Safety Critical Systems Project Report
Predictive Maintenance in Vehicle Systems

Pushpita Sarkar(Matriculation No: 1384152), Nidhi Nayak (Matriculation No: 1404524), Deepak Kumar (Matriculation No: 1400489), and Ashlesh Mithur (Matriculation No: 1386367)

Frankfurt University of Applied Sciences

1 Introduction

Predictive maintenance is a technique that uses data analysis tools and techniques to detect anomalies in your operation and possible defects in equipment and processes so you can fix them before they result in failure. Ideally, predictive maintenance allows the maintenance frequency to be as low as possible to prevent unplanned reactive maintenance, without incurring costs associated with doing too much preventive maintenance. When predictive maintenance is working effectively as a maintenance strategy, maintenance is only performed on machines when it is required. That is, just before failure is likely to occur. This brings several cost savings:

Minimizing the time, the equipment is being maintained Minimizing the production hours lost to maintenance Minimizing the cost of spare parts and supplies These cost savings come at a price, however. Some condition monitoring techniques are expensive and require specialist and experienced personnel for data analysis to be effective.

2 Process Model

Agile-Scrum software development: Scrum is an agile methodology where products are developed iteratively. Planning, sprints, stand-ups, and retrospectives are integral parts of scrum methodology. In this model we are going to use Vmodel XT in combination with Agile scrum.

3 Team Organization

- Scrum Master
- Front-end team
- Backend team
- Testing team
- Deployment team
- Maintenance team

4 Task Distribution

Since we are following Agile-Scrum methodology, we will be having bi-weekly scrums wherein we will discuss updates related to the tasks assigned and blockers if any. Since we are a small team of 4 people, every member will contribute in all the phases of the project life-cycle.

5 Requirement Management

5.1 Functional Requirement

- The application shall display statistical inferences drawn from the data along with the threshold that will be used to trigger predictive maintenance warning.
- The application shall be able to start/stop on user request.
- The application shall predict whether gearbox is healthy or maintenance is required due or gearbox has broken once it is started.
- The application shall graphically display the status of the gearbox based on the sensor values.

5.2 Non-functional Requirement

- The application shall continuously log information so that it can be used to diagnostics in case of system failure.
- The application should be able to operate irrespective of the size of the data.
- The application shall be platform independent and ensure the privacy and security of the data.
- The application shall be safeguarded against vulnerabilities.

6 Use Cases

A gearbox is a mechanical device that increases the output torque of a motor or changes its speed. The shaft of the motor is connected to one end of the gearbox, which delivers a particular output torque and speed dictated by the gear ratio according to the internal gear design of the gearbox.
The physical components of gearboxes vary depending on the type of gearbox, as well as between manufacturers. The majority of gearboxes are made of steel, such as iron, aluminum, and brass. As a result, it is open and susceptible to wear and tear throughout its lifespan.

When the gearbox fails, it is because the gears are worn out.

Vibration condition monitoring can be utilized as a technique for early detection of a wide range of defects in mechanical parts. Using the same technique, we can predict failures in gearboxes. For this we have used a dataset which was derived

using four vibration sensors placed in four different direction inside the gearbox subjected to variable load. The data was recorded in 2 gear conditions: healthy and broken tooth, each under variable load.

6.1 Use-Case 1

Title: Application Start
Description: The application will start at the 'Statistical Inferences' page smoothly
Primary Actor: Client
Precondition: None
Postcondition: Application starts with 'Statistical Inferences' page.
Main Success Scenario: Application starts smoothly.

6.2 Use-Case 2

Title: Loading of 'Statistical Inferences' page
Description: The user should be notified with proper GUI if there is any delay in loading this page.
Primary Actor: Database
Precondition: Database has valid data and the user starts the application
Postcondition: 'Statistical Inferences' page loads up properly or the user gets notified about the delay if any
Main Success Scenario: 'Statistical Inferences' page loads up properly

6.3 Use-Case 3

Title: Starting 'Gearbox Maintenance & Failure Prediction'
Description: The application will inform the user if the gearbox needs maintenance.
Primary Actor: Client
Precondition: Client has triggered 'Gearbox Maintenance & Failure Prediction'
Postcondition: The user gets notified if the gearbox needs maintenance with a pop-up window.
Main Success Scenario: The user gets notified if the gearbox is healthy / needs maintenance or has broken and needs replacement

6.4 Use-Case 4

Title: Pop-up window for notifying the client
Description: The application will inform the user about the maintenance status of the gearbox with a pop-up.

Primary Actor:
Precondition: Client has triggered 'Gearbox Maintenance & Failure Prediction' using the Start button
Postcondition: The user gets notified with a pop-up window display the current status of the gearbox.
Main Success Scenario: The user gets notified if the gearbox is healthy / needs maintenance or has broken and needs replacement

6.5 Use-Case 5

Title: Loading of 'Historical Logs' page
Description: The application should display timestamped logs to the user
Primary Actor: Database
Precondition: 'Gearbox Maintenance & Failure Prediction' has already been triggered by the client at least once
Postcondition: The application should display relevant logs to the client
Main Success Scenario: Logs are displayed to the client

6.6 Use-Case 6

Title: Display of gearbox status using an GUI Indicator
Description: The application should display the gearbox status to the client with the help of a GUI Indicator
Primary Actor: -
Precondition: 'Gearbox Maintenance & Failure Prediction' has already been triggered by the client at least once
Postcondition: The application should green for healthy gearbox, yellow for a gearbox which needs maintenance and red for a gearbox that has broken and needs replacement
Main Success Scenario: The application displays the gearbox status to the client

7 List of Deliverables

- Requirement Specification Document
- Functional Specification Document
- Design Document
- Unit Test Results
- Integration Test Results
- System Test Results
- Deployment Document
- User Manual

8 Milestone

Activities	Week47	Week48	Week49	Week50	Week51	Week52	Week01	Week02	Week03	Week04
Feasiblity and Planning										
Design and Validation										
Build and Validation										
Testing and Validation										
Customer Acceptance and Validation										
Deployment										

Fig. 1. Scheduled Milestone

9 Risk involved in the Project

- Improper interpretation of customer requirements
- Inadequate requirement analysis
- Inter component interactions

10 Requirement Analysis

10.1 Functional Requirement

1. The application shall display statistical inferences drawn from the data along with the threshold that will be used to trigger predictive maintenance warning.
2. The application shall be able to start/stop on user request.
3. The application shall predict whether gearbox is healthy / maintenance is due or gearbox has broken once it is started.
4. The application shall graphically display the status of the gearbox based on the sensor values

10.2 Non-Functional Requirement

1. The application shall continuously log information so that it can be used to diagnostics in case of system failure.
2. The application should be able to operate irrespective of the size of the data.
3. The application shall be platform independent and ensure the privacy and security of the data.
4. The application shall be safeguarded against vulnerabilities.

11 Effort Estimation

Calculating Unadjusted Functional Point

Function type	Simple	Average	Complex	Considered For Project	Count	Total
External Inputs	3	4	5	3	2	6
External Output	4	5	7	5	2	10
External Inquiries	3	4	6	5	3	12
Internal Logical Files	7	10	15	10	3	30
External Interface Files	5	7	10	5	0	0
						58

UFP = 58

Sr. No	Characteristics	$(0-5)$
1	Data communications	3
2	Distributed data processing	0
3	Performance	3
4	Heavily used configuration	0
5	Transaction rate	5
6	On-Line data entry	0
7	End-user efficiency	0
8	On-Line update	0
9	Complex processing	5
10	Reusability	2
11	Installation ease	2
12	Operational ease	2
13	Facilitate change	3
14	Multiple sites	0
		25

CAF $= 0.65 + (0.01 * \text{Fi})$

CAF $= 0.65 + (0.01 * 25) = 0.9$

FPC $= UFP * CAF$ $= 58 * 0.9 = 52.2$ 52.

12 Constructive Cost Model

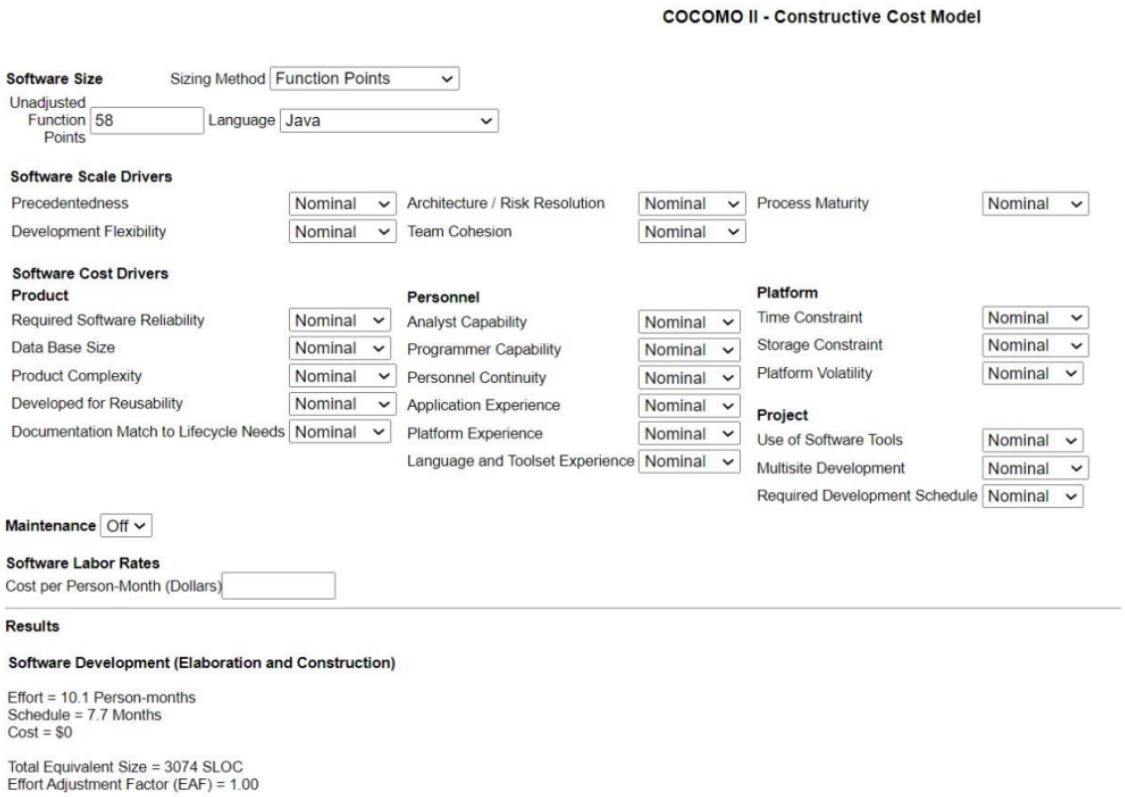

Fig. 2. COCOMO II- Constructive Cost Model

13 Data Flow Diagram

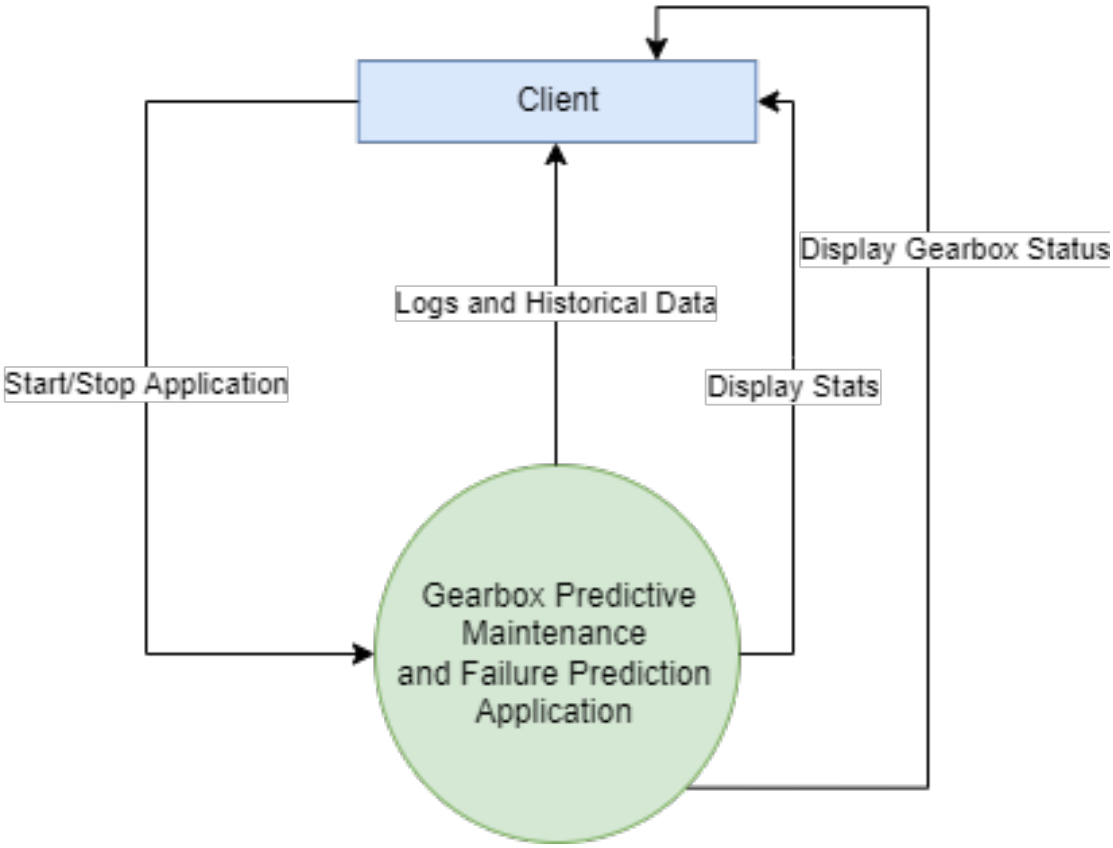

Fig. 3. Data Flow Diagram Level-0

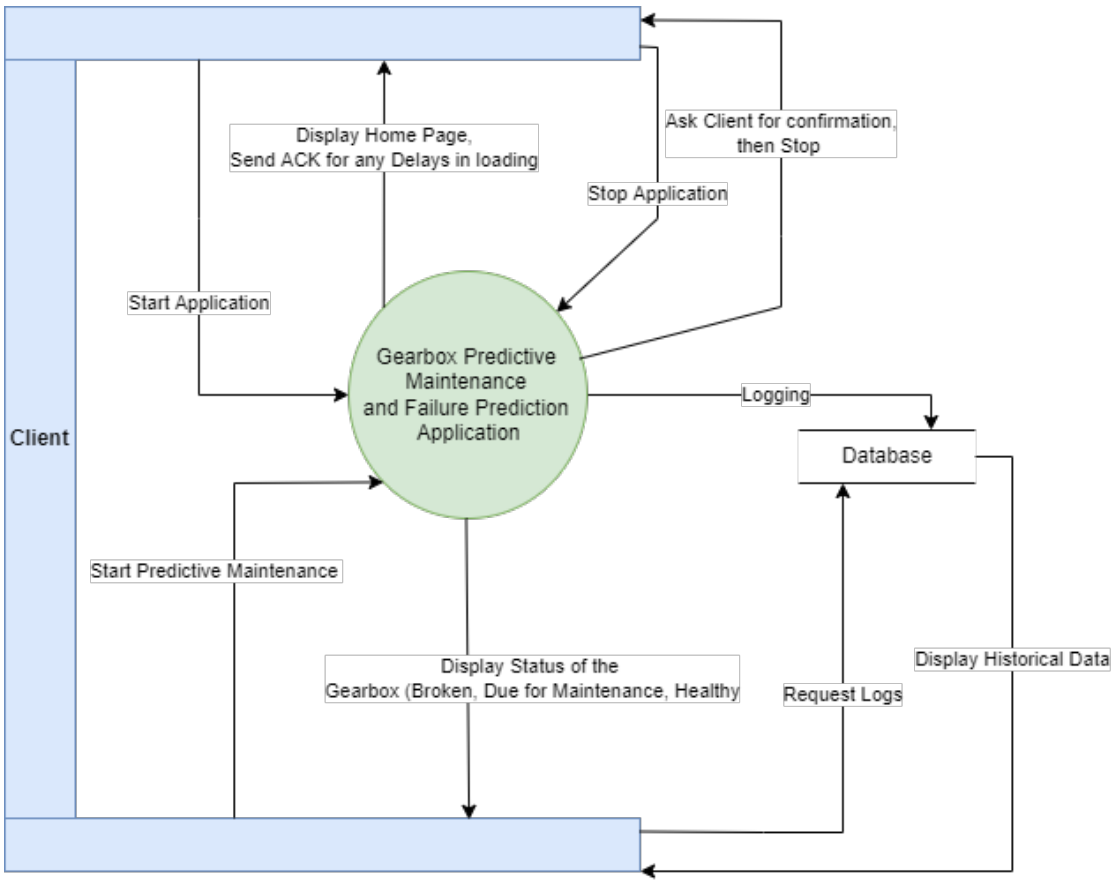

Fig. 4. Data Flow Diagram Level-1

14 Control Flow Diagram

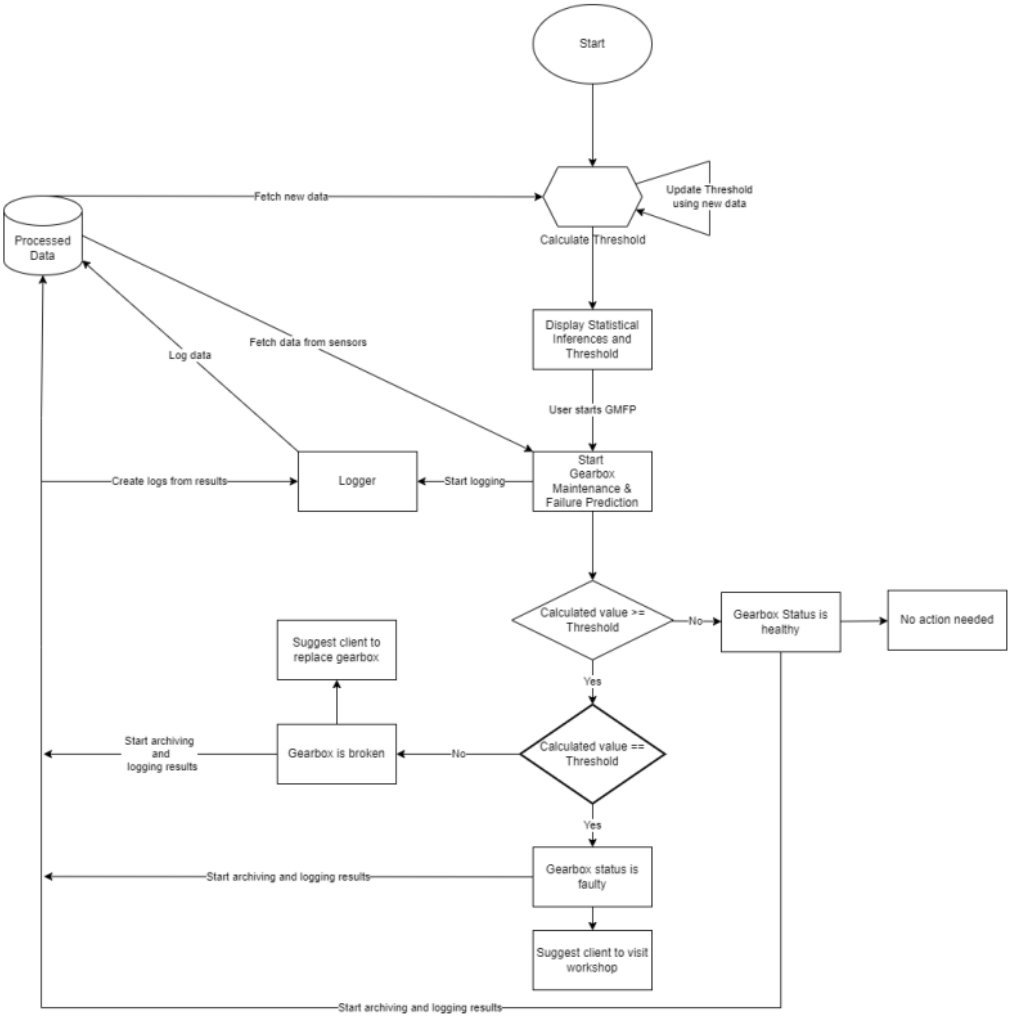

Fig. 5. Control Flow Diagram

15 Use Case Diagram

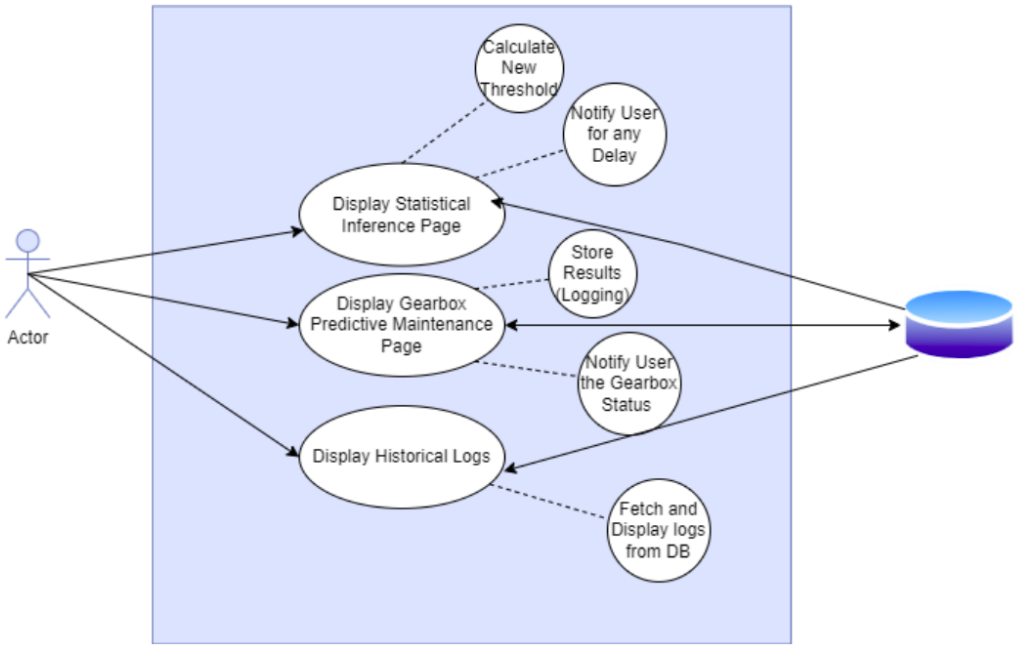

Fig. 6. Use Case Diagram

16 Sequence Diagram

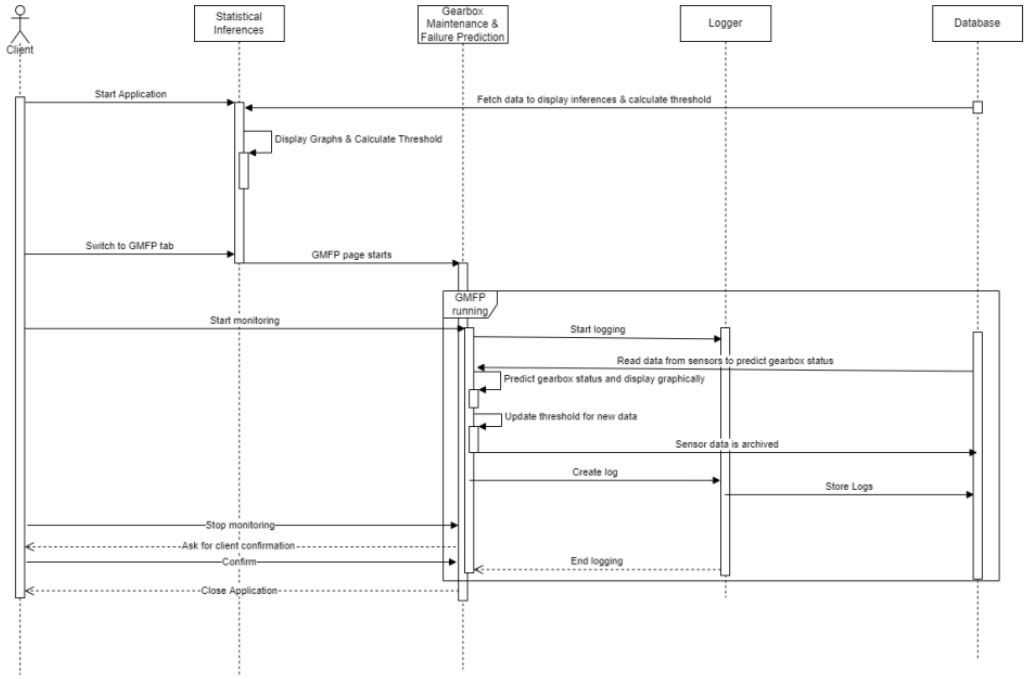

Fig. 7. Sequence Diagram

17 System Architecture

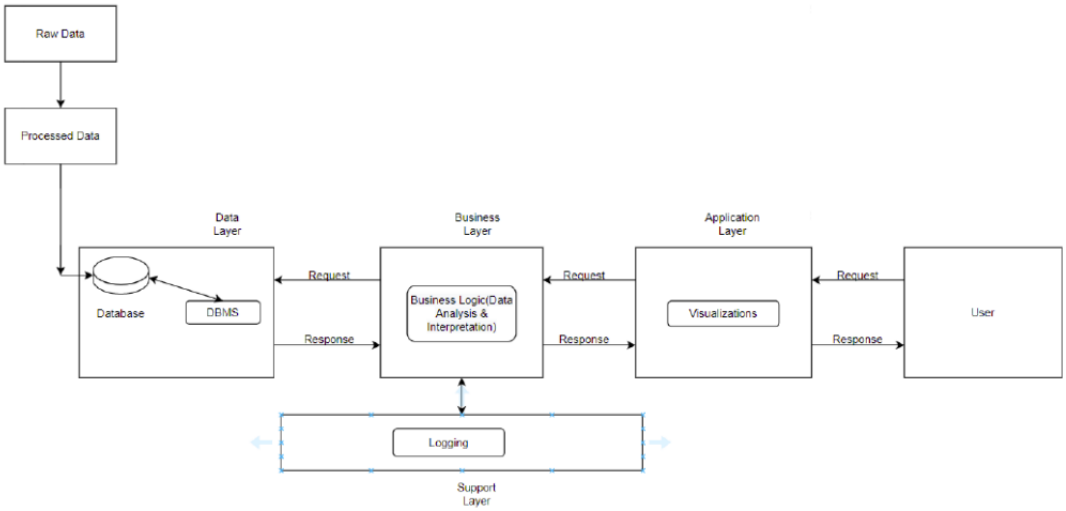

Fig. 8. System Architecture

18 Hazard Analysis

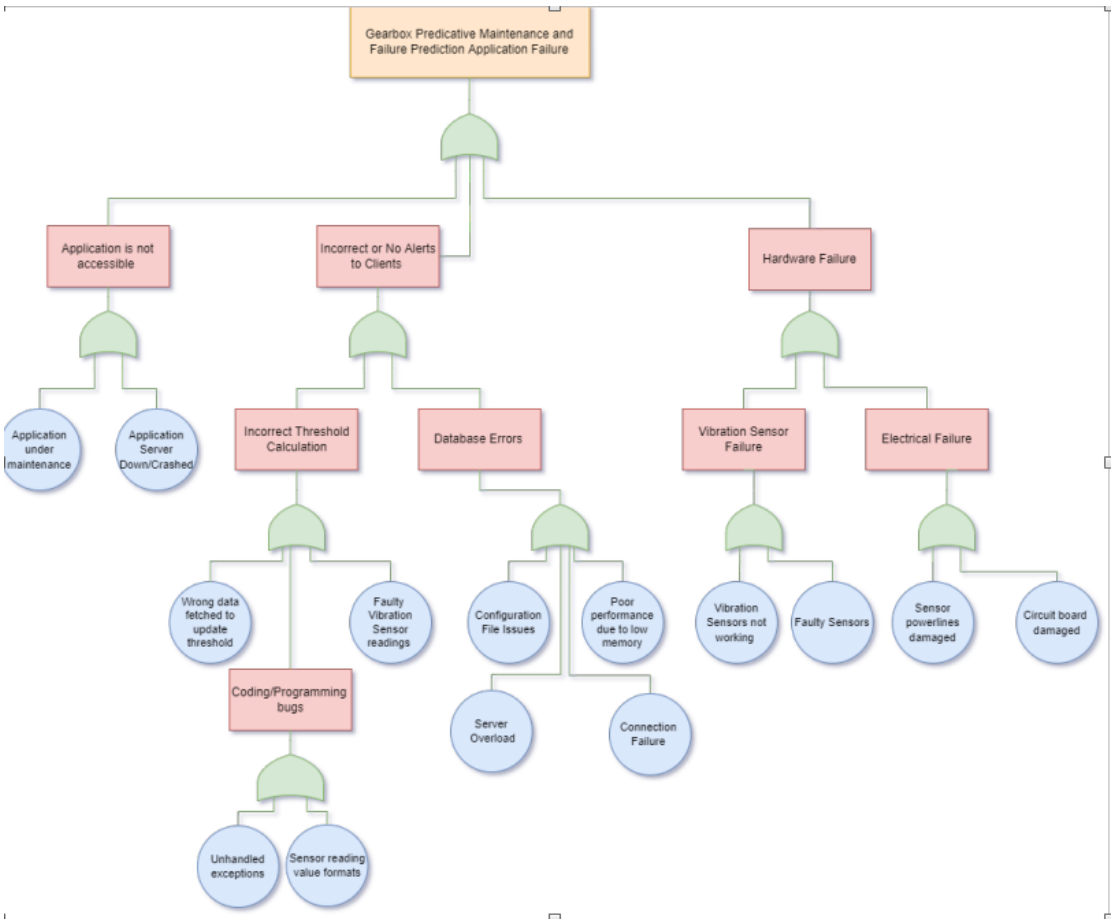

Fig. 9. Use Case Diagram

19 Safety Plan

The safety plan was created following a thorough examination of the system's hazard analysis results when it is subjected to specific environmental and technical conditions. We incorporated several safety mechanisms that were considered safety constraints within the system if the system failed, because a hazard describes a system's state that should not occur for safety reasons. We conducted a preliminary hazard analysis to identify probable hazards and their contributing flaws in a systematic manner. We then chose which flaws to fix and how to mitigate them after discussing with each other. The mitigations were outlined as mandatory safety measures for the system.

19.1 SID: 1

Hazard Category: Application is not accessible **Hazard:**

1. Application under maintenance.
2. Application server down/crashed

Root Cause:

1. Scheduled or planned maintenance
2. Issues at the client side such as file system corruption

Risks: If the application is unavailable for a considerable amount of time, the data coming from vibration sensors will be lost thus resulting the low reliability. The user will not be able to actively see the status of the gearbox.
Safety Plan:

1. Notify the user in prior about the downtime due to scheduled maintenance. Once the maintenance is over, the application shall publish the updated status of the gearbox using the data logged during downtime.
2. While the system is inaccessible, the data coming from the vibration sensors will be directly persisted until it is used to update the threshold. The user will also be provided with certain troubleshooting tips in the user manual. Once the system is available again, the application shall publish the updated status of the gearbox using the data logged during downtime.

19.2 SID: 2

Hazard Category: Incorrect threshold calculation **Hazard:**

1. Wrong data fetched to update threshold
2. Faulty vibration sensor readings (Out of scope)

Root Cause:

1. Improper transaction management of inconsistent data handling

2. Wear and tear of vibration sensors

Risks: Inaccurate gearbox status predictions due to incorrect threshold updates
Safety Plan:

1. Ensure data archival during downtime and orderly transaction execution to update the threshold
2. Ensure regular maintenance checks of vibration sensors

19.3 SID: 3

Hazard Category: Coding or Programming bugs **Hazard:**

1. Unhandled Exceptions
2. Sensor reading value formats

Root Cause:

1. Code smells
2. Conversion between different units of vibration

Risks:

1. Application crashes
2. Incorrect gearbox status predictions due to incorrect threshold updates

Safety Plan:

1. Ensure proper coding guidelines is followed during development and testing
2. Ensure conversion between different units of vibration is implemented during development phase.

19.4 SID: 4

Hazard Category: Database errors **Hazard:**

1. Configuration file issues
2. Poor performance due to low memory
3. Server overload
4. Connection failure

Root Cause:

1. Improper database deployment
2. Hardware issues or deadlocks arising due to improper transaction management
3. Hardware issues or server crashes

Risks: Unavailability of the database might result in loss of data coming from the vibration sensors thus affecting the reliability of the system and incorrect predictions of gearbox status
Safety Plan:

1. Ensure database is deployed properly
2. Ensure orderly execution of optimized queries to avoid deadlocks and scheduled maintenance checks should be carried out to ensure proper functioning of the database
3. Scheduled maintenance checks should be carried out to ensure proper functioning of the database

19.5 SID: 5

Hazard Category: Vibration sensor failure (Out of scope) **Hazard:**

1. Vibration sensor not working
2. Faulty sensors

Root Cause:

1. Wear and tear
2. Sensor malfunction

Risks: Data from one or more sensors will not be logged and thus will result in incorrect gearbox status prediction
Safety Plan: User should be notified if the system is not receiving data from all 4 vibration sensors. The application should not predict gearbox status if at least 1 of the sensors are not working

19.6 SID: 5

Hazard Category: Electrical failures (Out of scope) **Hazard:** Circuit board damaged

1. Sensor powerline damaged
2. Circuit board damaged

Root Cause: Wear and tear
Risks: Data from one or more sensors will not be logged and thus will result in incorrect gearbox status prediction
Safety Plan: User should be notified if the system is not receiving data from all 4 vibration sensors. The application should not predict gearbox status if at least 1 of the sensors are not working.

20 Question 2

New technology is causing fundamental changes in the etiology of accidents, necessitating adjustments in the explanatory processes now in use. The most effective models will go beyond assigning guilt and instead assist engineers in learning as much as possible about all of the elements at play, including social and organizational structures. Hazard analysis is essential for ensuring

the safety of smart systems that are often controlled by software. Systems-Theoretic Mischance Modelling and Processes (STAMP) has been utilized in different zones to obtain more causal components amid risk examination as a novel causality model. However, the use of STAMP to date has been random, with no formal mechanism in place to efficiently study system threats, and the quality of the analysis results cannot be assured. Additionally, the time element has received less attention in STAMP-based analysis as a key source of risks. This work proposes a systematic technique for hazard analysis based on STAMP in order to overcome these drawbacks. And the Hazardous Control Action Tree (HCAT) is offered as a model and analytical tool for all circumstances that should be addressed in hazard analysis.

Leveson published Systems-Theoretic Accident Modeling and Processes (STAMP) to capture a broader variety of causative elements than event-chain models and increase the efficacy of software hazard assessments. The STAMP views the mishaps as the result of a lack of control or enforcement of safety limits during the system's development, design, and operation. And, based on STAMP, the System Theoretic Process Analysis (STPA) approach is created to identify the Hazardous Control Action (HCA) as the causes in the hazard analysis, which can violate safety-related constraints and contribute to system hazard. STPA's application, on the other hand, is ad hoc and lacks rigid processes, necessitating greater work, time, and associated deep expertise in the STPA process.

Although certain support tools, such as SARRA, have been given to aid with the STPA hazard analysis, these tools focus on preserving the consistency or traceability of the data rather than providing a systematic technique to detect the HCAs in the STPA process. The expanded STPA techniques with context table were presented by Thomas and Asim for incorporating all potential environmental factors in the HCA identification of STPA, but these methods do not focus on temporal conditions, and the analysis process still relies on the analysts.

Why do we need something new? The current problem is that, our tools are 40 to 65 years old. But out technology is very different today. FMEA was introduced between the year 1945-1950. FTA, HAZOP and ETA came in the year 1960-1970. In the late 1970's people started using computers in control which gave rise to exponential increase in complexity, lots of new technology, new ways of doing things and changes in the way humans work.

Accident models are used to investigate and analyse accidents, as well as to prevent future ones and to determine whether systems are safe to use (risk assessment). They impose patterns on accidents in accident investigations, influencing both the data collected and the elements identified as responsible. They also serve as the foundation for all hazard analysis and risk assessment methods. They may operate as a filter and bias toward evaluating only particular occurrences and conditions, or they may broaden activities by demanding examination of aspects that are normally overlooked, because they alter the factors considered in any of these activities.

Most accident models consider accidents to be the outcome of a series of events. For losses induced by physical component failures and for relatively simple systems, such models function effectively. However, since WWII, the types of systems we've attempted to develop, as well as the context in which they've been built, have shifted. These changes, are pushing the limitations of current accident models and safety engineering procedures, necessitating new approaches.

The changes include-

- Increasing technological changes: Technology is evolving at a quicker rate than engineering strategies to deal with it are being developed. When outdated technology are replaced with new ones, lessons accumulated through ages about designing to prevent accidents may be forgotten or rendered worthless.
- Variants of Hazards: The most prevalent accident models are founded on the premise that accidents are caused by an uncontrolled and unwanted release of energy or an interruption in the usual flow of energy. Our growing reliance on information systems, however, raises the risk of data loss or incorrect data, which might result in unacceptably large physical, scientific, or financial losses.
- Decrease in Single-accident tolerance: As the cost and potential destructiveness of the systems we design rises, so do the losses resulting from accidents. Our new scientific and technological discoveries have not only created new or increased hazards (such as radiation exposure and chemical pollution), but they have also provided the means to harm an increasing number of people as the scale of our systems grows, as well as to negatively impact future generations through pollution and genetic damage. In an age when a spacecraft, for example, might take ten years and cost a billion dollars to develop, financial losses and wasted opportunities for scientific advancement are on the rise. Accident lessons must be augmented with a greater emphasis on preventing the first one.
- Changing Nature of Accidents: While digital technology has ushered in a quiet revolution in most sectors of engineering, methodologies in system engineering and system safety engineering have lagged behind. New "failure modes" introduced by digital systems are altering the character of accidents.
- Changing public and regulatory awareness of safety: The duty for safety is moving from the individual to the government in our increasingly complicated and interconnected society framework. Individuals no longer have control over the hazards they face, and they are asking that the government take a stronger role in regulating behavior through laws and other types of supervision and regulation. As firms face rising time-to-market and financial challenges, the government will be forced to step in to offer the protection that the public expects. The alternative is for people and groups to appeal to the courts for protection, which might have even worse consequences, such as restricting innovation unnecessarily due to fear of legal action.
- ncrease in complexity and coupling: The majority of the elements of complexity, particularly interactive complexity, are rising in the systems we are constructing.

We're creating systems with possible interactions between components that can't be completely planned, understood, predicted, or avoided. Some systems are so complicated that they are beyond the comprehension of all but a few specialists, and even they have only a partial knowledge of their possible behavior. Software has helped us implement more integrated, multi-loop control in systems with a large number of dynamically interacting components, where tight coupling allows disruptions or dysfunctional interactions in one part of the system to have far-reaching rippling effects.

– Complexity in relationship between Humans and Automation: Humans are increasingly sharing system control with automation and rising into higher-level decision-making roles, with automation carrying out the choices. These changes are leading to various new types of human errors.

These developments are putting our accident models, as well as the accident prevention and risk assessment methodologies that are based on them, to the test. There is a need for new paradigms.

An Accident Model Based on Systems Theory Systems-Theoretic Accident Model and Processes (STAMP) is a new conception of safety in which accidents occur when external disturbances, component failures, or dysfunctional interactions among system components are not adequately handled by the control system. The hypothesis underlying STAMP is that system theory is a useful way to analyze accidents. In this framework, understanding why an accident occurred requires determining why the control structure was ineffective. Safety then can be viewed as a control problem, and safety is managed by a control structure embedded in an adaptive socio-technical system. A control structure is designed to enforce constraints on system development and system operation that result in safe behavior.

In STAMP, systems are viewed as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control. The original design must not only enforce appropriate constraints on behavior to ensure safe operation, but the system must continue to operate safely as changes occur. Safety management is no longer simply about preventing component failure events. Instead, it is defined as a continuous control task to impose the constraints necessary to limit system behavior to safe changes and adaptations. Accidents can be understood in terms of why the controls that were in place did not prevent or detect mal-adaptive changes.

In STAMP, accidents are defined in terms of violations of safety constraints, which may result from

1. system component failure(s),
2. environmental disturbances, and
3. dysfunctional interactions among components

Constraints, control loops and process models, and degrees of control are the fundamental principles of STAMP. Each of them is now detailed, followed by an accident factor categorization based on the new model and core systems theory ideas.

The Central Role of Constraints in System Safety Safety is an emergent feature in systems theory that occurs when system components interact with their surroundings. A collection of restrictions (control rules) connected to the behavior of the system components govern or enforce emergent features. Accidents are caused by a lack of suitable interaction restrictions. In STAMP, a constraint, not an event, is the most fundamental idea. Accidents are said to occur as a result of a lack of appropriate system limitations. The system engineer, also known as a system safety engineer, is responsible for identifying the design constraints that must be adhered to in order to maintain safety, as well as ensuring that the system design, which includes not only the physical but also the social and organizational aspects of the system, does so.