems performance we need to find out the individual performance of the different components. The reliability of a system is highly essential to know because this reliability factor will decide how much load we need to act on that system when the preventive maintenance will be done for the system. Ignore the reliability of unity is not achieved for the system so if you are achieving the reliability of unity we assume that the system is currently has not started working. The reliability of a system is a combination off the function of the reliability of the individual components present in the system as well as the configuration of the components that are present in the system. So it is a highly complex task to find out stop the system as there may be different components having different reliabilities as well as the configurations will be highly different from each other.

Techniques of reliability evaluation of the system depend upon the configuration of its components. The various configurations present in a system are:

- Series configuration
- Parallel configuration
- Mixed configuration
- k-out-of-n configuration
- Standby configuration
- Complex configuration\

Let us understand a simple system. A simple system contains a combination of series and parallel configuration show to understand total simple system we need to understand first series and parallel configurations as well.

1.3.1 Series Configuration

Let us consider a system having n components. All the n components of the system are said to be in series configuration if the components are connected to each other in such a manner that:

- Only one component needs to fail for the system failure.
- For the successful operation of the whole system all the n components present in the system must be able to perform their intended functions.

To understand a series configuration based system we can take the example of rice lights that are hanged during the festival seasons for decoration purposes. In this the system entirely performs well when all the bulbs/ LEDs are working in the string of lights. If any of them fails to stop working the entire system fails.

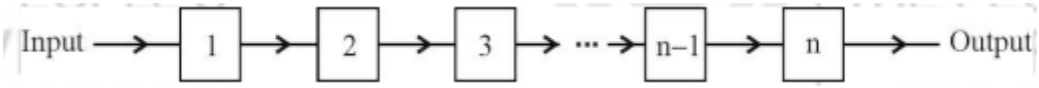

Figure 1-3 Series Combination

https://www.egyankosh.ac.in/bitstream/123456789/35168/1/Unit-14.pdf

1.3.2 Parallel Configuration

Similarly, let us consider a system having n components. All the n components are said to be in parallel configuration if the components are connected to each other in such a manner that:

- The system fails if all the components fail.
- Only one component among the n components needs to work for the successful operation of the system.

To understand the parallel configuration based system, we can take the example of electrical components present in a room. For example in a room there consists of a TV, refrigerator, air-conditioner, tube light and fan. If any of the above components fail, that doesn't guarantee that the whole setup in the room will fail. This is due to the parallel configuration of all the components.

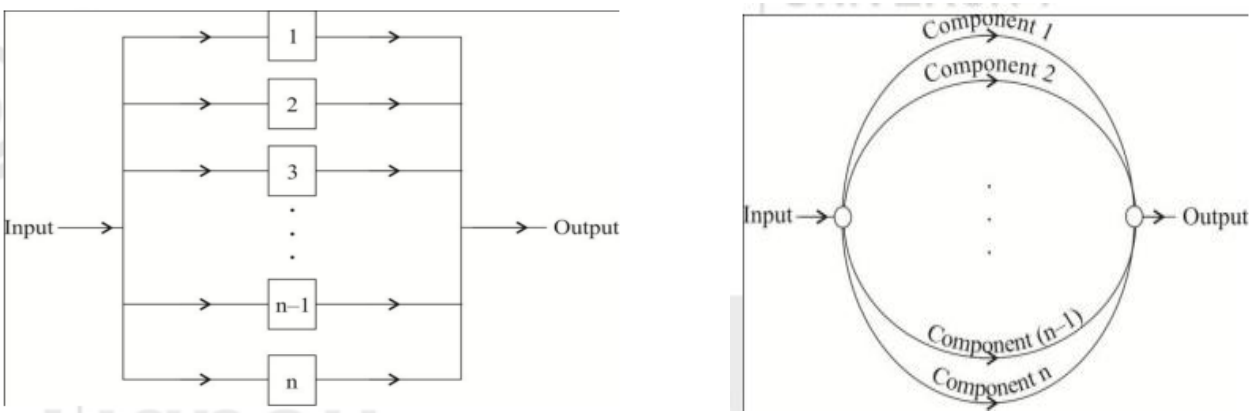

Figure 1-4 Parallel Combination Arrangement (Source: https://www.egyankosh.ac.in/bitstream/123456789/35168/1/Unit-14.pdf)

To evaluate the reliability of a system, we need to find out the number of elementary components present in the system and their configurations. This will help us decide what rules to apply to the different traditions of probability theory. These rules include the addition rule, multiplication rule, independence or dependence of various events or their combinations or conditional reliability.

The various steps involved in finding out the reliability of the system consisting of n components inside it but have different configurations are:

Step 1: We identify the n components present in the system and if some of them build up to form a subsystem or not. Makeup units comprise the system. Each unit can be a component or a subsystem.

Step 2: Evaluate the reliabilities of each unit. If the individual component reliability is not there, try to evaluate it.

Step 3: Draw the reliability block diagram of the entire system where the units present inside it are connected to make it logical and understandable for the system.

Step 4: Determine the constraints that should be fulfilled for the system's successful operation.

Step 5: Determine the probability theory rules that need to be applied based on the configuration such as addition rule, multiplication rule, independence or dependence of various events, or their combinations or conditional reliability.

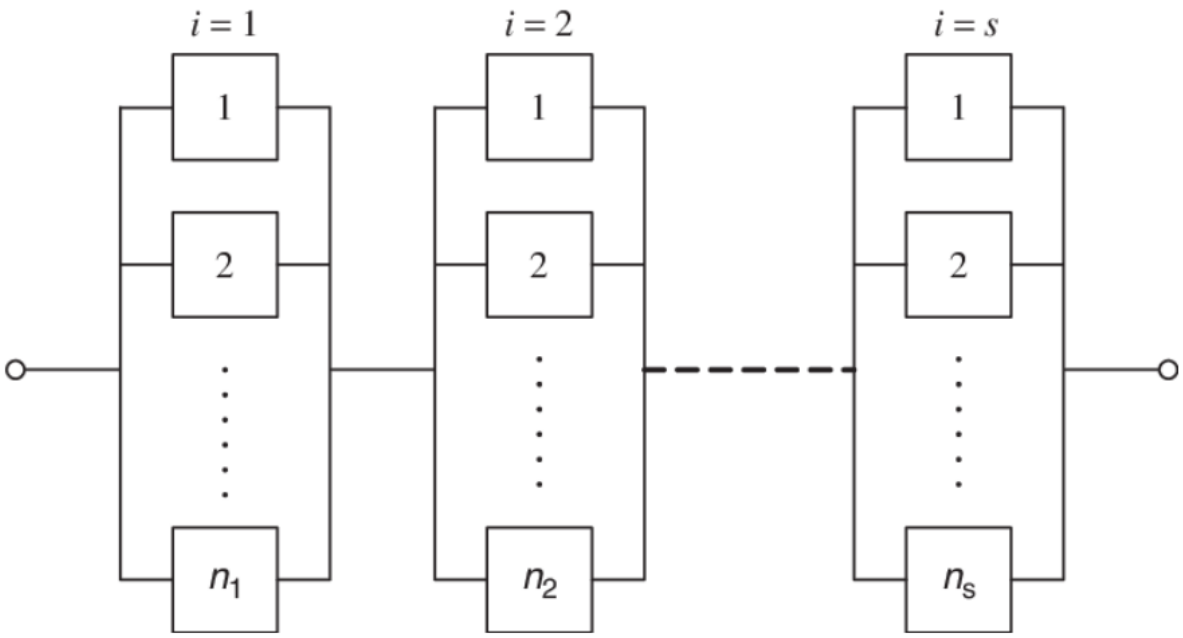

Figure 1-5- Combination of Series and Parallel Combinations

Source: https://www.egyankosh.ac.in/bitstream/123456789/35168/1/Unit-14.pdf

2 Literature Review

2.1 Repairable Systems

Every sophisticated system in the globe, including military vessels, aircraft, cars, and medical equipment, is constantly fixed when it malfunctions. They are never replaced because doing so would be very expensive logistically. Instead, the system's internal components that failed at the lowest level and caused the component to fail are replaced. For instance, if a water pump in a car breaks, the car is fixed and the water pump is replaced. Instead of being replaced, the system is fixed during this process. The dependability and other performance characteristics of these systems in customer use

Every systems are frequently of great interest when they are fielded or subjected to these environments. The projected number of failures throughout the warranty period, maintaining a minimum level of mission reliability, analysing the rate of wear out, choosing when to replace or overhaul a system, and lowering life cycle costs are possible areas of interest. In general, these problems cannot be solved using a lifetime distribution like the Weibull distribution. A procedure rather than a distribution is frequently employed to address the dependability traits of complicated repairable systems. The Power Law model of a process is the most widely used model. This model is well-liked for a number of reasons. One is that it has a very solid basis in terms of minimal repair, which is a scenario in which a failed system only requires enough repair to make it functional once more. Second, if the Weibull distribution describes the time to first failure, the Power Law model governs each consecutive failure, just as it would in the event of minimal repair. The Power Law model is, in this sense, a continuation of the Weibull distribution.

Let's consider a system with a specific number of components so that we can fully comprehend the distribution. When a component malfunctions, a replacement one of the same kind is installed right away. Each time a replacement is made, the system is restored to like-new condition. The time-to-failure for each component is set by the underlying distribution. A distribution refers to a single failure, which is a critical distinction to make. A renewal process is the term used to describe the system's failure sequence.

Let's suppose that a system is made up of numerous components in order to broaden this procedure even more. Assuming all the components are connected in series, a failure in any one of them results in a failure of the entire system. Each component goes through a renewal process that is controlled by the distribution function that corresponds to it. When one of the internal components breaks and the system malfunctions as a result, the internal component is replaced, and the system is then fixed. The system is often not restored to a brand-new state after replacing a single component because there are numerous other working components of varying ages. For instance, an automobile is not as good as new after a broken water pump has been replaced. The other parts, such as the engine and fuel pump, are still ageing because they have accumulated some experience. Therefore, the failures of a complex system like an automobile do not correspond to distribution theory. A sophisticated repairable system's failure intervals typically do not follow the same distribution. Distributions only apply to the individual system components; they do not apply to the entire system. A distribution is applied to the initial failure at the system level. A distribution has one failure attached to it. For instance, if a system's first failure is caused by a component that follows the Weibull distribution and its second failure is caused by a component that follows a degradation-based distribution, the

system's failure behaviour is inconsistent and, as a result, doesn't follow any of the distributions that the components inside of it carry.

In the real world, a repair could only be necessary to restore the functionality of numerous systems. If the car's water pump breaks, all that has to be done to fix it is to replace it by installing a new one. Similar to this, if a seal leaks, it is replaced but no further maintenance is carried out. The idea of minimal repair is this. It differs significantly from overhaul repair, which restores the system to its original condition. Repairing a single failure mode does not significantly increase the system's reliability compared to shortly before the failure for a system with multiple failure modes. When a complicated system with multiple failure modes undergoes minimal repair, the system's reliability after the repair is the same as it was right before the failure. A non-homogeneous Poisson process governs the progression of failures at the system level in this instance (NHPP).

The system age when the system is first put into service is time 0. Under the NHPP, the first failure is governed by a distribution F(x) with failure rate r(x). Each succeeding failure is governed by the intensity function u (t) of the process. Let t be the age of the system and Δt is very small. The probability that a system of age t fails between t and t+Δt is given by the intensity function u (t) Δt. Notice that this probability is not conditioned on not having any system failures up to time t, as is the case for a failure rate. The failure intensity u (t) for the NHPP has the same functional form as the failure rate governing the first system failure. Therefore, u (t) =r (t), where r (t) is the failure rate for the distribution function of the first system failure. Under minimal repair, the system intensity function is:

$$u(t) = \alpha\beta t^{\beta-1}$$

This is the power law model/ Weibull Power law model since the first failure is governed by Weibull distribution and for succeeding failures it is governed by the Power law model. If the system has a constant failure intensity u (t) =λ, then the intervals between system failures follow an exponential distribution with failure rate λ. If the system operates for time T, then the random number of failures N (T) over 0 to T is given by the Power Law mean value function.

$$E[N(T)] = \alpha t^{\beta}$$

Therefore the probability N (T) = k for period (0,t) is given by the Poisson probability

$$P(n = k) = \frac{N(T)^k\, e^{-N(T)}}{k!}$$

As the intensity function remains constant, this is known as a homogeneous Poisson process. For β =1, this is a specific instance of the Power Law model. The intensity function can alter as the repairable system gets older thanks to the Power Law model, which is a generalisation of the homogeneous Poisson process. For the Power Law model, the failure intensity is increasing for β>1 (wear out), decreasing for β<1 (infant mortality) and constant for β=1 (useful life).

2.2 Maintainability

To deal with repairable systems effectively, we must first comprehend how their components are restored (i.e., the maintenance actions that the components undergo). In general,

maintenance is described as any action that keeps working units from breaking down or restores broken ones to a working state. Maintenance is essential to extending the life of repairable systems. It has an impact on the system's overall dependability, availability, downtime, operational costs, etc. The three categories of maintenance actions are generally corrective maintenance, preventative maintenance, and inspections.

Corrective Maintenance

The action(s) taken to return a malfunctioning system to operating status constitute corrective maintenance. This typically entails changing or fixing the component that caused the system as a whole to malfunction. Because a component's failure time cannot be predicted in advance, corrective maintenance must be done at erratic intervals. In the shortest amount of time possible, corrective maintenance aims to return the system to satisfactory operation. Three steps are commonly taken while performing corrective maintenance:

- Diagnosis of the issue. The maintenance worker must spend the necessary time to identify the damaged components or to otherwise correctly determine what went wrong with the system.
- Defective component replacement or repair (s). Once the root cause of a system failure has been identified, a solution must be found, typically by replacing or repairing the system's problematic components.
- Verification of the repair procedure. After the problematic components have been fixed or replaced, the maintenance expert must make sure the system is working properly once more.

Preventive Maintenance

Contrary to corrective maintenance, preventive maintenance involves replacing parts or subsystems before they malfunction in order to ensure ongoing system operation. Based on previous system behaviour, component wear-out mechanisms, and knowledge of which components are essential to continuous system performance, a preventative maintenance schedule is created. Preventive maintenance schedules are always influenced by cost. In many situations, it is more cost-effective to replace parts or components at predetermined intervals even when they have not yet failed than to wait for a system breakdown that could cause a costly interruption in operations.

Inspection- Based Maintenance

Inspections are used to find hidden failures (also called dormant failures). Unless the component is discovered to have failed, in which case a corrective maintenance activity is started, no maintenance is typically conducted on the component during an inspection. However, there may be instances where an inspection entails doing a partial restoration of the inspected object. For example, when checking the motor oil in a car between scheduled oil changes, one might occasionally add some oil in order to keep it at a constant level.

2.3 Down- Time

Maintenance actions (preventive or corrective) are not instantaneous. There is a time associated with each action (i.e., the amount of time it takes to complete the action). This time is usually referred to as downtime and it is defined as the length of time an item is not operational. There

are a number of different factors that can affect the length of downtime, such as the physical characteristics of the system, spare part availability, repair crew availability, human factors, environmental factors, etc. Downtime can be divided into two categories based on these factors:

Waiting downtime: Waiting downtime is defined as the time where we cannot operate the system as well as the system is not getting repaired. There are several reasons for this. Some of them could be time to ship the parts that are to be replaced, administrative clearances time and so on.

Active Downtime: Active downtime is defined as the time where we cannot operate the system but the system is getting repaired. Alternatively, the active downtime is the time it takes repair personnel to perform a repair or replacement. The length of the active downtime is greatly dependent on human factors, as well as on the design of the equipment. For example, the ease of accessibility of components in a system has a direct effect on the active downtime.

The length of time it takes to repair or restore a particular item is typically variable due to the impact of numerous distinct factors on downtime. That is, just like the time to failure, the time to repair is a random variable. The claim that it takes five hours on average to repair suggests that there is a probabilistic distribution at play. Repair distributions (also known as downtime distributions) are used to separate time-to-repair distributions from failure distributions. However, technically speaking, the techniques used to quantify these distributions are identical to those used to quantify failure distributions. The distinction lies in their method of employment (i.e., the events they describe and metrics used). The likelihood that the event (failure) will occur by that time is provided by unreliability when employing a life distribution with failure data (i.e., the event modelled was time-to-failure), but the probability that the event (failure) will not occur is provided by reliability. When it comes to downtime distributions, the data set is made up of times-to-repair, thus what we previously referred to as unreliability is now the likelihood that the event will occur (i.e., repairing the component).

2.4 Maintainability and Availability

The possibility of completing a successful repair action in a specific amount of time is what is meant by maintainability. In other words, maintainability assesses how quickly and easily a system may be made operational following a breakdown. For instance, if it is said that a certain component can be maintained in an hour with a 90% success rate, it is likely that the component will be fixed in an hour with a 90% success rate. In the same way that time-to-failure is the random variable in reliability, time-to-repair is the random variable in maintainability. Take the maintainability equation for a system with exponentially distributed repair times as an illustration. Its M (t) maintainability is determined by:

$$M(t) = 1 - e^{-\mu t}$$

Where μ = repair rate

We can observed how this equation and the one for a system with exponentially distributed failure times are comparable. The maintainability statement, however, is equal to the unreliability expression, (1-R), since maintainability reflects the probability of an event occurring (repairing the system), whereas reliability indicates the probability of an event not occurring (failure). Additionally, the one model parameter is now known as the repair rate, which is equivalent to the failure rate, used in reliability with an exponential distribution. The mean of the distribution can be called as mean time to repair.

$$\text{Mean Time to Repair (MTTR)} = \frac{1}{\mu}$$

We can see the analogy between mean time to failure (MTTF) and mean time to repair (MTTR)

Availability

When reliability (the likelihood that an item won't break) and maintainability (the likelihood that an item can be effectively repaired after failing) are taken into account, a third metric—the likelihood that the component or system is operational at a specific time, t—is required (i.e., has not failed or it has been restored after failure). This statistic measures availability. The availability of a component or system takes into account both its reliability and maintainability as a performance criterion for repairable systems. It is described as the likelihood that the system will function properly when called upon to do so. In other words, availability is the likelihood that a system will not be down or requiring maintenance when it is needed.

2.5 Condition- Based Maintenance Implementation

The physical description, functional description, component prioritisation, data handling, diagnosis, and prognosis are the four primary steps that make up the CBM implementation process. Consideration should be given to data gathering, data analysis, decision-making, and execution as gathered through eight pertinent postulates. The first phase is understanding information sources, which is followed by physical structure and functional elements. These latter results are essential for deciphering the symptoms and providing interpretive context for them. Prior to describing maintenance activities and general actions, such as diagnostic and prognosis, which call for recurring updates for optimal configuration, the most crucial systems and components are first chosen. The idea of Reliability Centered Maintenance (RCM), which strengthens CBM with a particular focus on failure modes analysis, is connected to CBM through the examination of critical components. Based on these factors, data fusion levels—starting with signal-level fusion and extending through feature-level fusion and decision-level fusion—remain pertinent. The ranking of maintenance tasks according to priority is referred to as decision-making. Every piece of equipment has unique maintenance requirements in terms of reliability, safety levels, and failure impacts. Costs, safety, added value, feasibility, and risk are also taken into consideration during the decision-making process. CBM is now implemented using neural networks even in terms of decentralised artificial neural networks. These latter ones make use of a hierarchical strategy built on numerous neural networks, each of which is tailored for a particular purpose. The neuro-fuzzy method, which also uses self-organizing map approaches, can reduce the digression in fuzzy models. These latter can be triggered under scenarios when the prior neural classifier loses their validity because they can learn even in the absence of the corresponding class labels for the input pattern.

The most essential part in a CBM implementation is collection of data. Data management and data collection are of importance, and related issues with data quality can be seen. As demonstrated by a number of extensive studies of data-driven techniques (from data preparation to sensitivity analysis), poor raw data management can have an impact on the implementation of CBM. CM, which refers to ICT (Information Communication Technology) use, specifically of web and agent technologies, is crucial for guaranteeing proper data management. A corresponding technology support development is needed after the data integration, especially in situations where the assets are dispersed geographically. In response to the monitoring of particular parameters (especially vibration, a widely used metric for

CBM), and usability issues, sensor technologies play a significant role in CM (portability, non-contact and reduced size). Prognostics and health management are closely related to the development of sensor technology, making it possible for more efficient and affordable data collecting. Prognostics and health management are techniques and technology used to assess a product's dependability, comprehend possible problems, and reduce risks. They represent a sophisticated field of CBM research, influencing everything from item behaviour forecasts to the collection of raw data. Recently, artificial intelligence has improved them.

2.6 Recurrent Event Data Analysis

Recurrent Event Data Analysis (RDA) is employed in many practical domains, including criminology, social sciences, business, economics, and reliability. Unlike repairable system data or situations where the analyst is interested in modelling the number of occurrences of events over time rather than the amount of time prior to the first event, as in Life Data Analysis (LDA, where it was assumed that events (failures) were independent and identically distributed (iid), there are many cases where events are dependent and not identically distributed. There are 2 types of approaches to analyse such data: (1) Parametric Recurrent Event Data Analysis (2) Non-Parametric Recurrent Event Data Analysis

Parametric Recurrent Event Data Analysis

This analysis is used to identify the trends, estimate the rate and predict total number of recurrences. One kind of recurrence data is the failure and repair data of a repairable system. The process of future failure may be impacted by previous and ongoing fixes. Time (distance, cycles, etc.) is a crucial element for the majority of recurring events. The rate of recurrence could increase, decrease, or stay the same over time. For other recurring events, the process of recurrence can be influenced by both time and the number of events (e.g., the debugging process in software development). The General Renewal Process (GRP) model is applied in the parametric analysis technique. The repair time is considered to be small in this paradigm, allowing the processes to be seen as point processes. This model offers a means of describing the pace at which events occur over time, as in the case of information gleaned from a repairable system. This model is especially effective for simulating the behaviour of a given system's breakdown and comprehending how repairs affect the system's age. Consider a system that has experienced a failure and has since been restored, but the system is not now in an as-good-as-new or as-bad-as-old condition. In other words, after the repair, the system is only partially revived. Rarely, a Weibull distribution has also been applied in situations where the system is nearly as good as new following the repair, commonly known as a perfect renewal process (PRP). Even though numerous models have been suggested in literature, there hasn't been a commercially viable model for the transitional states following the repair.

Non- Parametric Recurrent Event Data Analysis

Non-parametric RDA provides a non-parametric graphical estimate of the mean cumulative number or cost of recurrence per unit versus age. A mean cumulative function (MCF) is implemented for non-parametric RDA. It can be used for:

- Analysing whether the population's repair (or expense) rate rises or falls as people get older (this is useful for product retirement and burn-in decisions).
- Calculating the average number of repairs that each unit will require, and their cost, within the warranty period.

- Comparing two or more sets of information from various designs, production cycles, maintenance guidelines, settings, operational situations, etc.
- Determining future figures and repair costs, such as the anticipated number of failures in the upcoming month, quarter, or year.
- Providing surprising details and insights.

Mean Cumulative Function

Each population unit can be characterised by a cumulative history function for the total number of recurrences in a non-parametric study of recurrent event data. It is a staircase function that shows the total number of times an event—like repairs—has occurred over time. The cumulative history function of a unit is shown in the figure below:

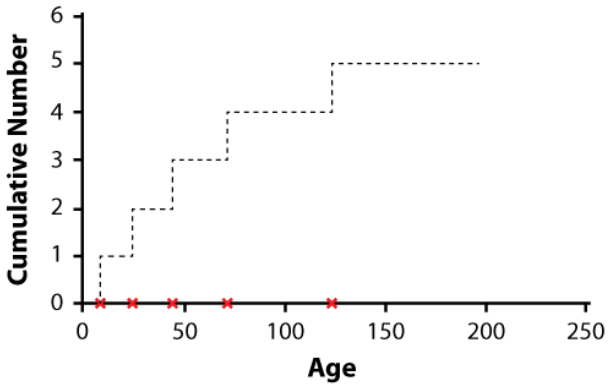

Figure 2-1 Events occured over time (Source:Reliawiki)

The population of cumulative history functions is used to define the non-parametric model for a population of units (curves). Every unit in the population's population of staircase functions makes up this population. The units have a distribution of their total number of events at age t. In other words, a portion of the population has experienced 0 recurrences, a portion has experienced 1, a portion has experienced 2, etc. The mean M (t) of this distribution, which varies at various ages t, is known as the mean cumulative function (MCF). The point-wise average of all population cumulative history functions is known as the M (t) (see figure below):

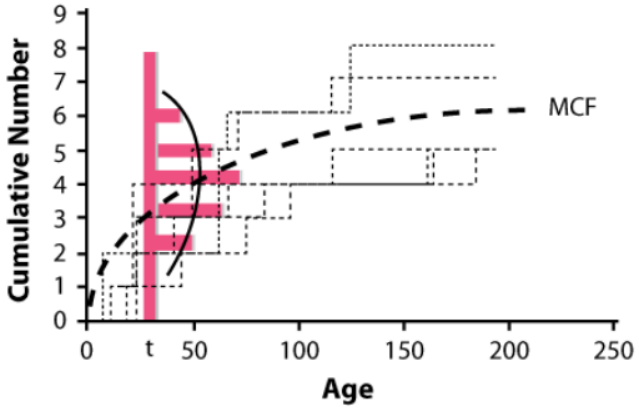

Figure 2-2 Mean Cumulative Function (Source:Reliawiki)

2.7 Reliability Growth Analysis

Design, manufacturing, and/or engineering flaws are typically present in the initial prototypes created during the development of a new complicated system. These flaws may cause the prototypes' initial dependability to fall short of the system's reliability objective or requirement. The prototypes are frequently put through a rigorous testing regimen in order to spot these flaws and fix them. Testing identifies issue areas, which are then corrected (or redesigned) as necessary. A reliability metric (or parameter) of a product (component, subsystem, or system) that has improved over time as a result of modifications to the product's design and/or manufacturing procedure is referred to as reliability growth. **Reliability growth** is defined as the positive improvement in a reliability metric (or parameter) of a product (component, subsystem or system) over a period of time due to changes in the product's design and/or the manufacturing process. A reliability growth programme is a methodical procedure for identifying reliability issues through testing, implementing corrective measures, and tracking the improvement in the product's reliability throughout the test phases. The word "growth" is employed because it is believed that as design modifications and corrections are made, the product's reliability would eventually rise. However, in actuality, there may be no increase or even deceleration. A reliability growth programme typically has reliability goals attached to it. A programme may have many dependability objectives. For instance, there might be a reliability objective for faults that force unforeseen maintenance and a different goal for failures that result in catastrophic failure or mission abort. Safety-related failure modes may be connected to other reliability objectives. A crucial part of achieving these goals is keeping track of how the product's reliability grows over the course of succeeding testing phases. The goal of **reliability growth analysis (RGA)** is to measure and evaluate various factors (or metrics) that affect a product's reliability as it develops through time. The planning and management of the reliability growth process is addressed in reliability growth management in order to achieve the reliability objectives.

Duane Model

J. T. Duane produced a study in 1962 that included failure statistics from various systems that were under development. The cumulative MTBF vs cumulative operating time was observed to follow a straight line when plotted on log-log paper during data analysis, as illustrated in the following figure:

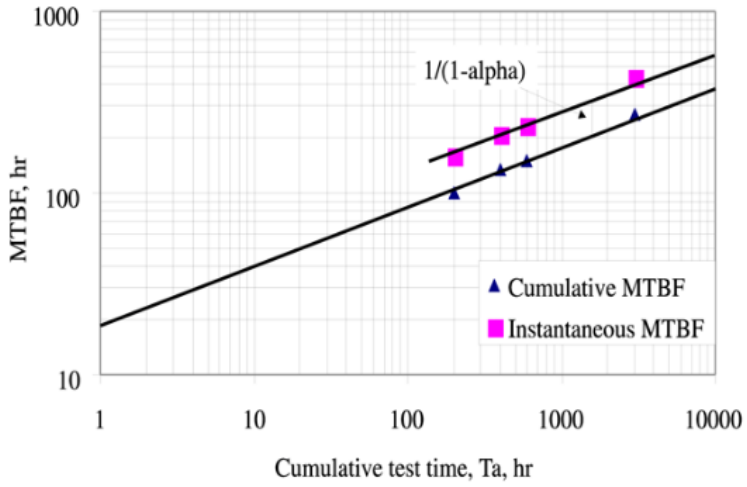

Figure 2-3 Cumulative Operating Time VS MTBF (Source:Reliawiki)

Based on that observation, Duane developed his model as follows. If N(T) is the number of failures by time T, the observed mean (average) time between failures, MTBFc, at time T is:

$$MTBF_c = \frac{T}{N(T)}$$

The equation of a straight line can be expressed as:

$$y = mx + c$$

Setting:

- $y = ln(MTBF_c)$
- $x = \ln(T)$
- $m = \alpha$
- $c = \ln(b)$

yields:

$$ln(MTBF_c) = \alpha \ln(T) + (b)$$

Equating $(MTBF_c)$ to its expected value assuming exact linear relationship: $E(MTBF_c) = bT^{\alpha}$

Assuming a constant failure intensity, the cumulative failure intensity λ_c is: $E(\lambda_c) = \frac{T^{-\alpha}}{b}$

The expected number of failures upto time T is: $E\big(N(T)\big) = \lambda_c T = \frac{T^{1-\alpha}}{b}$

where:

- λ_c is the average estimate of the cumulative failure intensity, failures/hour.
- T is the total accumulated unit hours of test and/or development time.
- 1/b is the cumulative failure intensity at T=1 or at the beginning of the test, or the earliest time at which the first λ is predicted, or the λ for the equipment at the start of the design and development process.
- α is the improvement rate in the λ, $0 \leq \alpha \leq 1$.

Where b= cumulative MTBF at T=1 or at the beginning of the test, or the earliest time at which the first $MTBF_c$ can be determined, or the $MTBF_c$ predicted at the start of the design and development process (b>0).

The cumulative MTBF, $MTBF_c$, and λ_c tell whether m is increasing or λ is decreasing with time, utilizing all data up to that time. You may want to know, however, the instantaneous $MTBF_i$ or λ_i to see what you are doing at a specific instant or after a specific test and development time. The instantaneous failure intensity, λ_i, is:

$$\lambda_i = \frac{d(E(N(T))}{dT} = \frac{1}{b}(1-\alpha)T^{-\alpha} = (1-\alpha)\lambda_c$$

The instantaneous failure intensity improvement line is produced by moving the cumulative failure intensity line downward, parallel to itself, by a distance of (1- α), as demonstrated in these derivations. The cumulative MTBF line is moved up, parallel to itself, by a distance of 1/(1- α) to produce the present or instantaneous MTBF growth line, as seen in the figure below:

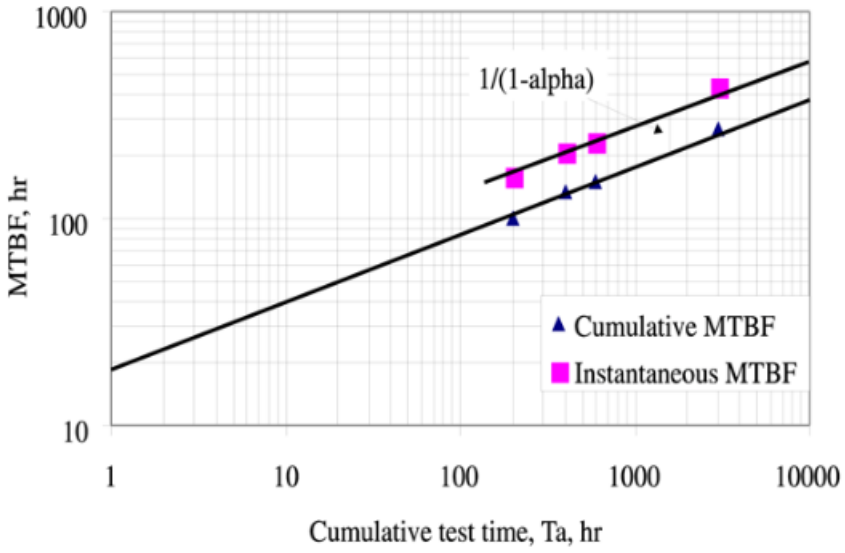

Figure 2-4 Cumulative Test Time VS MTBF (Source:Reliawiki)

Crow-AMSAA

According to Dr. Larry H. Crow [17], the Duane Model might be stochastically represented as a Weibull process, enabling the use of statistical techniques in this model's application to reliability increase. As a result of this statistical expansion, the Crow-AMSAA (NHPP) model was created. The U.S. Army Materiel Systems Analysis Activity(AMSAA) was where this technique was initially developed. On systems where utilisation is gauged on a continuous scale, it is widely used. When there is great dependability and a lot of trials, it can also be used to analyse one-time items.

Typically, test procedures are carried out phase by phase. Instead of tracking reliability across test phases, the Crow-AMSAA model is developed for tracking reliability within a single test phase. Multiple distinct test phases may be included in a development testing programme. The

Crow-AMSAA model can analyse this sort of testing and the related data if corrective measures are implemented during a specific test phase. The following can be ascertained with the use of the model, which assesses the reliability growth progress within each test phase:

- Reliability of the configuration currently on test
- Reliability of the configuration on test at the end of the test phase
- Expected reliability if the test time for the phase is extended
- Growth rate
- Confidence intervals
- Applicable goodness-of-fit tests

The cumulative number of failures is linear when shown on the ln-ln scale, and the reliability growth pattern for the Crow-AMSAA model is exactly the same pattern as for the Duane postulate. The Crow-AMSAA model is statistically grounded, in contrast to the Duane postulate. The failure rate on an ln-ln scale is linear according to the Duane postulate. However, when shown on an ln-ln scale for the statistical structure of the Crow-AMSAA model, the failure intensity of the underlying non-homogeneous Poisson process (NHPP) is linear.

3 Problem Statement

<u>Objective:</u> To prove whether traceability of the system affects the performance parameters like mean time between failures, next expected failure, reliability of the system, number of spares required for various components and failure rates of system.

3.1 Description of a System:

A system is combination of number of sub-systems present inside it. A system fails generally when one or more sub-systems present inside the system fails and it depends upon the configuration of the sub-system.

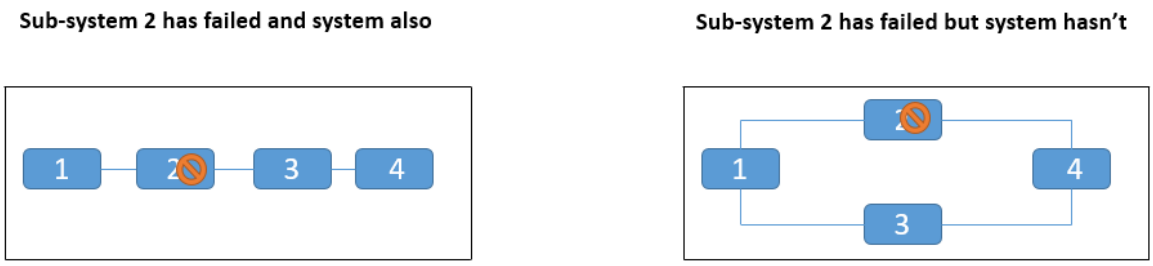

Figure 3-1-System Failure

3.2 What happens after a system is failed?

The 1st thing is obvious that identify the sub-system that has resulted in the failure of the system. Then we have 2 more options to follow: 1st is to remove out the sub-system sent it to the repair house where it actually gets repaired, make the system wait until that time until the sub-system comes out after getting repaired. 2nd is to send the failed sub-system to the repair house and bring out similar sub-system present in the repair house/ spare house as spare. The time to repair in the 1st cause is not certain as repairing the sub-system by taking into the repair house is a quite tedious task. It involves the transportation time to and fro from the sub- system and repair house. Along with that it involves the time to identify the component that has failed. So it is highly time consuming way of repairing. The 2nd option seems to highly feasible as the system will halt for a very short time which is the sum of time to bring the sub-system from the repair house/ spare house to the system and time of replacing the sub-systems with the newly bought sub-system.

3.3 What happens in a repair house?

The repair house accepts the failed sub-system from the system. The experts there try to identify out the component that has failed. Then they replace the component as we know components are non-replaceable entities and hence needs to be replaced. After performing this task they forward out the sub-system to the spare house where the sub-system is kept for further use whenever similar sub-system has failed in a different system. In this process the other components present in the system are old and have acquired some age. The spare house keeps all the spare sub-systems for immediate use if similar sub-systems have failed for any other systems. The spare house doesn't contain all fully new sub-systems, rather they are all repaired sub-systems.

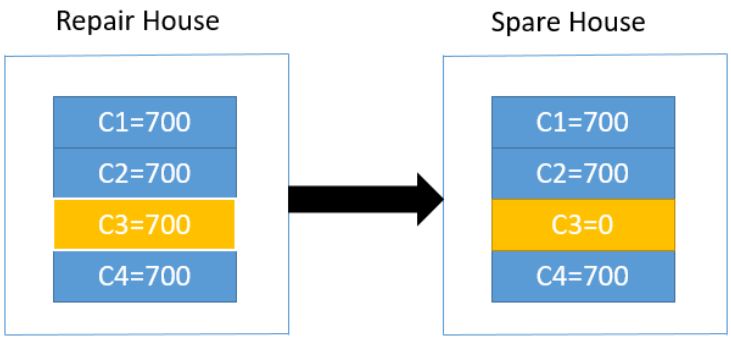

Figure 3-2-Repair house and Spare house

3.4 Traceability and No-Traceability:

Traceability is the process of continuous tracking of the sub-system throughout its entire movement from one system to another where the entire history of the sub-system is present along with it and it comes to the system along with its history. While No-Traceability is the process that where there is no history of the sub-system, the system assumes that the sub-system is as good as new but it is not the actual case.

To take up an example let us consider the sub-system that was already there in our spare house:

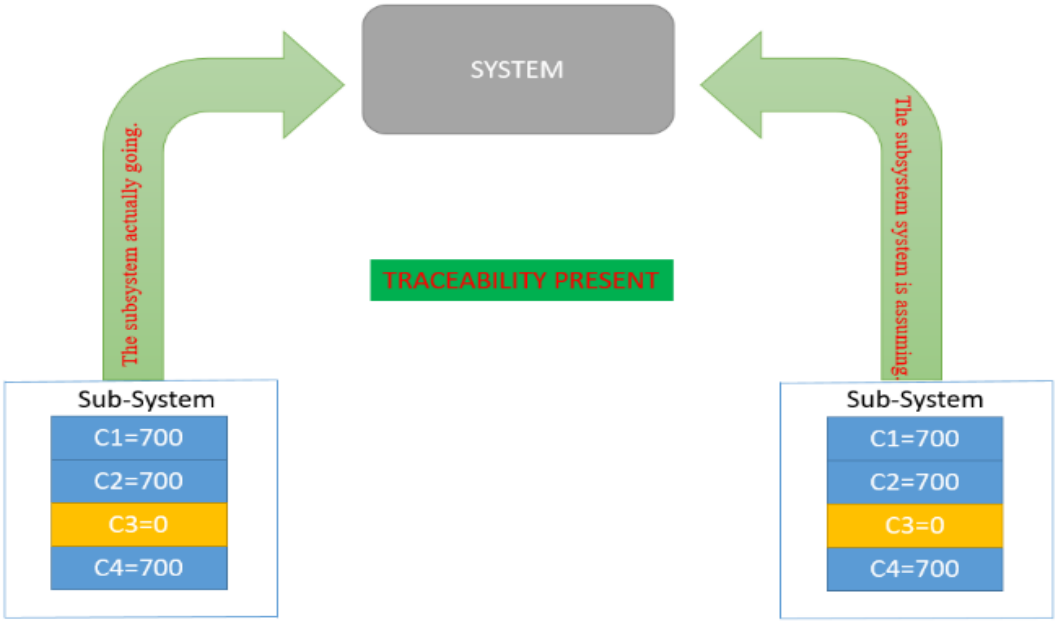

Figure 3-3-Traceability Present

The system accepted the history that the sub-system carries along with it. Hence the sub-system actually going and the sub-system the system is assuming collides with each other. This is the example of a traceable system.

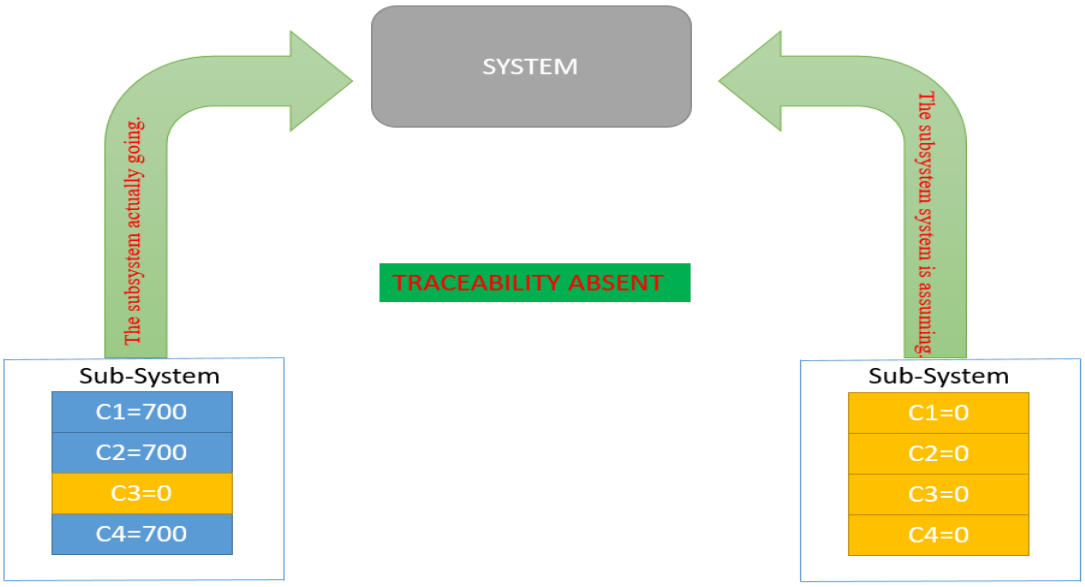

Figure 3-4-No traceability

The system has no idea about the history that the sub-system carries along with it. The system assumes that the sub-system is as good as new. But there may be components present in the system that are old. Hence the sub-system actually going and the sub-system the system assuming doesn't collides with each other. This is the example of a non-traceable system.

3.5 How to ensure traceability?

The important task here is to define the process to ensure traceability of the sub-systems that travel from one system to another or from repair house to spare houses. Every sub-system has to be given unique identity which it carries with itself. Along with the ID of the sub-system it carries along with it the failure behaviours of the component present in it as well as the models that each of the components follow. This reflects upon the age of the components that are present inside it. In addition to that we can fit in a power law model to each of the sub-systems and estimate the parameters of that model which can be used for future predictability of failure.

To easily demonstrate this let us take an example.

A system has failed and we got to find out it happened due to sub-system **B** present in the system. Now it's the task to send the sub-system to the repair house and bring in a spare sub-system from the spare house. As we have IDs for each of the sub-systems the sub-system that has failed has an ID "B412S". Before going to the repair house it goes with its history from the system like {ID: "B412S", C1:700, C2:700, C3:700, C4:700, C5:700}. It can be number of hours the sub-system B has operated in the system-1. Here we assumed it to be 700hrs. The repair house experts try to find out the component (Here C2) that led to failure of the sub-system and replace the component. After doing that they need to update the sub-system details where they would reset the component C2's age zero and others' as it is. After updating these they send it to the spare house. The spare house will have the sub-system in the form of: {ID: "B412S", C1:700, C2:0, C3:700, C4:700, C5:700}.And in the meantime from the spare house a similar sub-system of category **B** goes into the system-1. It will have its own history like in our case {ID: "B722F", C1:100, C2:50, C3: 0, C4:250, C5:350}

After some time let us take that another system's (here system-2) sub-system –B which was brand new has failed after 800 hrs of operation {ID: "B611F", C1:800, C2:800, C3:800, C4:800, C5:800}. This will again go to repair house where the repair experts find out that component C4 was responsible for the failure and they just replace that. In addition to that they need to update the history of the sub-system before sending it to the spare house to ensure the traceability so the spare house will have the sub-system in form of {ID: "B611F", C1:800, C2:800, C3:800, C4: 0, C5:800}. And in the meantime from the spare house a similar sub-system of category **B** goes into the system-1. It will have its own history like in our case {ID: "B412S", C1:700, C2:0, C3:700, C4:700, C5:700}. It was the same sub-system that had gone to the previous system (system-1).

In the spare-house we have spares of all the possible sub-systems(Here-2) that can be there and number of spares would depend on a large number of factors like mean time to repair of the sub-system, mean time of failure of the sub-system, space and cost constraints.

Hence to ensure the traceability there needs some work to be done at the system, repair house and spare house side. The system should update the time between failures of the sub-system before sending it to the repair house. The repair house should accept the sub-system along with its ID identify the component that has failed update the failure model and reset the age of the component in the sub-system data to zero and make no changes to others.

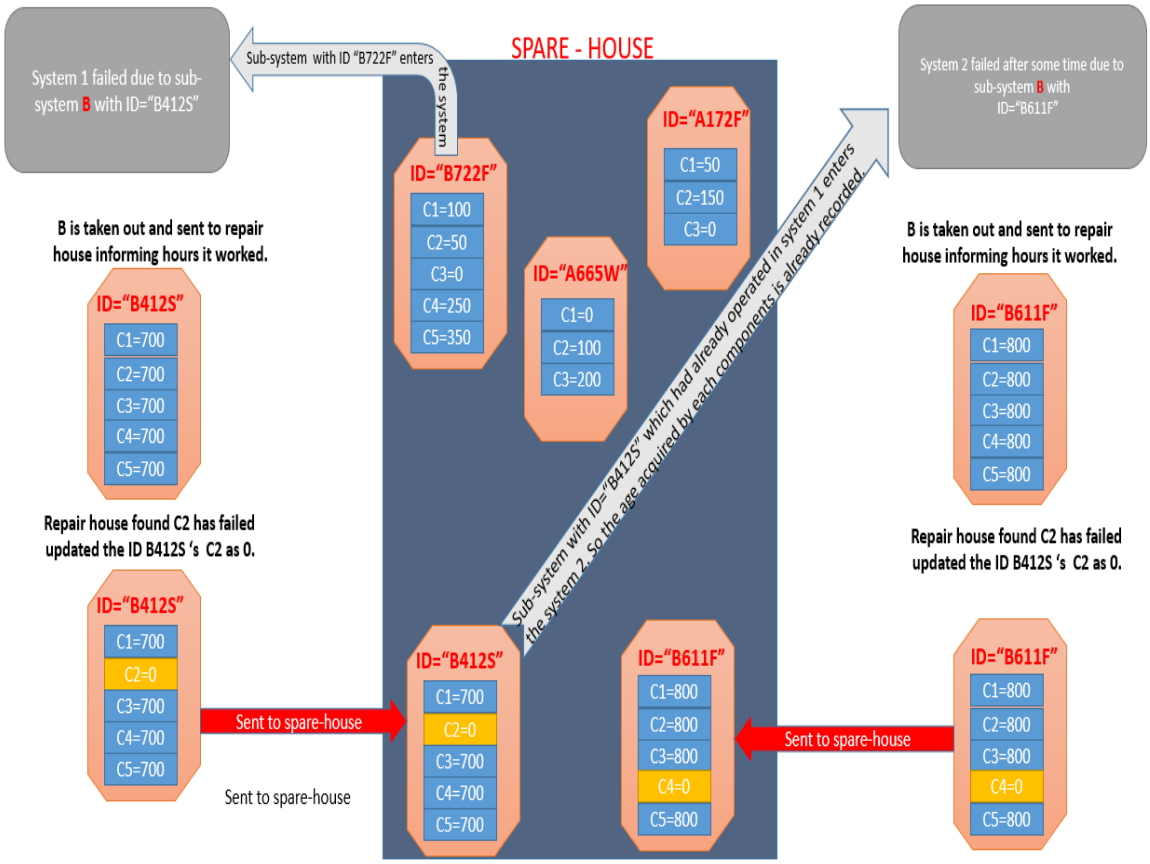

Figure 3-5- Methodology to ensure traceability

3.6 Advantages of Ensuring Traceability:

Ensuring traceability will imply better predictability of various performance measures like the next expected failure, mean time between failures, failure rate, and number of spares required, etc. It can be demonstrated through the sub-system **B** from the system-1 that was used earlier. After the 1st failure of the system-1 due to sub-system B it is now replaced with another similar sub-system **B** that has all history details as follows: {ID: "B722F", C1:100, C2:50, C3: 0, C4:250, C5:350}. If there is no traceability the incoming sub-system is as good as new. Though it won't have an ID and this things we can assume as {ID: "B722F", C1:0, C2:0, C3: 0, C4:0, C5:0}.If we know the mean time between failures (MTBF) we can predict the next expected failure. In case of traceability for MTBF of 600hrs we predict that the next expected failure is about 300hrs and in case of no traceability it is about 600hrs. And the actual failure for the sub-system will happen in around 300hrs and hence we are more correct in our predictions if we ensure traceability.

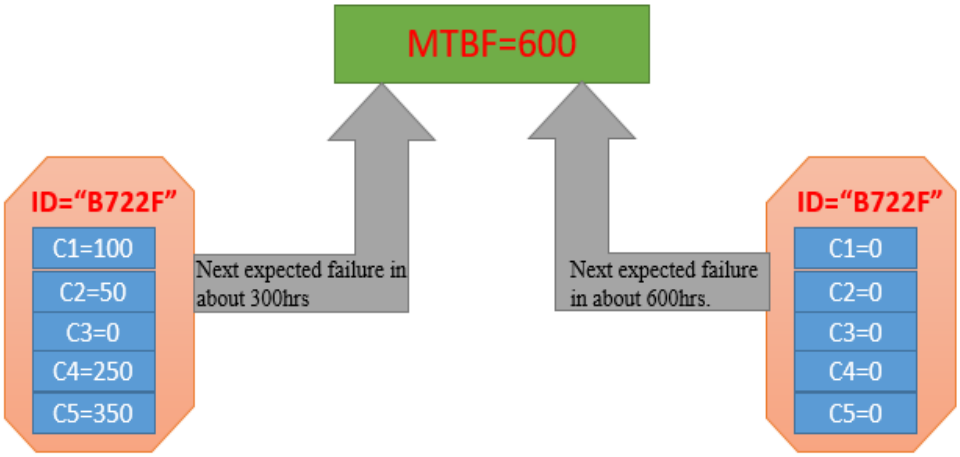

Figure 3-6- Advantages of ensuring Traceability

Generally the organisations only deal with failure data at a system level. They note down the failure data points at a system level. But noting them at a sub-system level will help achieving better accuracy in predictability and this will improve when traceability is done as better results will come out. As the system is a repairable entity, we can fit a generalized non-homogeneous Poisson Process (NHPP) and estimate the parameters to predict the next failure and to check whether the system is improving or degrading with time. Similarly even the sub-system is a repairable entity and hence the same thing can be done with it for predictability. But if traceability is not there we could go horribly wrong in our predictions.

For example: Based on the past historical data for the sub-system with ID= "B412S" we have got a NHPP model as $\mathbf{0.36T^{0.49}}$ and when it fails leading to go to the repair house a different subsystem comes from the spare house (here subsystem with ID= "B722F"). The sub-system with ID="B722F" has got a NHPP model as $\mathbf{0.76T^{0.65}}$ .If traceability is not there then we would assume the model that repairable sub-system model B has $0.36T^{0.49}$ and we predict performance measures according to it. But it will actually show incorrect values as the actual model the sub-system follows is $0.76T^{0.65.}$ We hope to prove that traceability is important but the extent to which it is important needs to find out.

4 WORK DONE

4.1 Need For Simulation

To perform any kind of reliability based modelling and analysis we need data. The data will mostly be failure times or time between failures. For gathering data we have some options to choose. The 1st one is to collect the data from industries that keep a track on. The 2nd one is to refer handbooks that have a list of similar components and parameter values associated with them. The 1st one is currently quite tricky to get while handbooks give generalised values. So data collected from it won't we always correct. Hence we are going to rely on simulations to generate the data and work on it.

4.2 Models Used:

To model an entire system we need to model to root of the system. Every repairable system consists of non-repairable components. The components are always replaced when failed. But as the system is not replaced we regard the system to be repaired in form of component replacement.

There are generally 3 types of models that are used to quantify the reliability of individual components:

1. Statistical Models
2. Shock Based Models
3. Degradation Based Models

4.2.1 Statistical Models:

The statistical based models consider time to failure as their important criteria for reliability prediction. The statistical models are basically classified into 2 types of probability based models one is discrete model and other is continuous probability model. Generally the discrete probability best model are classified into 2 types of distributions: binomial and poison distribution. Call you similarly the continuous best distributions are classified into 5 different types: 1: Exponential distribution 2: normal distribution 3: lognormal distribution 4: Gamma distribution 5: Weibull distribution. Here we have just considered the **2-parameter Weibull Model** for our system. It is one of the widely used reliability analysis models because it has a widely used application in hazard rate curve modelling. The Weibull distribution is applicable to many engineering products as well as for reliability testing material strength and warranty analysis. β is the shape parameter. The shape parameter indicates the failure mechanism a component can undergo. It also helps in determining the shape of the hazard rate function. We have assumed the shape parameter to be greater than 1 for all the components. As the value of β>1 then the hazard rate is increasing and the component can undergo wear-out. η is the scale parameter. It decides the effect of scaling the time axis. For a constant value of β if we increase the value of η to the distribution will stretch to the right while maintaining the starting location of it as well as the shape. There are total **8** out of **20** components in the system that belong to Weibull 2-parameter model.

4.2.2 Shock Based Models:

The reliability models in statistical failure consider time as the criteria of failure and reliability across time. In some cases the criteria for failure may not be time. It may be the number of events occurring and the failure will occur at an event and not in between 2 events. The usage unit may be just a raw number. Here the unit of usage is no of the times the component has been used irrespective of the number of hours it is used. If there are sudden shocks or voltages surges, this criteria needs to be considered while modelling a component as well.

- Basic Shock Model: The basic shock model is modelled where the system fails when number of shocks exceed a given number. 2 components of the system belong to basic shock model.
- Extreme Shock Model: The extreme shock model is modelled considering the factor that the system fails when the magnitude of a shock exceeds a given level or the number of shocks exceed a given number, the 1st occurrence among the two. 2 components of the system belong to extreme shock model.
- Cumulative Shock Model: The cumulative shock model is modelled considering the factor that the system fails when the cumulative sum of all shock magnitudes exceed a given level or the number of shocks exceed a given number, the 1st occurrence among the two. 2 components of the system belong to cumulative shock model.
- Run Shock Model: The run shock model is modelled considering the factor that the system fails when ‘k’ consecutive shocks with critical magnitude occur. 2 components of the system belong to cumulative shock model.

4.2.3 Degradation Based Models:

Degradation based models are modelled in a way that the component follow a degrading pathway with respect to time that are specified by certain equations. The equation determines the amount of degradation the component has undergone until a specified time. There is a threshold for each one of the degradation model. For some components the threshold has to be exceeded in order to declare the component as failed while for other components the threshold has to be undervalued to declare the component as failed. There are 4 components in the system that follows a degradation based mechanism as their inherent failure model. The description of failure of each of models along with their parameters as well as thresholds are written along with the components.

4.3 System Definition:

The virtual system can be considered as a combination of various sub-systems. Here we have considered a repairable system that consists of 5 sub-systems and a total of 20 components are present in the system where the number of components are non-uniformly divided within the different sub-systems where we have a maximum of 6 components in a sub-system and a minimum of 2 components in a sub-system. The system break- down structure is quite simple where we have sub-systems inside the system hierarchy and it ultimately boils down to components level which is the end and the components are obviously non-repairable and hence

once they are failed they are needed to be replaced. Although the components are non-repairable but the sub-system as well as the system itself is a repairable system where the system repair can be performed by replacing the components which are failed and responsible for system failure. The outline of the system is given in the below figure.

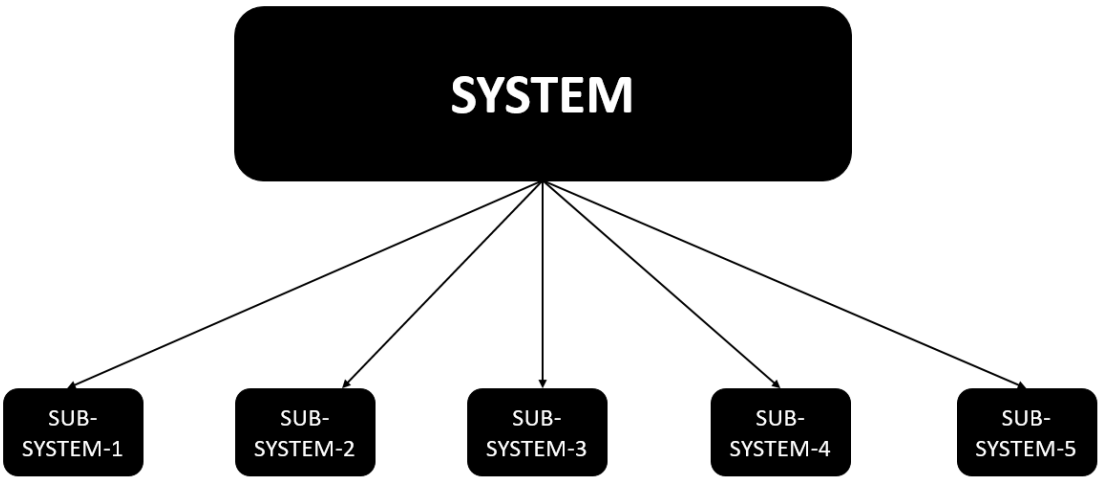

Figure 4-1 System Breakdown

Each of the sub-system present is assumed to be in series connection with every other sub-system. This implies that the system gets failed when any one of the sub-system gets failed. This can be inferred from the reliability block diagram.

Figure 4-2-System Configuration

4.3.1 Sub-System-1:

Sub-system-1 is one of the repairable sub-systems where we have 3 components in it. The 3 components follow all the 3 different types of models i.e. Statistical Models, Shock Based Models and Degradation models. The 3 components were connected in series that means every components failure is regarded as the failure of the sub-system. The reliability block diagram is drawn as follows:

Figure 4-3-Sub system 1 configuration

C11:

This component follows 2-parameter Weibull model where the shape parameter β =1.5 and the scale parameter η =1800. The probability density function for the function can be described as follows:

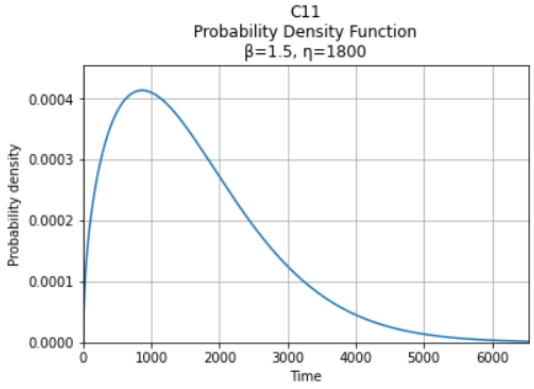

Figure 4-4-C11 failure behaviour

C12:

This component follows extreme shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	200hrs
Number of shocks permissible:	9
Distribution for magnitude of shocks:	Normal Distribution
Mean of the magnitude:	14
Standard deviation :	0.45
Threshold magnitude permissible:	15

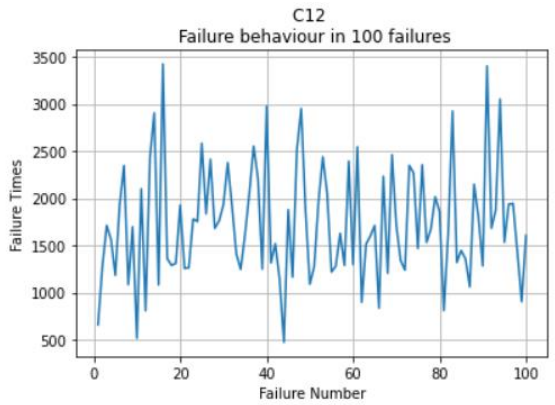

Figure 4-5-C12 failure behaviour

C13:

This component follows a degradation based failure mechanism. The parameters and distribution for the failure of this component are as follows:

Distribution:	exp(z*t)
Parameter:	Z=(0.005,0.0053)
Threshold of degradation:	500

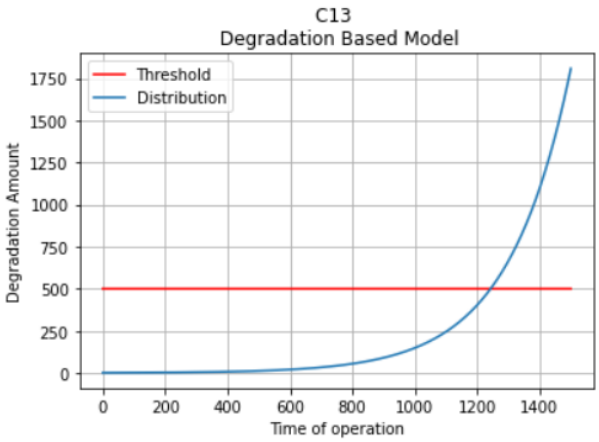

Figure 4-6-C13 failure behaviour

4.3.2 Sub-System-2:

This is the smallest sub-system in the entire system consisting of only 2 components each belonging to the class of shock- based and statistical model respectively. Both the components are arranged in series configuration. The reliability block diagram is as follows:

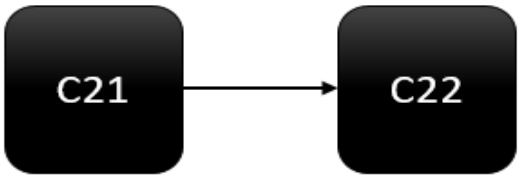

Figure 4-7-Sub-system 2 configuration

C21:

This component follows basic shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	250
Number of shocks permissible:	8

Figure 4-8-C21 failure behaviour

C22:

This component follows 2-parameter Weibull model where the shape parameter β =1.1 and the scale parameter η =2200. The probability density function for the function can be described as follows:

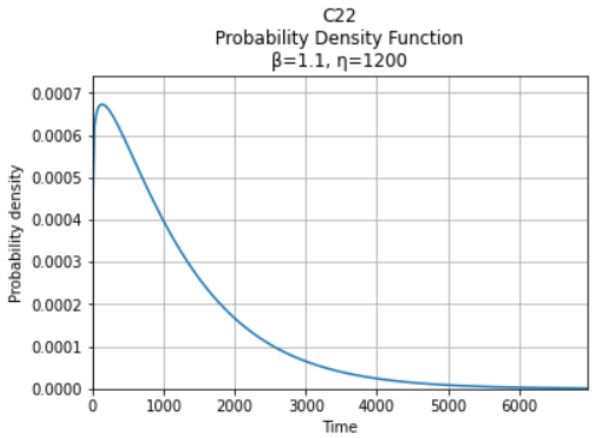

Figure 4-9-C22 failure behaviour

4.3.3 Sub-System-3:

This is the largest sub-system in the entire system consisting of **six** components out of which two belong to class of shock- based models, three belonging to statistical model and one belongs to degradation based models respectively. All the components are arranged in series configuration. The reliability block diagram is as follows:

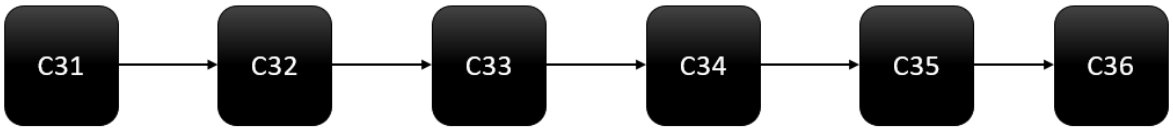

Figure 4-10-Sub-system 3 configuration

C31:

This component follows basic shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	220
Number of shocks permissible:	9

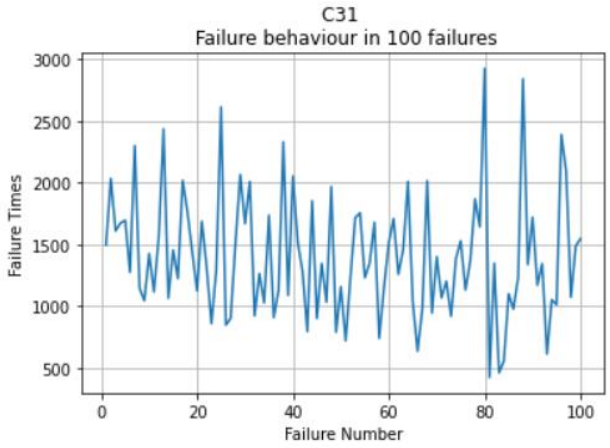

Figure 4-11-C31 failure behaviour

C32:

This component follows 2-parameter Weibull model where the shape parameter β =1.45 and the scale parameter η =1800. The probability density function for the function can be described as follows:

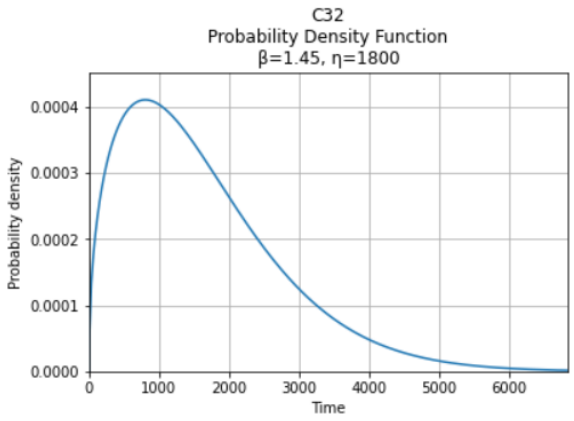

Figure 4-12C32 failure behaviour

C33:

This component follows extreme shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	170hrs
Number of shocks permissible:	8
Distribution for magnitude of shocks:	Normal Distribution
Mean of the magnitude:	10.5
Standard deviation :	0.9
Threshold magnitude permissible:	12

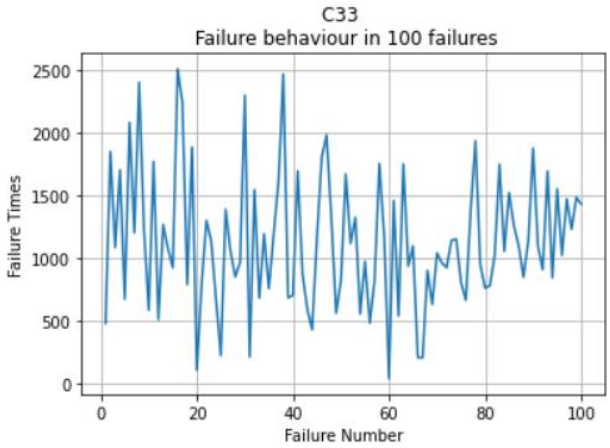

Figure 4-13-C33 failure behaviour

C34:

This component follows 2-parameter Weibull model where the shape parameter β =1.8 and the scale parameter η =2000. The probability density function for the function can be described as follows

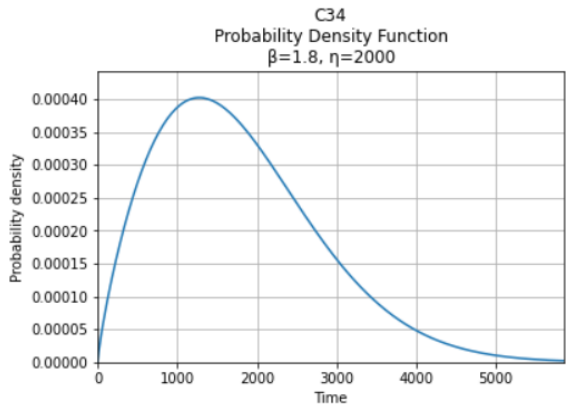

Figure 4-14-C34 failure behaviour

C35:

This component follows a degradation based failure mechanism. The parameters and distribution for the failure of this component are as follows:

Distribution:	log(1+z*t)
Parameter:	Z=(0.005,0.0055)
Threshold of degradation:	1.9

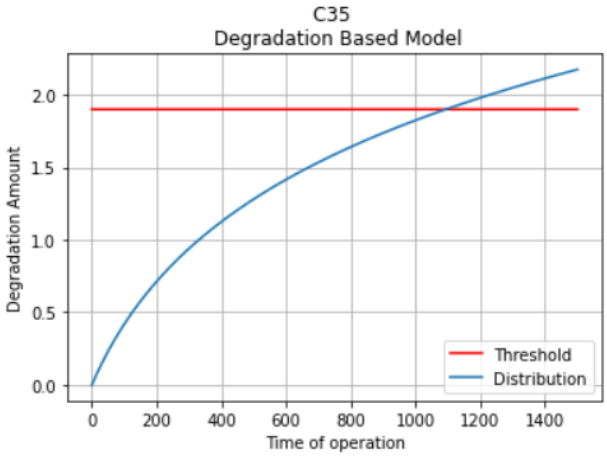

Figure 4-15-C35 failure behaviour

C36:

This component follows 2-parameter Weibull model where the shape parameter β =1.7 and the scale parameter η =1700. The probability density function for the function can be described as follows:

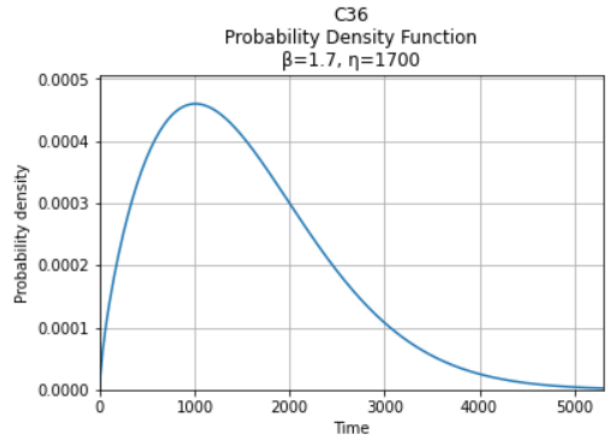

Figure 4-16-C36 failure behaviour

4.3.4 Sub-System-4:

This sub-system consists of **five** components out of which three belong to class of shock- based models, one belonging to statistical models and one belongs to degradation based models

respectively. All the components are arranged in series configuration. The reliability block diagram is as follows:

Figure 4-17-Sub-system-4 configuration

C41:

This component follows cumulative shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	120hrs
Number of shocks permissible:	8
Distribution for magnitude of shocks:	Normal Distribution
Mean of the magnitude:	6
Standard deviation :	0.2
Threshold cumulative magnitude permissible:	45

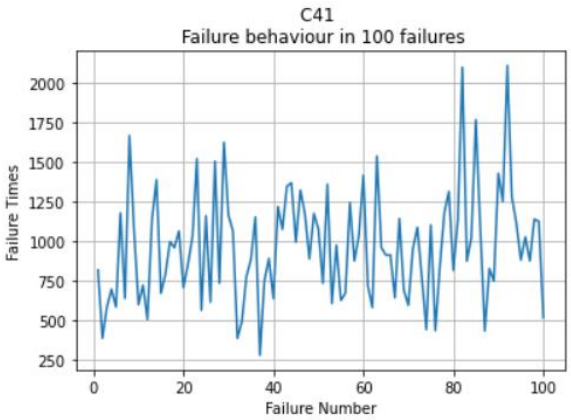

Figure 4-18-C41 failure behaviour

C42:

This component follows cumulative shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	130hrs
Number of shocks permissible:	11
Distribution for magnitude of shocks:	Normal Distribution
Mean of the magnitude:	2
Standard deviation :	0.9
Threshold cumulative magnitude permissible:	35

Figure 4-19-C42 failure behaviour

C43:

This component follows run shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	100hrs
Number of shocks permissible with magnitude :	5 shocks with magnitude 8
Distribution for magnitude of shocks:	Normal Distribution
Mean of the magnitude:	6
Standard deviation :	2

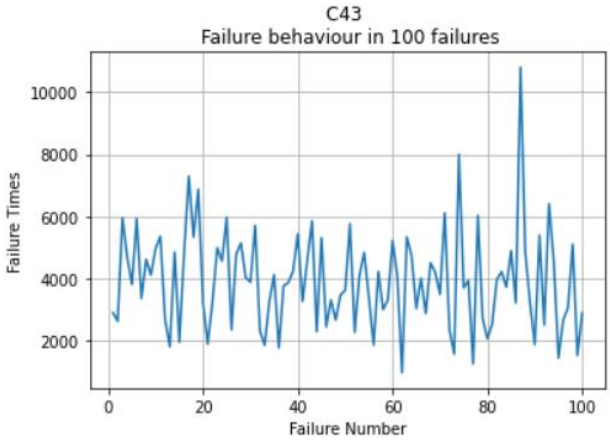

Figure 4-20-C43 failure behaviour

C44:

This component follows a degradation based failure mechanism. The parameters and distribution for the failure of this component are as follows:

Distribution:	exp(z_0-z_1*t)
Parameter:	z_0=5 ; z_1=(0.002,0.0025)
Threshold of degradation:	15

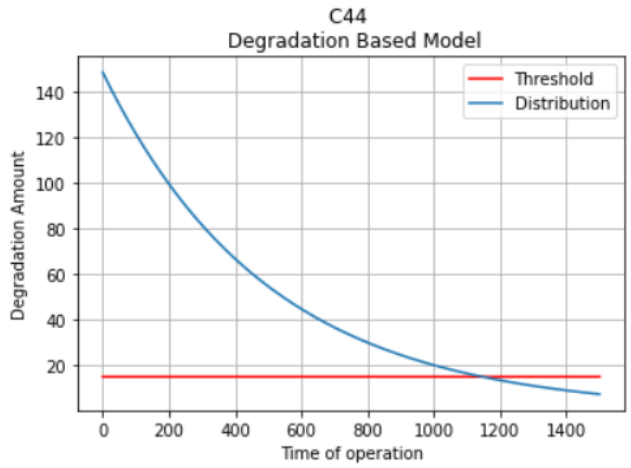

Figure 4-21-C44 failure behaviour

C45:

This component follows 2-parameter Weibull model where the shape parameter β =1.57 and the scale parameter η =1850. The probability density function for the function can be described as follows:

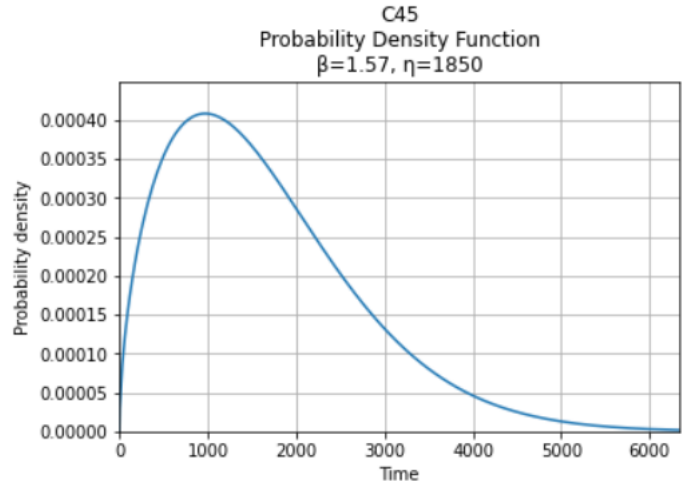

Figure 4-22-C45 failure behaviour

4.3.5 Sub-System-5:

This sub-system consists of **four** components out of which one belong to class of shock- based models, two belonging to statistical models and one belongs to degradation based models respectively. All the components are arranged in series configuration. The reliability block diagram is as follows:

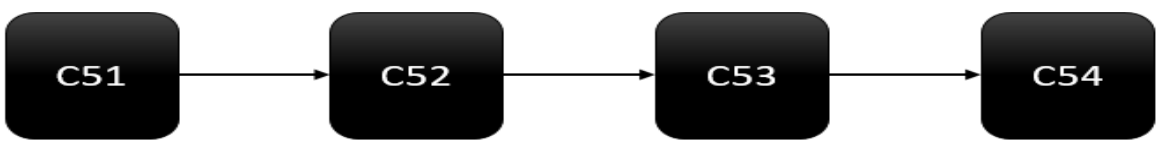

Figure 4-23-Sub-system-5 configuration

C51:

This component follows 2-parameter Weibull model where the shape parameter β =2.2 and the scale parameter η =1325. The probability density function for the function can be described as follows:

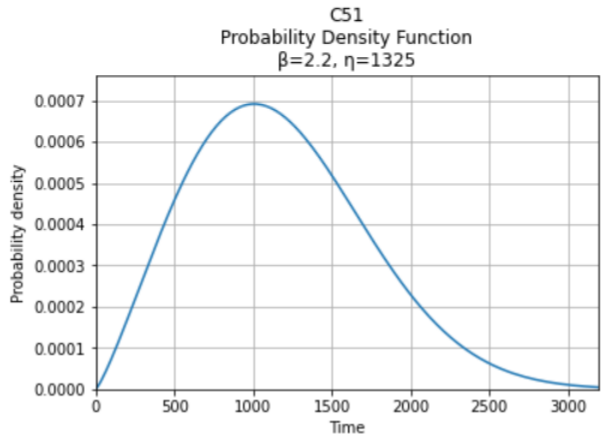

Figure 4-24-C51 failure behaviour

C52:

This component follows 2-parameter Weibull model where the shape parameter β =1.62 and the scale parameter η =1550. The probability density function for the function can be described as follows:

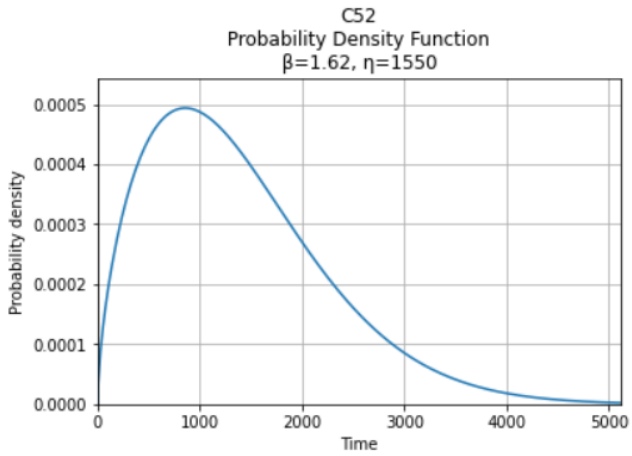

Figure 4-25-C52 failure behaviour

C53:

This component follows run shock model. The distributions, parameters and failure behaviour of the model for 100 failures are:

Distribution for the time between arrival of shocks:	Exponential Distribution
Mean time between arrival of shocks:	80hrs
Number of shocks permissible with magnitude :	6 shocks with magnitude 7
Distribution for magnitude of shocks:	Normal Distribution
Mean of the magnitude:	6.5

Standard deviation :	0.65

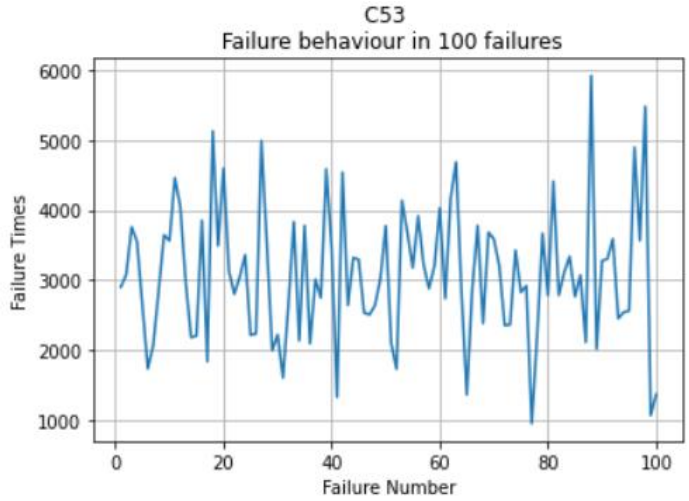

Figure 4-26-C53 failure behaviour

C54:

This component follows a degradation based failure mechanism. The parameters and distribution for the failure of this component are as follows:

Distribution:	z1-z2*(t^0.45)
Parameter:	z1=6 ; z2=(0.08,0.12)
Threshold of degradation:	3.6

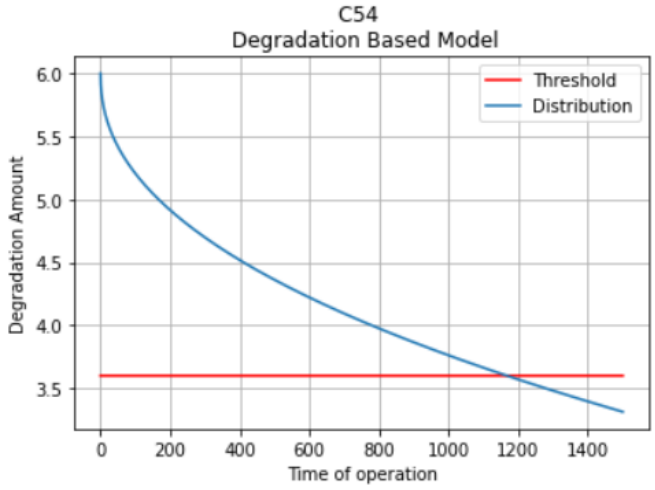

Figure 4-27-C54 failure behaviour

4.4 Common Causes:

Common causes are such events that arrive randomly and they affect multiple components present in the system. Considering an example: Inadequately rated rectifying diodes on identical twin printed circuit boards failing simultaneously due to a voltage transient. Here the voltage transient is the common cause.

Typically, causes arise from:

- Requirements: incomplete or conflicting
- Design: Software or noise
- Manufacturing: batch-related component deficiencies
- Maintenance/operations: human-induced or test equipment problems
- Environment: temperature cycling or electrical interference

Common causes are classified generally into 2 categories:

1. Discrete common causes: These events are discrete in nature and the performance of the system depends on whether they come or not. The time between occurrences of discrete common cause is assumed to have exponential distribution with a certain mean. Every time these events happen they don't affect the system directly. There is a conditional probability associated with them for every components denoted as Pr (affect|occurred). If the component gets affected by the common cause it will reflect in the parameters of the model that the component holds. The magnitude has an effect on the residual life of the component after which the common cause arrives on the component.
2. Continuous common causes: These events are continuous in nature and the performance of the system depends on the magnitude of the event. The time between occurrences of continuous common causes is assumed to have exponential distribution having a certain mean. The probability that the component gets affected if the magnitude of the event exceed a certain value is always 1. The magnitude of the effect that the event carries follows a certain distribution and hence it gets correlated with the underlying distribution of the component that it has affected.

Effects of Common Causes:

The effects of common causes can be divided into 3 types of models that we are using and effects would be different in all the 3 cases:

1. Statistical Models: In statistical models especially Weibull model the effect of common cause is reflected on the scale parameter eta where it comes out to be a multiplier to the scale parameter and it reflects on the residual life of the component. In mathematical terms: If T_CC is the time of arrival of the common cause and the component hasn't failed until this time T_CC
    - Until the time T_CC the component operates having a scale parameter η.
    - After that period for the remaining residual life of the component the scale parameter gets affected due to effect of the common cause and it becomes x* η

where 0<x<1. This value depends on the intensity of the common cause and a lot of other factors.

The effects of the common cause can be easily reflected where it can be seen from the below graph for the failure of a component for about 10 times with and without the effect of common cause with demonstrating the cumulative hours of failure.

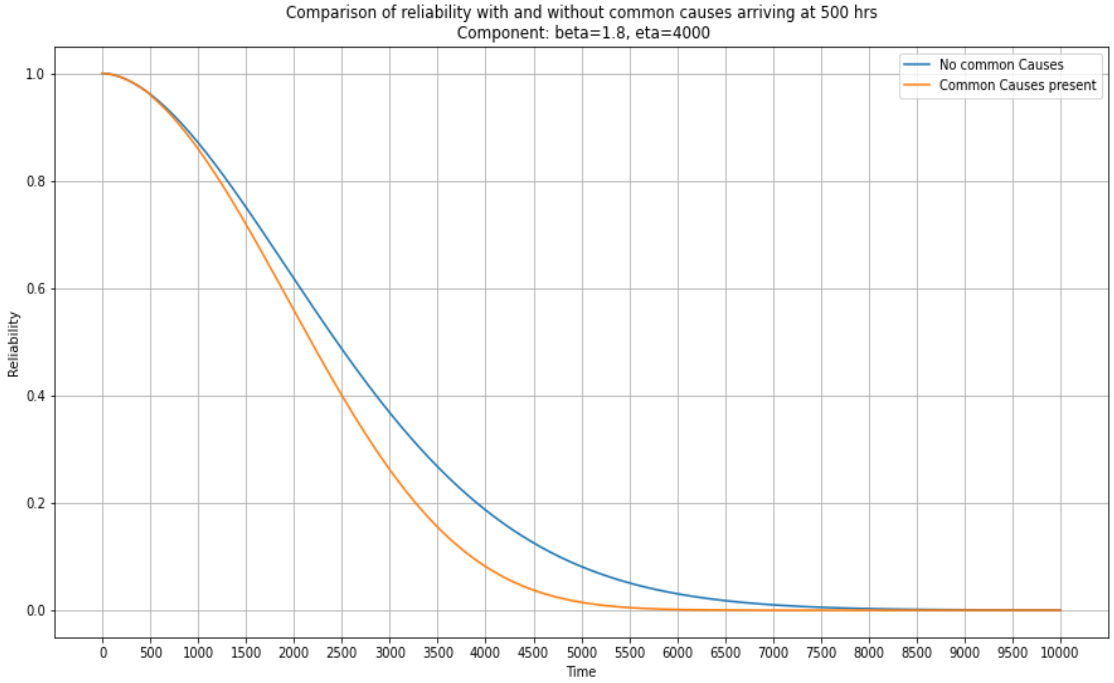

Figure 4-28- Effect on Reliability with and without common causes

2. Shock based Models: In shock based models the effect of common causes depends on the type of shock based models we are using. In general if a common cause occurs before the failure of a component then the number of shocks that the component can resist is decreasing a positive integer. If there is an involvement of the magnitude of the shock then there are 2 possibilities: Firstly the mean/standard deviation of the shock may get increased. Secondly, the threshold magnitude of component that the component can hold gets decreased by a multiplier.
   In mathematical terms: If T_CC is the time of arrival of the common cause and the component hasn't failed in until this time. Then:
   - Find the factor (a positive integer) that the threshold of shocks is subtracted from.
   - Check whether the threshold shock number is already achieved. If YES, then T_CC is the time to failure.
   - Else find out the multipliers for mean, standard deviation or threshold magnitude of shock.
   - If multiplier is there for threshold magnitude (M) then x* M where 0<x<1. This value depends on the intensity of the common cause and a lot of other factors.
   - If multiplier is there for mean (m) or standard deviation (std) then (1+x)* mean or (1+x)* std where 0<x<1. This value depends on the intensity of the common cause and a lot of other factors.

3. Degradation based Models: In degradation based models the effect of common causes is directly seen on the parameters of the degradation model we are using. It is sometimes reflected on the threshold magnitude of degradation permissible. In general if the component hasn't failed yet and common cause has arrived in between. After that time the parameter value gets increased so that the required degradation is achieved faster leading to quicker failure of the component. There is a different scenario as well where the threshold has decreased leading to quicker failure.
   In mathematical terms If T_CC is the time of arrival of the common cause and the component hasn't failed in until this time. Then:
   - Find the multiplier to the parameter (z). Let x be that. After T_CC magnitude=(1+x)*z where 0<x<1
   - If there is a multiplier(x) to the threshold magnitude. The new_thres=thres*x where 0<x<1.
   - If both are there implement both

The diagram below shows the comparison between the presence and absence of common cause for a degradation based model. Here the common cause arrived at 500hrs and it can be evident from the graph the parameters of the degradation model has changed after 500hrs that leaded in the variation of magnitude of degradation after that. Along with that the threshold has decreased for the component undergoing the effect of common cause. This resulted in early arrival of failure for the component that had faced the effect of common cause.

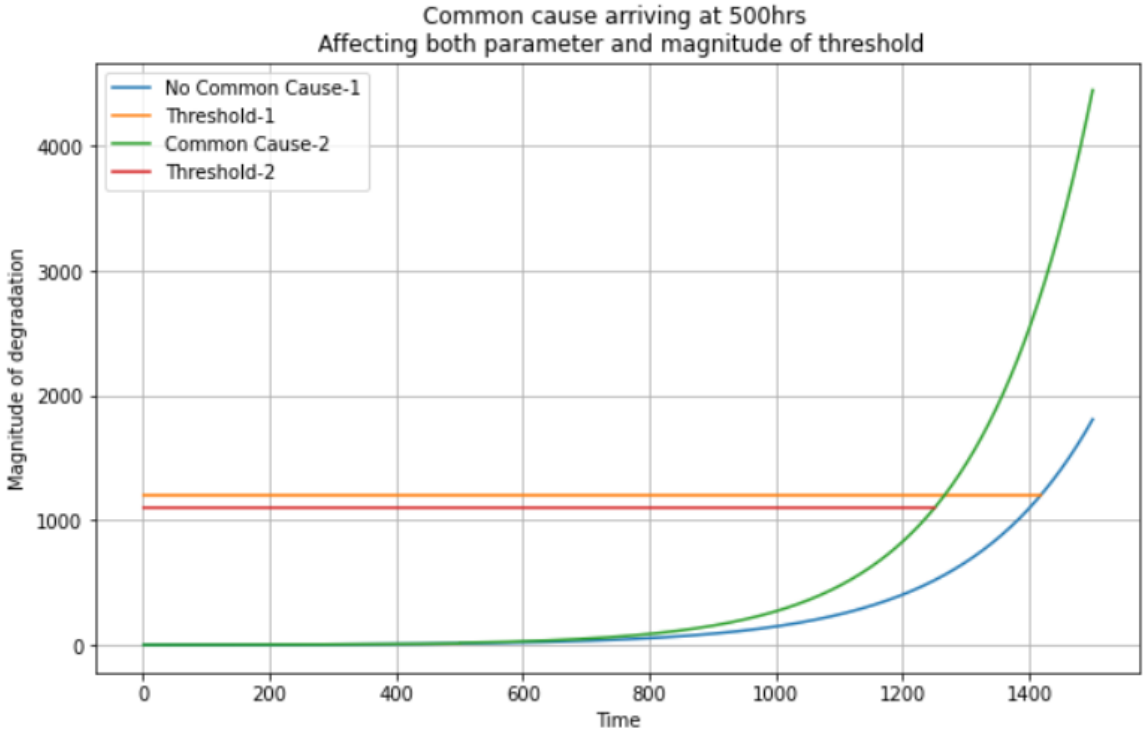

Figure 4-29- Degradation impact with and without common causes

4.5 Failure Simulation Procedure:

To generate the data we are preparing simulators. The simulators are quite specific to our system and components present in the system. The failure simulation procedure needs to be correctly defined in order to generate perfect data that can be used for further analysis. We need to define the failure simulation from the lowest level to the topmost level i.e. from the component to the entire system. For components we have 3 broad classification of models:

4.5.1 Statistical Models:

This models based components are failure simulated by random number using inverse transformation approach. The rule of probability integral transform states that if X is a continuous random variable with cumulative distribution function F_X, then then the random variable $Y= F_X (X)$ has a uniform distribution on [0,1]. The inverse probability integral transform is just inverse of it.

Let T be the random variable (time) whose distribution can be described by the cumulative distribution function F (T). We need to generate values of T which are the **failure times** of the component as per this statistical model distribution and apply the inverse transform sampling method where:

- Generate a random number y from a uniform distribution within the interval inside [0, 1].
- Compute the value of T such that F(T)=y
- Consider T to be the random number that is drawn from the distribution F(T)

For the Weibull distribution $R(t) = \exp(-t/\eta)^{\beta}$ where R (t) is reliability. We can consider this as our distribution and generate T from that by the inverse probability integral transformation approach we can find out the time to failure of the component which we are simulating on.

4.5.2 Shock Models:

The shock based models are simulated in a different way as we have a different type of shock based models. The time between arrivals of shocks is always exponential distribution.

- Generate time of 1st shock and continue it till Nth shock where N is the number of shocks permissible if it is basic shock model. Calculate the cumulative time until Nth shock which gives the time to failure of that component belonging to basic shock model.
- If the component follows extreme shock models for time of each shock generated generate the magnitude of the shock. If at any shock until the Nth shock the magnitude of the shock has exceeded the threshold magnitude that declare the component to have failed and the cumulative time of arrival of that shock is the time to failure of system else the system fails at Nth shock cumulative time.
- If the component follows cumulative shock models for time of each shock generated generate the magnitude of the shock. Add up the magnitude of previous shock to the next shock which results in cumulative magnitude of shocks. If at any shock until the

Nth shock the magnitude of the shock has exceeded the cumulative threshold magnitude that declare the component to have failed and the cumulative time of arrival of that shock is the time to failure of system else the system fails at Nth shock cumulative time.

- If the component follows run-shock model, generate the time of arrival of shock along with its magnitude. If the magnitude exceeds a certain threshold then count that shock. Perform the count operation until we get the desired threshold number of shocks. Return the cumulative time until that to denote the failure of the component.

4.5.3 Degradation Models:

Though there are a different types of degradation models the models that we have used are a bit easier to simulate.

- Take a comparatively large time frame and divide them into smaller time steps.
- For every time step find the magnitude of degradation that component has undergone.
- Check whether the magnitude of degradation has exceeded or undervalued the threshold magnitude of degradation.
- Declare the time step where magnitude has exceeded as the time to failure of the component

4.6 System Failure Simulations:

System's failure simulation depends upon the components and the sub-systems present in the system. The steps involved in simulation procedure are as follows:

- Initially assuming the age of components to be zero run the simulation of all components until failed for every sub-system.
- Note down the time to failure of all the components an find the minimum time to failure. This will denote the actual failure time of the system.
- Do this for every sub-systems and you would have failure times of every sub-systems present.
- System failure time is the minimum of failure times of each of the sub-systems present.
- Reset the failure time of every sub-system until component level to the system's failure time.
- Perform the next simulation assuming that the system including sub-systems and components has acquired age except failed component/ sub-system.

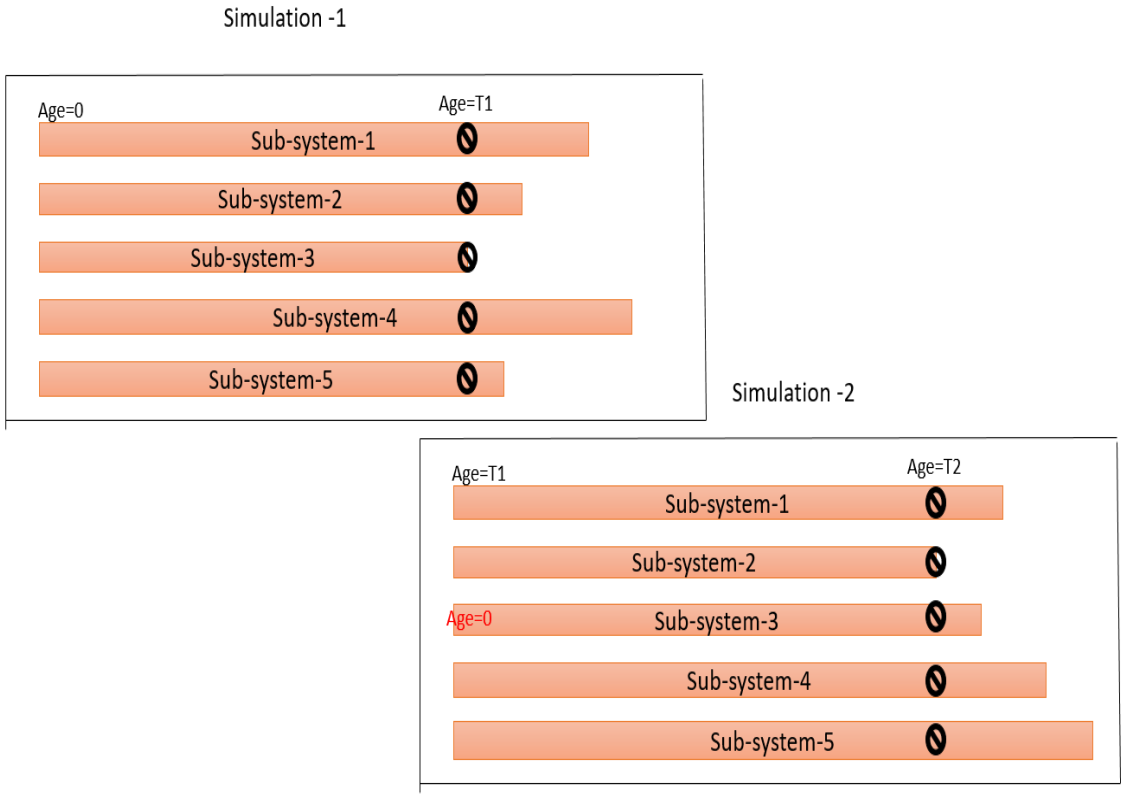

Figure 4-30-System Simulation Procedure

In the above figure the sub-system-3 gets failed 1st hence the simulation is stopped when the sub-system has failed. The age of the entire system including the sub-system is set to T1. Then assuming that the sub-system-3 is replaced completely the age of that sub-system is reset to zero. For the next simulation considering the initial age acquired by each of the sub-systems the simulation is performed. Then it is observed that sub-system-2 has failed and hence is replaced setting its age to 0 while others have age.

4.7 Fitting Power Law Model:

As we already know that the power law model used for repairable systems' failure rate is of the form:

$$u(t) = \frac{\beta}{\eta^{\beta}} \beta t^{\beta-1}$$

And considering $\alpha = \frac{\beta}{\eta^{\beta}}$

$$u(t) = \alpha \beta t^{\beta-1}$$

The parameters of the model are need to be estimated to calculate the various parameters like mean time between failures, next expected failure, etc. They are estimated using maximum

likelihood estimation (MLE). The probability density function (pdf) for i^{th} event given $(i-1)^{th}$ event occurred at T_{i-1} is:

$$f(T_i|T_{i-1}) = \frac{\beta}{\eta}\frac{T_i}{\eta}^{\beta-1} \; e^{\frac{-(T_i{}^{\beta} - T_{i-1}{}^{\beta})}{\eta^{\beta}}}$$

The likelihood function is of the form: $L = \alpha^n \beta^n e^{-\alpha T^{*\beta}} \prod_{i=1}^{n} T_i{}^{\beta}$

Where T* is termination time and given by:

T^*= {T_n if the test is failure terminated | $T>T_n$ if the test is time terminated}

Taking natural logarithm on both sides on the likelihood function:

$$\ln(L) = n\ln(\alpha) + n\ln(\beta) - \alpha T^{*\beta} + (\beta - 1)\sum_{i=1}^{n} \ln(T_i)$$

Differentiating with respect to α and equating it to zero we get $\alpha = \frac{n}{T^{*\beta}}$

Differentiating with respect to β and equating to zero we get $\beta = \frac{n}{n\ln T^* - \sum_{i=1}^{n} \ln(T_i)}$

Power law can be used to evaluate whether a system is improving or deteriorating. This can be demonstrated by the failure intensity function.

Figure 4-31- Comparing failure rates with number of components in a system

Let us take the above example for comparing failure rates for different systems. As we can see increasing the number of components in the system smoothens the failure rate behaviour. For system containing 5 components with age initially failure rate increases and after some time some components fail and hence they are replaced that leads to drop in failure rate and this process continues. With 100 components present in the system the randomness of the graph decreases and a constant failure rate is observed and expected number of failures is constant.

Power Law can also be used to observe the failure dependencies of various components present. In the figure below we can observe 3 different cases (1) 1 component is slightly dependent on other (2) 1 component is highly dependent on other (3) 2 components are dependent on each other. The failure behaviour is seen through Power law: In case of less dependency the failure rate becomes constant after a period of time, while increase in dependency increases failure rate. In case of both ways dependency failure rate increases at a rapid rate.

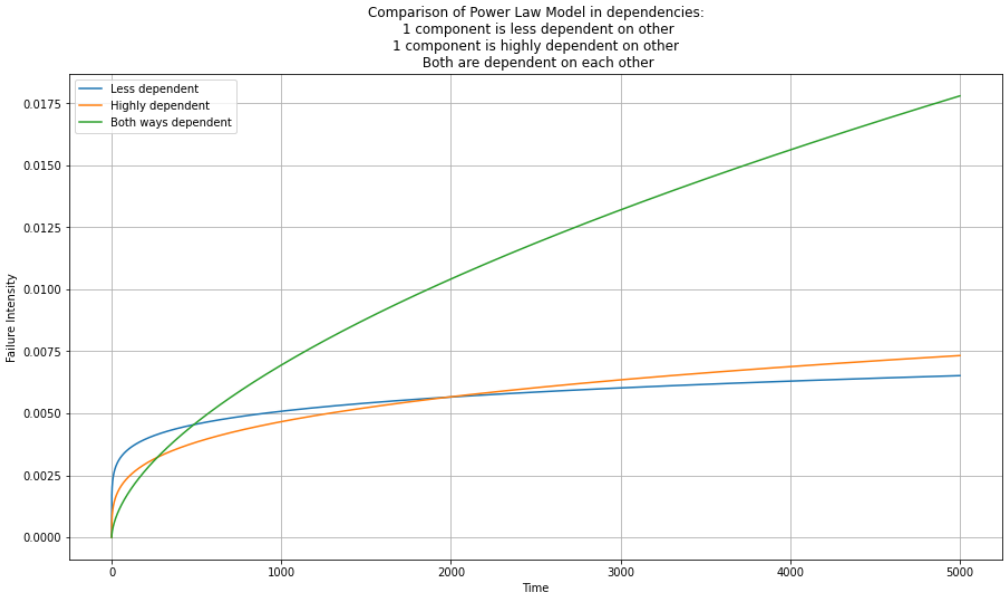

Figure 4-32-Comparing failure rates with dependencies

4.8 N-gram Model

The contiguous collection of n items from a given sample of text or speech is known as an "n-gram." Depending on the application, the objects could be letters, words, or base pairs. The N-grams are usually gathered from a corpus of text or voice (A long text dataset).

N-gram is tried to implement to predict the next failed component given that earlier a certain component has failed. It will also work to check for failure dependencies between components as in this way we can find out the maximum likelihood of failing of a component given that the previous component has already failed.

Probability distribution of components

$$\boldsymbol{p(c1\ c2\ \dots\ cn) = p(c1)p(c_2|c_1)p(c_3|c_1,c_2)\dots p(c_n|c_1,c_2,\dots,c_{n-1})}$$

N-gram language model assumes each word depends only on the last n-1 words (Markov assumption).

Unigram model is of the form: $\boldsymbol{P(c_1)P(c_2)P(c_3)\dots P(c_n)}$

Bigram model is of the form: $\boldsymbol{P(c_1)P(c_2|c_1)P(c_3|c_2)\dots P(c_n|c_{n-1})}$

N-gram model is of the form: $\boldsymbol{P(c_1)P(c_2|c_1)\dots P(c_n|c_{n-1}c_{n-2}\dots c_{n-N})}$

For training an N-gram model: $\boldsymbol{P(c_k|c_{k-1}) = \frac{Y(c_{k-1}c_k)}{Y(c_{k-1})}}$

Where, $\boldsymbol{Y(\emptyset)}$ *= Length of document or total number of components in a system*

N-gram modelling is done on a system consisting of 5 components each following Weibull distribution of different shape and scale parameters and probability of failing of 1 component with and without failure dependencies are calculated.

Dependency do not exist	C1	C2	C3	C4	C5
C1	0.09385	0.161276	0.287927	0.173121	0.272437
C2	0.219495	0.018934	0.267181	0.183029	0.300842
C3	0.257702	0.155704	0.061615	0.186095	0.330974
C4	0.2164	0.165845	0.310111	0.009864	0.288533
C5	0.279406	0.164756	0.306582	0.215711	0.022505

Dependency Exists	C1	C2	C3	C4	C5
C1	0.063348	0.111425	0.226244	0.354638	0.238688
C2	0.163007	0.007601	0.216216	0.35473	0.249155
C3	0.203382	0.124638	0.038164	0.3657	0.258937
C4	0.199308	0.138062	0.265398	0.108651	0.273702
C5	0.205939	0.148946	0.257184	0.364943	0.014847

This is an outcome of a n-gram model build with and without dependencies. Here the component C4 is a dependent component and in case of existing dependency it is seen that C4 arrives and almost every component's failure. While in case of no dependency we can can see there is a random allocation of failure depending on parameters of components.

5 Prediction of Time to Failure

5.1 Introduction

In many businesses, predicting when a system may fail is essential since it can reduce costly downtime and boost productivity. The next failure time has been predicted using a variety of methods over the years, with technological developments encouraging the creation of more complex models.

The power law model was one of the initial methods used to forecast the next failure time. This model makes the assumption that a system's failure rate is proportional to a power of time. This model has drawbacks, such as the assumption that the failure rate stays constant over time, despite the fact that it can be effective in specific circumstances.

Time series models have made it possible to estimate the next failure time with greater accuracy. The Autoregressive Integrated Moving Average (ARIMA) model is one such example. This model utilises the time series data to forecast potential failures in the future. Many industries, including manufacturing, healthcare, and finance, have effectively adopted ARIMA models.

However, as technology has developed further, recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM), have gained popularity. The ability of LSTM networks to capture long-term dependencies in the data makes them particularly helpful for estimating the time before the next failure. This is crucial because system failures may be influenced by occurrences that took place weeks or even months before. To update a concealed state, LSTM networks process incoming data incrementally over time. In order to forecast upcoming failures, this hidden state, which contains data about the prior inputs, is utilised. Manufacturing, energy, and transportation are just a few of the industries that have effectively deployed LSTM networks.

5.2 Use of Time Series Models like ARIMA:

ARIMA (Auto Regressive Integrated Moving Average) is one of the process to forecast the next time to failure in systems with multiple components. ARIMA offers a powerful framework for time series analysis, allowing us to extract valuable insights and make informed decisions regarding system maintenance and reliability. ARIMA combines autoregressive (AR), differencing (I), and moving average (MA) components. It is widely used for modelling and forecasting time-dependent data, making it applicable to the prediction of time to failure in systems with multiple components.

An observation's link to several lag observations is taken into account by the autoregressive part of ARIMA. The AR component records any innate patterns or dependencies in the failure histories of separate components in terms of time to failure. Each component's failure patterns can be examined in order to learn important details about the system as a whole. The time series data's trend or seasonality are intended to be eliminated by ARIMA's differencing step. It aids in removing any progressive variations in failure rates brought on by ageing or other influences in the case of time to failure. We can concentrate on the residual patterns that can point to impending failures by diffusing the time series. The moving average part of ARIMA takes into account the relationship between an observation and a residual error from a model that uses moving averages. It aids in the time series' ability to detect transient fluctuations and

abnormalities. The MA component can spot rapid changes or unexpected occurrences that might have an impact on the failure patterns of certain components in terms of time to failure.

5.3 Replacing ARIMA with LSTM:

Initially, predicting the time to failure in complex systems involving multiple components has traditionally relied on statistical methods like ARIMA. However, with the advancements in deep learning, specifically Long Short-Term Memory (LSTM) networks, a paradigm shift has occurred in time series analysis.

LSTM networks are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. They excel at modelling complex temporal relationships, making them an excellent choice for time series forecasting tasks. In contrast, ARIMA models are statistical models that analyse the autocorrelation and moving average components of a time series to make predictions.

There are numerous difficulties with utilising ARIMA to predict time to failure in complicated systems:

- Relationships that are not linear: ARIMA bases its predictions on linear relationships between variables, which may not be the case in complex systems. A more flexible modelling strategy is needed when there are many components interacting in complex ways.
- Managing Temporal relationships: ARIMA finds it difficult to accurately capture long-term temporal relationships. LSTM networks are particularly well suited for modelling such dependencies due to their memory cells.
- ARIMA needs manual feature engineering, which can be time-consuming and error-prone. By automatically picking up pertinent elements from the data, LSTM networks can reduce the need for user intervention.

The steps involved in using LSTM as the method to find out the time to next failure are as:

A. Pre-Processing

   This is the most important step in helping the model learn better and better. The data given by the user is in the form of time. The time may vary around a quite long range. The range needs to be reduced. To reduce that we use Minimax Scaling where the data in form of time is restricted between 0 and 1 where 0 indicates the lowest possible failure time and 1 indicates the highest possible failure time and all the values lie in between the two values. This gives the model an easy feed input to learn from as there is no much more deviation in the data. After the pre-processing is done the data is filtered in such a way that a perfect input format is supplied to the model. The inputs to the LSTM is as is classified into X and Y terms where X is an array while Y is a single element. Here we have used 80 elements in X array while Y contains a single element. The model takes those 80 inputs as input and predict the next element by using the LSTM architecture. In this process it tries to reduce to loss to the model to as low as possible.

B. Model Architecture and Training

The architecture of an LSTM network consists of input layers, LSTM layers, and output layers. The LSTM layers' memory cells retain information about past states, while the output layer predicts the time to failure for each component or the system as a whole. The architecture used for the failure prediction had 3 hidden layers, 1 input and output layer. Each layer has 50 units in it for better learning. Dropout is a regularization technique introduced by Srivastava et al. (2014) that helps prevent overfitting in neural networks. It randomly sets a fraction of the input units to zero during each training iteration, effectively "dropping out" those units. This technique encourages the network to learn more robust features and reduces the network's dependence on specific units, thus preventing over-reliance on individual neurons. Adding dropout layers in LSTM networks can enhance their generalization capabilities and reduce overfitting. So with every hidden a dropout layer was added to take advantage of the features it provides for better training and modelling. The architecture of the model finally used is as shown in the figure below for better understanding.

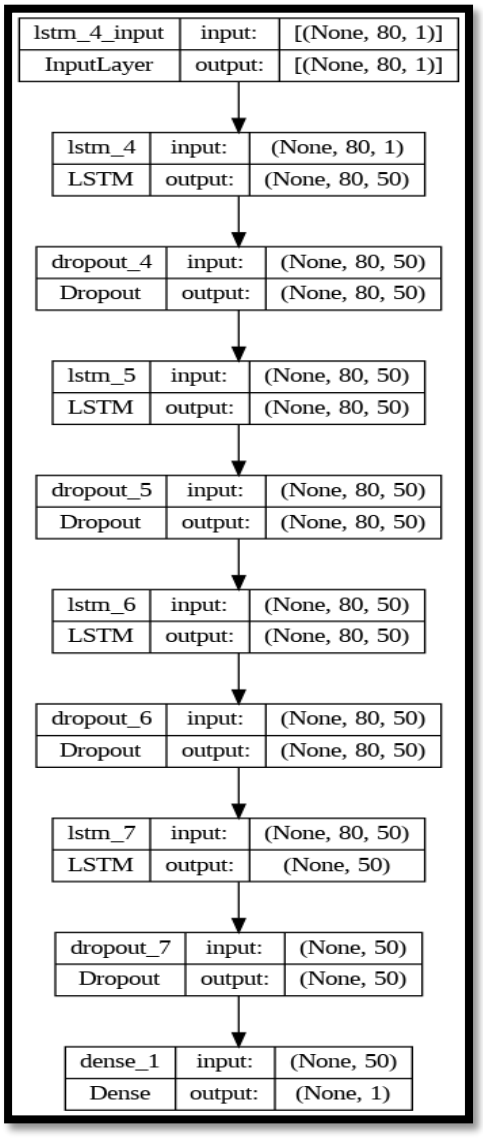

Figure 5-1Architecture for predicting TTF

C. Model Evaluation and Prediction

After training, the LSTM model's performance is evaluated using the testing dataset. The predicted time to failure is compared against the actual failure instances to assess the model's accuracy. Iterative improvements can be made by tuning the model's hyper parameters, such as the number of LSTM layers, the number of memory cells, and the learning rate, to achieve better predictions.

For testing the model we have to pre-process the test as well in the same format as the training data. We will have an array of 80 elements as inputs and the model will give the output accordingly. But as we have scaled the input between 0 and 1 for inputs to the model hence the model will give output between 0 and 1 and we have to reverse scale this to its original form.

5.4 Results:

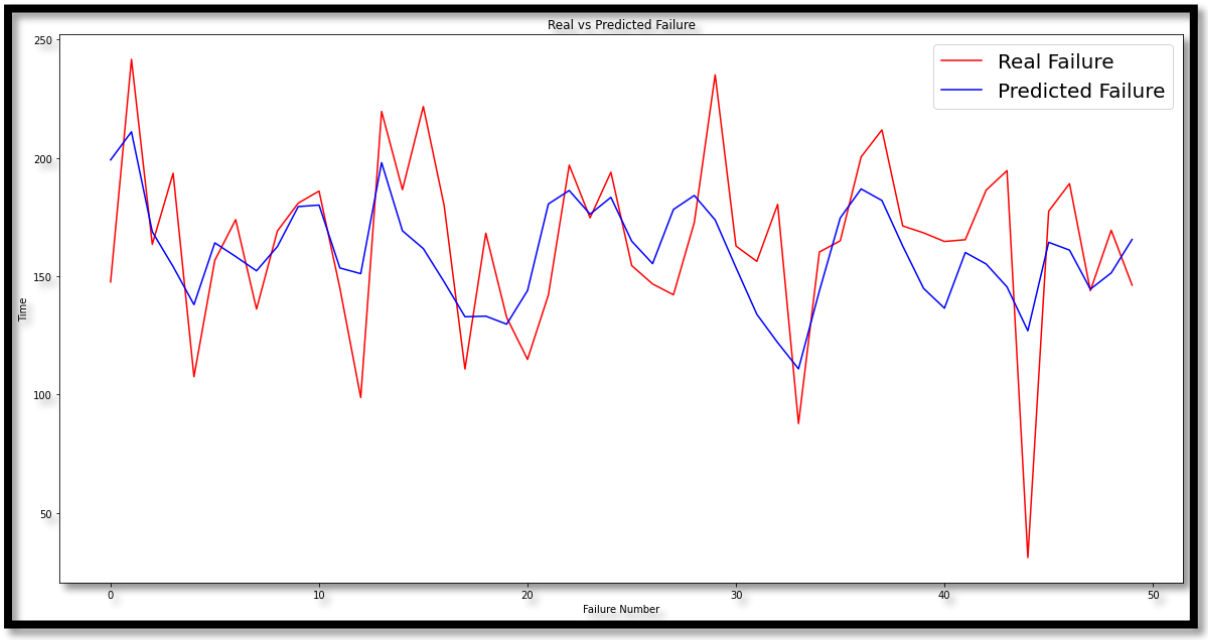

Figure 5-2Predition V/S Actual Results

The above is the result of testing the data after training using the above LSTM with a dataset containing about 1 lakh failures of the system removing the last 50 failures and testing on those failures. After learning the model seems to have almost learned the trend of failures and able to predict the future failures as well.

The results are compared with the results that ARIMA has provided and it is observed that LSTM has shown far better predictions as compared to ARIMA. The loss for both functions were root mean square error. In case of ARIMA the error touched about 35 while in case of LSTM the error was below 30. This is a good indicator that LSTM has performed better as compared to ARIMA

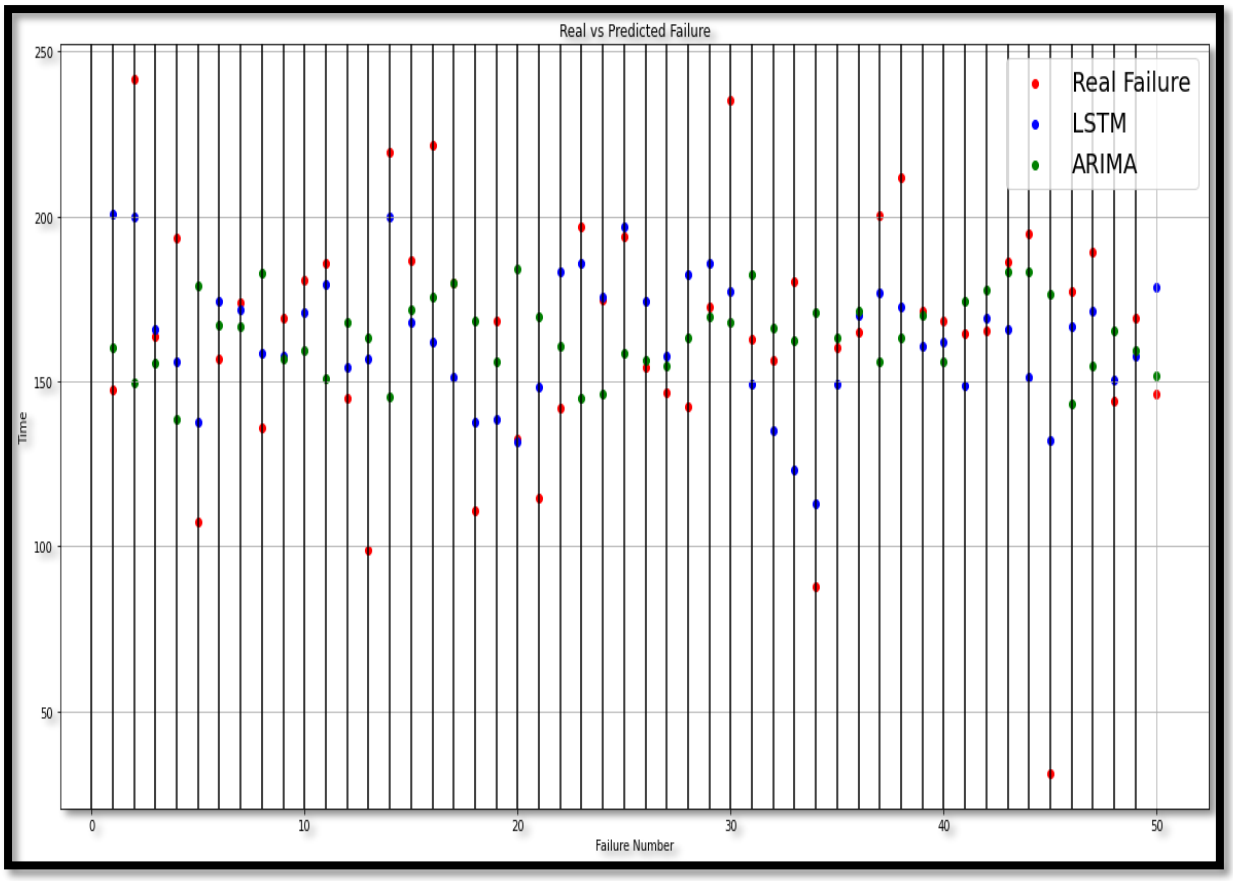

Figure 5-3Comparison between Actual , LSTM and ARIMA results

The dropout value was tuned to avoid overfitting the model as less as possible. So a range of dropout values was used. For the dataset it was found that 0.15% dropout was the ideal value for dropout as the root mean square error is the minimum in that case. There is a graph comparing the predicted values using 0.15% and 0.2%.

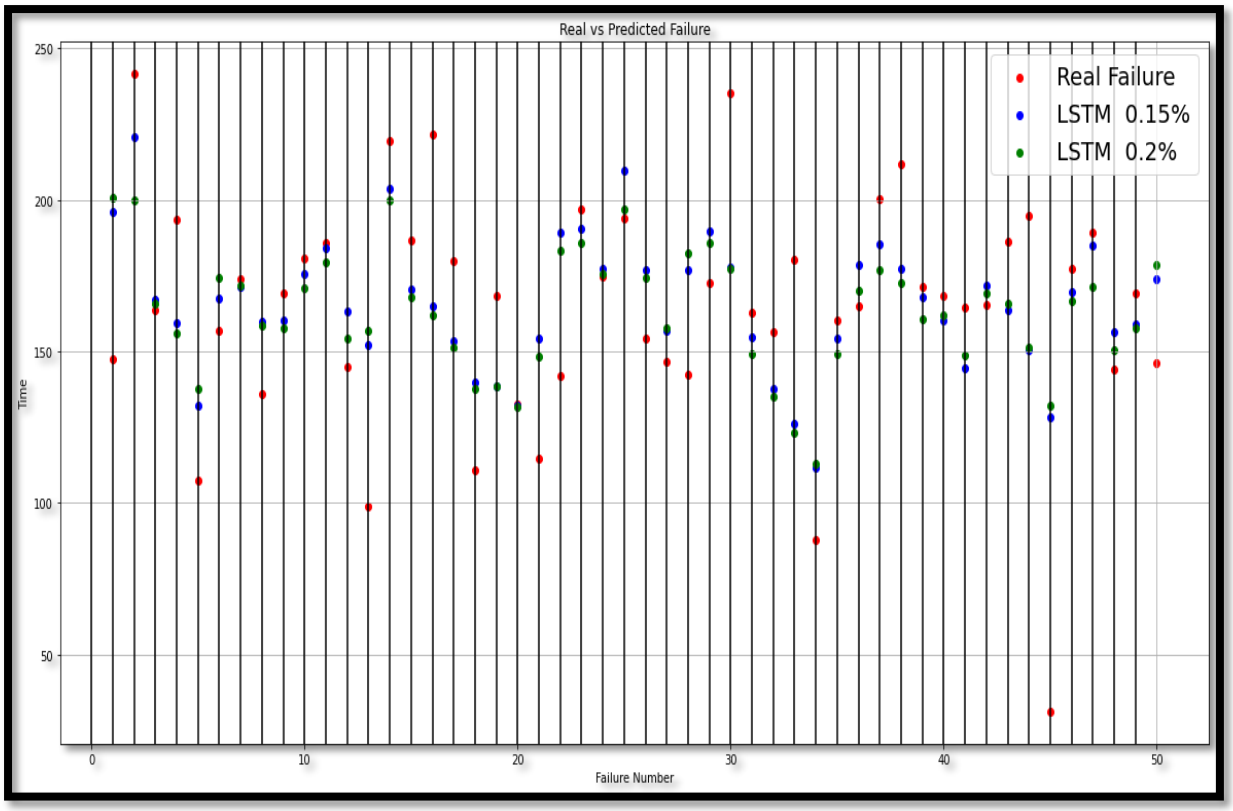

Figure 5-4 Comparison between actual and LSTM with 2 different dropoutsi

However the model carries the noise that the failure behaviour contains. As the simulator had all the independent failures in it the failure rate is almost constant and the slight deviations from that can be regarded as the noise. But this approach of predicting the time to next failure will be completely valid in all the cases even when there is dependency of multiple components on each other whether it is one-sided or both-sided. However the only constraints in prediction is that the data needs to be trained on a large dataset containing at least greater than 1000 data points which would enable the deep neural network to learn better the trends which would further help in accurate or near accurate predictions.

6 Prediction of Next Failed Component

In our interconnected and technology-driven world, multi-component systems play a vital role in numerous industries. Whether its transportation, manufacturing, healthcare, or telecommunications, these complex systems rely on the delicate balance of their components. The failure of even a single component can have far-reaching consequences. Hence, the need to accurately predict the next component failure becomes paramount. In this article, we will delve into the pressing requirement for predicting the next component failure in complex systems and the significant benefits it offers.

A. Minimizing Downtime and Service Disruptions:
   Unplanned downtime can result in substantial financial losses and disruptive impacts across various sectors. By accurately predicting the next component failure, organizations can proactively plan maintenance activities, replace or repair the faulty component before it fails, and prevent unexpected system outages. Implementing predictive maintenance strategies enables companies to avoid costly emergency repairs, reduce downtime, and ensure uninterrupted service delivery to their customers.
B. Optimizing Maintenance Operations:
   Traditional maintenance approaches in complex systems typically rely on preventive or reactive methods, such as routine inspections or addressing failures as they occur. This allows companies to allocate resources effectively, focusing on critical components and performing targeted interventions at the most opportune time.
C. Cost Reduction and Resource Optimization:
   Predicting the next component failure empowers organizations to optimize resource allocation, resulting in significant cost reduction. By identifying components that are likely to fail, companies can strategically plan and procure replacement parts, thereby minimizing excess inventory and storage costs. Moreover, targeted maintenance actions prevent unnecessary servicing of components in good condition, avoiding unnecessary expenses. The ability to allocate resources efficiently enables organizations to achieve considerable cost savings and enhance their overall financial performance.
D. Enhancing System Reliability and Safety:
   The failure of a single component in a complex system can trigger a domino effect, leading to system-wide malfunctions or even catastrophic failures. Predicting the next component failure allows organizations to take proactive measures, such as implementing redundancy, redesigning components, or enhancing monitoring systems, to mitigate risks and improve system reliability. This aspect holds utmost importance in industries like aerospace, healthcare, and energy, where component failure can endanger lives or result in severe environmental consequences.

6.1 Previous Approaches and its Disadvantages:

Initially we have used n-gram in order to predict the next component that is going to be failed in the upcoming time. But n-gram has some limitations which led to the use of RNN in the further work.

RNNs are capable of capturing long-term dependencies in sequential data. They have a recurrent structure that allows them to retain information from previous time steps, which is essential for understanding the context and making accurate predictions. N-gram models, on the other hand, consider only a fixed number of previous words, typically limited to a few preceding tokens, and cannot capture long-range dependencies effectively.

RNNs have a parameter sharing mechanism that allows them to efficiently model sequences of arbitrary lengths. Instead of learning separate parameters for each possible n-gram, RNNs learn a set of shared weights that are applied at every time step. This parameter sharing greatly reduces the number of parameters compared to n-gram models, making RNNs more efficient in terms of memory and computational requirements.

6.2 Use of RNN to predict next component:

Language modelling refers to the development of models capable of predicting the next word in a sequence of words based on prior context. RNNs, a type of neural network, excel at capturing sequential information and dependencies over time. By training an RNN on a large dataset comprising component failure patterns and historical data, it may become possible to leverage the network's ability to learn complex patterns and make accurate predictions.

The process involved while using RNN is somewhat similar as compared to n-gram for the same all independent components simulator. The dataset to the RNN model is the list of all the components that have failed in order. Al the components list have been converted into a single sentence such that the pre-processing can be done in well manner. The three steps involved in this process are:

A. Data Pre-processing
    a. The corpus of components is converted to lowercase to have uniformity and split to generate a list of tokens.
    b. A dictionary is created to map each unique component in the tokens into a numerical index.
    c. Each token in the tokens list is replaced with its corresponding numerical value derived after mapping. This token is now used for training the model.
    d. A sequence length is decided. The sequence length captures the past information that needs to be captured before the next component prediction. The sequence length is decided to be 100. So the component learns about the last 100 components after deciding the next component. This is done in order to ensure that maximum information is captured by the model and during prediction it uses the maximum amount of information before.
B. Model Building:
    a. The data had 2 components one an array of 100 elements as input and other the next element (single) as output.
    b. The size of the vocabulary used is the number of unique components used for the model.
    c. The model architecture contains 1 input layer, 4 hidden layers and 1 single output layer.
    d. An embedding layer is added to the input layer and whose input was the number of components in the system while the output dimensions was set to 128.

e. 2 LSTM layers are added to the model so that each can capture all short term and some long term dependencies.
f. 1 dense layer is attached that to the output dense layer. The former dense layer has **ReLU** activation function while the later dense layer which behaves as the output layer has **softmax** activation function.
g. The loss parameter is set as **sparse_categorical_crossentropy** indicating that the model will be trained to predict a single integer label for each input sequence. The 'optimizer' parameter is set to '**adam**', which is a popular optimization algorithm. The 'metrics' parameter is set to ['**accuracy**'], specifying that the accuracy metric will be used to evaluate the model during training.

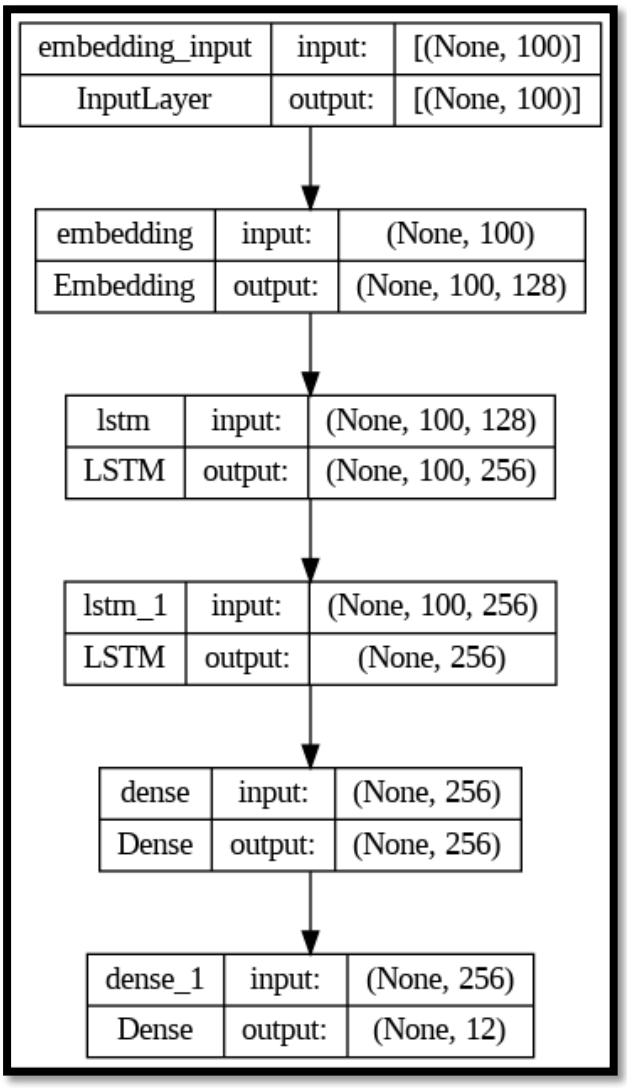

Figure 6-1 Architecture to predict next component

C. Model Testing and Results:
The model is thoroughly validated by tuning several hyper parameters like number of layers and number of units in each layer. The prediction results of the LSTM layer were quite poor for the data generated by the independent components simulator. We

generated around 1 lakh data points so we had a 1 lakh word sentence. Except the last 50 components rest 99950 components were fed into the RNN model to train. And the last 50 components were kept for testing.

The model has predicted correctly only 9 times out of 50 which component is going to fail. Fore rest of the times it almost predicted the components which have failed the maximum in the course of simulation.

After completing this model building task we have realised at least for independent components the use of RNN to predict the next failed component is not a good approach. Because it just captures the sequence in which the component fails. It doesn't contain any information regarding the current state the component is in, number of hours the component has already worked and the current condition in which the component is working in. Hence using RNN seems to be failure for predicting next component. So we have explored an alternative approach.

6.3 Implementing a Classification Algorithm

After completing the modelling using RNN to predict the next component that is going to be failed it was completed that the sequence in which components fail won't help a model learn the characteristics that make the component fail. So a different approach was tried where the history of the components are stored. After learning through multiple literatures it was inferred that a classification tool can be implemented where the input of the classification is the current age of various components while the output of the algorithm is the component. This classification algorithm takes the age of the component and decides which component is going to fail.

6.3.1 Data and its pre-processing:

The data generated from the simulator initially had the information of the system itself where the time of failure of the system and the component responsible for the failure of the system was extracted. But this data was not enough for this classification as we must have the age of different components at the time of failure of the system. So altering the data extraction process from the simulator we had the extracted the data in such a form that the age of different components is there at the time of failure of the system.

	C11	C12	C13	C21	C22	C31	C32	C33	C34	C35	C36	C41	C42	C43	C44	C45	C51	C52	C53	C54
1	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	258.7751	0	258.7751	258.7751	258.7751	258.7751
2	388.3201	388.3201	388.3201	388.3201	388.3201	388.3201	388.3201	388.3201	388.3201	0	388.3201	388.3201	388.3201	388.3201	388.3201	129.545	388.3201	388.3201	388.3201	388.3201
3	567.9674	567.9674	567.9674	567.9674	567.9674	567.9674	567.9674	567.9674	567.9674	179.6473	567.9674	567.9674	567.9674	567.9674	567.9674	0	567.9674	567.9674	567.9674	567.9674
4	721.1376	721.1376	721.1376	721.1376	721.1376	721.1376	721.1376	721.1376	721.1376	332.8175	0	721.1376	721.1376	721.1376	721.1376	153.1702	721.1376	721.1376	721.1376	721.1376
5	826.4614	826.4614	826.4614	826.4614	826.4614	826.4614	826.4614	826.4614	826.4614	0	105.3237	826.4614	826.4614	826.4614	826.4614	258.494	826.4614	826.4614	826.4614	826.4614
6	937.6431	937.6431	937.6431	937.6431	937.6431	937.6431	0	937.6431	937.6431	111.1817	216.5055	937.6431	937.6431	937.6431	937.6431	369.6757	937.6431	937.6431	937.6431	937.6431
7	1056.818	1056.818	1056.818	1056.818	1056.818	1056.818	119.1752	1056.818	1056.818	230.3569	335.6807	1056.818	1056.818	1056.818	1056.818	0	1056.818	1056.818	1056.818	1056.818
8	1194.283	1194.283	1194.283	1194.283	1194.283	1194.283	256.64	1194.283	1194.283	0	473.1454	1194.283	1194.283	1194.283	1194.283	137.4647	1194.283	1194.283	1194.283	1194.283

Figure 6-2 Data Format

Above is the representation of how the data was stored. At the 1st failure of the system C45 had failed and hence replaced. So the age of C45 was set to 0 while other attained the age that the

system has attained. Similarly at the 2nd failure C35 has failed after 129.5 hours of 1st failure. So this time is added to all the components except C35. This way the entire data was extracted for continuing into the classification task.

But this data was not enough to feed into a classification model. The data needed for a classification model was the input in the form of arrays containing the age of different components at a given stage and output to be a single index indicating the index of component that is going to fail. To generate data in a form like this once again a small **simulation exercise** is done using the extracted data. For every 20 time steps the current state of the component is checked from the extracted. If it is not replaced then 20 is added to current age and the output will have the index of the component that is going to fail next.

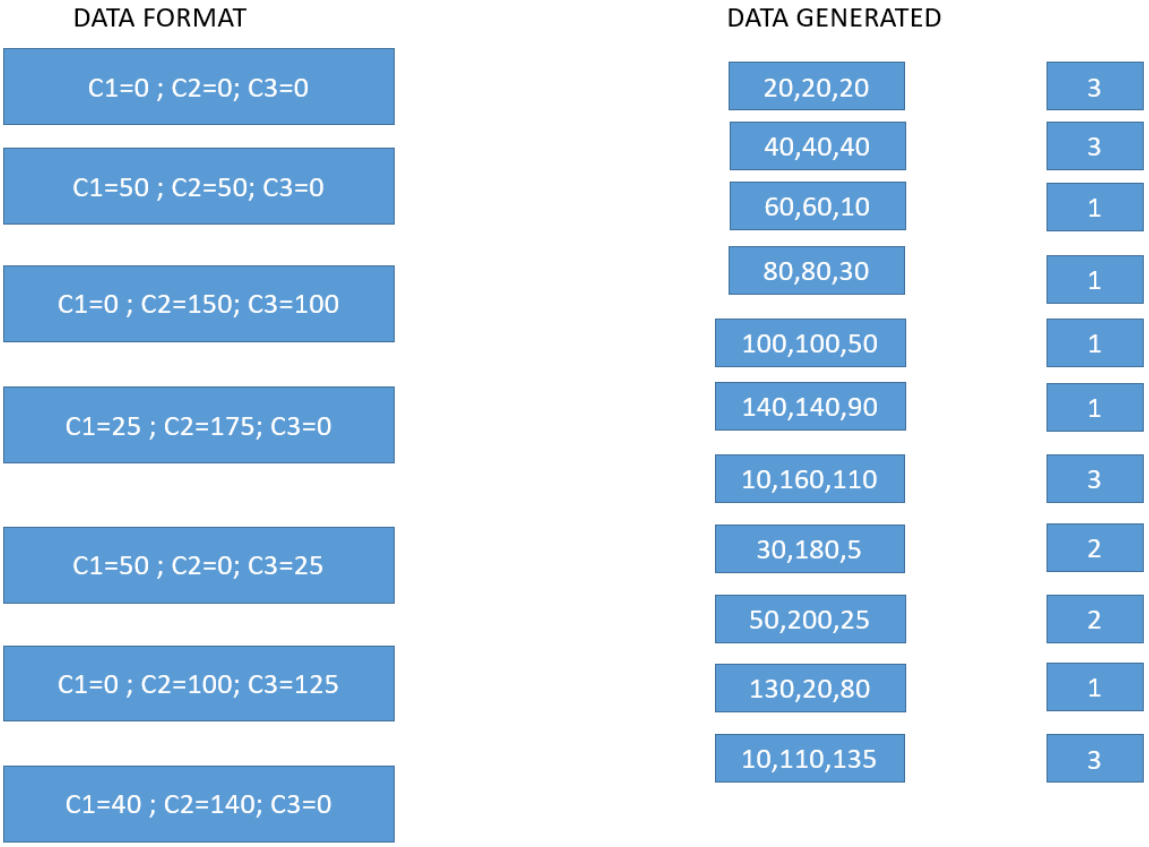

Figure 6-3 Example of creating the data from extracted data

The above is a sample how the data is generated. As for the example format during the 1st failure C3 had failed so the output of classification would be index of C3 which is 3 here. And for the input an array of age is generated until C3 fails. Here as C3 fails at 50hrs, until 40 hrs input will be the multiples of 20 while output will be 3. After 50 the nearest multiple of 20 is 60. During that time the age of C3 is 10 and the next incoming failure is C1. Hence the output will have 1 as the index of C1 is 1.

6.3.2 Building the model:

Building a classification model using deep learning involves several essential steps. First, collect a labelled dataset that contains input features and corresponding target labels for training and evaluation. This dataset was generated from our new simulation process. Next, pre-process

the data by cleaning it and handling missing values, normalizing features, and performing any necessary transformations. As the data is generated there is no need of pre-processing as the data is generated in such form.

The model architecture is designed by selecting the appropriate neural network type and determining the number and configuration of layers. The model architecture is defined using the Sequential API from the Keras library. The model consists of several dense layers connected sequentially. The first layer has 64 units with a ReLU activation function and applies L2 regularization with a regularization strength of 0.001. It takes input with a shape of (20,), indicating that the input data has 20 features. Here the 20 features are the **20 components** present in the system. The subsequent layers also have 64 and 32 units, respectively, with ReLU activation functions and L2 regularization. The final layer has 20 units and uses the **softmax** activation function, which is suitable for multiclass classification tasks.

The model is compiled by specifying the optimizer, loss function, and evaluation metrics. The chosen loss function is '**sparse_categorical_crossentropy**', which is commonly used for multiclass classification problems with integer-encoded labels. The optimizer used is '**adam**', a popular optimization algorithm that adjusts the model's parameters during training. Additionally, the 'accuracy' metric is specified to evaluate the model's performance during training and evaluation. The model was trained feeding the pre-processed data, allowing it to learn patterns and make predictions.

6.3.3 Model Evaluation and Results

The model is thoroughly validated by tuning several hyper parameters like number of layers and number of units in each layer. But even then the results were quite poor for the above modelling procedure. The classification model also gave similar results as the language modelling gave. This was because the same output was fed for the component that has failed the most. Hence the model tries learning that there is a higher probability of that component failing with whatever the current state of other components have.

Hence both the approaches have failed to predict the next component which is going to fail given the past state and current state of different components at least for a system containing components which are independent of each other. There might be some method to find that out but it wasn’t been able to figure out until then. N-grams were suitable to find out dependencies but even that wasn’t been able to find out the next failed component.

7 Creating an Interface

7.1 Need of an interface for failure simulations:

One compelling reason to use an interface instead of code for failure simulations is to enhance flexibility and accessibility. Interfaces provide a user-friendly and intuitive means of interacting with complex systems, allowing users with limited programming knowledge or expertise to conduct failure simulations effectively.

By utilizing an interface, individuals who are not proficient in coding can still navigate and utilize the simulation software. This inclusivity promotes collaboration among multidisciplinary teams, where domain experts, engineers, and stakeholders can all contribute to the simulation process. They can easily adjust parameters, input various scenarios, and analyse the simulation results without the need for extensive coding skills.

Moreover, interfaces offer a visual representation of the simulation, enabling users to easily interpret and understand the outcomes. This visual feedback enhances comprehension, allowing users to identify failure points, analyse the impact of different variables, and make informed decisions based on the simulation results.

Another advantage of using an interface is the potential for faster iteration and experimentation. With a user-friendly interface, users can quickly modify input parameters, run simulations, and observe the outcomes in real-time. This iterative process enables rapid testing and refinement of failure scenarios, facilitating a more efficient and accurate simulation workflow.

Interfaces can also provide a higher level of abstraction, shielding users from the complexities of the underlying code. This abstraction enables users to focus on the problem at hand rather than the technical details of the simulation implementation. Consequently, users can dedicate more time and energy to analysing the failure scenarios and devising strategies to mitigate risks and improve system resilience.

Lastly, interfaces can incorporate interactive features, such as data visualization, interactive controls, and customizable dashboards. These features enable users to explore and manipulate simulation data in a more engaging and interactive manner. It facilitates the identification of patterns, trends, and correlations within the simulated failures, leading to deeper insights and more informed decision-making.

In summary, using an interface for failure simulations promotes flexibility, accessibility, collaboration, faster iteration, abstraction, and interactive data exploration. By removing barriers to entry and simplifying the simulation process, interfaces empower a broader range of users to engage in simulations effectively and efficiently, ultimately leading to better risk assessment, improved system design, and enhanced decision-making.

7.2 General Procedure

Web frameworks are essential tools for creating user interfaces (UI) using Python. They provide a structured and efficient way to build web applications by handling various tasks, such as routing, request handling, and template rendering. Here are some key reasons why web frameworks are beneficial for UI development in Python:

- Simplified Development: Web frameworks provide a set of pre-built components and libraries that simplify the development process. They offer a structured approach to

designing and organizing your UI code, making it easier to create and maintain complex applications.

- Routing and URL Handling: Web frameworks handle routing, which determines how different URLs or paths are mapped to specific functions or views. This allows you to define URL patterns and associate them with appropriate UI components, such as HTML templates or Python functions.
- Request Handling: Web frameworks handle incoming HTTP requests and provide tools for parsing request data, such as form inputs or JSON payloads. This simplifies the process of extracting data from user input and incorporating it into your UI.
- Template Rendering: Many web frameworks include template engines that enable you to separate your UI logic from the backend code. Templates allow you to define the structure and appearance of your UI using HTML or other markup languages, and dynamically insert data from your Python code.
- Security Features: Web frameworks often include security features, such as request validation, protection against common vulnerabilities (e.g., cross-site scripting or SQL injection), and user authentication mechanisms. These features help ensure the security and integrity of your UI and protect against common web threats.
- Database Integration: Web frameworks often provide tools for integrating with databases, allowing you to store and retrieve data for your UI. This makes it easier to create dynamic UIs that interact with a backend database.
- Community Support and Ecosystem: Popular web frameworks like Django, Flask, and Pyramid have large and active communities. This means you can find extensive documentation, tutorials, and community support to help you overcome challenges and find solutions when building your UI.

Some widely used Python web frameworks for UI development include:

- Django: A high-level, batteries-included framework that provides a robust set of tools and features for building complex web applications.
- Flask: A lightweight framework that focuses on simplicity and flexibility, making it suitable for small to medium-sized projects or prototyping.
- Pyramid: A flexible and scalable framework that emphasizes modularity and allows developers to choose the components they need for their specific project.
- Bottle: A minimalist framework with a small footprint that is ideal for building simple applications or APIs.
- Tornado: A powerful, asynchronous framework that excels in handling high-performance, real-time web applications.

These frameworks offer different features and levels of abstraction, so you can choose one that best matches your project requirements and development preferences.

Among the web frameworks present I chose **Django** framework in order to build my application. Django is a powerful web framework that offers several advantages over other frameworks. Here are some reasons why Django was a good choice:

- Batteries included: Django follows the principle of "batteries included," meaning it provides a comprehensive set of tools and libraries out of the box. It includes functionalities for handling routing, database interactions, form handling, authentication, and more. This reduces the need for additional third-party libraries and simplifies the development process.

- High-level abstractions: Django provides high-level abstractions that enable developers to build complex web applications with ease. It follows the Model-View-Controller (MVC) architectural pattern, where models represent the data structure, views handle user requests and responses, and templates manage the presentation layer. This separation of concerns makes it easier to maintain and test code.

- Scalability: Django is designed to handle high-traffic websites and scale as needed. It includes built-in caching mechanisms, support for database replication, and tools for load balancing. Django's scalability has been proven by its usage in large-scale applications such as Instagram and Pinterest.

- Security: Django takes security seriously and incorporates numerous features to help developers build secure web applications. It includes protection against common web vulnerabilities like cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection. Django also provides secure password hashing, user authentication, and authorization mechanisms.

- Community and ecosystem: Django has a large and active community of developers, which means there are plenty of resources and support available. The official documentation is comprehensive and well-maintained. Additionally, Django has a rich ecosystem of third-party packages that can extend its functionality further.

- Rapid development: Django emphasizes rapid development by following the DRY (Don't Repeat Yourself) principle. Its built-in admin interface allows for quick creation of an administrative interface for managing data. Django also provides a robust ORM (Object-Relational Mapping) layer, which simplifies database interactions and reduces the amount of boilerplate code.

- Versatility: Django can be used for a wide range of web development projects. It supports different databases, including PostgreSQL, MySQL, SQLite, and Oracle. It also offers flexibility in terms of template engines, allowing developers to use popular choices like Django's built-in templating language, Jinja, or others.

Creating a user interface (UI) for failure simulations using Django involves several steps. Django is a high-level Python web framework that simplifies the process of building web applications. Here's an outline of the process:

A. Set up the Django project:
    a. Install Django using pip or your preferred package manager.
    b. Use the Django-admin command to create a new Django project.

    c. Use the Django-admin command to create a new Django project.

B. Design the UI:
    a. Determine the requirements and goals of your failure simulation UI.
    b. Sketch out the layout and components of the UI, considering usability and user experience.
    c. Choose a design framework like Bootstrap to assist with styling and responsiveness.

C. Define the Django models:
    a. Identify the data models necessary for your failure simulations.
    b. Define Django models using Python classes that represent the entities and relationships in your application.
    c. Add fields and relationships to the models based on the data you need to store and manipulate.

D. Create Django views:
    a. Write Django view functions or class-based views that handle the logic for different UI pages or actions.
    b. Map URLs to the appropriate views using Django's URL routing mechanism.
    c. Retrieve data from the models or other sources, process it, and pass it to the templates for rendering.

E. Implement Django templates:
    a. Create HTML templates that define the structure and content of each UI page.
    b. Use Django's template language to dynamically insert data and generate HTML based on the view's context.
    c. Incorporate CSS and JavaScript to enhance the UI and make it interactive as needed.

F. Add forms and validation:
    a. Create Django forms to handle user input and data submission.
    b. Add form validation logic to ensure data integrity and validate user inputs.
    c. Render the forms in the templates and handle form submissions in the appropriate views.

G. Include static files and media:
    a. Organize and serve static files like CSS, JavaScript, and images using Django's static files handling.
    b. Configure Django to handle media uploads and serve user-generated content if needed.

H. Test and debug:
    a. Perform thorough testing of the UI to ensure it functions as expected.
    b. Debug any issues that arise during testing, using Django's built-in debugging tools or logging.

I. Secure the UI:
    a. Implement appropriate security measures, such as authentication and authorization, to protect sensitive functionality and data.
    b. Follow best practices for securing Django applications, including input validation and protection against common vulnerabilities.

J. Deploy the application:

a. Choose a deployment strategy, such as deploying to a cloud platform or setting up a dedicated server.
b. Configure the production environment, including web server settings, database connections, and any necessary dependencies.
c. Deploy the Django application and ensure it is accessible to users.

7.3 Interface Developed:

7.3.1 Home Page

The process followed to create a user interface (UI) homepage was by using Bootstrap. The objective was to design an appealing and responsive homepage featuring a navigation bar with four sections: Home, About, Models, and Contact Us. The report outlines the steps taken to achieve these goals.

A. Bootstrap Setup
   a. Bootstrap, a popular framework for building responsive web interfaces, was chosen for this project which was included through a HTML file.
B. HTML Structure
   a. The HTML structure for the homepage was created to establish the foundation.
   b. The navigation bar was defined using the **<nav>** tag.
   c. Within the **<nav>**, an unordered list (**<ul>**) was created, and list items (**<li>**) were added for each section of the navbar.
   d. Each list item was assigned a class or ID for styling purposes.
C. Styling and Customization:
   a. Bootstrap classes and custom CSS styles were applied to enhance the visual appeal of the homepage.
   b. Bootstrap classes like **navbar**, **navbar-expand-lg**, and **navbar-light** were used to define the appearance and behaviour of the navigation bar.
   c. The search bar was customized using the Bootstrap class **form-control** for the input field and a button class for the search button.
   d. The button labelled "Start Creating System and Simulate" was given a class to match the primary colour theme.

Additionally, a button labelled "Start Creating System and Simulate" was included in the homepage. This button serves as a gateway to the System Creating Homepage specifically designed for conducting failure simulations. When users click on this button, they are redirected to a dedicated page where they can create a system with components inside the system as many as can. By seamlessly integrating this button into the homepage, users can effortlessly transition from exploring general information to actively creating the system and start the failure simulation and system analysis.

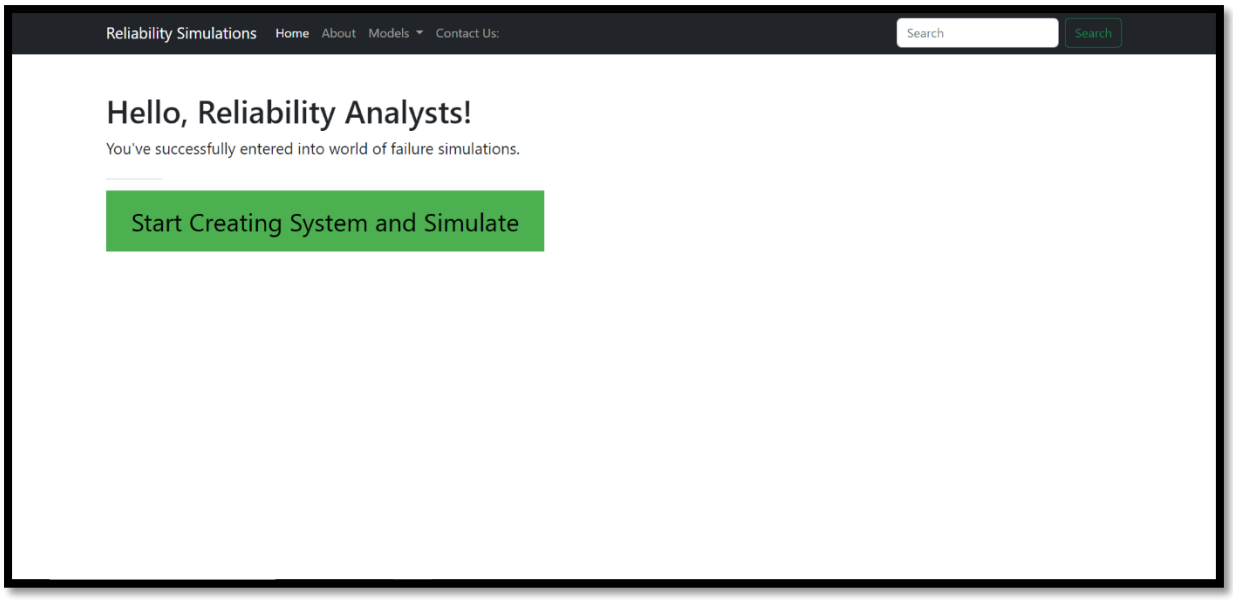

Figure 7-1 HomePage of the UI

7.3.2 Create your System:

The system creation webpage is the most critical stage in the failure simulation process as we are going to add components to the system one by one. Each of the individual components in the system follow any kind of statistical or reliability based models which subsequently have some parameters given as input by the user for the specific component. In this process we add each of the components to the system by selecting suitable models for the component and add them one by one to create the system.

A dynamic dropdown component using HTML and JavaScript was developed for adding the components. The dropdown encompasses a comprehensive range of statistical models that can be utilized in various analytical scenarios. The key innovation lies in the ability to create a system consisting of multiple components, each representing a statistical model.

The process begins with the dynamic dropdown, which allows users to select from a wide array of statistical models. Upon selecting a specific model, a set of inputs automatically appears, tailored to the selected model's characteristics and requirements. These inputs capture essential information such as the component's name and the parameters associated with the chosen statistical model.

One of the most remarkable aspects of this work is its versatility. It is ensured that the statistical models can be replicated multiple times, enabling the creation of complex analytical setups. Each system comprises several components, enabling users to design sophisticated statistical models by combining various elements. This flexibility empowers analysts and researchers to create tailored solutions specific to their needs.

The brilliance of this work lies in its attention to detail and the seamless integration of statistical models into a user-friendly interface. By automating the process of generating inputs based on the selected statistical model, users are relieved of the burden of manually configuring each component. This not only saves time but also minimizes the potential for errors.

Furthermore, it has been taken into account the nuances of different statistical distributions. The inputs that appear after selecting a model are intelligently determined based on the distribution present in the statistical model. The number and types of parameters required for each model are dynamically adjusted, ensuring an intuitive and efficient user experience.

Information Retrieved From User:

The models each component can possess are generally classified into 3 broad categories based on the criteria in which their failure will occur.

- Time as failure criterion (Weibull Model)
- Shocks as failure criterion (Shock Based Models)
- Degradation as failure criterion (Degradation Based Models)

The key element of this work lies in the creation of individual models as classes within the **models.py** file of a Django project. Each model represents a distinct statistical distribution used for failure simulations. By structuring the models as classes, the object-oriented capabilities of Python are leveraged to encapsulate the characteristics and behaviour of each distribution.

The use of classes for modelling statistical distributions allows for a highly modular and extensible approach. Each class in **models.py** contains a unique set of variables that represent the parameters of the respective distribution. This design choice not only provides a clear representation of the statistical properties but also enables easy modification and addition of new distributions as needed.

One of the remarkable aspects of this work is the utilization of Django's inbuilt SQLite3 database for storing the values associated with the statistical models. By leveraging this feature, it is ensured efficient data storage and retrieval within the Django framework is done. This integration enables seamless interaction between the statistical models and the underlying database, providing a solid foundation for further analysis and simulation.

The dropdown is as follows. A total of 9 models been used to create any kind of system with each model representing a component. There is option of creating more models which is offered easily by Django by creating multiple classes in models.py. The models that are present are as follows in form of dropdown as shown through the interface.

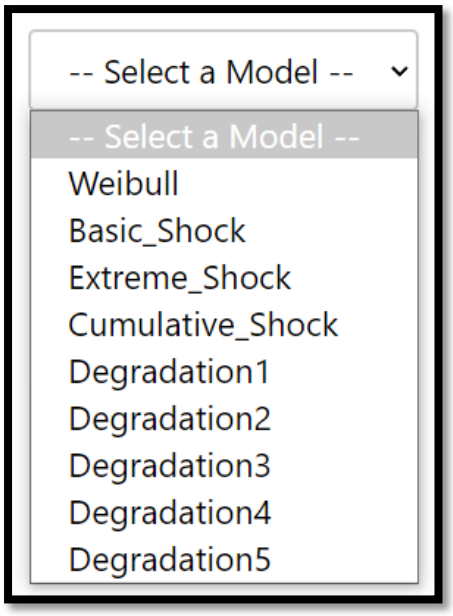

Figure 7-2 Models Present

As seen above there is a single time to failure model, 3 shock based models and 5 degradation based models present inside our simulator so that each component can be of any of the following models. It is quite easy to add more models to this based on the user's requirement. For creating a model there are barely 2-3 steps that are needed to be followed. The 1st step involves creating a database for a model in models.py file. The 2nd step involves creating a function depicting the way the failure criterion the model follows. The 3rd step involves adding the model in the dropdown list with all functionalities involving the parameters inputs.

The 1st model present in the dropdown list is the standard Weibull Model. While selecting this option you will have 3 more inputs to give. The inputs are: the name of the component, Eta and Beta which are parameters. The display from the page is shown below.

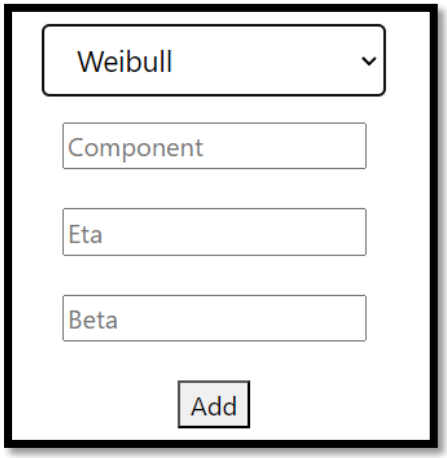

Figure 7-3 Options when you select Weibull

The 2nd model present in the dropdown list is the Basic Shock Model. While selecting this option you will have 3 more inputs to give. The inputs are: the name of the component, expected arrival of shock and threshold shocks. The display from the page is shown below.

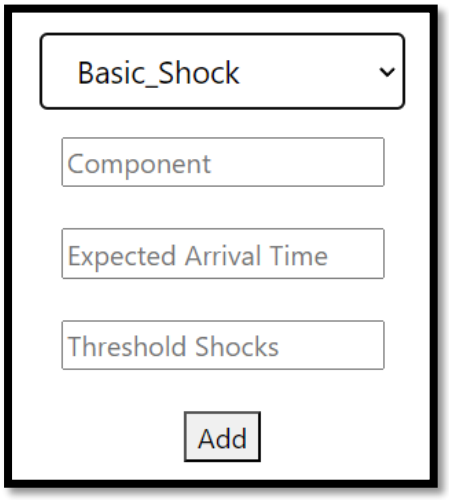

Figure 7-4 Basic Shock Attributes

The 3rd model present in the dropdown list is the Extreme Shock Model. While selecting this option you will have 6 more inputs to give. The inputs are: the name of the component, expected arrival of shock, threshold shocks, mean and standard deviation of magnitude and threshold magnitude for failure. The magnitude of the shock is assumed to have been derived from a normal distribution. The display from the page is shown below.

The 4th model present in the dropdown list is the Cumulative Shock Model. While selecting this option you will have 6 more inputs to give. The inputs are: the name of the component, expected arrival of shock, threshold shocks, mean and standard deviation of magnitude and threshold magnitude for failure. Here the threshold magnitude is the cumulative magnitude. The magnitude of the shock is assumed to have been derived from a normal distribution. The display from the page is shown below.

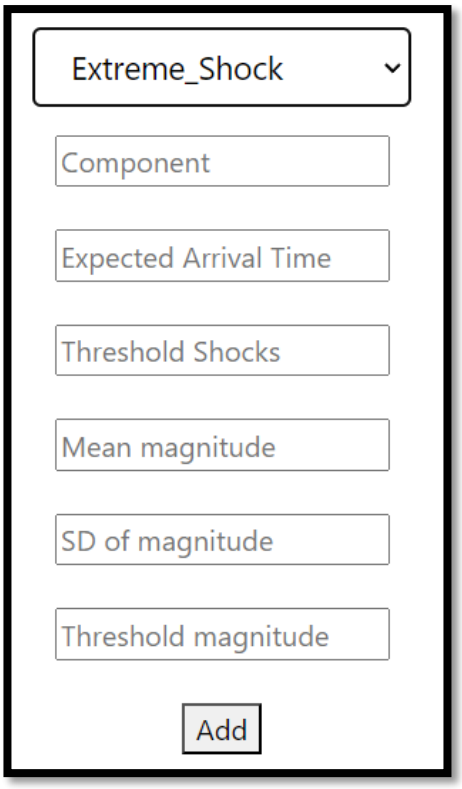

Figure 7-5 Extreme Shock Attributes

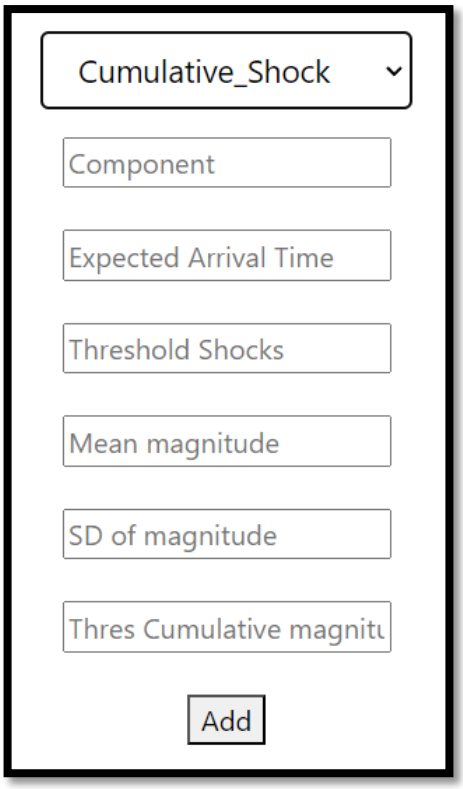

Figure 7-6 Cumulative Shock Attributes

The models ranging from Degradation -1 to Degradation -5 have no inbuilt name as they follow a degradation mechanism where the component degrades with respect to time following a distribution and a threshold degradation is permitted after which the component gets failed. Each of them follows a certain distribution and the parameters depend on the distribution that it follows. The **initial degradation** value follows a **uniform distribution** by picking a random value from ranging between minimum and maximum values given by the user

For each of the degradation models the bare minimum inputs are: name of the component, the minimum initial value, maximum initial value for degrade, threshold and distribution for the parameter. To let the user know about what distribution the degradation model is following, a representation of the equation along with a possible graph for a random parameter value is depicted below to get the user a clear idea. An example of the same is demonstrated below showing the inputs that are being asked after selecting Degradation -1 as the model for the component.

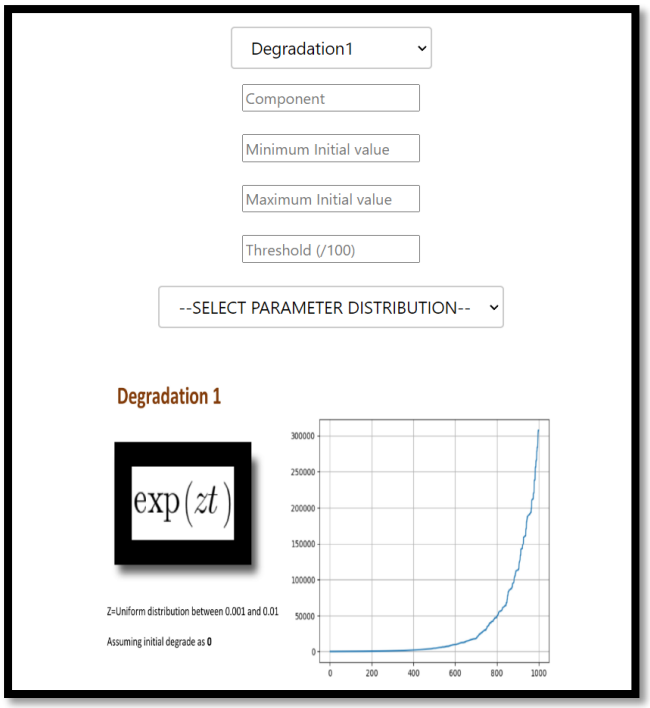

Figure 7-7 Degradation model attributes

The parameters that drive the distribution are random number generated from a distribution it gets added up with each time unit being passed. The 2 distributions out of which the parameter values can be picked up here are Normal Distribution and Uniform Distribution. When selected any one of them the values required comes up as inputs. For normal distribution the mean and standard deviation is asked to user while for uniform distribution minimum and maximum values are asked.

Degradation- 1 follows the path of **exp (zt)** where z is the parameter and t is the time which increases by 1 unit every time. As there is a single parameter the values will be asked accordingly. It is an increasing degradation distribution.

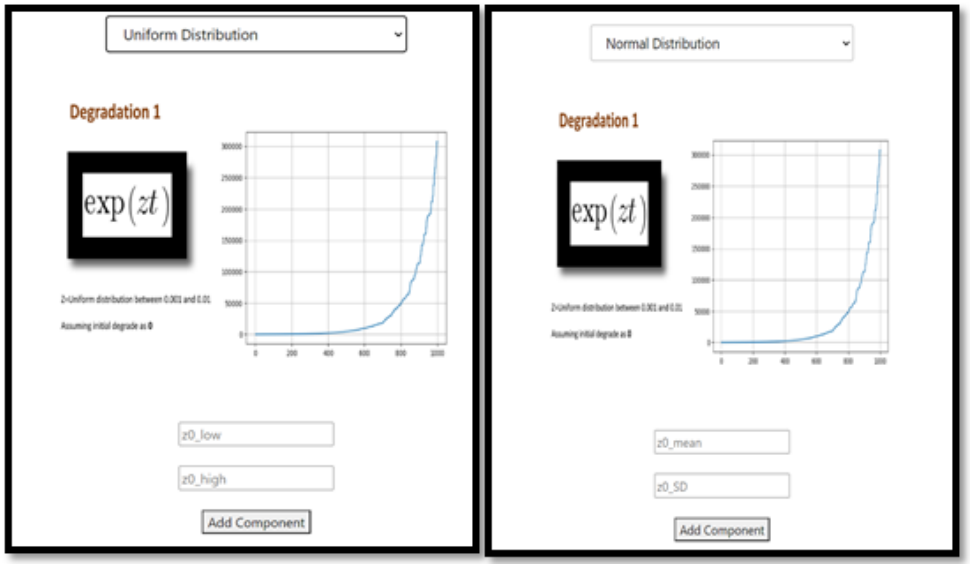

Figure 7-8 Degradation1 with Uniform And Normal Dist Parameters

The behaviour of multiple degradation paths for the same component having Degradation1 as the model to their failure have their failure patterns as follows:

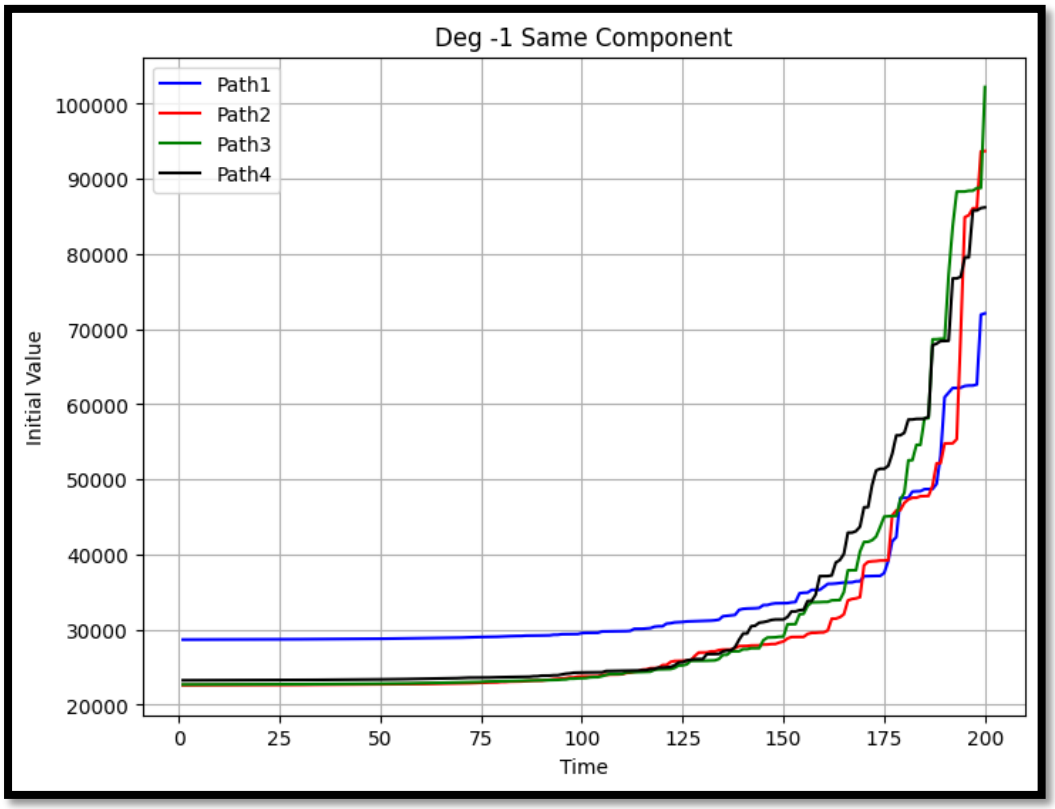

Figure 7-9 D1 Component multiple failure paths

Degradation- 2 follows the path of **log (1+zt)** where z is the parameter and t is the time which increases by 1 unit every time. As there is a single parameter the values will be asked accordingly. It is an increasing degradation distribution.

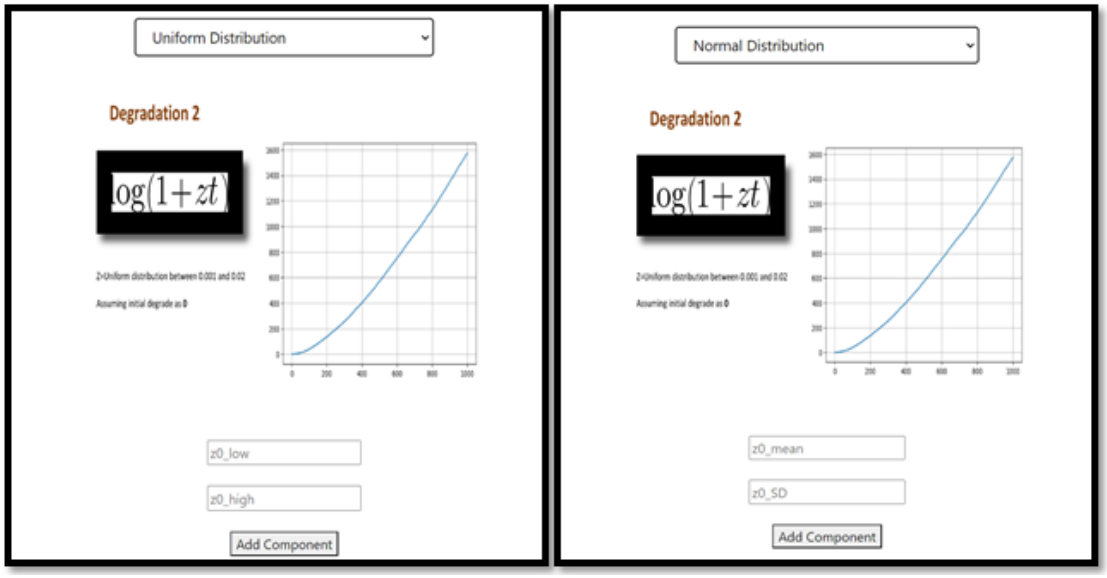

Figure 7-10 Degradation2 with Uniform And Normal Dist Parameters

The behaviour of multiple degradation paths for the same component having Degradation2 as the model to their failure have their failure patterns as follows:

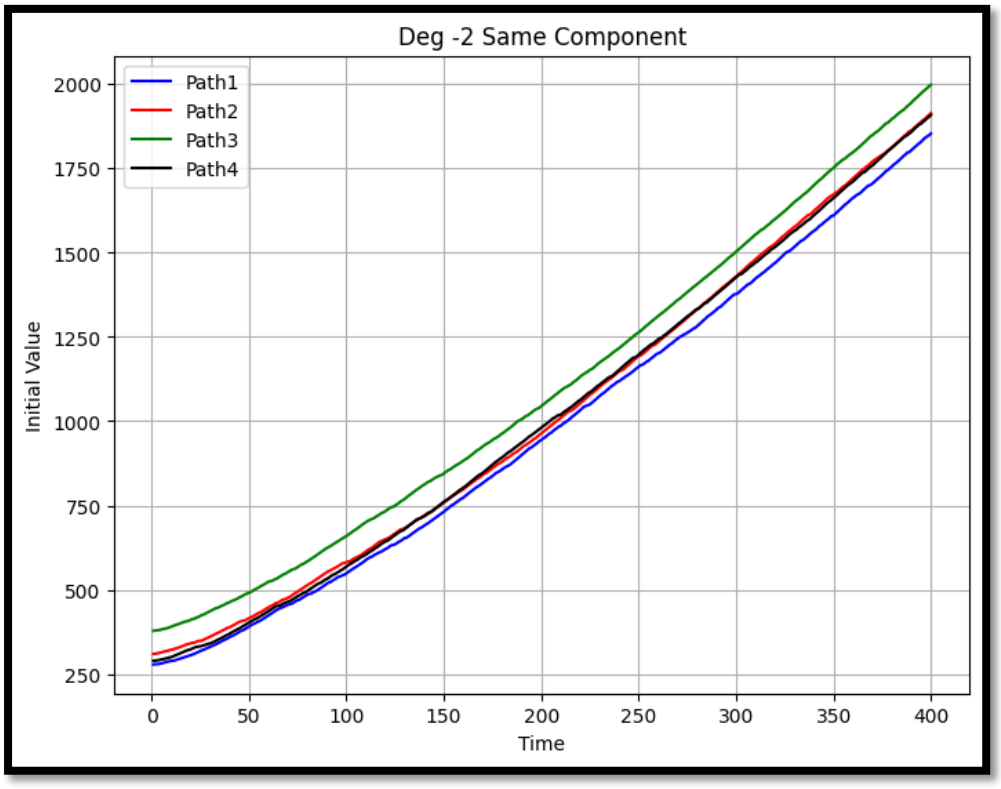

Figure 7-11 D2 Component multiple failure paths

Degradation- 3 follows the path of $\mathbf{z_0+z_1t+z_2t^{0.75}}$ where z_0, z_1, z_2 are the 3 parameters and t is the time which increases by 1 unit every time. As there are three parameters the values of distribution will be asked for each of them. It is an increasing degradation distribution.

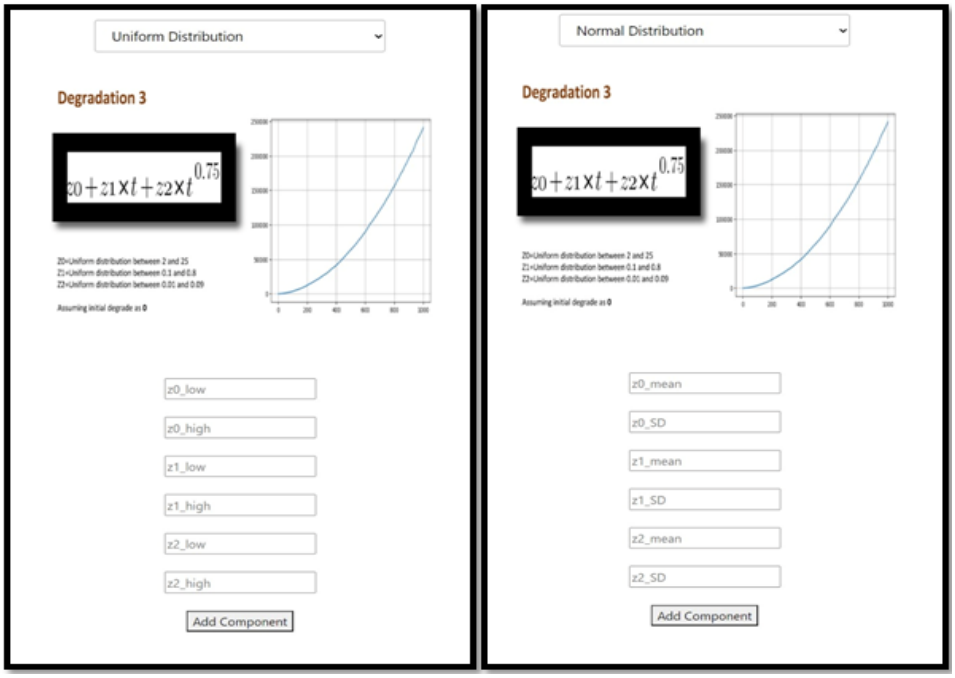

Figure 7-12 Degradation3 with Uniform And Normal Dist Parameters

The behaviour of multiple degradation paths for the same component having Degradation3 as the model to their failure have their failure patterns as follows:

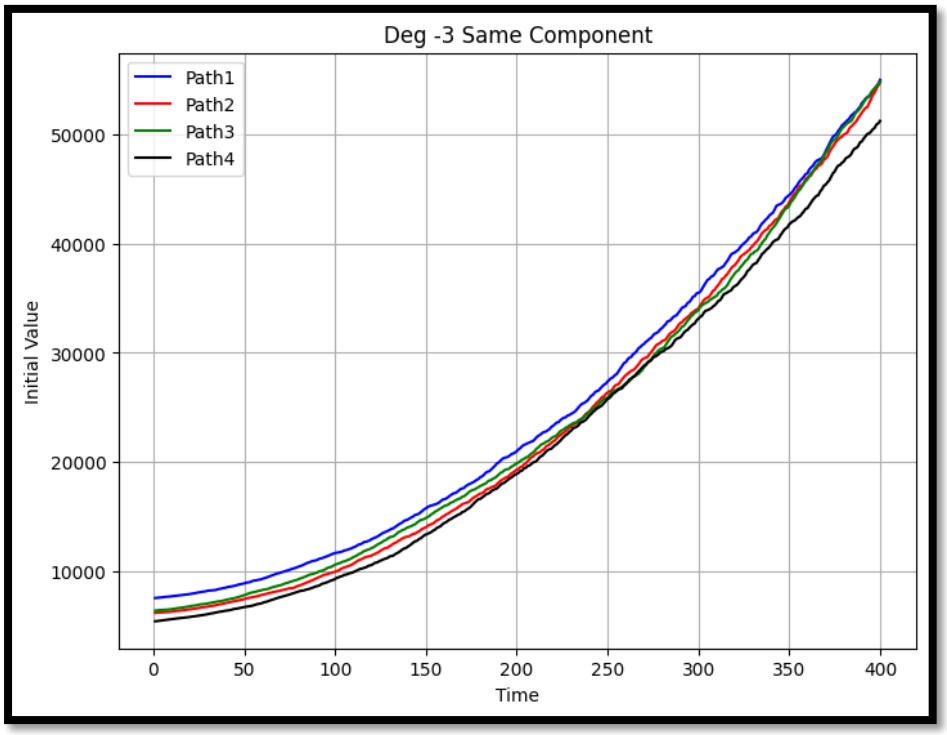

Figure 7-13 D3 Component multiple failure paths

Degradation- 4 follows the path of $\mathbf{z_0 + z_1 t^{0.95}}$ where z_0, z_1 are the 2 parameters and t is the time which increases by 1 unit every time. As there are two parameters the values of distribution will be asked for each of them. It is an increasing degradation distribution.

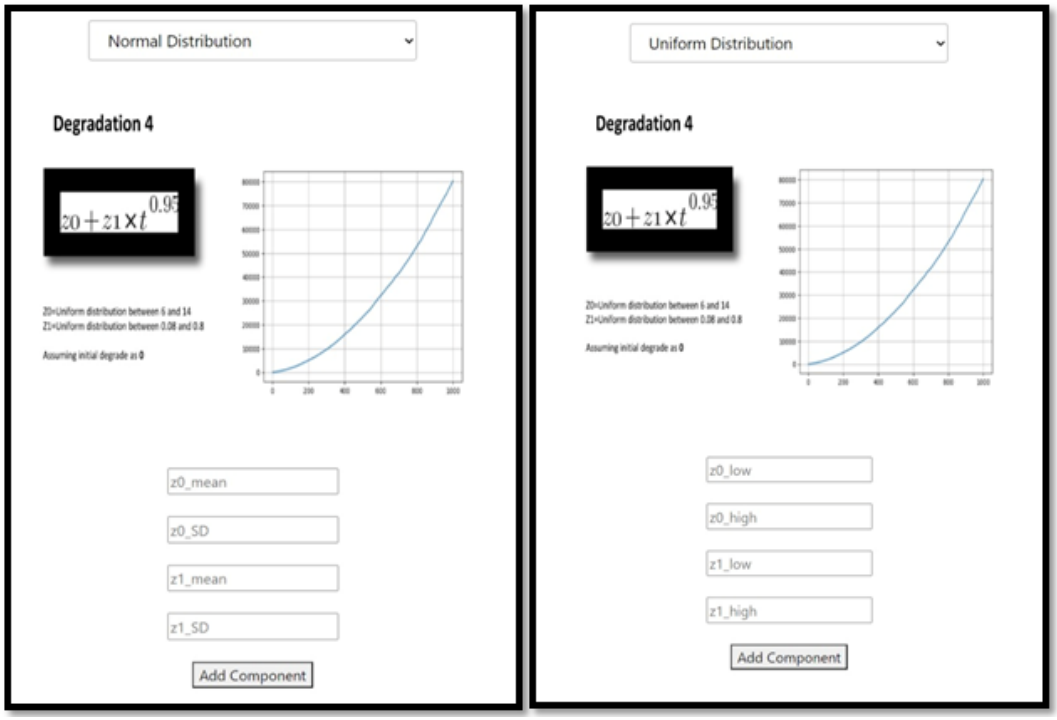

Figure 7-14Degradation4 with Uniform And Normal Dist Parameters

The behaviour of multiple degradation paths for the same component having Degradation4 as the model to their failure have their failure patterns as follows:

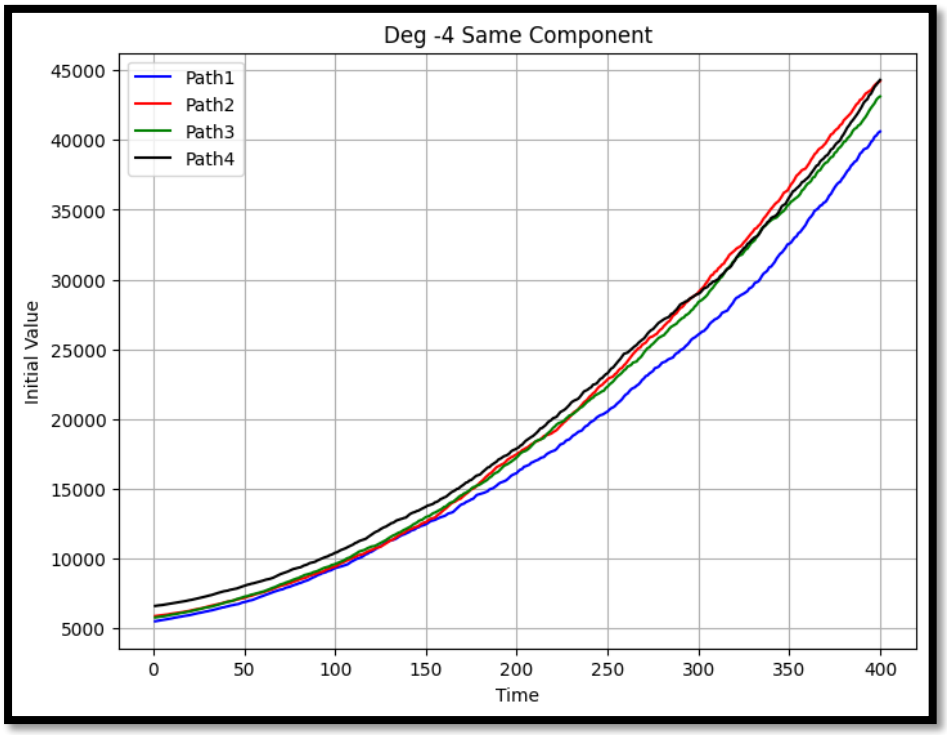

Figure 7-15 D4 Component multiple failure paths

Degradation- 5 follows the path of **100/ (zt)** where z is the parameter and t is the time which increases by 1 unit every time. As there is a single parameter the values will be asked accordingly. It is a decreasing degradation distribution.

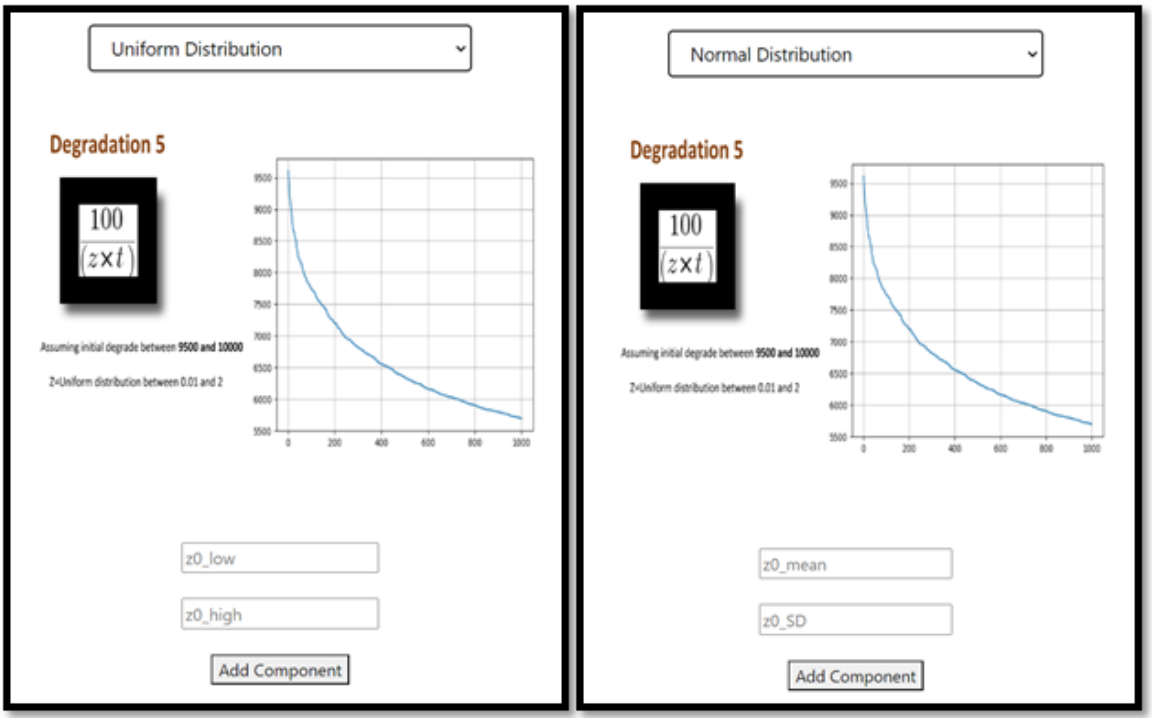

Figure 7-16 Degradation5 with Uniform And Normal Dist Parameters

The behaviour of multiple degradation paths for the same component having Degradation4 as the model to their failure have their failure patterns as follows:

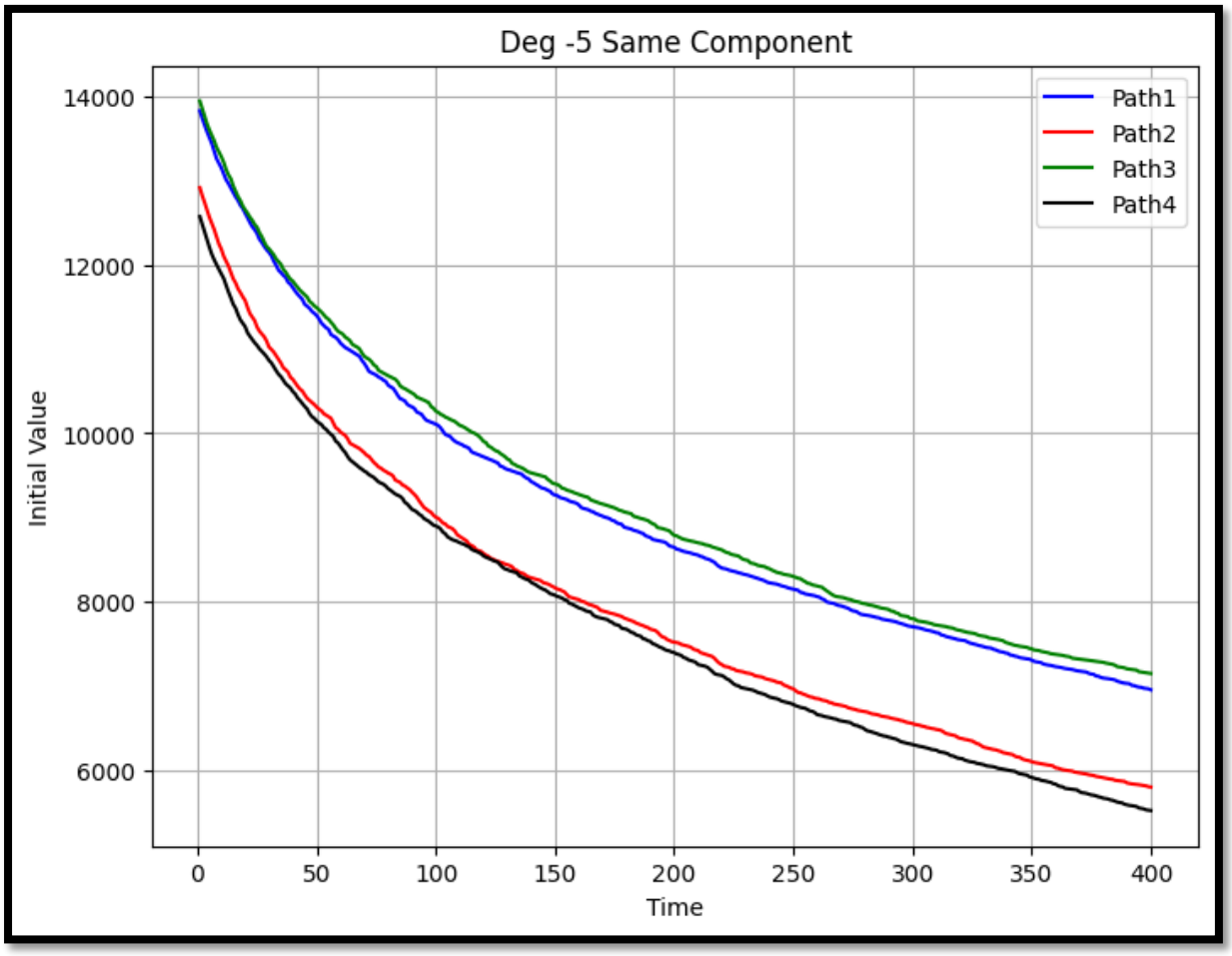

Figure 7-17 D5 Component multiple failure paths

After the user enters all the components it wants to add to the system the user needs to enter the button '**Show the System'.** Clicking the button enables the user to see the list of components it has added to the system. The components will be clearly differentiated with respect to the model each component possess. To show a depiction after adding 1 component to each of the 9 models present the components list for the system will be shown as the below. In case of degradation models the necessary values for the distribution of parameters are demonstrated as p1, p2 and so on. Since the distribution of the parameters whether normal or uniform are already given in the data so the values will be shown correspondingly. The odd p indices like p1, p3, p5 will indicate the minimum limit in uniform distribution while it will indicate the mean in normal distribution. The even p indices like p2, p4, p6 will indicate the maximum limit in uniform distribution while it will indicate the standard deviation in normal distribution. And 2 combined values denote the value of a single parameter. For example p1, p2 indicates the parameter z0 while p3, p4 indicates the values for the parameter z1 and so on.

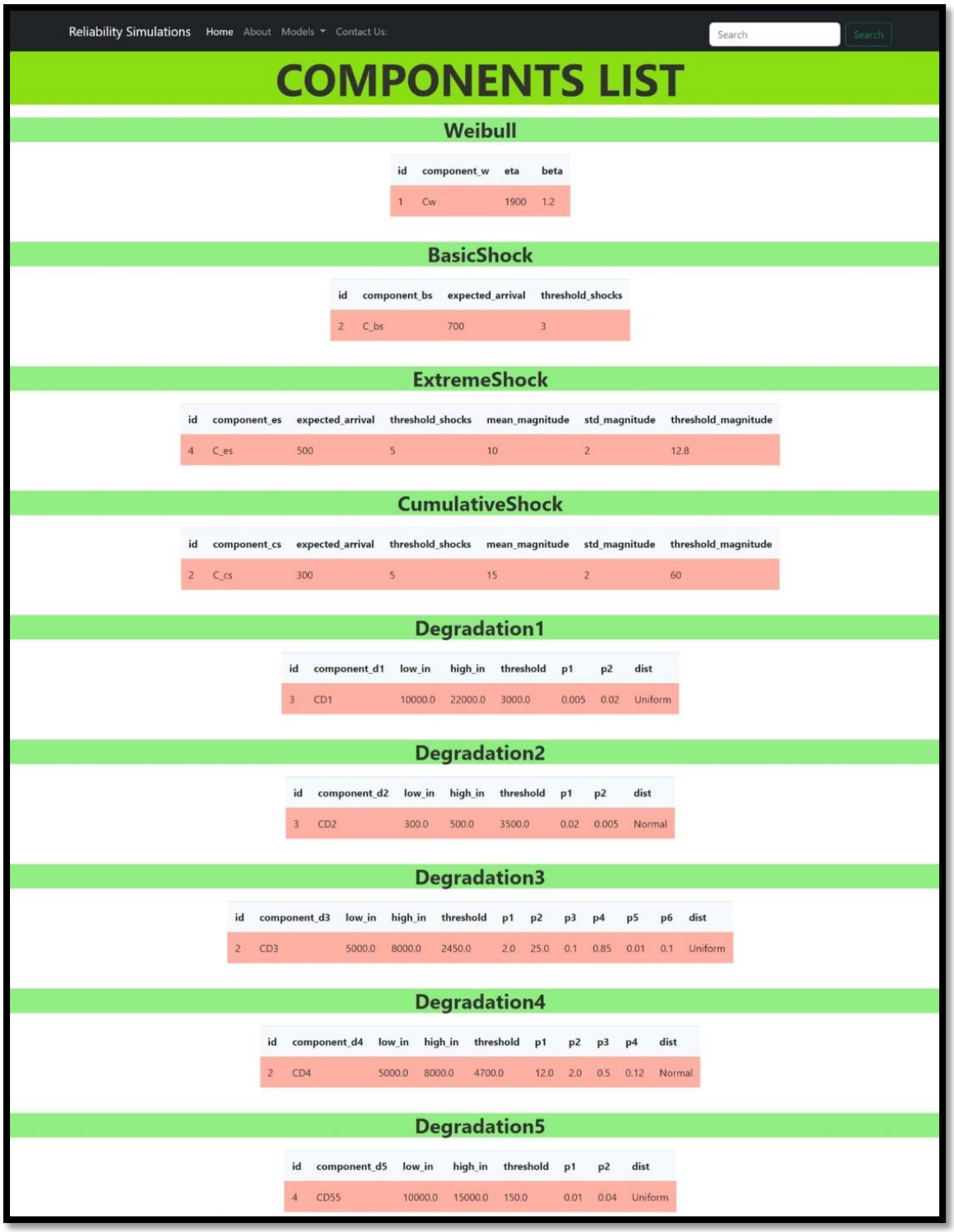

Figure 7-18 Showing the system

7.3.3 Implementing Dependencies:

In complex systems, it is common to find dependencies between different components or subsystems. These dependencies can play a significant role in determining the overall reliability and performance of the system. One type of dependency that can have a profound impact is failure dependency, where the failure of one component affects the functionality or reliability of another component within the system. That implies that when a component gets

failed another component rate of failure increases and hence failure will arrive early for the component who is dependent on the failed component. The pace at which the failure arrives is decided by the user in the interface. It can be demonstrated by user by giving a ratio to it. The user interface has an option to above or include failure dependencies in the interface. In case of clicking to avoid the failure dependencies the simulation final step arrives. In case the user wants to avoid failure dependencies. The inputs to the user for failure dependency arrives. The inputs come up in a matrix format where the rows and columns will have the number of components and the user has to fill the cell showing the dependency of row index on column index. The dependency value can be in between 0 and 0.99 denoting 0.99 to be most dependent while 0 means there is no dependency. The inputs asked by the user are in the format shown below. As the component won't have dependency on itself the value is chosen 1 by default. Once the failure dependencies values are entered we will have start simulations webpage.

In case of Weibull model components the value of eta parameter is reduced by (1-magnitude) that is defined by the user. While in case of shock models the time to arrival of shock is decreased by dividing (1+magnitude) while for extreme/cumulative shocks the the mean magnitude is increased by multiplying (1+magnitude). For degradation models the value of threshold is mutltiplied by (1-magnitude) and values of parameters by (1+magnitude).

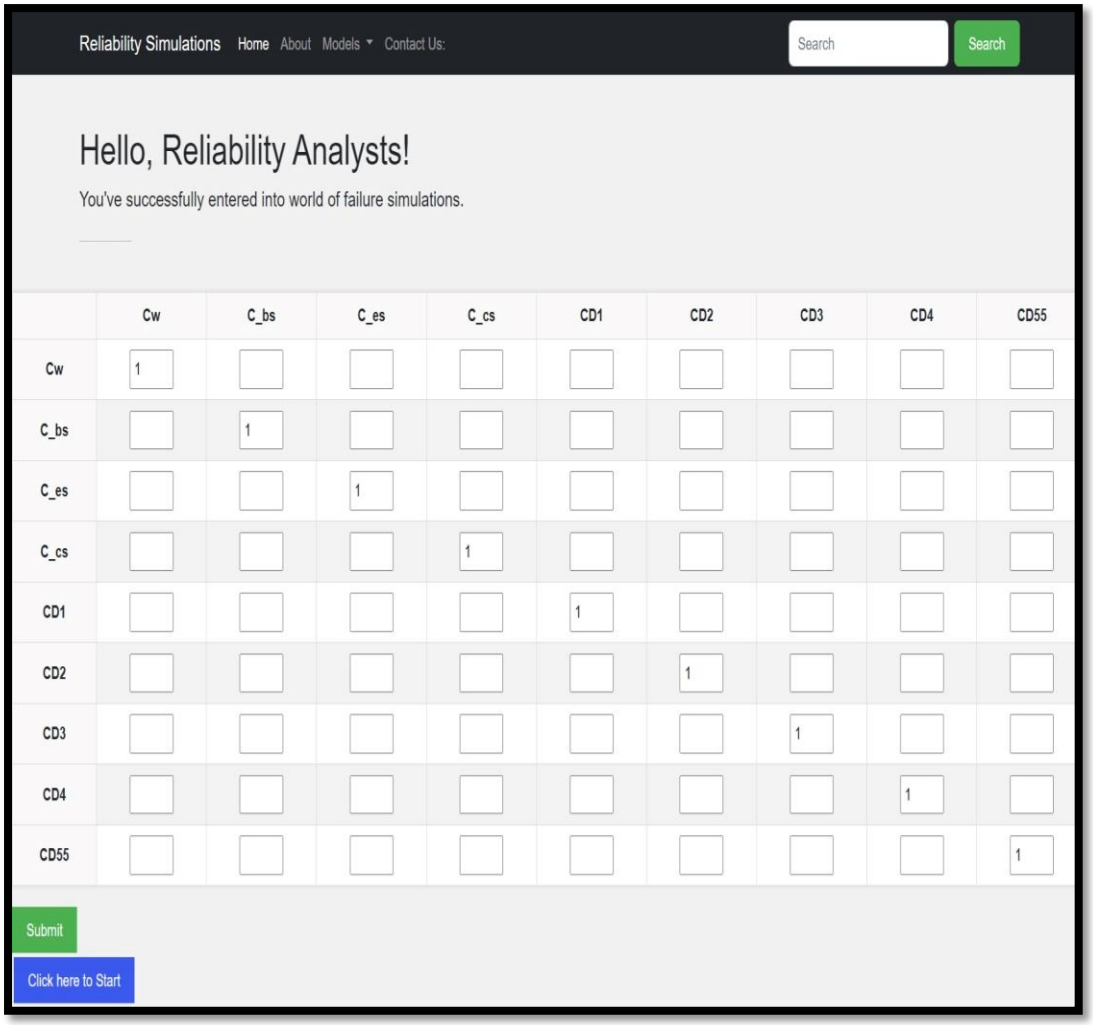

Figure 7-19 Accepting dependencies

7.3.4 Start Simulations

The most simple page in the interface which arrives directly whenever the user enters the button of no failure dependencies and arrives indirectly when the user enters the values of failure dependency and submits it and later clicks the button Click here to Start. The user here needs to enter how long the simulations need to run. To help the user to have a clear idea, it is designed that the user needs to enter the number of times the system needs to fail before the simulation stops. So basically the number of failures of system are to be entered by the user until simulation stops. After submitting the value the user enters Start Simulations button in order to start the simulations which run at the backend to produce the simulation results.

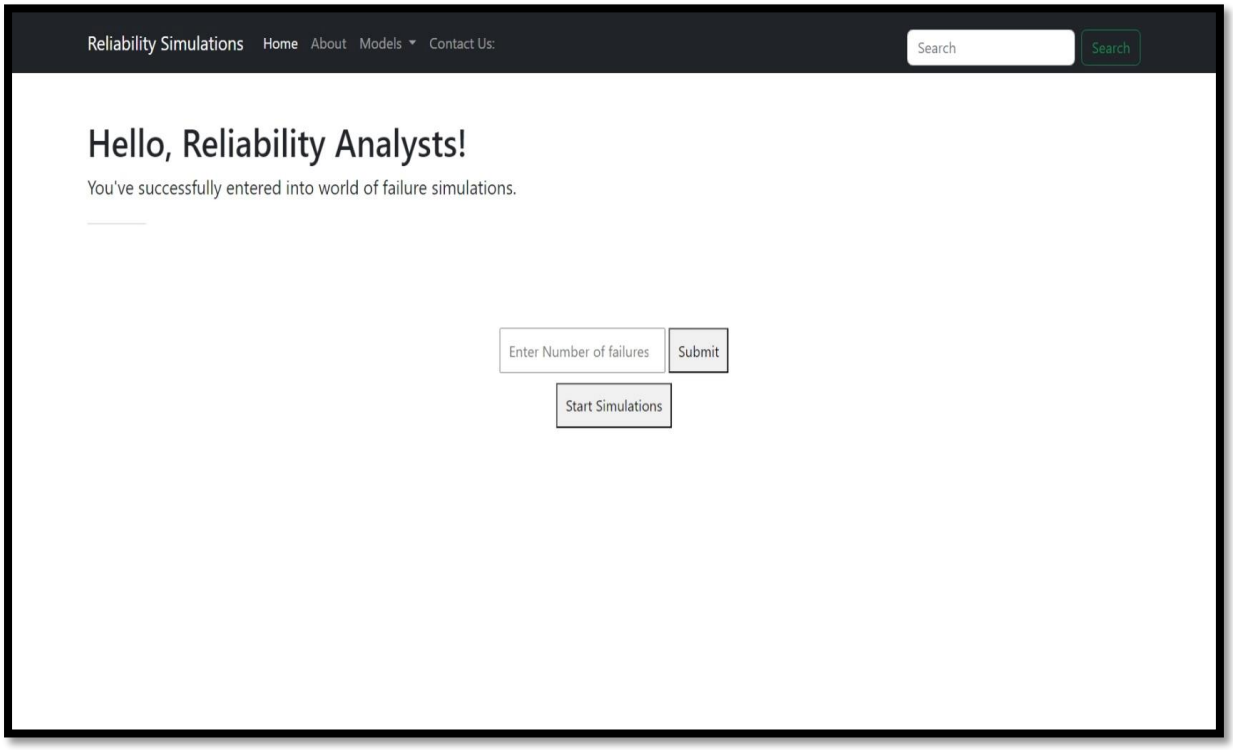

Figure 7-20 Accepting the number of failures to simulate

7.3.5 Post Simulation Results

Once the user enters the number of failure until which the simulation needs to be done and has clicked Start Simulations then at the backend the simulation process is done and in form of output the time to failure of the system and the component responsible for the failure of the system gets displayed. After clicking Start Simulations the biggest function inside the views.py file of the project named **process_data ()** gets executed. Inside the function the components, models and parameters of the model are extracted in form of an array. Each component is individually simulated for each failure and the failed component and time is detected every time the system gets failed. If it has dependency on other components then before the next simulation the failure dependency parameter is established and the next simulation occurs. This process continues until the number of failures given by the user is achieved. The output of the simulation is time to failure and failed component.

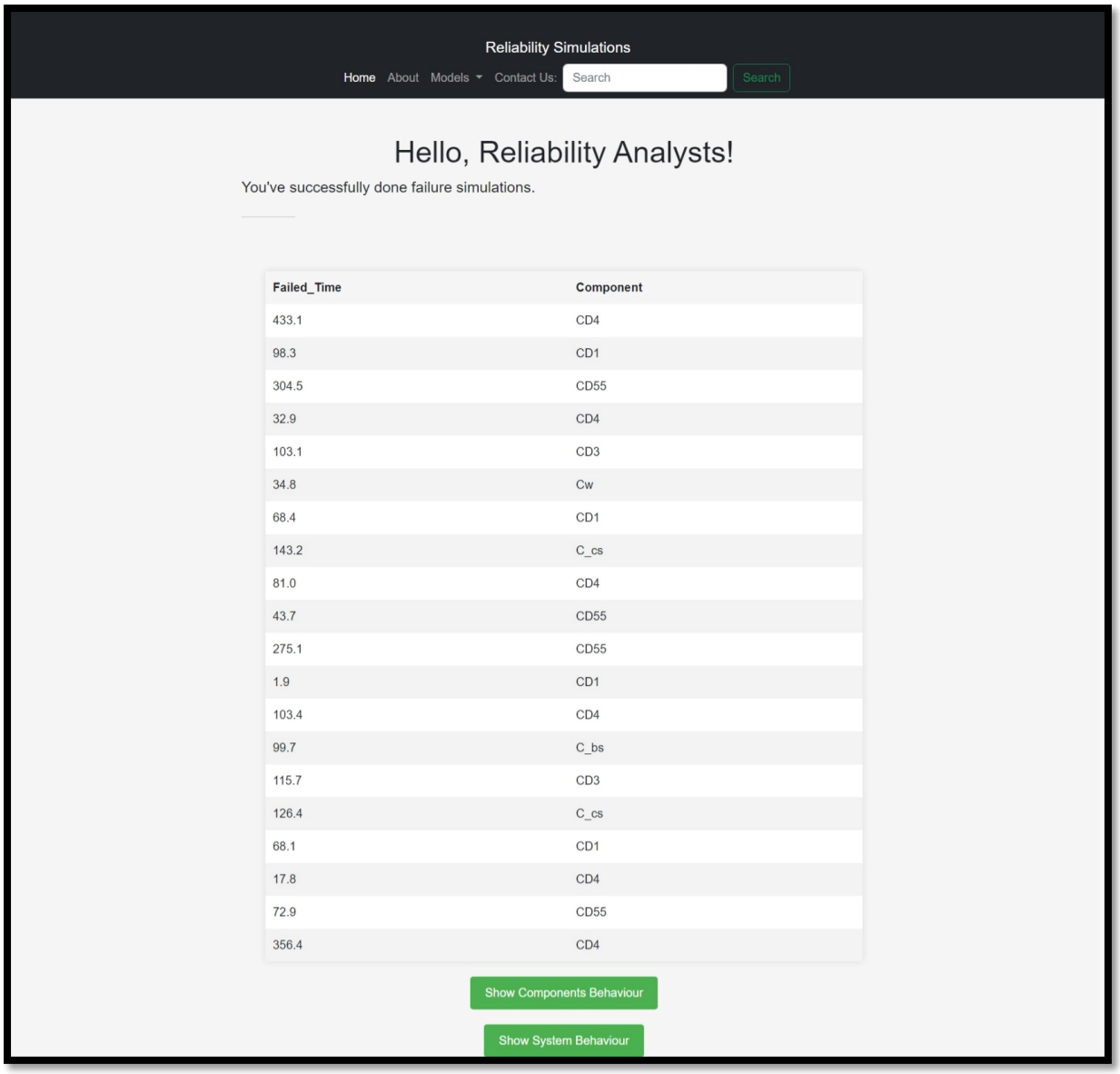

Figure 7-21 Example Simulation Result

The above is a result of a failure simulation involving 8 components and no dependency in between them and for 20 failures of the system. The failed time is not a cumulative number rather it denotes the time between failures. After the simulation is done user is asked whether it needs to know about the behaviour of all components in this process of simulation or behaviour of the system as a whole. If the system behaviour is chosen the overall behaviour of the system assuming it to be repairable is shown. A NHPP model is fit to the system to get the parameters of the model as well. For components behaviour the plots of individual components are shown depending upon the type of model the component follows.

7.3.6 System Behaviour

The system is a repairable system while the components present inside are non-repairable. Once the component gets failed even the system is also failed but the component is replaced itself thus enabling the system getting repaired. For modelling of repairable systems the most preferred model is the Non-Homogeneous Poisson Process (NHPP) otherwise called as the

power law model. The parameters of the model are found out by maximum likelihood estimation and graph is plot between the number of failures and cumulative time of failure of the system. An example graph is shown after conducting 50 simulations for a the system.

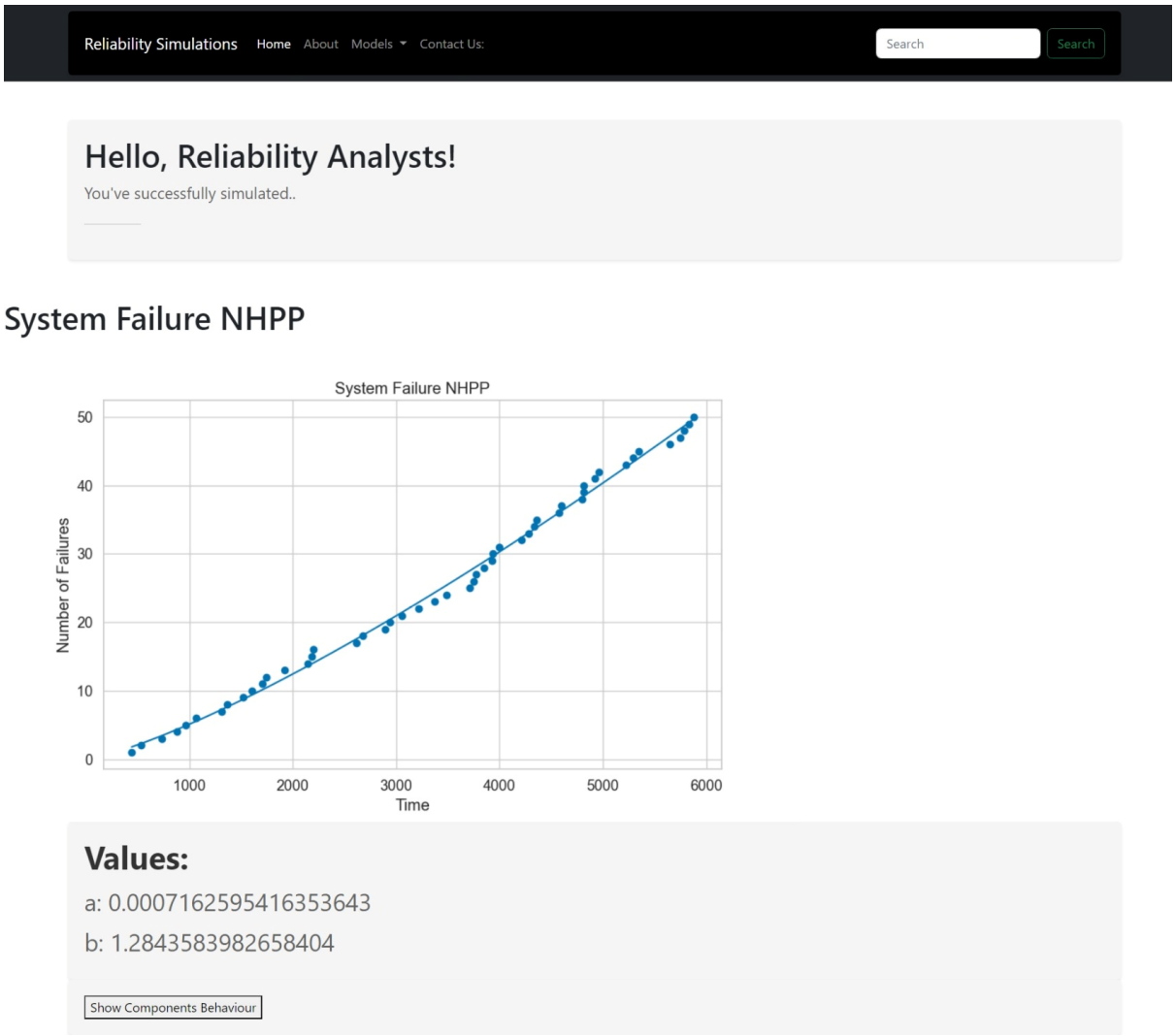

Figure 7-22 System Behaviour Displayed in the WebPage

The system above has an increasing failure rate as the value of b is greater than 1. These all insights are quite helpful for the analysts to design a better maintenance plan and all. The points are the actual failure times and the graph is the power law fitted plot.

7.3.7 Component Level Behaviour

The components level behaviour are also essential as they mean that complete traceability is ensured. The components behaviour is shown considering the type of model each component follows. As the component are broadly classified into 3 types: Shock Based, Degradation Based and Time to failure based there are 3 broad ways in which they are plotted individually.

For **time to failure** based or Weibull model there is a proper straight line parallel to x-axis and at the time at which the component failed is denoted by a height or straight line parallel to the y-axis.

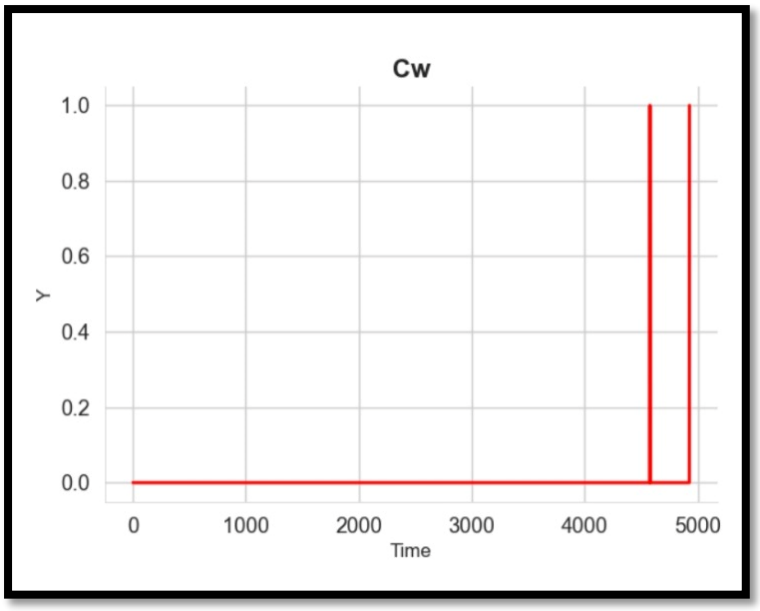

Figure 7-23 Weibull models failure representation

For **shock based models** there are broadly 2 categories at which component gets failed. One when the number of shocks exceed a certain threshold and other when the magnitude exceeds a certain threshold as decided. For basic shock model the magnitude is assumed to be 1 as there is no necessity of magnitude to decide the failure of the component. At every point where the shock has arrived there is a vertical line present for each time scale. The height of each vertical line represents the magnitude of the shock that has arrived. For basic shock the height is denoted as 1. There is a red dot attached to the shock which was at the end responsible to fail the component so that user can easily know the failure point along with the shocks.

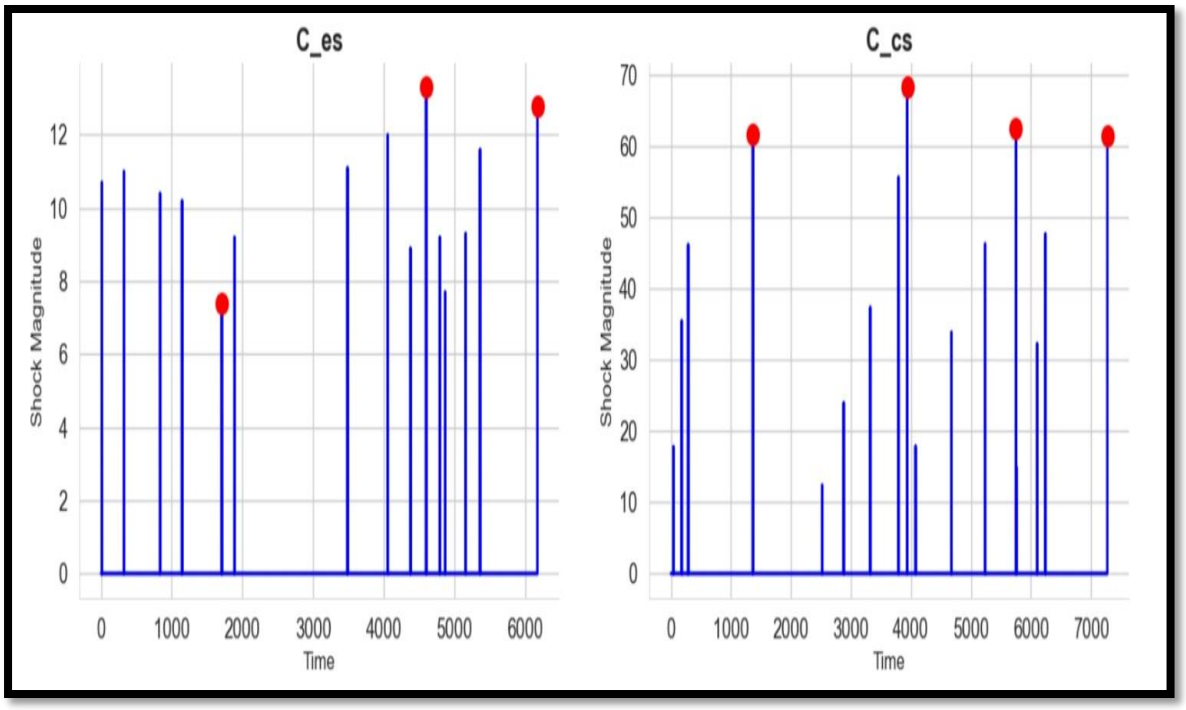

Figure 7-24 Shock Models Failure Representation

These are the representation of 2 of the components where one is of extreme shock model and the other one is of cumulative shock model. As there are only 3 dots in C_es the component has failed thrice during the course of simulation whereas due to 4 dots in C_cs the component has failed 4 times in the course of the simulation process. A lot of insights can be concluded by the user by looking into the plots.

For degradation based models as the only criteria for failure is the amount of degradation the component has undergone. The amount of degradation is one of the important criterion for the analysts to look into and hence while plotting it is taken into account that while in x-axis the time period until which the simulation is run is shown in y-axis the amount of degradation at each time scale is shown. Once the component fails, depending upon whether it is an increasing or decreasing degradation model the amount of degradation resets back to its original state that can give us an idea that the component is replaced and a new component has arrived in place.

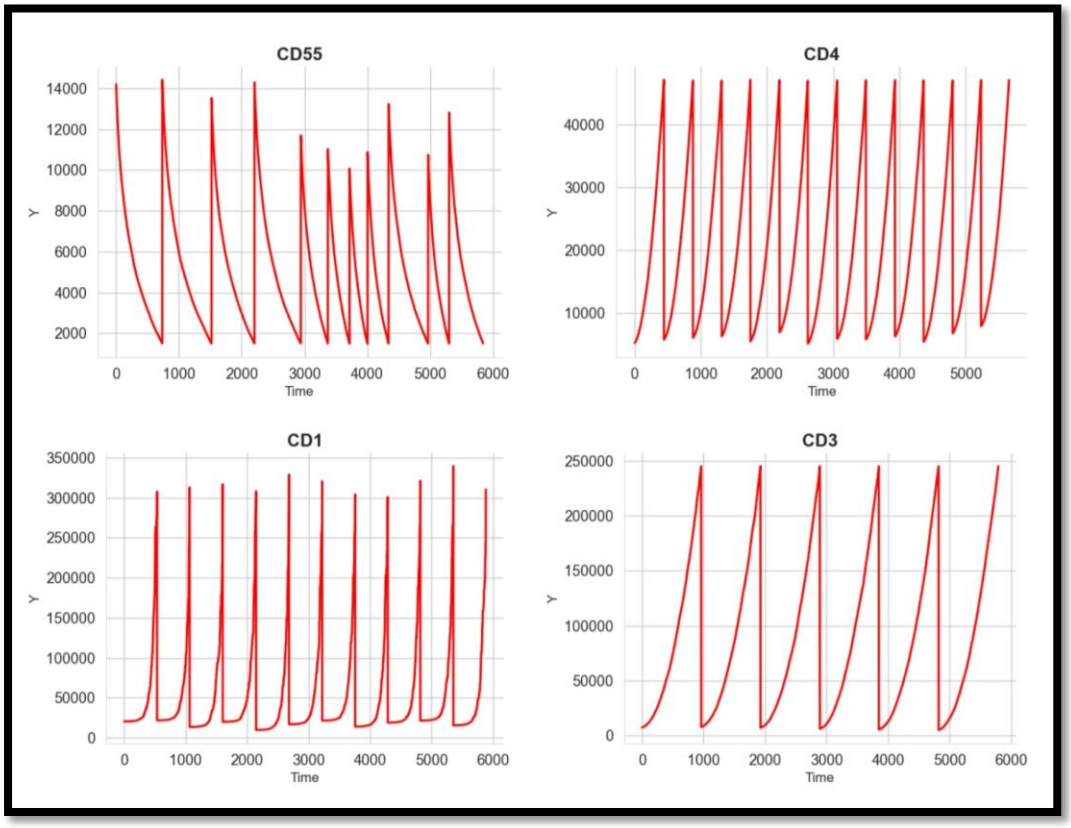

Figure 7-25 Degradation Models Failure Representation

Here except CD55 all other component follow increasing degradation model failure mechanism. Once the component gets failed there is vertical line going down which coincides with the initial degrade value for the new component. While in case of CD55 once the component fails there is a vertical line going from bottom to the top to the point that is the initial degrade for the new incoming component. The failure behaviour of all the components undergoing degradation failure mechanism are shown in the similar manner.

While clicking the Show Components Behaviour Button the behaviour of all the components present in the system are shown simultaneously, one such example of webpage that shows the behaviour is as shown below.

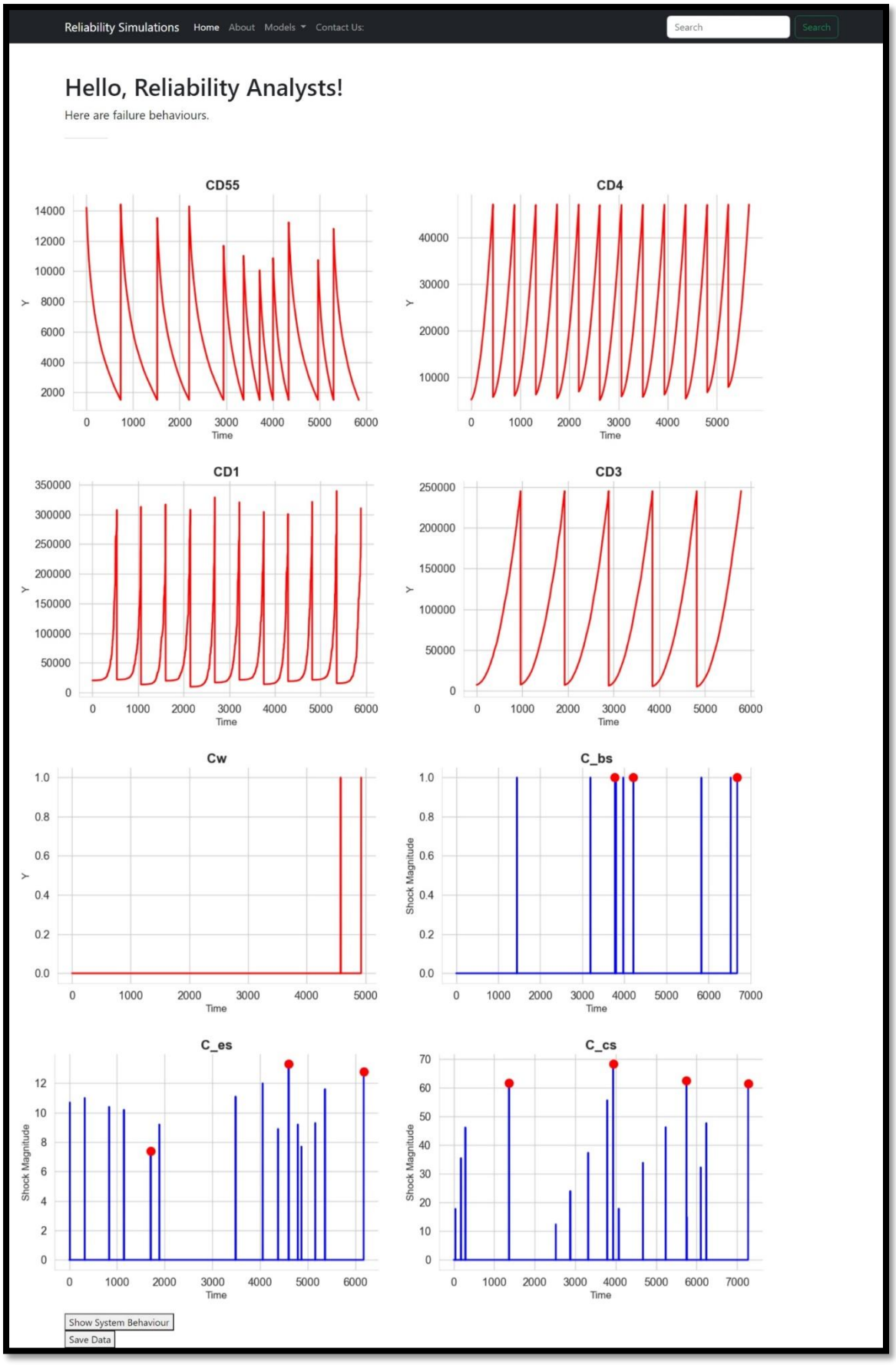

Figure 7-26 Components Behaviour as Displayed in WebPage

7.3.8 Save Data

The main objective of creating an interface is to provide a tool to the analysts who want to analyse the behaviour of systems, components inside the system and plan further action regarding the maintenance plan and predicting further failures for the system in terms of time and component responsible for that. To give a clear idea about the data that has been derived after simulating the entire system even we are also providing the data in csv format along with some self-analysing plots so that the analysts will get a better idea of what part of data to analyse. Each component of the system is saved to a single csv file and the type of data stored depends upon the type of model the component follows. Along with component wise data the data at a system level is stored.

In case of Weibull model, the time of failure of the component in a cumulative scale with respect to failure is stored.

In case of Shock Based models, the time of arrival of shocks along with the magnitude of the shock is stored. The state of the component at the time of arrival of shocks is also stored. If the component fails at that shock then the state is 1 else the state is 0.

In case of Degradation Based models, the amount of degradation as well as the state of the component at that time is stored. As storing the data in each time step makes the data more long and complicated, hence the data is stored for every 20 time steps. We have assumed the time step to be hours. So the degradation is measured in every 20 hours. If the component fails in between two time steps the state will be shown as 1 in the latter time step. In all the other cases the state is 1.

In case of system level, two things are recorded and stored as system level. One is the component that was responsible for the failure of the system. Other is the cumulative time of failure of the system in the course of simulation.

These all csv files are generated are given to the analysts who can derive other inferences from the simulator. After the data is saved a webpage indicating that the data is saved and simulation process is complete in total is displayed.

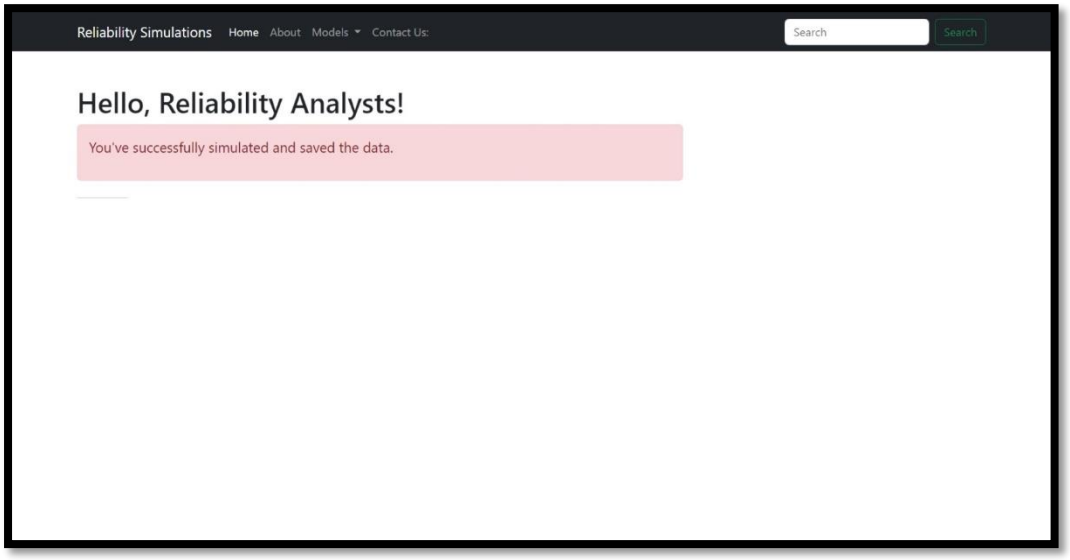

Figure 7-27 After completing the simulation and saving the data

8 Conclusion

In the beginning of the project our objective was to prove that traceability is important in component and sub-sytem level to find out the various performance parameters and improve the functioning time and reducing the down-time of the system. There was no failure data available so that it can be proved or not due to which failure simulators were created so that using those failure simulated data we can analyse the system and come into our hypothesis whether we will be rejected or not. The initial aspect was to build a model that would help predicting the next time to failure. This work has been successfully done and was tested on an independent components simulator and it has given perfect results even capturing the noise along with. But the most important outcome to prove this was to build a solid machine learning or deep learning model that can most correctly predict the next failed component using the data available. If the model would have been successfully able to predict the next failed component and we already have the model to predict the next time to failure then based on the values we can have a clear judgement whether to replace other components based on their history and other information. But the prediction of next failed component was unsuccecssful both by language modelling and also through a classification algorithm. Howeveer during language modelling (specially n-gram model) we were able to pick dependencies but couldn't jump into conclusion that finding the probabilities will help us in capturing dependencies among different components if there in the system.

The other part of the project included creating a simple, interactive user interface for failure simulations. The user interface will serve as a tool for analysts in the field of reliability and maintainability where they can a system of their own consisting of multiple components quite easily who have no coding background. Even dependencies are included in the simulator where the dependency between components can be easily established. After conducting failure simulations interactive plots are shown which show the failure behaviour of system and various components present in the system. Along with that the data is saved for each component level as well as system level so that the experts can use the data for any kind of analysis, preparing maintenance plans or design systems accordingly.

9 References

[1] Ech-chhibat, M.E., Bahatti, L., Raihani, A. and Fentis, A., 2018, April. Estimation of a repairable system reliability. In 2018 4th International Conference on Optimization and Applications (ICOA) (pp. 1-5). IEEE.

[2] Alsina, E.F., Chica, M., Trawiński, K. and Regattieri, A., 2018. On the use of machine learning methods to predict component reliability from data-driven industrial case studies. The International Journal of Advanced Manufacturing Technology, 94(5), pp.2419-2433.

[3] Rafiee, K., Feng, Q. and Coit, D.W., 2014. Reliability modeling for dependent competing failure processes with changing degradation rate. IIE transactions, 46(5), pp.483-496.

[4] Olteanu, D. and Freeman, L., 2010. The evaluation of median-rank regression and maximum likelihood estimation techniques for a two-parameter Weibull distribution. Quality Engineering, 22(4), pp.256-272.

[5] Yousefi, N., Tsianikas, S. and Coit, D.W., 2022. Dynamic maintenance model for a repairable multi-component system using deep reinforcement learning. Quality Engineering, 34(1), pp.16-35.

[6] Yousefi, N., Tsianikas, S., Zhou, J. and Coit, D.W., 2020. Inspection plan prediction for multi-repairable component systems using neural network. arXiv preprint arXiv:2001.09015.

[7] Zheng, R., Chen, B. and Gu, L., 2020. Condition-based maintenance with dynamic thresholds for a system using the proportional hazards model. Reliability Engineering & System Safety, 204, p.107123.

[8] Brown, P.F., Della Pietra, V.J., Desouza, P.V., Lai, J.C. and Mercer, R.L., 1992. Class-based n-gram models of natural language. Computational linguistics, 18(4), pp.467-480.

[9] Sen, A., 1998. Estimation of current reliability in a Duane-based reliability growth model. Technometrics, 40(4), pp.334-344.

[10] Mathlouthi, W., Fredette, M. and Larocque, D., 2015. Regression trees and forests for non-homogeneous Poisson processes. Statistics & Probability Letters, 96, pp.204-211.

[11] Brito, É.S., Tomazella, V.L. and Ferreira, P.H., 2022. Statistical modeling and reliability analysis of multiple repairable systems with dependent failure times under perfect repair. Reliability Engineering & System Safety, 222, p.108375.

[12] Peng, H., Wang, Y., Zhang, X., Hu, Q. and Xu, B., 2022. Optimization of preventive maintenance of nuclear safety-class DCS based on reliability modeling. Nuclear Engineering and Technology.

[11] https://www.reliawiki.com

[12] https://chat.openai.com