Online Fault Diagnosis System for an Autonomous Guided Vehicle

A thesis submitted to the Division of Graduate Studies and Advanced Research of the University of Cincinnati

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In the Department of Mechanical, Industrial and Nuclear Engineering
Of the College of Engineering
2000

by

Sampath Kanakaraju
B.E (Production Engineering), PSG College of Technology, Coimbatore
India, 1997

Thesis Advisor and Committee Chair: Dr. Ernest L. Hall
Table of Contents

Chapter 1   Introduction
  1.1  Objective ................................................................. 10
  1.2  Organization of the thesis.......................................... 11
Chapter 2   Literature Survey
  2.1  Methodologies of fault diagnosis system............ 12
      2.1.1  Fault Tree Analysis........................................ 12
      2.1.2  Reliability Analysis........................................ 12
      2.1.3  Robotic Self - Diagnostic methods............ 12
      2.1.4  Expert Systems............................................... 13
Chapter 3   Grouping the components of Bearcat II
  3.1  Power System............................................................ 16
  3.2  Vision System.............................................................. 17
  3.3  Mechanical System........................................................ 18
  3.4  Charging System............................................................ 18
  3.5  SONAR system............................................................... 19
Chapter 4   Database Design
  4.1  Why we need a database............................................ 20
  4.2  Benefits................................................................. 20
  4.3  Relational database terms........................................... 20
  4.4  Selecting a database development methodology 22
  4.5  High-level database development methodology.......... 23
  4.6  Database for online fault diagnosis system...... 25
<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Dynamic web application development</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Static web pages.............................. 28</td>
</tr>
<tr>
<td>5.2</td>
<td>What are dynamic web pages....................... 28</td>
</tr>
<tr>
<td>5.3</td>
<td>What are Active Server Pages..................... 28</td>
</tr>
<tr>
<td>5.4</td>
<td>How ASP works...................................... 29</td>
</tr>
<tr>
<td>5.5</td>
<td>General Structure of Active Server Pages........ 31</td>
</tr>
<tr>
<td>5.6</td>
<td>Active Server Pages object model................ 32</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Request object..................................... 34</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Response object..................................... 35</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Server Object....................................... 35</td>
</tr>
<tr>
<td>5.6.4</td>
<td>Application object.................................. 36</td>
</tr>
<tr>
<td>5.6.5</td>
<td>ObjectContext object.............................. 36</td>
</tr>
<tr>
<td>5.6.6</td>
<td>Session object...................................... 36</td>
</tr>
<tr>
<td>5.7</td>
<td>Active Database Component........................ 37</td>
</tr>
<tr>
<td>5.8</td>
<td>Active data object - Overview..................... 38</td>
</tr>
<tr>
<td>5.9</td>
<td>ADO object model................................... 39</td>
</tr>
<tr>
<td>5.9.1</td>
<td>Connection object.................................. 40</td>
</tr>
<tr>
<td>5.9.2</td>
<td>Command object..................................... 41</td>
</tr>
<tr>
<td>5.9.3</td>
<td>Recordset object................................... 41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Architecture and working of online fault diagnosis system</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>The Application architecture.............................. 43</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Client / Server Model..................................... 43</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Services and tiers......................................... 44</td>
</tr>
</tbody>
</table>
6.1.3 Two-Tier Client / Server architecture ............ 44
6.1.4 Three-Tier Client / Server architecture .......... 45
6.2 Application infrastructure .................................. 46
6.3 Architecture for Online fault diagnosis system .. 47
6.3.4 Client ................................................................ 49
6.3.5 Server ................................................................ 52
Chapter 7 Conclusion and future recommendations ....... 55
References ................................................................ 56

Appendix A (This contains the programs and scripts)

Appendix B (This contains the contents of the database table)
**ABSTRACT**

This work describes the development of an Online Fault Diagnosis System. Though designed apparently for fault diagnosis of the University of Cincinnati’s autonomous, unmanned vehicle Bearcat II, the online system can be configured to be adapted for any fault diagnosis application. Using a top-down approach the robot is sub-divided into different functional units [major units], such as the power unit, mechanical unit, vision guidance unit, the ultrasonic obstacle avoidance unit, and the charging unit. The technique of Potential Failure Mode and Effects Analysis (PFMEA) is based as standard to analyze faults, potential effects, potential causes and solutions. The information then obtained is designed as a database table, which could be handled by any database system like Microsoft Access, Microsoft SQL server or Oracle. This formatted information is then made available to the team via a web client interface, which maybe any web browser such as Internet Explorer or Netscape Navigator. The online system is developed on the technology of Active Server Pages, which provides the functionality of active web content. The online system could be accessed anytime from any part of the world with a web browser and the connection to the Internet. The provision is made such that user authenticity is obtained before allowing the user to access the data.

The Fault Diagnosis System thus provides a handy trouble-shooting tool that cuts down the time involved in diagnosing failures in the complex system like robots, which may otherwise need experts from various disciplines. This system could be of great use in diagnosing the robot during the stressful phase of the contest.
Acknowledgements

I am grateful to my advisor Dr. Ernest Hall for giving me an opportunity to work on this new concept. He has been the constant source of support for gathering relevant resources and information to complete this thesis.

I am thankful to the team members of the robotic team for providing their valuable suggestions and supporting my work by adding their experiences which formed the primary source of the database. I would take this occasion to appreciate Jaiganesh for his support in helping me to use Photoshop.

I wish to thank my committee members Dr. Richard L. Shell and Dr. Ronald L. Huston for providing their invaluable suggestions and making my work more vivid and valid.

I take this opportunity to thank the entire faculty and staffs of MINE department for helping in pursue my masters at University of Cincinnati.

Finally, I give immeasurable thanks to my family –parents and sister- who have been everything in my life to achieve this work.
List of Figures
Chapter 3
3.1 Elements of power system.............................................. 17
3.2 Elements of vision system.............................................. 17
3.3 Elements of mechanical system..................................... 18
3.4 Elements of charging system....................................... 18
3.5 Elements of SONAR system.......................................... 19

Chapter 4
4.1 Unary relationship..................................................... 21
4.2 Binary relationship.................................................... 21
4.3 Ternary relationship.................................................. 22
4.4 Diagrammatic representation of database design............. 24
4.5 Database table showing sample data............................ 27

Chapter 5
5.1 Browser requesting an ASP page................................. 29
5.1.1 Processed ASP sent back to web browser.................. 30
5.1.2 Browser receives the requested page......................... 30
5.2 More closer look at the web browser........................... 31
5.3 General Script syntax in a typical ASP page.................. 32
5.4 Object model of ASP.................................................. 33
5.5 Overview of ADO object............................................. 38
5.6 ADO object model.................................................... 39

Chapter 6
6.1 Two-Tier model....................................................... 44
6.2 Three-Tier model...................................................... 45
6.3 Architecture of online fault diagnosis system.............. 48
6.4 User authentication.................................................. 49
6.5.1 Major unit: power system.................................... 51
6.5.2 Major unit: SONAR system................................... 51
6.6 Final output showing fault diagnosis data for power and fuse 52
Chapter 1

Introduction

The ever-growing demand for excellent operational efficiency and safety in the design and development of Automated Guided Vehicle (AGV) leads to the development of diagnostic strategies that could cover the major potential faults of the AGV.

Fault diagnosis, is an answer to the well-known fact – If anything can go wrong, it will – coined by Murphy. How well a system maybe designed, developed, and tested, it still may have breakdowns and needs routine maintenance. Routine maintenance aided by the fault diagnosis system, can help assure the safety, reliability and quality of product and processes. Fault diagnosis as an engineering discipline has spread into the engineering design and development practices of general aerospace and hi-tech industries of the present. Robotics, one such hi-tech industry needs a sophisticated system for analyzing the failures, storing the related information in an integrated repository and retrieving the same via standard user-friendly interface.

Failure is a fundamental concept of any reliability analysis. According to accepted standards failure is defined as ‘the termination of the ability of an item to perform a required function.’ Failures are classified into failure modes. The failure mode concept is generally recognized as a useful and necessary concept in failure analysis. Failure mode is defined as ‘the effect by which a failure is observed on a failed item.’
Many systems in use today are not performing as intended, nor are they cost effective in terms of their operation and support. Benjamin, S. Blanchard [1] has analyzed that a large percentage of the total cost of doing business is due to maintenance-related activities in the factory. Further, these costs are likely to increase even more in the future with the added complexities of factory equipment through the introduction of new technologies, automation, the use of robots, and so on.

Automated Guided Vehicles (AGV) play a major role in the automation of many manufacturing processes. These AGVs may be subject to failures during their normal course of operation. The unavailability of an expert no doubt contributes to the downtime of the AGV. Rather than actually fixing the failure, the major portion of maintenance is effected by the process of fault fixing (fault diagnosis). The maintenance process consists of two discrete steps: Diagnosis and Repair. In most cases, Repair time associated with a failure is constant and independent of time. However, the diagnosis phase can be optimized.

To aid the users, commercially available robots come with Maintenance and Trouble-shooting manuals. These provide instructions on maintenance procedures as well as trouble-shooting hints for most commonly observed problems. However, manuals tend to be too detailed and hence most technicians are reluctant to use them under the pressure of the productivity targets.
Considering the competitive nature of business, there exists a tremendous potential for utilizing a non-human expert that would be readily available and accurately diagnose the problem with the least loss of time. A tool, that allows to quickly and easily analyzing the potential failure modes of a system and the resulting effects of those failures could greatly reduces the cost of maintenance.

Fault diagnosis system provides a systematic review of the components, assemblies and subsystems of a product to identify single point failures and the causes and effects of such failures. It identifies and tabulates the potential modes by which equipment or a system might fail and the consequences such a failure would have on the equipment or system being studied. The fault diagnosis system determines the criticality of each failure mode and provides the necessary remedies.

Enterprise computing is undergoing another revolution as Internet and Web-based technologies dramatically change and simplify the computing architectures within and between companies. I believe that this revolution, while impacting many types of software applications, will be most powerful in any Intranet applications where in entire organization can access the integrated information as and when needed. An online dynamic web application would ease the job of maintaining documents, improve accessibility and provide faster approach to problem solving. The future of electronic computing will be interactive, dynamic and real-time - within the enterprise and between enterprises.
The work presented in this thesis concentrates on the development of such a dynamic web-based Fault Diagnostic System. Though designed apparently for University of Cincinnati’s - the Robotics Research - Autonomous Unmanned Vehicle Bearcat II, it has the scalability to be implemented for any commercial applications as well.

1.1 Objective

The online fault diagnosis system is aimed at providing online fault diagnostic support to the University of Cincinnati, robotics team. This web-based system is targeted at analyzing the faults, storing the information in the form of relational database and then retrieving the desired solution set for the faults that pop-up during the operation of Bearcat II in the AUVS (Autonomous Unmanned Vehicle Systems) competition. This AUVS competition is held annually and UC Robotics team has been a contender for the past 8 years. The University of Cincinnati has so far used 2 versions of their AGVs. The Bearcat II Robot weighs approximately 450 lbs., and is 2 feet wide and 4 feet long. In the 2000 competition, the track was 10 feet wide. The course was filled with obstacles, and challenging barriers like a ramp with 15 degrees inclination, an asphalt segment, a sand segment etc. Due to these challenges, the robot had a chance of potential failures. Hence to overcome the competition pressure, this online fault diagnosis system delivers a user-friendly web-application for the most recent information on solving those problems. All these objectives are implemented as dynamic web-application that could be accessed anytime, anywhere with a PC and Internet connection. Presently the system is designed for the team use alone and this is ensured with secure login.
Features of the Online Fault Diagnosis

- A web user interface implemented in a web browser
- Provides functionality for user authentication
- Provides options to choose between major units and its sub-classifications
- Provides a comprehensive data table showing the information for fault diagnosis and fault fixing
- Provides round-the-clock assistance to the Robotics team for fixing the various faults encountered during the operation of the Bearcat II

1.2 Organization of the Thesis

I have organized my documentation in separate chapters, which differentiates the ideas and technologies that I have used to develop the web application.

Chapter 2 discusses the various fault diagnosis systems that are available in the market and their brief outline. Chapter 3 describes the idea for grouping the components of the robot into major units and its sub units. Chapter 4 deals with the database design for storing the information about various faults and their solutions. Chapter 5 deals with the development of the web-application, stresses on the active server pages (ASP) technology and design of the web pages. The working of the web application is explained in chapter 6, while the final outputs and future recommendations are discussed in chapter 7.

The appendix A shows the programming codes and appendix B shows database tables used to obtain the results.
Chapter 2

Literature Survey

The fault diagnosis system has been based on the following traditional and modern methodologies for predicting the fault and fixing them effectively and efficiently.

2.1 Methodologies of fault diagnosis system [2]

Traditional methodologies include

- Fault Tree Analysis
- Reliability block diagrams.
- Cause Consequence Analysis

Modern techniques that have been formulated for the fault diagnosis are

- Expert Systems
- Robotic Self-Diagnosis and Self-Repair

2.1.1 Fault Tree Analysis:

The Fault Tree Analysis, also known as the Fault Propagation Tree, is a top-down technique that graphically represents the equipment state as a function of the component.

2.1.2 Reliability Analysis:

Reliability analysis is based on the slight assumption that the system under consideration is made up of components with independent failure behavior.

2.1.3 Robotic Self-Diagnostics and Self-Repair

“The technique of robotic self-diagnosis and repair lends itself to taking care of the most commonly observed faults that have fully automated measurement and repair tools. For e.g. Pattern recognition - based diagnostics can be used to locate the source of
vibration on a robot arm by moving the arm along different axes one at a time to better distinguish between inherent and induced vibrations.

2.1.4 Expert Systems

Expert system is a software system, which displays expert behavior in a particular domain of application. A human expert is a rare commodity. An expert system consists of principally two parts - an inference engine and a knowledge base. Commercially, available software tools, known generically as expert system shells, facilitate the development of expert systems. An expert system shell contains an inference engine and software to build and incorporate knowledge base.

Expert systems have been developed for various areas of applications including fault diagnosis and recovery for mechanical, electrical and electronic equipment as well commercial robots.

Fault Diagnosis has been done in almost all areas of engineering and development. Let us examine the various expert systems for the fault diagnosis. The fault prediction and fault compensation functions have been applied to hydraulic systems [3]. It is seen that the hydraulic systems exhibit a sensitive dynamic behavior, which is highly influenced by leakages. Considering the complexity of these systems and high working pressures, an automatic on-line fault prediction and fault compensation are necessary for these systems. This fault prediction and fault compensation functions are incorporated in a real time expert system that diagnoses faults in a hydraulic system.
This system incorporates a data acquisition and control module, an expert system, and a mathematical model which together is interfaced with the actual hydraulic system to diagnose the fault and fix the same.

An expert system for engine fault diagnosis known as EXEDS[^4](Expert Engine Diagnosis System) is another good work for the fault prediction and repair. This system has been developed using KnowledgePro, an expert system tool, and run on PC. The purpose of the prototype is to assist auto mechanics in fault diagnosis of engines by providing systematic and step-by-step analysis of failure symptoms and offering maintenance or service advice. This system replaces the Diagnostic trouble codes (DTCs) which although pinpoint the faults, is focused only for experts who have the know-how. Expert Systems, a major part of AI technology, can capture the knowledge of an expert in a particular area, computerize it then transfer it to others.

The objective of this EXEDS is to help mechanics and car drivers in systematic diagnosis of their daily engine failures. This system is intended to provide low-level expert assistance in fixing engine malfunctions and consists of rules that progressively lead to possible causes, together with proposed courses of action to cure the problem. This expert system is structured to assist the users with a neat and standard GUI (Graphical User Interface) interface with menus. This system gives the user the following functionality like

- Show the list of possible symptoms
- Serve as information inputs in response to the system queries
- Provide on-line explanations
- Provide diagnosis result(s)
- Provide recommended actions to cure the problems
- Provide graphical illustrations of engine systems

With this background information, the online diagnosis system poses itself to be a unique system by its own. It takes advantage of the latest cutting edge technologies for implementing an interactive web based active information content.
Chapter 3

Grouping the components of Bearcat II

In order to obtain the failure modes for individual subsystems as well as the entire robot, a thorough study and grouping of the components is essential. This chapter outlines system description of Bearcat II, by grouping the components into major units and a sub classification of the major units. For the robot to operate efficiently at the contest not only should the individual subsystems work satisfactorily but they also should work in tandem. This thorough analysis has been tremendously useful in understanding the architectural, functional and behavioral details of the system.

According to the functionality the robot has been categorized into major units as

- Power System
- Vision System
- Mechanical System
- Charging System
- SONAR system

3.1 Power System

The Power system is further classified into seven finite elements. Power system of the robot consists of the components that help to power the electrical components of the robot. The heart of the power system being the Solenoid acts as a switch, which can be controlled to cut off the power during emergency.

The database consists of the Power System as the attribute value in the major unit column.
3.2 Vision System

The vision system defines the components that assist in the line following of Bearcat II. It consists of two JVC CCD cameras mounted on either side of the robot, such that a clear line tracking can be achieved. The sub classification the vision system yields the following components for fault diagnosis.
3.3 Mechanical System:

The mechanical system as a whole serves as steering control for the robot. The components include 40:1 reduction gearbox, two pairs of flexible couplings, two 36 volts servomotors and two sets of wheels with shafts, couplings and keys. The computer through Galil motion controller controls the servomotors.

3.4 Charging System

The charging system helps in charging the three 36 volt batteries for providing the power to the various components of the Bearcat II. The charger charges the batteries in 12 hours.
3.5 SONAR System

SONAR system is used for obstacle avoidance, apart from the vision system. For accurate path navigation it is essential that the obstacle avoidance system function properly. This system consists of a rotating transducer, which makes mirror stops on either side of the centerline for obstacle avoidance.

The major units are then sub classified into specific units for the purpose of building the relational database, which holds the information about the fault analysis of each component. Each specific unit is then analyzed for its failures and the required actions are then recorded which becomes an important source of information during the break down of the robot.
Chapter 3

Database Design

4.1 Why we need a database?

In any business, certain pieces of information about customers, products, prices, financial status and so on must be saved. These pieces of information are called data. Information philosophers say that data is just data until it is organized in a meaningful way, at which point it becomes “information”. The three main things you do with data is acquire it, store it and retrieve it. A relational database management system (often called an RDBMS for short) gives you a way of doing these tasks in an understandable and reasonably uncomplicated way.

4.2 Benefits

Using proven database development methodologies on projects can reduce the complexity that is inherent in constructing relational database solutions. Methodologies are an excellent example of best practices and project lessons. Use of methodologies, therefore, reduces risk on development projects. Methodologies define activities and deliverables that are constructed in projects that were successful.

4.3 Relational Database Terms

Because relational technology is a mature technology with many practical applications, it is useful to consider some of the important terms that pertain to the relational model. Many of these terms are straightforward and generally unambiguous, whereas some terms have specific definitions that are not always understood.
A data set represented in the form of a table containing columns and rows is called a relation. The columns are called attributes, and the rows are called tuples.

The relational model views data in a tabular format. Here is an example.

<table>
<thead>
<tr>
<th>PART #</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>UMBRELLA STAND</td>
</tr>
<tr>
<td>32</td>
<td>SPITTOON</td>
</tr>
<tr>
<td>48</td>
<td>BUGGY WHIP</td>
</tr>
</tbody>
</table>

Relational databases require duplicate data among tables but don't permit duplication of data within tables. The relationships between the tables are categorized under three different types. The three different relations are described as follows:

- **Unary Relationship**[^5]

Unary Relationship is a relationship between the instances of a single entity type. This can be of one-to-one relation or one-to-many relation.

![Fig 4.1: Unary Relationship](image)

- **Binary Relationship**[^5]

![Fig 4.2: Binary Relationship](image)
Binary Relationship is a relationship between the instances of two entity types and is the most common type of relationship encountered in data modeling. This relationship has three degree of relationship, one-to-one, one-to-many, and many-to-many.

- **Ternary Relationship**[^5]

A ternary relationship is a simultaneous relationship among the instances of three entity types.

![Fig 4.3: Ternary Relationship](image)

### 4.4 Selecting A Database Development Methodology

The following list identifies some of the features that should be included in any database development methodology that is being considered for deployment in an organization.

- **Techniques:** Many popular development methodologies support a combination of techniques to streamline development of deliverables. The traditional waterfall approach involves producing deliverables in a sequential fashion. A rapid application development (RAD) approach has gained popularity in the past 10 years. RAD produces deliverables in a much smaller timeframe than the older...
waterfall approach. Iteration and prototyping are cornerstones of most RAD approaches.

- **Proven Success**: One of the surest ways of selecting a suitable methodology is to check the references of similar organizations that have used it successfully on development projects.

- **Templates/Deliverables**: Reusable templates and deliverables are a good source of best practices that provide the means for quick starting development projects.

### 4.5 High-Level Database Development Methodology

This section defines a high-level methodology for database development. The various phases that are involved in the database design are as follows:

- **Define Business Requirements**: Business requirements are captured for any system development effort. The requirements also should be used to build the logical data model. They will feed such things as the number of entities, attribute names, and types of data stored in each attribute.

- **Build Logical Data Model**: The logical data model is built iteratively. The first view usually is done at a high level, beginning with a subject area or conceptual data model. Subsequent levels contain more detail. The process of normalization also is applied at this stage.

- **Verify the Data Model**: The logical data model is validated iteratively with users, the fields of the user interface, and process models.
- **Build the Physical Data Model**: The logical data model is converted to a physical data model based on the specific database that is used. The physical data model will vary with the choice of database products and tools. The physical data model also contains such objects as indexes, foreign keys, triggers, views, and user-defined data types.

- **Refine the Data Model**: The physical data model is refined continuously as more information becomes available, and the results of stress testing and benchmarking become available to the database development team.

- **Populate the Data**: After the database structure is established and the database is created, it is necessary to populate the database. This can be done through data scripts, applications, or data conversions.

![Diagrammatic Representation of Database Design](image-url)
4.6 Database for online fault diagnosis system

The database was first created in the form of Microsoft Excel data and then imported into Microsoft Access database by utilizing the import data facility in the Microsoft Access.

The database is named as FMEA.mbd in Microsoft Access. This database consists of one table named as “pfmea” with thirty-eight records and eight columns.

The attribute definition of the table is shown as below.

<table>
<thead>
<tr>
<th>Seq_ID</th>
<th>Failure No</th>
<th>Major Unit</th>
<th>Specific Unit</th>
<th>Failure Mode</th>
<th>Potential Effects</th>
<th>Potential Causes</th>
<th>Solutions</th>
</tr>
</thead>
</table>

The Seq_ID contains values of the sequence of records and it forms the primary key column of the table. Failure number denotes the failure and is based on the major system. An instance value of Failure number is PFMEA_POU_1, which denotes Power Unit. Similarly for each major system the Failure number is designated to denote that particular major unit. The major unit consists of the major units that were discussed earlier (Power System, Mechanical System, SONAR system, Vision System, Charging System). The specific unit stores the sub-classification of the major system. Failure mode, potential effects, potential causes and solutions pertain to that particular major unit and the specific unit.

Failure modes for the major system have been divided so that it would match the sub-classified units. Each failure mode is then analyzed for its potential effects depending on how the failed component is related to the rest of the components in the system. It is
seen that for the same sub-classified unit there maybe more than one potential mode of failure and hence this accounted as unique set of records in the database.

  Once potential effect is known, then potential causes are analyzed so that the source of problem could be targeted and concentrated for fixing up the problems. The solution that are provided for each set of failure modes are not just theoretical knowledgebase but a comprehensive collection of experienced solutions gained by testing the Bearcat II. The failure number is assigned based on the type of the major system, which it represents. For example the failure number for the power system is assigned as PFMEA_POU whose middle set of literals denote the system for which it is assigned.

<table>
<thead>
<tr>
<th>Failure Number</th>
<th>Major System which it denotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFMEA_POU_1 to 9</td>
<td>Power System</td>
</tr>
<tr>
<td>PFMEA_OAU_1 to 9</td>
<td>Obstacle Avoidance System</td>
</tr>
<tr>
<td>PFMEA_MEU_1 to 12</td>
<td>Mechanical System</td>
</tr>
<tr>
<td>PFMEA_VSU_1 to 7</td>
<td>Vision System</td>
</tr>
<tr>
<td>PFMEA_CHU_1</td>
<td>Charging System</td>
</tr>
</tbody>
</table>

This relation is used in retrieving the information from the database to the web application client. The database is queried using SQL (Structured Query Language).

  This database forms the central repository for accessing the information pertaining to the failures for the particular components and their potential effects and causes. The solutions provided for each failure has been obtained over the time with experience and also widely known facts.
An instance of the database table shows the various values that it holds. This information could be upgraded to any database system with third party software’s available in the markets.

<table>
<thead>
<tr>
<th>Seq_ID</th>
<th>Failure NO</th>
<th>MajorSystems</th>
<th>SpecificUnit</th>
<th>Failure Modes</th>
<th>Potential Effects</th>
<th>Potential Causes</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>PFMEA_POU_1</td>
<td>Power System</td>
<td>Battery</td>
<td>No Main power</td>
<td>Robot does not start</td>
<td>Battery down</td>
<td>Check voltage by the voltmeter which is mounted on the side panel</td>
</tr>
<tr>
<td>1001</td>
<td>PFMEA_POU_2</td>
<td>Power System</td>
<td>Battery</td>
<td>Low battery power</td>
<td>Robot stops at jog greater than 10000</td>
<td>Battery down</td>
<td>Check voltage by the voltmeter which is mounted on the side panel</td>
</tr>
<tr>
<td>1002</td>
<td>PFMEA_POU_3</td>
<td>Power System</td>
<td>Fuse</td>
<td>No Power</td>
<td>Camera &amp; ISCAN tracker do not work</td>
<td>Fuse blown</td>
<td>Check and replace fuse</td>
</tr>
<tr>
<td>1003</td>
<td>PFMEA_POU_4</td>
<td>Power System</td>
<td>Battery</td>
<td>Low power</td>
<td>Computer Reboots</td>
<td>Battery down</td>
<td>Check voltage by the voltmeter which is mounted on the side panel</td>
</tr>
<tr>
<td>1004</td>
<td>PFMEA_POU_5</td>
<td>Power System</td>
<td>Inverter</td>
<td>No Power</td>
<td>Computer doesn't power on</td>
<td>Loose connection</td>
<td>Check connection at the inverter</td>
</tr>
<tr>
<td>1005</td>
<td>PFMEA_POU_6</td>
<td>Power System</td>
<td>Main Switch</td>
<td>No power</td>
<td>Computer doesn't power on</td>
<td>Main switch in off position</td>
<td>Switch the main on</td>
</tr>
</tbody>
</table>

The next chapter deals the concepts of development of web application. Deals with various architectures that form the basis for e-development.
Chapter 4

Dynamic Web application development

5.1 Static Web Pages

When a web user clicks a URL or types a web address in the address box of the browser, the browser from the server requests a page. This is just a file that resides in the server and hence is transported back by the appropriate Internet Protocols. The user sees the contents as a rendered HTML page, but the source is the same as the file that is stored on the server’s disk.

5.2 What are Dynamic Web pages?

The big attraction with active server pages, of course, is the ability to include script directly in the file that’s referenced by the browser, and thereby creates dynamic pages.

Dynamic web pages are pages that interact with the user. Dynamic pages change their appearance or behavior based on user feedback or other inputs. Examples of dynamic web pages include amortization schedules, weather reporting, stock quoting services, user authentication and logins, data entry, and more.

5.3 What are Active Server Pages?

- ASP is the latest technology from Microsoft that allows web site developers and service providers to make their pages even more interactive and dynamic.
- Like any technology, Active Server Pages (ASP) has their place in business and software development. The strength of ASP is the ability to dynamically build a
pure HTML web page based on a user's input and profile, the time and location the user accesses the page or the type of browser and/or operating system that is running on the user's computer.

- ASP is a way of building secure transaction, server-based applications and web sites, to allow people to obtain information based on their requirements. As far as interactive web server technologies go, Active Server Pages is probably the hottest product ever for building multitier Internet and intranet applications.

5.4 How ASP works?

Whenever a user requests a file whose extension is .ASP, the ASP ISAPI (Internet Server Application Programming Interface) filter handles the interpretation. ASP then loads any required scripting language interpreter DLLs (Dynamic Link Libraries – A program, which is written for specific applications and stored in the server/workstation as DLLs) into memory, executes any server-side code found in the Active Server Page, and passes the resulting HTML to the web server, which then sends it to the requesting browser.

![Fig 5.1A: Browser requesting an ASP page](image)
More detailed look at the server side processing of the active server pages gives a better understanding of the technology. When a user requests a page, either by typing the URL in the web browser address box or clicks a link/URL, this request is sent to the web server (Internet Information Server – IIS - a software program provided by a Microsoft for NT servers). The server then recognizes the .ASP extension and invokes the ASP DLL, which then launches the interpreter based on the script that is written in the ASP file. The appropriate interpreter then processes the page and then the dynamic page is then sent to the server, which then adds a HTTP wrapper for transporting over the Internet. The
request page is then sent to the client where the HTTP is decoded and then displayed in the client browser.

Where can you find Active Server Pages?

Active Server Pages is part of Microsoft Internet Information Server (IIS), and thereby part of the windows NT4 server operating system.

5.5 General Structure of Active Server Page

The server must be able to differentiate between script that is to be executed while the dynamic page is being created, and script that is to be sent back to the browser as it. ASP uses two tags for placing the script, so that the server can understand it where the script has to be executed. `<% %>` tags are used for the placing those scripts that are processed in the server, while `<Script> </Script>` tags are used for placing those scripts that are to be processed by the client browser. However the `<% %>` tags don’t allow functions to be placed. Hence ASP provides another attribute called `RUNAT`, for placing the functions.
Lets have a look at the general tags that are used in ASP in the illustration as shown in the figure.

![Diagram of ASP script syntax](image)

**Fig 5.3: General Script syntax in a typical ASP page**

5.6 *Active Server Pages Object Model*[^6]

Scripting alone doesn’t make Active Server Pages. The scripts that use the rich collection of objects that are part of the Active Server Pages environment, by calling their methods and setting their properties provide a new outlook to the web sites. There are five main built-in objects provided automatically by Active Server Pages, which we can

[^6]: Reference to the section or page number.
use in the interactions between the client’s browser and our server. These objects form a hierarchy. At the top of the pile is the server object. The methods and properties of this object offer general utility functions that we can use throughout our scripts. The server object represents the environment in which our pages run, and the remaining four objects fit together to make up an Active Server Application.

These objects are

- Application
- Session
- Request
- Response
Each object has its own methods and properties, which could be utilized in making the web content, an active content. These objects are pre built with predefined methods, which give the flexibility to implement those methods without rewriting the entire code. These collections of objects could be represented in the form of diagram as shown below.

Let us have a look at the brief description of each of the object collection and a detailed description of those objects and collection that I have used in my web application development.

### 5.6.1 The Request Object

The Request Object provides us with all the information about the user’s request to our site, or application. The request object has five different collections such as:

- Cookies
- Form
- QueryString
- ServerVariables
- ClientCertificate

A collection is data structure, rather like an array, which can store values by linking each one to a unique key. If the key is known, then the values can be set and retrieved from the collections.

The Form collection of the request object is extensively used in the application. While using the form collection the data is wrapped in the HTTP header and then transferred to the Active Server Pages. Since it’s a collection, we can retrieve values from it, and iterate through it. A typical way of getting the information from the form collection is using the following syntax: `StrName = request.form (“txtName”).`
StrName is the variable name used in the script, txtName is the name of the item given during the design time of the HTML forms.

5.6.2 Response Object:

The second main object in the hierarchy is the response object. The response object is responsible for sending output from the server to the requesting client. The three most common response object methods are Write, Redirect and response methods. The response object uses the Write method to send HTTP output to the browser. The redirect method provides the capability to redirect the client browser to a different URL. The response object enables a Cookie collection to set cookie values on the client’s browser. While the request object provides all means to get the data from the client to the server, the response object provides interfaces to pass information from server to the client. The collections of the response object can be grouped under the following functionality.

<table>
<thead>
<tr>
<th>Description</th>
<th>Response Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inserting information into a page</td>
<td>Write, BinaryWrite</td>
</tr>
<tr>
<td>Sending the cookies to the browser</td>
<td>Cookies</td>
</tr>
<tr>
<td>Redirecting the browser</td>
<td>Redirect</td>
</tr>
<tr>
<td>Buffering the page as it is created</td>
<td>Buffer, Flush, Clear, End</td>
</tr>
<tr>
<td>Setting the properties of a page</td>
<td>Expires, ExpiresAbsolute, ContentType</td>
</tr>
<tr>
<td></td>
<td>AddHeader, Status</td>
</tr>
</tbody>
</table>

5.6.3 Server Object

This is the base of the object model hierarchy. This object provides some basic properties and methods that are used in almost every Active Server Page that we create.

The interface provided by the server object can be tabulated as follows

<table>
<thead>
<tr>
<th>Property/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScriptTimeOut property</td>
<td>Amount of time a script can run, before an error occurs</td>
</tr>
<tr>
<td>CreateObject</td>
<td>Creates an instance of an object or server component</td>
</tr>
</tbody>
</table>
5.6.4 Application Object

The Application object is used to manage tall information in the ASP application, the information can be accessed and passes between different users in the application. Because multiple people can try to simultaneously change the application variables, the Application object enables locking and unlocking of its variables. When a variable is locked, other users cannot modify the properties of an Application object.

<table>
<thead>
<tr>
<th>Method/Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock Method</td>
<td>Prevents other clients from modifying application properties</td>
</tr>
<tr>
<td>Unlock method</td>
<td>Allows other clients to modify application properties</td>
</tr>
<tr>
<td>OnStart event</td>
<td>Occurs when the application first starts</td>
</tr>
<tr>
<td>OnEnd event</td>
<td>Occurs when the web server is stopped</td>
</tr>
</tbody>
</table>

5.6.5 ObjectContext Object

ASP to control transaction processing using the Microsoft Transaction Server (MTS) uses the ObjectContext object. This direct processing within MTS environment enables Active Server Pages to explicitly control, commit, and rollback features of objects managed by MTS.

5.6.6 Session Object

Before talking about the session object, first we need to know the concept of SCOPE of variables, whether we need to have them as local variables or global variables. If we need to share the information between different pages, but, not necessarily between different clients then we can make use of the session object. Each client who request information from our server is assigned a Session Object. A
Session is created when the client first makes a document request and destroyed, by default, twenty minutes after their last request was received.

The interface to the session object is as shown below:

<table>
<thead>
<tr>
<th>Interface Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SessionID</td>
<td>Returns the session identification for this user</td>
</tr>
<tr>
<td>Timeout property</td>
<td>The timeout period (minutes) for sessions in this application</td>
</tr>
<tr>
<td>Abandon method</td>
<td>Destroys a session</td>
</tr>
<tr>
<td>OnStart event</td>
<td>Occurs when a user first requests a page in the application</td>
</tr>
<tr>
<td>OnEnd event</td>
<td>Occurs when the session ends</td>
</tr>
</tbody>
</table>

These objects are like template libraries, which can be customized for the necessary requirements. These objects form the crux of the Active Server Pages application model. Any web application built for active content using Active Server Pages most likely utilize these functionalities for easy building of the pages. Knowing the various objects provided by ASP lets take a look at how the ASP and database are work together to create an active web content.

5.7 The Active Database Component

The Active Server Pages is supplied with a component called the Active Database Component. This collection is very much necessary for database connectivity and accessing the database. This provides with a whole hierarchy of objects, collectively known as Active Data Object (ADO) – which is the link between the web pages and the database of any kind. To give a better understanding of the database connectivity, its necessary to know what ADO is, the structure of ADO and brief description of the ADO methods to manipulate the database.
5.8 Active Data Object – Overview

The Active Data Object is really a connection mechanism that provides access to data of all types. The most common usage of ADO will be accessing a relational database from a client interface. The accessing mechanism is implemented in the form of VBScript or Jscript, thereby giving the flexibility to develop web-based client-server applications. ADO is designed to interface with databases through Open Database Connectivity (ODBC), which allows to access not only ‘proper’ databases like Oracle, Access, SQL server, but also spreadsheet files and text files.

However ADO is built on another technology called OLE-DB. This provides a uniform data interface through the methods and properties it maintains internally. ODBC is one kind of data ADO and ASP can access, but it isn’t the only kind. Since ADO is built upon another layer, OLE-DB, we also need to be able to specify another layer of connectivity in our links to a data source.
Hence it's no longer sufficient to think of just the driver software (such as ODBC), but we need to consider what the actual provider of data is. ODBC is just the most popular of the OLE-DB providers.

### 5.9 ADO Object Model

ADO is altogether the easiest way to access a database than those included in Microsoft Access, and other object-based applications. For instance, the hierarchy has only three main objects, Connection, Recordset, and Command, and several collections of subsidiary objects, parameters, properties, and Errors.

![Fig 5.6: ADO object model](image)

Using the Connection object we can establish an active connection between the database and our program that accesses the data. To obtain data from the data source,
execute SQL queries, or manipulate the data directly, we can use the Command object. Recordset object provides the functionality to retrieve data from the SQL queries, stored procedure, or by opening a table.

The Connection, Command and Recordset objects have their own collection of properties as we have seen for the ASP objects. The Connection object also has a collection of error objects, which contains information about the errors that occur while using the objects. Let us go over the brief description of the main three objects of the ADO objects.

### 5.9.1 The Connection Object

Defining a connection to a database is a straightforward process. The first step is to create an instance of the Connection object. Here is how a Connection is setup with the database: Set oConn = Server.CreateObject(“ADODB.Connection”)

By using the CreateObject method of the Server object, we can instantiate a variable to hold a reference to the newly created Connection object, just as we did with other components. The Connection could have different levels of scope, meaning it can either at the session level or application level depending on the application needs.

The important methods and the properties of the Connection object are listed below:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Opens a new connection to a data source</td>
</tr>
<tr>
<td>Close</td>
<td>Closes an existing open connection</td>
</tr>
<tr>
<td>Execute</td>
<td>Executes a query, SQL statement or stored procedure</td>
</tr>
<tr>
<td>Properties</td>
<td>Description</td>
</tr>
</tbody>
</table>
The above listed methods and properties are the ones that are mostly used in the online fault diagnosis system.

A general rule of thumb to connect to the database would be

- Declare a connection variable of type ADODB.connection
- Open the connection using open method
- Execute the connection string to fetch the data from the database
- Close the connection

### 5.9.2 The Command Object

The command object is a specific command executed against the data source, and it returns a recordset. The command object is optional, but is particularly valuable when dealing with stored procedures. The command object’s most important method is the ActiveConnection method, which specifies the connection object the command object, belongs to.

### 5.9.3 The Recordset Object

The Recordset object is the key of ADO. It is the object that is more often used than any other object in the ADO collection of objects. For queries that return some records, we must assign the results to a Recordset Object. The recordset object can be used with the conjunction of the Field object to get most functionality out of it. The most common methods and properties of the Recordset object are listed as follows in the following table.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddNew</td>
<td>Creates a new record in an updateable recordset</td>
</tr>
<tr>
<td>Close</td>
<td>Closes an open recordset and any dependent objects</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the current record in an open recordset</td>
</tr>
<tr>
<td>Move</td>
<td>Moves the position of the current record</td>
</tr>
<tr>
<td>MoveFirst, MoveLast</td>
<td>Moves to the first, last, next or previous record in the recordset, and makes that the current</td>
</tr>
<tr>
<td>MoveNext, MovePrevious</td>
<td>record the current record</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOF</td>
<td>Returns true if the current record position is before first</td>
</tr>
<tr>
<td>EOF</td>
<td>Return true if the current record position is after last</td>
</tr>
<tr>
<td>RecordCount</td>
<td>Number of records in the current Recordset</td>
</tr>
</tbody>
</table>

The online fault diagnosis system is based on the above concepts and objects. The next section will explain the architecture of the fault diagnosis system, database behind the system and the how to use the system from any corner of the world.
Chapter 6

Architecture and working of Online Fault Diagnosis System

Online Fault Diagnosis System is a web based client server architecture that is based on the cutting edge technologies such as ASP, Internet Information Server, ADO and Microsoft Access database. This system is aimed for the use of the University of Cincinnati, Robotics team for analyzing the fault that could occur during the course of AGV operation.

6.1 The Application Architecture:

There are various types of architecture in the development of applications, depending on the cost, technical requirement, efficiency and effectiveness. They are grouped under the category called the client/serve models.

6.1.1 The Client/Server Model

Client/server describes physical deployment models where the client computer makes a request to the server computer and the server computer services or responds to the request.

Client/server is a model of computing whereby client applications running on a desktop or personal computer access information on remote servers or host computers. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized functionality for
multiple users. Client/server is also used to refer to a request and service-the-request relationship

6.1.2 Services and Tiers

Tiers are a logical concept. The three tiers are generally described as user (first), business (second or middle), and data (third) service tiers. A service is a unit of application logic that implements operations, functions, or transformations that are applied to objects. For example, a reusable business service can be implemented as a COM component that ensures a purchase does not exceed the buyer's credit limit.

The concept of tiers emphasizes segmenting applications into the three types of services, and is not about implementing the services or about the number of physical computers involved in deploying the solution.

6.1.3 Two-Tier Client/Server Architecture

Two-tier applications represent a first step in separating database-specific access logic out of the client application and placing it on a server.

Usually, this is done by implementing services as stored procedures that execute on the server against a database. This makes maintenance, upgrades, and general
administration of that code easier because the code only exists on the server and not on every client.

Also the database management system (DBMS) provides the centralized functionality required when supporting many users.

6.1.4 Three-Tier Client/Server Architecture

Three-tier client/server architecture provides additional flexibility for building applications as they grow in complexity. It describes both the application model (user, business, and data services) and the actual deployment of services. An application model is a conceptual view of an application that establishes the definitions, rules, and relationships that structure the application. Some services reside on client computers, typically user services, while business and data services generally reside on server computers that provide centralized, multi-user functionality.

Fig 6.2: Three-Tier model
The following illustration describes a general scenario. User services run on a client computer, shared business services represent the middle tier and run on a server computer, and shared data services are the third tier, also running on a server computer.

The decision to build a two-tier or three-tier application depends on the application requirements. This course focuses on three-tier systems that serve enterprise-wide applications.

6.2 Application Infrastructure

A server computer provides the centralized, multi-user functionality in a three-tier architecture. Other terms for this functionality are application infrastructure and plumbing. The application infrastructure is the software that supports concurrent access to a shared service, usually business and data services. For example, a Relational Database Management System (RDBMS) controls access to stored procedures and data on a database server.

Microsoft Transaction Server (MTS) is software that works with the operating system to provide the application infrastructure for sharing business services implemented as components. Recent studies show that as much as 30–40 percent of corporate IT budgets for application developments are targeted at developing that application infrastructure. MTS reduces those costs because it provides centralized, multi-user functionality.

The Internet represents another potential form of three-tier client/server architecture. Early Internet technologies did not take advantage of the processing
power of the client computer. Client browsers simply translated the HTML files. Developers who wanted to provide enhanced features had to rely on technologies such as server-side scripting through the Common Gateway Interface (CGI).

Newer technologies can now exploit client-processing power and extend server capabilities. Dynamic HTML (DHTML), Java applets, client-side scripting, and ActiveX controls now take advantage of the client processor. Web clients can access shared business services through Active Server Pages (ASP). ASP files both pass information to the client, and take information from the client and pass it to business rules in the middle tier. The ASP files call COM components that enforce business rules. The COM components run in an MTS environment that provides the application infrastructure required by multiple users accessing the ASP files.

6.3 Architecture for Online Fault Diagnosis System

This Application is designed as a two-tier Client/Server architecture. Client maybe a Internet Explorer or Netscape Navigator or any other browser which supports VBScript and JavaScript. Internet Information Server 5.0 (IIS) serves as the web server, which processes the request and connects with the database for appropriate data, which needs to be sent along with the HTML pages.

The system is could be accessed from any part of the globe, with just a PC, web browser and Internet connection. Whenever a user types the URL for the application he is directed to the Welcome page where there is a brief description of the application. This page also contains a Java Applet, which scrolls the news about the Robotics team. Apart from this the user could also view randomly changing pictures about the taken in the contest. He has the facility to enter the login page, where he is
asked to provide the information for authentication. Based on the right information he is then allowed to access the system for the required data for fixing the fault that may occur during the operation of the Automated Guided Vehicle.

In order to understand the architecture behind the fault diagnosis system, here is the diagrammatic representation of the same

Fig 6.3: Architecture of Fault Diagnosis System
6.3.1 Client

The knowledgeable user of the fault diagnosis system, make his request through a web interface using Internet Explorer or Netscape Navigator. The user is made to enter the username and password for authentication. This information is then sent to the Internet information server. But before explaining the process in the serve, let see what features are incorporated on the client side. If the user tries to hits a wrong username and password then he is redirected back to the login page. If he hits the login button without entering either the username or password then the client side scripting checks for the flaw and prompts the user to enter them.

This could be seen from the following screen shot.

![Fig 6.4: User Authentication](image)

Here the user tries to enter the Online System but the browser prompts him to enter the password, since it’s empty, which would obviously redirect him back to this login page because of the unauthentic data. This feature helps in avoiding the Active Server Page processing the data by making connection with the database and unnecessarily wasting the resources. This would increase the efficiency of the system by allowing
more number of users to access the information, since only those users who entered complete information is sent for further processing of the request.

A program module takes care of this functionality and is embedded in the HTML page where the user login is prompted. This module consists of a function which check the validity of the form before it is sent to the server for processing the user information. The script is shown as below

```javascript
<script Language = "JavaScript">
function Form1_Validator(theForm)
{
    // check to see if the field is blank
    if (theForm.username.value == "")
    {
        alert("You must enter the username.");
        theForm.username.focus();
        return (false);
    }
    // require at least 3 characters be entered
    if (theForm.userpassword.value =="")
    {
        alert("Please enter password.");
        theForm.userpassword.focus();
        return (false);
    }
}
</Script>
```

Once the user is verified for the authenticity, a HTML page is sent back to the client for submitting the required options for providing data about the fault diagnosis for that system.
The user is given the option to select the major system of the robot, which then provides a different set of options for choosing the specific unit, which fall under the respective major categorization. The robot system is categorized under five main systems, which then has its own set of specific units. The combination of the major unit and its selected specific unit would give us a unique set of fault diagnosis data containing possible fault, potential effects, potential causes and solutions for rectifying the fault. This functionality is implemented with the JavaScript, which produces the required options. Here are few examples showing the functionality explained above.

<table>
<thead>
<tr>
<th>Major Unit: Power System</th>
<th>Major Unit: SONAR System</th>
</tr>
</thead>
</table>

The two samples show the Major subsystem selected by the user and how the specific Units option set is changed based on the major system selected. Once the user selects the required option, he then submits the request, which is then processed and then returned in the form of a table containing data for fault diagnosis.
6.3.2 Server

The web server is used to process the request sent from the clients and provide required results. Internet Information Server (IIS 5.0) a product provided by Microsoft Corporation installs and runs only on the Windows NT/95/98 platform.

The entire application related files are stored in the virtual directory of the web server and is processed when requested by the user.

The database is stored in a different folder that is meant for storing the database information. Microsoft Access Database is used for storing and retrieving relevant information for the system. This database can be easily exported as Excel file for easy understanding of the data.

The application on the server is always in the active state as long as the server is up and running. So whenever a user types the specified URL, the client sends a request to the IIS, which then process the request and sends back a HTML file. The user then hits the login button and then he is directed to the Login page of the application. Once the user submits the required information for user authentication then the server receives the information, which in turn calls the necessary Active
Server Page for processing the request. The program then sends a request to the database through the connection object of the ADO object. The query is the processed by the database and the results of the query are sent back to the program, which then validates the given information with the stored information. If the validation is true then the user is allowed to enter the area where he can select the criteria for the particular faults that occur in the system to obtain the rich collection of data for fixing the fault. The program that validates the username and password with the data, which is stored in the database, is as follows

```html
<!--
myname = request.form("username")
mpassword = request.form("userpassword")
set oconn = server.createobject("ADODB.Connection")
oconn.Open("DRIVER=Microsoft Access Driver (*.mdb); DBQ=" & Server.MapPath("\ucrobotics\db\Security.mdb")")
strSQL = "Select * from users where user = " & myname & " ;" 
set rsMyrec = oconn.execute(strSQL) 
if rsMyrec.eof then <%
    Please check the Username. The username is case sensitive.
    Try <a href = 'securitylogin.asp'> Logging In </a> Again.
<% response.end
If rsMyrec("password") = mpASSWORD then 
    response.redirect "subsystem.asp"
else <%
    Password Unrecognized<br>
    Try <a href = 'securitylogin.asp'> Logging In </a> Again.
<% response.end
rsMyrec.close
oconn.close
set rsMyrec="nothing
set oconn="nothing %>

Script used to authenticate the User
```
The script uses the connection object of the ADO object collection. A variable of type connection object is declared and then its properties and methods are used to connect to the database and execute the SQL query. The data retrieved is collected using the Recordset object, which then check for the End of File condition to check if the user has provided the right kind of data. If the data is incorrect he is then redirected to the login page by using the response objects redirect method. The IIS maintains the log about the number of users, has the option to limit the number of users for any time period and much more which are out of scope of this thesis.

Hence it could be observed that the server acts both as web server and a database server. As the system gets more complicated – more users, high volume of data flow, tight business rules – then the architecture could be upgraded to a n-tier architecture.

Advantages of the Online System when compared to stand alone system

- Flexibility to access the data globally and at anytime
- Reduces the inventory of carrying additional storage medias
- Could be upgraded to n-tier architecture
- Allows multiple user to access the data at the same time from different locations
- Single source of integrated data

With these advantages the online system could help the robotics team in analyzing the faults that come along during the operation of the AGV, Bearcat II. Although the Online fault diagnosis system provides comprehensive information about the failures of the Bearcat II, unexpected failures certainly exist and hence the database needs to be updated.
Chapter 7

Conclusion and Recommendations for Future Enhancements

An online fault diagnosis system has been developed and deployed at www11.ewebcity.com/ucrobotics for the Robotics team to efficiently manage the fault diagnosis and fixing the same. This system provides the best source of the fault analysis that the team has experienced and the ones that could really take a long time without this system.

The online system could be viewed from web browser like Internet Explorer or Netscape Navigator by either typing http://www11.ewebcity.com/ucrobotics or by navigating to the links page from www.robotics.uc.edu.

This online e-diagnosis system could be developed as an customized portal for different category of people who use the system for different purposes. The best comparison for the future recommendation would be enhancing this system that could be made to have different look and feel depending on the users taste, need similar to what Yahoo.com or MSN.com provides for its’ users.

Now the system is hosted on a private web server provided by ewebcity.com. It would be a good idea to install an Internet Information Server at the Robotics Research laboratory. This would give the freedom to develop, test and implement the customized portal by providing facilities for session management and application management which otherwise is not possible.
References


Appendix A

Program Scripts that are used in the Online Fault Diagnosis System

Program to query the database when the options are submitted

```html
<html>
<! -- Program to query the database for the required data about the fault analysis of the system -- >

<head>
<title>dbtable.asp</title>
</head>

<body bgcolor =black>

<% 
strSQL = "Select * from pfmea where MajorSystems = " &
request.form("MajorSys") & " and " 
strSQL = strSQL & "SpecificUnit = " & request.form("Specificunits") & 
";"
showblank = " &nbsp;
shownull = " - null - 

set oconn = server.CreateObject ("ADODB.Connection")
oconn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("\ucrobotics\db\PFMEA1.mdb")

set rstemp = oconn.execute(strSQL)
if rstemp.eof then
    response.write "No Records Matched<br>"
oxconn.close
    set oconn = nothing
else
%>

<DIV align=center>
P align=center>
<table align=center border=1 cellPadding=1 cellSpacing=1 width="581"
id=TABLE1 style="height: 71px" borderColorLight="#FFFFFF"
borderColorDark="#FFFFFF" background="bluebackground.gif">
<tr>
    <td width="19"><b><font color="#FFFFFF" face="Verdana">ID</font></b></td>
    <td width="63"><b><font color="#FFFFFF" face="Verdana">Failure Number</font></b></td>
    <td width="69"><b><font color="#FFFFFF" face="Verdana">Major Systems</font></b></td>
    <td width="62"><b><font color="#FFFFFF" face="Verdana">Specific Unit</font></b></td>
    <td width="55"><b><font color="#FFFFFF" face="Verdana">Failure Modes</font></b></td>
    <td width="74"><b><font color="#FFFFFF" face="Verdana">Potential Effects</font></b></td>
    <td width="109"><b><font color="#FFFFFF" face="Verdana">Potential Causes</font></b></td>
</tr>
```

58
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Login Page</title>
</head>
<body bgcolor="#000000">
<table border="0" width="75%" height="51" cellspacing="0" cellpadding="0">
<tr>
<td width="100%" height="47" background="gray.gif" bordercolor="#000000">

Program to get the username and password from the login page

```
<% ' Now lets grab all the records
Do UNTIL rstemp.eof
%>
<TR>
<% for each whatever in rstemp.fields
<%thisfield = whatever.value
if isnull(thisfield) then
   thisfield = shownull
end if

   if trim(thisfield) = ""then
      thisfield = showblank
end if%>
<TD Valign = top><font color="#FFFFFF" face="Verdana" size="2">
<%=thisfield%></font></td>
<%next%>
</TR>
<%rstemp.movenext
LOOP
rstemp.close
set rstemp=nothing
oconn.close
set oconn=nothing
end if%>
</font></TD>
</TABLE>
</body></html>
<p align="center"><img border="0" src="samrobologo.gif" align="right" hspace="15"></td></p>
</table></td>
</tr>
</table>
<p align="center">Enter the Username and password to use the Online Fault Diagnosis System</p><center></center></td></tr></table></td></tr></table>
<script Language = "javascript">
function Form1_Validator(theForm) {

// check to see if the field is blank
if (theForm.username.value == "")
{
    alert("You must enter the username.");
    theForm.username.focus();
    return (false);
}

// require at least 3 characters be entered
if (theForm.userpassword.value =="")
{
    alert("Please enter password.");
    theForm.userpassword.focus();
    return (false);
}
}
</Script>

<tr>
    <td width="25%" height="126" valign="top" bgcolor="#000000" align="center">
        &nbsp;
    </td>
</tr>
Script that displays random images and the java applet in the welcome page

<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
</head>
</html>
Welcome to UC Robotics Online Fault Diagnosis System

<!-- The script here helps in rotating the image -->

<script language="JavaScript">
<!-- Begin
var interval = 2.0; // delay between rotating images (in seconds)
var random_display = 1; // 0 = no, 1 = yes
interval *= 1000;

var image_index = 0;
image_list = new Array();
image_list[image_index++] = new imageItem("bearcatasphalt.jpg");
image_list[image_index++] = new imageItem("designpresentation.jpg");
image_list[image_index++] = new imageItem("UCTeam2.jpg");
image_list[image_index++] = new imageItem("robot_ramp.jpg");
var number_of_image = image_list.length;
function imageItem(image_location) {
    this.image_item = new Image();
    this.image_item.src = image_location;
}
function get_ItemItemLocation(imageObj) {
    return(imageObj.image_item.src)
}
function generate(x, y) {
    var range = y - x + 1;
    return Math.floor(Math.random() * range) + x;
}
function getNextImage() {
    if (random_display) {
        image_index = generate(0, number_of_image-1);
    }
    else {
        image_index = (image_index+1) % number_of_image;
    }
    var new_image = get_ItemItemLocation(image_list[image_index]);
    return(new_image);
}
function rotateImage(place) {
    var new_image = getNextImage();
document[place].src = new_image;
var recur_call = "rotateImage('"+place+'")";
setTimeOut(recur_call, interval);
}
// End -->
</script>

<body onload="rotateImage('rImage')" bgcolor="#000000">
<p>&nbsp;</p>
<div align="left">
<table border="0" width="75%" height="51" cellspacing="0" cellpadding="0">
<tr>
<td width="100%" height="47" background="gray.gif" bordercolor="#000000">
<p align="center"><img border="0" src="samrobologo.gif" align="right" width="205" height="70" hspace="15"></p>
</td>
</tr>
</table>
</div>

<div align="left">
<table border="0" width="75%" height="196" cellspacing="0" cellpadding="0" bordercolor="#FFFFFF" bgcolor="#D8D8D8">
<tr>
<td width="24%" height="192" bgcolor="#000080" background="gray.gif" bordercolor="#000000">
<applet code="OCVscroll.class" onmouseover="window.status=' '; return true" align="baseline" height="110" width="177">
<! -- General / Default Settings -->
<param name="bgcolor" value="0,0,0">
<param name="textcolor" value="255,255,255">
<param name="font" value="Helvetica, plain, 12">
<param name="halign" value="Left">
<param name="lrmargin" value="5">
<param name="hlcolor" value="0,0,255">
<param name="delay" value="2">
<param name="scrolldelay" value="55">
<param name="loadwhere" value="_self">
<! -- Specific Settings -->
<param name="desc0" value="UC Robotics Center is where innovative and path-breaking technologies are conceived, tested, implemented and most importantly, put into practical use.">
<param name="textcolor0" value="255,255,255">
<param name="desc1" value="">
<param name="desc2" value="">
<param name="desc3" value="The UC Robot Team is a multi-disciplinary team conducting original research in the areas of Robot Vision, Robot Controls and Integration of Robot Systems.">
<param name="desc4" value="">
<param name="desc5" value="">
<param name="desc6" value="Bearcat II came third in the 2000 International Ground Robotics Competition at Disneyworld, Florida.">
<param name="desc7" value="">
<param name="desc8" value="">
<param name="desc9" value="The Robotics Team also involves itself in a variety of outreach activities. Recently, the team demonstrated the Bearcat II for high school children at ----.">
<param name="desc10" value="">
<param name="desc11" value="">
<param name="desc12" value="Get more information about the UC Robotics Team at http://www.robotics.uc.edu">
</applet>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
</param>
Welcome to the Online Fault Diagnosis System of the UC Robotics Center.

This online system is an aide to help robot designers in improving their productivity and making better design decisions - by providing the right information at the right time.

The system enhances information visibility to the users in a secure environment by providing a single input interface.

This system is actually an online repository of information collected over a period of 6 years and organized under meaningful subsystems.
Program to check the username and password against the database

```html
<html><head>
<title>SecurityLoginRespond</title>
</head>

<%
myname = request.form("username")
mypassword = request.form("userpassword")

set oconn = server.createobject("ADODB.Connection")
oconn.Open("DRIVER={{Microsoft Access Driver (*.mdb)}}; DBQ=" & Server.MapPath("\ucrobotics\db\Security.mdb"))

strSQL = "Select * from users where user = '"
strSQL = strSQL & myname & "'

set rsMyrec = oconn.execute(strSQL)
if rsMyrec.eof then %>
    Please check the Username. The username is casesensitive.
    Try <A href ='loginsecurity.asp'> Logging In </a> Again.
<%else

if rsMyrec("Password") = mypassword then
    session("name") = rsMyrec("user")
    response.redirect"Subsystem.asp"
else %>
    Password Unrecognized<br>
    Try <A href ='loginsecurity.asp'> Logging In </a> Again.
<% end if
end if

rsMyrec.Close
oconn.Close
set rsMyrec=nothing
set oconn=nothing
%
</body></html>
```

---

Script that generates the frames page while searching the database

```html
<html>
<head>
    <title>SubSystem Search</title>
</head>
```
Program to dynamically load the drop down list for the set of values

<HTML>
<HEAD><TITLE>Fault Diagnosis System</TITLE>
<base target="main">
</HEAD>

<BODY OnLoad="BuildSystem(0);">
<font align="justify">
<table border="0" width="107%" height="593" cellspacing="0" cellpadding="0" align="left">
<tr>
<td width="100%" bgcolor="#E6E6E6" valign="top" colspan="3" height="107" background="gray.gif">
&nbsp;
<FORM Name="SearchForm" method="post" action="dbtable.asp">
</font>
<font color="#FFFFFF">
</font>
<font face="Verdana" size="2">
Major System:<font> </font></font>

<SELECT Name="MajorSys" OnChange="BuildSystem(this.selectedIndex);">
<Option Value="Power System">Power</Option>
<Option Value="SONAR System">SONAR</Option>
<Option Value="Mechanical System">Mechanical</Option>
</SELECT>
</font>
</BODY>
</HTML>
<OPTION Value="Vision System">Vision
<OPTION Value="Charging System">Charging
</Select>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</font>
<p><font color="#FFFFFF" face="Verdana" size="2">Specific Units:</font>
<! --  We want to define at least one option so that the select tag is
created
with the correct dimensions--> </p>
<SELECT Name="SpecificUnits">
<OPTION Value="">
--------
</Select></p>
<p><font face="Verdana" size="2" color="#FF0000"><input type="submit" value="Submit" name ="Submit" ></font>
</p>
</Form>

</td>
</tr>
<tr>
<td width="13%" bgcolor="#000000" valign="top" height="285" background="gray.gif">
<center><p> &nbsp;</p></center></td>
<td width="65%" bgcolor="#000000" valign="top" height="285">
<p align="left"><font face="Verdana" color="#FFFF00" size="2"><u>How To Use the
Menu:</u></font></p>
The online Fault Diagnosis is based on two key elements.&nbsp;
<p align="left"><font face="Verdana" size="2" color="#8D9902">The Major System option gives
the flexibility of choosing the required major unit.&lt;/font&gt;&lt;/p&gt;
<p align="left"><font face="Verdana" color="#8D9902" size="2">The Specific Units option lists the
sub classification for the major unit selected in the first drop-down list.&lt;/font&gt;&lt;/p&gt;
<p align="left"><font face="Verdana" size="2" color="#8D9902">So select both the options and then
&lt;/font&gt;&lt;font face="Verdana" size="2" color="#FFFF00">submit</font>&lt;/font&gt;&lt;font face="Verdana" size="2" color="#8D9902">to get the
failures modes and required actions for the failures.&lt;/font&gt;&lt;/p&gt;</td>
<td width="22%" bgcolor="#B0BE03" valign="top" height="478" background="gray.gif" rowspan="2">&nbsp;</td>
</tr>
<SCRIPT Language="JavaScript"> 

//Build arrays for each Major System 
Contacts=new Array(5);
Contacts[0]=new Array(7);
Contacts[1]=new Array(7);
Contacts[2]=new Array(8);
Contacts[3]=new Array(4);
Contacts[4]=new Array(1);

//Power System 
Contacts[0][0]="Battery";
Contacts[0][1]="Fuse";
Contacts[0][2]="Invertor";
Contacts[0][3]="Main Switch";
Contacts[0][4]="SONAR Motor";
Contacts[0][5]="Servo Motor";
Contacts[0][6]="Solenoid";

//SONAR System 
Contacts[1][0]="PID";
Contacts[1][1]="Polakit";
Contacts[1][2]="SONAR Wire";
Contacts[1][3]="SONAR Height";
Contacts[1][4]="Polaroid";
Contacts[1][5]="Program";
Contacts[1][6]="Fastening";

//Mechanical System 
Contacts[2][0]="Wheel";
Contacts[2][1]="Fitting";
Contacts[2][2]="Alignment";
Contacts[2][3]="Lubricant";
Contacts[2][4]="Coupling";
Contacts[2][5]="Servo Motor";
Contacts[2][6]="Wheel Shaft";
Contacts[2][7]="Shaft Keys";

//Vision System 
</SCRIPT>
Contacts[3][0]="Vision Threshold";
Contacts[3][1]="CCD";
Contacts[3][2]="ISCAN";
Contacts[3][3]="Switching Unit";

//Charging System
Contacts[4][0] = "Charger";

//Call this to build the Contact list for the specified Salesperson
function BuildSystem(num)
{
    //Select the first Contact
document.SearchForm.SpecificUnits.selectedIndex=0;

    //For every contact in the array for this person, add a new option
    for (ctr=0; ctr<Contacts[num].length; ctr++)
    {
        document.SearchForm.SpecificUnits.options[ctr]=new Option(Contacts[num][ctr],Contacts[num][ctr]);
    }

    //Set the length of the select list
document.SearchForm.SpecificUnits.length=Contacts[num].length;
}

</Script>

HTML tags that generate the Main frame of the subsystem mod

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
    <meta name="GENERATOR" content="Microsoft FrontPage 4.0">
    <meta name="ProgId" content="FrontPage.Editor.Document">
    <title>Main</title>
    <base target="_self">
</head>

<body bgcolor="#000000">

</body>
</html>
Script that pops up a window on closing the application

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
    <meta name="GENERATOR" content="Microsoft FrontPage 4.0">
    <meta name="ProgId" content="FrontPage.Editor.Document">
    <title>New Page 4</title>
    <base target="contents">
    <!-- Begin LANGUAGE = "Javascript" -->
    <SCRIPT LANGUAGE = "Javascript">
      function leave() {
        window.open('thankyou.htm', '', 'toolbar=no,menubar=no,location=no,height=300,width=500');
      }
    </SCRIPT>
    <!-- END -->
  </head>
  <body onUnload ="leave()" background="gray.gif">
    <div align="center">
      <center>
        <table border="0" width="100%">
          <tr>
            <td width="24%"><a href="Javascript: onClick =top.close()"><img border="0" src="logout.jpg" align="left" width="89" height="22"></a></td>
            <td width="76%"><img border="0" src="samrobologo.gif" align="right" width="205" height="70"></td>
          </tr>
        </table>
      </center>
    </div>
    <p align="right">&nbsp;</p>
  </body>
</html>
Thank You for using the Fault Diagnosis System. Hope you found it Useful!!!
<table>
<thead>
<tr>
<th>Seq ID</th>
<th>Failure NO</th>
<th>Major Systems</th>
<th>Specific Unit</th>
<th>Failure Modes</th>
<th>Potential Effects</th>
<th>Potential Causes</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>PFMEA_POU_1</td>
<td>Power System</td>
<td>Battery</td>
<td>No Main power</td>
<td>Robot does not start</td>
<td>Battery down</td>
<td>Check voltage by the voltmeter which is mounted on the side panel</td>
</tr>
<tr>
<td>1001</td>
<td>PFMEA_POU_2</td>
<td>Power System</td>
<td>Battery</td>
<td>Low battery power</td>
<td>Robot stops at jog greater than 10000</td>
<td>Battery down</td>
<td>Check voltage by the voltmeter which is mounted on the side panel</td>
</tr>
<tr>
<td>1002</td>
<td>PFMEA_POU_3</td>
<td>Power System</td>
<td>Fuse</td>
<td>No Power</td>
<td>Camera &amp; ISCAN tracker do not work</td>
<td>Fuse blown</td>
<td>Check and replace fuse</td>
</tr>
<tr>
<td>1003</td>
<td>PFMEA_POU_4</td>
<td>Power System</td>
<td>Battery</td>
<td>Low power</td>
<td>Computer Reboots</td>
<td>Battery down</td>
<td>Check voltage by the voltmeter which is mounted on the side panel</td>
</tr>
<tr>
<td>1004</td>
<td>PFMEA_POU_5</td>
<td>Power System</td>
<td>Inverter</td>
<td>No Power</td>
<td>Computer doesn't power on</td>
<td>Loose connection</td>
<td>Check connection at the inverter</td>
</tr>
<tr>
<td>1005</td>
<td>PFMEA_POU_6</td>
<td>Power System</td>
<td>Main Switch</td>
<td>No power</td>
<td>Computer doesn't power on</td>
<td>Main switch in off position</td>
<td>Switch the main on</td>
</tr>
<tr>
<td>1006</td>
<td>PFMEA_POU_7</td>
<td>Power System</td>
<td>SONAR Motor</td>
<td>No main Power</td>
<td>SONAR Motor doesn't start</td>
<td>Battery down</td>
<td>Check the battery voltage and charge if below the 36V</td>
</tr>
<tr>
<td>1007</td>
<td>PFMEA_POU_8</td>
<td>Power System</td>
<td>Servo Motor</td>
<td>Low power</td>
<td>Whining noise &amp; Motor heats up</td>
<td>Less voltage input due to low battery voltage</td>
<td>Check the voltage for 36V and charge the batteries sufficiently</td>
</tr>
<tr>
<td>1008</td>
<td>PFMEA_POU_9</td>
<td>Power System</td>
<td>Solenoid</td>
<td>No power</td>
<td>Entire system down</td>
<td>Solenoid coil is burnt</td>
<td>Check the voltage across the solenoid energizing coil</td>
</tr>
<tr>
<td>Year</td>
<td>Code</td>
<td>System</td>
<td>Issue Description</td>
<td>Cause Description</td>
<td>Resolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>PFMEA_OAU_1</td>
<td>SONAR System</td>
<td>No intermittent stops</td>
<td>SONAR doesn't bring in proper feedback</td>
<td>Adjust the PID values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>PFMEA_OAU_2</td>
<td>SONAR System</td>
<td>Error in calculating the obstacle distance</td>
<td>Robot hits obstacle</td>
<td>Adjust parameters in Polakit program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>PFMEA_OAU_3</td>
<td>SONAR System</td>
<td>Error in calculating the obstacle distance</td>
<td>Robot goes off track</td>
<td>Adjust parameters in Polakit program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>PFMEA_OAU_4</td>
<td>SONAR System</td>
<td>Failure to detect obstacle</td>
<td>Robot hits obstacle</td>
<td>Secure connections, replace Sonar’s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>PFMEA_OAU_5</td>
<td>SONAR System</td>
<td>Failure to detect obstacle</td>
<td>Robot hits obstacle</td>
<td>Secure connections, refer Polaroid manual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>PFMEA_OAU_6</td>
<td>SONAR System</td>
<td>Failure to detect obstacle</td>
<td>Robot hits obstacle</td>
<td>Reset parameters in the program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>PFMEA_OAU_7</td>
<td>SONAR System</td>
<td>Failure to initiate change in steering</td>
<td>Robot hits obstacle</td>
<td>Secure connections</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>PFMEA_OAU_8</td>
<td>SONAR System</td>
<td>False feedback pulse</td>
<td>Robot goes off track</td>
<td>Adjust elevation to 18” from ground to Edge of SONAR polaroid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- **PID:** Proportional-Integral-Derivative
- **SONAR System:** System uses SONAR technology for obstacle detection.
- **Polakit:** Software used for robotic control.
- **SONAR Wire:** Component involved in the SONAR system.
- **Program:** Refers to robotic control program.
- **Height:** Component height in relation to the ground or obstacle.
<table>
<thead>
<tr>
<th>No.</th>
<th>PFMEA Code</th>
<th>System</th>
<th>Issue</th>
<th>Potential Failure</th>
<th>Effect</th>
<th>Recommended Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>PFMEA_OAU_9</td>
<td>SONAR System</td>
<td>Fastening</td>
<td>False feedback pulse</td>
<td>Improper maneuvering</td>
<td>Check if the SONAR is secured to the motor shaft</td>
</tr>
<tr>
<td>3000</td>
<td>PFMEA_MEU_1</td>
<td>Mechanical System</td>
<td>Wheel</td>
<td>Wheel rotating freely</td>
<td>Shaft may get damaged</td>
<td>Remove the wheel and place appropriate shaft key</td>
</tr>
<tr>
<td>3001</td>
<td>PFMEA_MEU_2</td>
<td>Mechanical System</td>
<td>Wheel</td>
<td>Wheel coming off the shaft</td>
<td>Damage to the entire robot</td>
<td>Check the size of the retainer screw size and secure it tightly</td>
</tr>
<tr>
<td>3002</td>
<td>PFMEA_MEU_3</td>
<td>Mechanical System</td>
<td>Fitting</td>
<td>Heating of the main wheel bearings</td>
<td>Wheels don't rotate smoothly and bearings damaged</td>
<td>Check the levels with the spirit levels and adjust the plumber block</td>
</tr>
<tr>
<td>3003</td>
<td>PFMEA_MEU_4</td>
<td>Mechanical System</td>
<td>Fitting</td>
<td>Wheels jammed</td>
<td>Robot doesn't run</td>
<td>A bulged shaft due to improper fit</td>
</tr>
<tr>
<td>3004</td>
<td>PFMEA_MEU_5</td>
<td>Mechanical System</td>
<td>Alignment</td>
<td>Wheels don't turn smoothly</td>
<td>Unwanted noise and the gearbox and drive shaft coupling gets heated</td>
<td>Improper lubrication and alignment</td>
</tr>
<tr>
<td>3005</td>
<td>PFMEA_MEU_6</td>
<td>Mechanical System</td>
<td>Lubricant</td>
<td>Increase in the gear meshing noise</td>
<td>Gearbox gets heated up</td>
<td>Lubricant level in the gearbox</td>
</tr>
<tr>
<td>3006</td>
<td>PFMEA_MEU_7</td>
<td>Mechanical System</td>
<td>Alignment</td>
<td>Heating of servo motors</td>
<td>Malfunction of the total drive system</td>
<td>Alignment between servo motor and the gear</td>
</tr>
<tr>
<td></td>
<td>PFMEA_MEU_8</td>
<td>Mechanical System</td>
<td>Coupling</td>
<td>Heating of servo motors</td>
<td>Inefficient transmission of power</td>
<td>Improper coupling alignment</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>-------------------</td>
<td>----------</td>
<td>-------------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>3007</td>
<td>PFMEA_MEU_9</td>
<td>Mechanical System</td>
<td>Servo Motor</td>
<td>Servo motors turn very slowly</td>
<td>Robot doesn't move</td>
<td>Amplifier output is insufficient</td>
</tr>
<tr>
<td>3008</td>
<td>PFMEA_MEU_10</td>
<td>Mechanical System</td>
<td>Servo Motor</td>
<td>Servo motor doesn't rotate according to the pulses</td>
<td>Robot goes off-track</td>
<td>Improper feedback</td>
</tr>
<tr>
<td>3009</td>
<td>PFMEA_MEU_11</td>
<td>Mechanical System</td>
<td>Wheel Shaft</td>
<td>Wheel Shaft slipping out of coupling</td>
<td>Bending of wheel shaft</td>
<td>Robot stops</td>
</tr>
<tr>
<td>3010</td>
<td>PFMEA_MEU_12</td>
<td>Mechanical System</td>
<td>Shaft Keys</td>
<td>Key moving out of its place</td>
<td>Shaft gets disengaged from the coupling</td>
<td>Robot stops</td>
</tr>
<tr>
<td>3011</td>
<td>PFMEA_VSU_1</td>
<td>Vision System</td>
<td>Vision Threshold</td>
<td>No points picked</td>
<td>Robot goes off-track</td>
<td>Threshold limit not reached</td>
</tr>
<tr>
<td>4000</td>
<td>PFMEA_VSU_2</td>
<td>Vision System</td>
<td>CCD</td>
<td>No points picked</td>
<td>Robot goes off-track</td>
<td>CCD camera not working</td>
</tr>
<tr>
<td>Code</td>
<td>PFMEA_VSU_3</td>
<td>PFMEA_VSU_4</td>
<td>PFMEA_VSU_5</td>
<td>PFMEA_VSU_6</td>
<td>PFMEA_VSU_7</td>
<td>PFMEA_CHU_1</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Vision System</td>
<td>ISCAN</td>
<td>Coordinates not available</td>
<td>Vision System</td>
<td>ISCAN</td>
<td>Cannot be calibrated</td>
<td>Vision System</td>
</tr>
<tr>
<td>ISCAN Coordinates not available</td>
<td>ISCAN tracker is not functioning</td>
<td>Vision System</td>
<td>Vision Threshold</td>
<td>Incorrect points picked</td>
<td>ISCAN tracker is not functioning</td>
<td>Switching Unit</td>
</tr>
<tr>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>Switching Unit</td>
<td>Switching Unit</td>
</tr>
<tr>
<td>4002</td>
<td>4003</td>
<td>4004</td>
<td>4005</td>
<td>4006</td>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>PFMEA_VSU_3</td>
<td>PFMEA_VSU_4</td>
<td>PFMEA_VSU_5</td>
<td>PFMEA_VSU_6</td>
<td>PFMEA_VSU_7</td>
<td>PFMEA_CHU_1</td>
<td></td>
</tr>
<tr>
<td>ISCAN</td>
<td>ISCAN</td>
<td>ISCAN</td>
<td>ISCAN</td>
<td>ISCAN</td>
<td>ISCAN</td>
<td>ISCAN</td>
</tr>
<tr>
<td>Coordinates not available</td>
<td>Vision System</td>
<td>Vision Threshold</td>
<td>Switching Unit</td>
<td>Camera switch fails</td>
<td>Camera doesn't switch properly</td>
<td>Batteries don't charge</td>
</tr>
<tr>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>Incorrect points picked</td>
<td>Robot doesn't follow the path</td>
<td>Robot goes off-track</td>
<td>Switching Unit</td>
<td>No power to the entire system</td>
</tr>
<tr>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>Points outside the line are brighter</td>
<td>Galil board controller not working</td>
<td>Switching unit not working</td>
<td>Switching unit not working</td>
<td>Circuit not closed</td>
</tr>
<tr>
<td>ISCAN tracker is not functioning</td>
<td>ISCAN tracker is not functioning</td>
<td>Readjust the threshold so that image on the line is seen as black image</td>
<td>Reset the Galil board</td>
<td>Check connections with respect to the schematic diagram</td>
<td>Adjust the ISCAN settings written as specified on the robot panel</td>
<td>Check to see the main switch is on and the emergency stop in its off position</td>
</tr>
</tbody>
</table>

- **4002**: PFMEA_VSU_3
  - Vision System: ISCAN Coordinates not available
  - ISCAN tracker is not functioning
  - Check the circuit and connections

- **4003**: PFMEA_VSU_4
  - Vision System: Vision System
  - Vision Threshold: Incorrect points picked
  - Points outside the line are brighter
  - Readjust the threshold so that image on the line is seen as black image

- **4004**: PFMEA_VSU_5
  - Vision System: Switching Unit
  - Camera switch fails
  - Robot doesn't follow the path
  - Reset the Galil board

- **4005**: PFMEA_VSU_6
  - Vision System: Switching Unit
  - Camera doesn't switch properly
  - Robot goes off-track
  - Check connections with respect to the schematic diagram

- **4006**: PFMEA_VSU_7
  - Vision System: ISCAN
  - Cannot calibrate
  - Robot doesn't run within the specified path
  - Image obtained doesn't reflect properly in the video monitors
  - Adjust the ISCAN settings written as specified on the robot panel

- **5000**: PFMEA_CHU_1
  - Charging System: Charger
  - Batteries don't charge
  - No power to the entire system
  - Circuit not closed
  - Check to see the main switch is on and the emergency stop in its off position