# Table of content

# List of figures



# List of Table

# Unit 1: INTRODUCTION

The globalization of the world's economies has provided both benefits and challenges to consumers and manufacturers. With an open global market, manufacturers have access to a stratified and larger supply chain with cheaper raw materials and even labor services. Unfortunately, such condition has also put a strain on the local manufacturing sector that now has to improve their competitiveness by boosting their productivity and in the process gain back market share.

There have been numerous advancements in various areas of technologies such as automation, computer aided engineering, information technology, etc. that have helped manufacturers reach their production goals. This project exposes a new are in the manufacturing sector which can help improve their production while minimizing their expenses. This method is Intelligent maintenance system.

Conventionally, the maintenance system used by manufacturers tend to be largely reactive. This is a corrective maintenance procedure carried out on an asset after breakdown. There is another proactive maintenance model called the preventive maintenance. This maintenance system schedules for maintenance practices on a periodic scale, like every 3 or 6 months. These two models tend to result in a lot of loss to the manufacturer. These losses might come in the form of downtime (asset not being able to function) or increased cost. The intelligent maintenance system tends to bridge these methods and produce a cost-effective model while also increasing production time.

## Goal

In this project, I developed an intelligent maintenance system for a factory asset based on its requirement and compliance policy. This contains a detailed knowledge base with the asset information. To make a prediction, the application takes into consideration various KPI's found in the knowledge base. It matches this with a set a matrix used as a neural network created from KPI intervals to determine the current state of the machine and make a prediction on when the asset should go on maintenance to return it back to its optimum performance.

Our main goal focuses on increasing production time through measurement and monitoring of its daily production, waste ratio and wear rate. The present condition of the machine I determined and a prediction is made based on the result of analysis.

## Objective function

In this project, I selected three performance indicators to hep me achieve my goal of maximizing production time. The are efficiency, Quality Tracking and Wear Rate.

$$f(P_t(Eff, QT, WR)) \rightarrow max$$
$$Eff = (H_p/M_p) \times 100$$
$$M_p = \frac{Speed}{12}; H_p = \frac{G_p}{24}$$
$$WR = k \times pressure \times Sd;$$
$$QT = (G_p/M_p) \times 100$$

Non controlled variable: Speed, $Sd$

Controlled carriable: Pressure, $G_p$ and $M_p$

Term definition
Speed is the speed of the asset;
$Sd$ is the sliding distance of surfaces in contact;
Pressure is the amount of force causing wear on asset;
$G_p$ is the amount of product that conform to standard;
$M_p$ is the possible maximum hourly production;
QT is quality rating of products from asset;
WR is wear rate;
Eff is the efficiency of this asset;
$H_p$ is average hourly production;
$P_t$ is production time;
K I the coefficient of wear for our asset.

## Tasks

There are several fundamental issues that have not been addressed by the manufacturing industry to advance its development and adoption.
There is a lack of a systematic methodology in the implementation of intelligent maintenance systems. Most research activities regarding intelligent maintenance found in literature are mostly ad-hoc in nature and there is a lack of validation in real factory environments. As a consequence, there is a challenge to propagate intelligent maintenance applications and tools to a factory-wide scale.

To achieve this goal of increasing production time, I considered the following;

- I had to survey all the possible reasons why production time reduces.
- I try to pick out the most important factors to measure and monitor.
- I try to determine with these factors, the state of the machine.
- Finally, I predict it should be due for maintenance.

To implement these, I collected sensor readings from various parts of the asset. These sensor readings help to generate a value for my important factors.

# Unit 2: APPROACH FOR SOLUTION

## IDEF0 Diagram

IDEF0 is a model that consists of a hierarchical series of diagrams, text, and glossary cross referenced to each other. The two primary **modeling** components are: functions (represented on a diagram by boxes), and. data and objects that interrelate those functions (represented by arrows).

A detailed pictorial description of the processes was carried out as IDEF0 diagrams. These diagrams show the various processes I used to achieve my goals. It also contains the resources as input used in this project. This is shown on the left side of each process. On the right-hand side of each process, it shows the output of the process. Sometimes, the output of one process serves as an input for another process, most likely, the next process. Shown below is my IDEF0 diagrams for my main process.

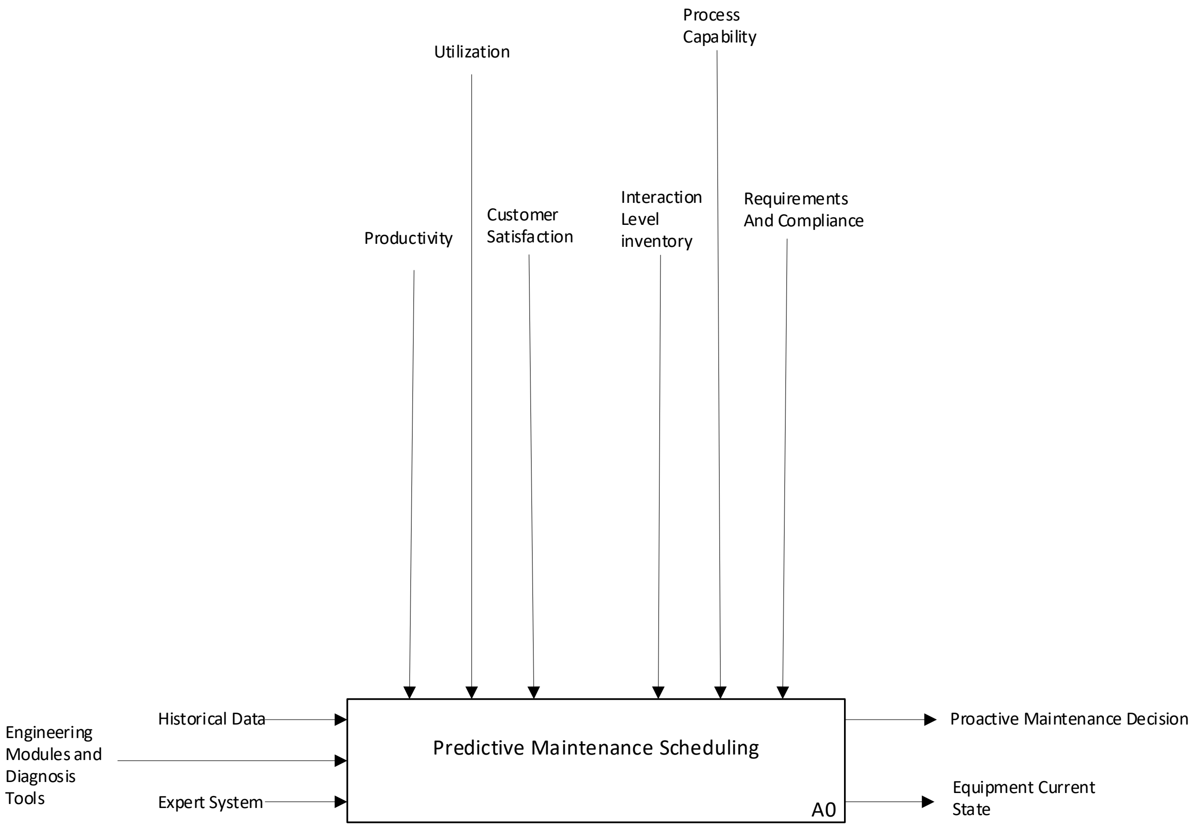


*Fig. 2.1. IDEF0 Diagram for Main Process*

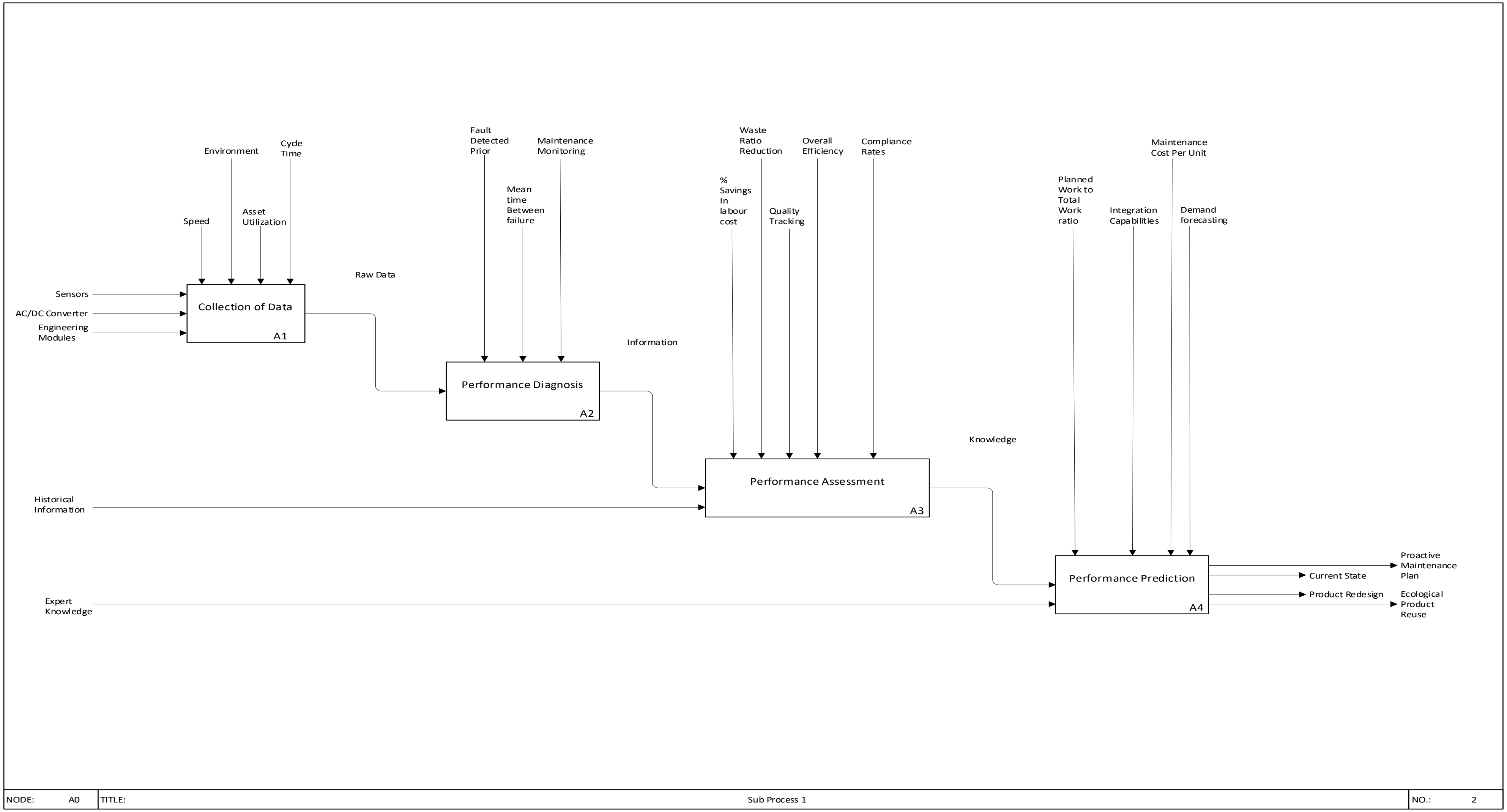


*Fig. 2.2. IDEF0 Diagram decomposing my main process*

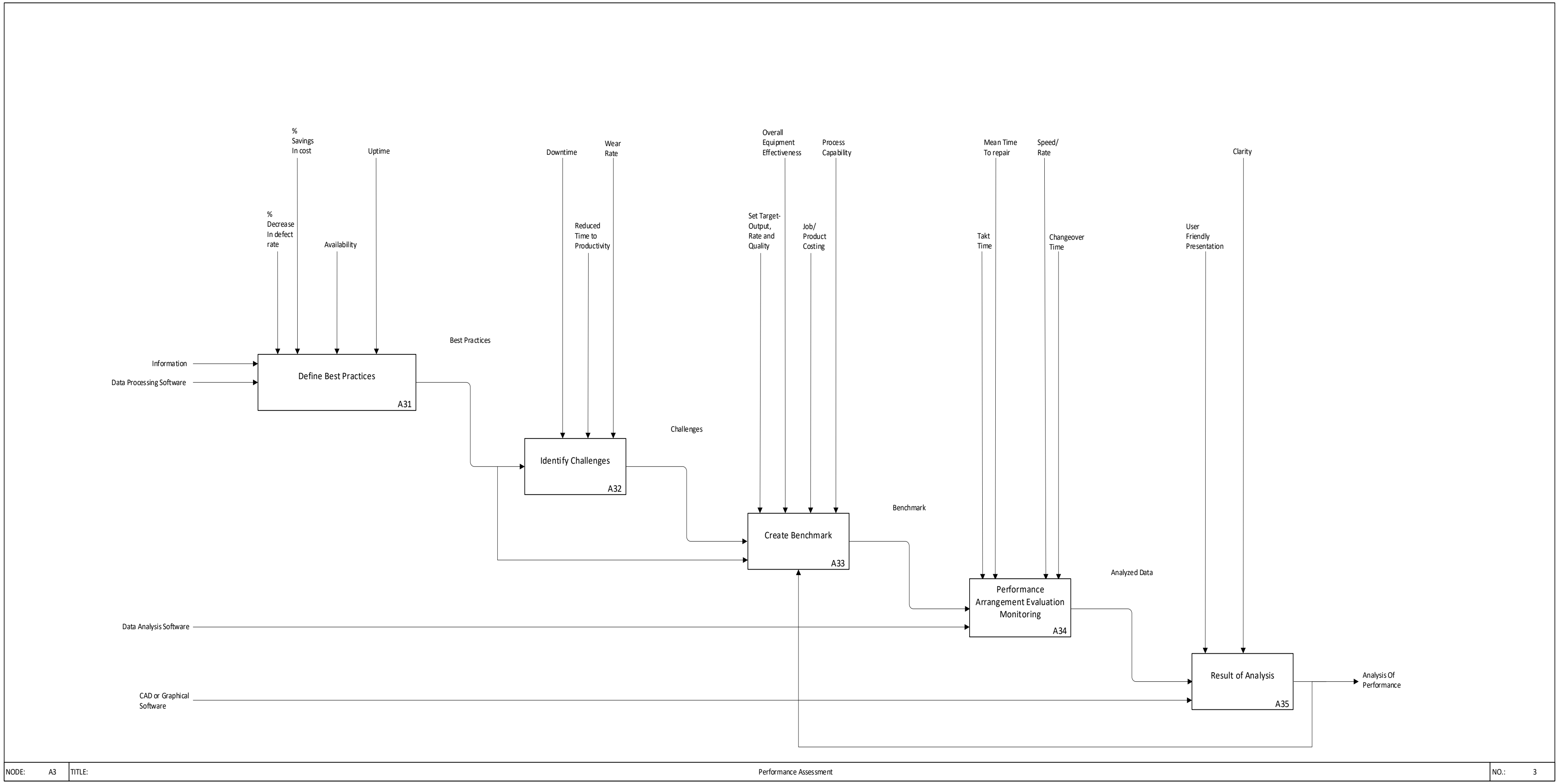


*Fig. 2.3. IDEF0 Diagram for Performance Assessment*

## SwimLane diagram

A swimlane diagram is a type of flowchart that delineates who does what in a **process**. Using the metaphor of lanes in a pool, a swimlane diagram provides clarity and accountability by placing **process** steps within the horizontal or vertical "swimlanes" of a particular employee, work group or department.

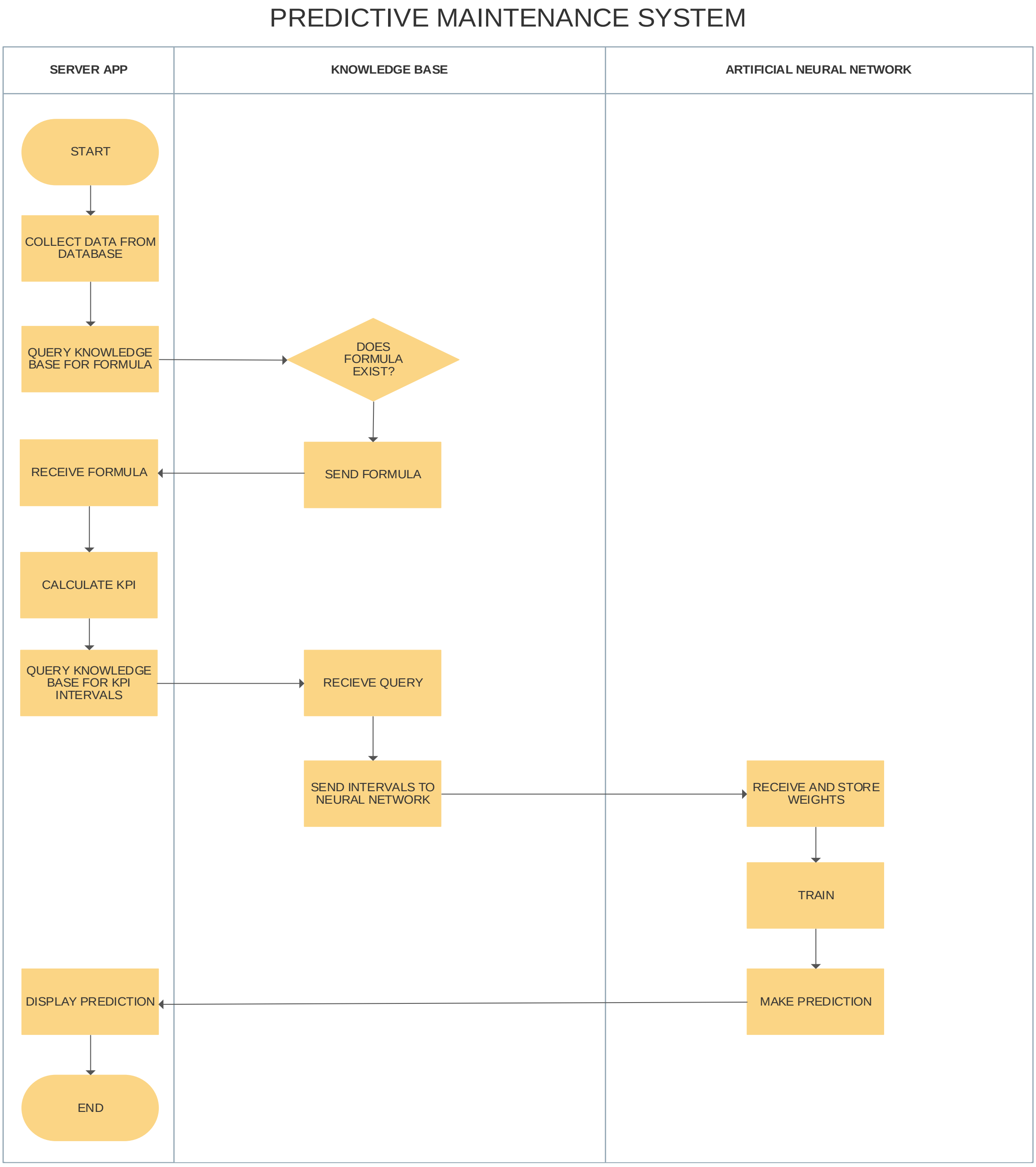


*Fig. 2.4. Swimlane diagram for Predictive Maintenance Scheduling*

## Uncertainty

A likely uncertainty is the situation uncertainty where there are insufficient details from the source or sensors but the algorithm makes the prediction based on the details provided. To solve this, the previous readings are used as a replacement.

## The Components needed for this project

### Knowledge base management System

The knowledge base for my project is stored in "Knowledge_Base.n3. In this file, all the information for my project can be gotten. The management system for my knowledge base is responsible for giving answers to queries from the algorithm. It also tells which function to use for each calculation.

For my knowledge base to be loaded into the project algorithm, I create a graph and a parser in my algorithm. Using this parse command, I then load my knowledge base into the algorithm.

```
using VDS.RDF.Query;
using VDS.RDF.Query.Builder;
using VDS.RDF.Parsing;

namespace NwiwuUzoma_Application
{

    }
    public partial class Form1 : Form
    {
        Graph Uzy = new Graph();
        Notation3Parser passe = new Notation3Parser();


        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            passe.Load(Uzy, @"Knowledge_Base.n3");
```

### Artificial Neural network

In this project, I worked with an artificial neural network that operates on binary threshold principle. Using the queries in my algorithm, I obtain the values for my input limits. I use these limits to form weights of the first layer neurons in my artificial neural network. To form these weights, I took the inverse of the limits gotten the knowledge base. For the minimum values, I used the positive inverse and for the maximum values, I used the positive inverse. The implementation is shown below;

```
private void getWeightValues(string[] MinVal, string[] MaxVal, double[] Weightmin,
double[] Weightmax)
        {
            for (int j = 0; j < MinVal.Length; j++)
            {
                double[] invW = new double[3];
                invW[j] = Convert.ToDouble(MinVal[j]);
                Weightmin[j] = 1 / (double)invW[j]; //derive the inverse of the limit
value
            }
```

```
            for (int j = 0; j < MaxVal.Length; j++)
            {
                double[] invW = new double[3];
                invW[j] = Convert.ToDouble(MaxVal[j]);
                Weightmax[j] = -1 / (double)invW[j]; //derive the inverse of the limit
value
            }
        }
```

The topology of my neural network is shown

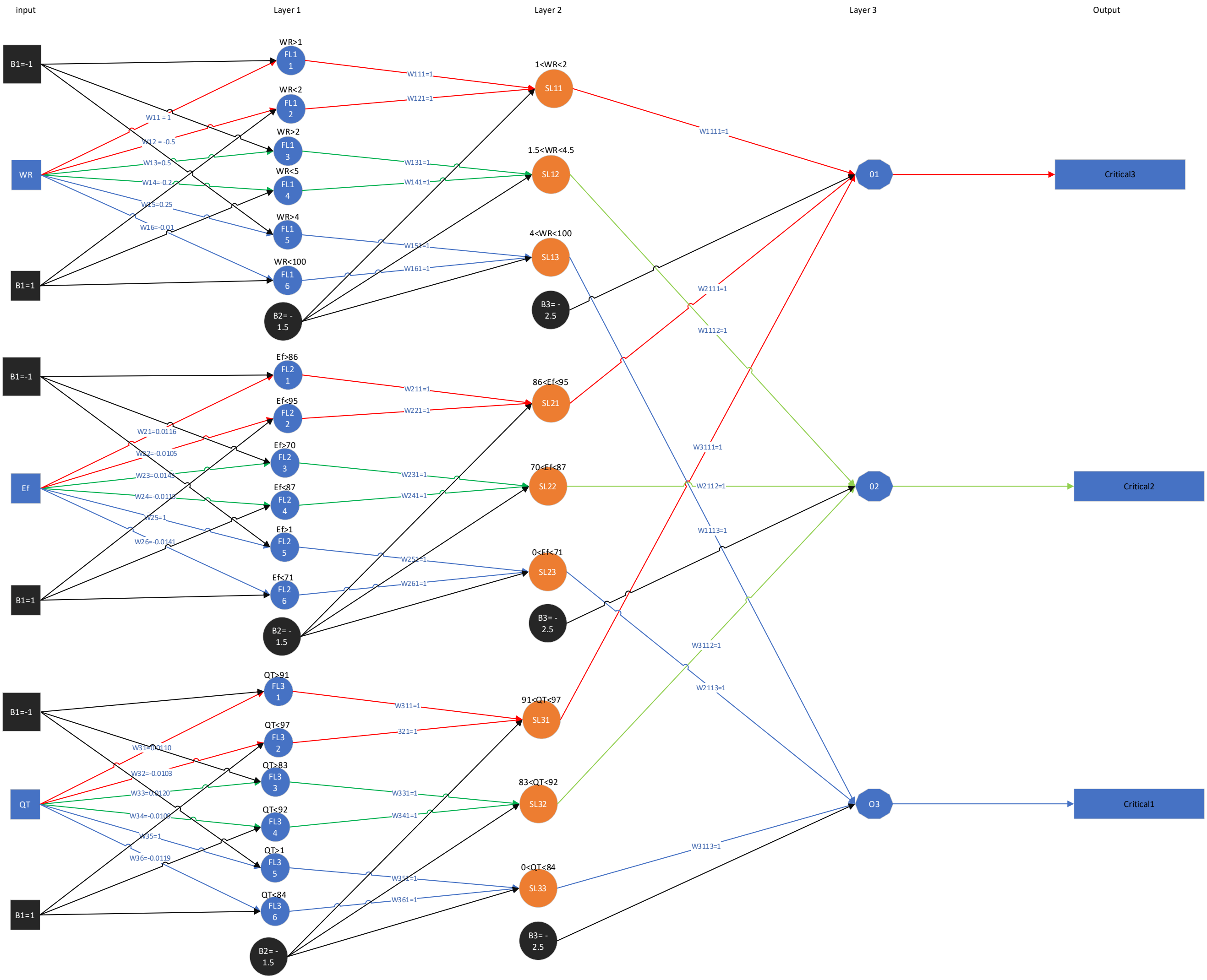


*Fig. 2.5. Network Topology for Artificial Neural Network*

The matrix of weight gotten from the limits stored in my algorithm is given below;

NNMAT= [ 1 -0.5 0.5 -0.2 0.25 -0.1
0.0116 -0.0105 0.0145 -0.0118 1 -0.0141
0.0110 -0.0103 0.0120 -0.0108 1 -0.0119]

For the first layer neurons, I used a value of +1 as bias for the maximum values and a value of -1 as bias for the minimum values. In the second layer neurons, I used a unity matrix as my weights and -1.5 as value of my bias. In the output layer, I used a unity matrix as my weight and -0.5 as the value of my bias. This is so that I can give priority attention to the most critical condition of my asset.

## Formula

In this project, I used a series of formulas to derive my key performance factors. These formula names are stored in the knowledge base. When they are needed, the algorithm sends a query to the knowledge base and it returns these formulas.

The formulas are written in python code. This is because of the flexibility and dynamism it brings to my code. An example is given below;

```python
def WearRate(Pressure, distance):
    k = 1.7 * (30 **-5)     #wear coefficient
    sumPressure = 0
    sumdistance = 0

    if len(Pressure) > 30:

        for i in range(len(Pressure) - 30, len(Pressure)):
            sumPressure += float(Pressure[i])

        for j in range(len(distance) - 30, len(distance)):
            sumdistance += float(distance[j])

        avPressure = sumPressure / 30.0
        avdistance = sumdistance / 30.0

    else:
        for i in range(len(Pressure)):
            sumPressure += float(Pressure[i])

        for j in range(len(distance)):
            sumdistance += float(distance[j])

        avPressure = sumPressure / len(Pressure)
        avdistance = sumdistance / len(distance)

    return k * avPressure * avdistance
```

To execute this code in C#, I used the IronPython library.

```csharp
using IronPython.Hosting;
using IronPython.Runtime;
using Microsoft.Scripting.Hosting;
```

With these libraries, I create a new Python Script Engine and a scope.

```csharp
private void useFormular_Click(object sender, EventArgs e)
        {

            ScriptEngine engine = Python.CreateEngine();
            ScriptScope scope = engine.CreateScope();
```

Subsequently, I can create calculate the values of my performance factors through by feeding in the scope values.
In my application, formulas can be changed to suite the prevailing conditions.

## Communication module

The communication module is used to exchange information. A TCP connection is part of my algorithm. This algorithm makes it possible for data to be collected from a client. This data can be used as a new formula or a model. It can also be used as knowledge or data.
To implement this, I followed the following steps

1. Create a server through the TCPListener function.
2. Define the address and Port number

```
using System.Net;
using System.Net.Sockets;                        //Import libraries
using System.IO;

namespace NwiwuUzoma_TCP
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        TcpListener server;                             //create server

        IPAddress address = IPAddress.Parse("192.168.11.181"); //define address
        int Port_Number = 1337;                                //ddefine port number

        private void Form1_Load(object sender, EventArgs e)
        {
            server = new TcpListener(address, Port_Number);

        }
```

With this in Place, we can start searching for a client for data exchange. Data gotten from this medium can be used and can influence the project.

## User Interface

This project uses the windows form as its user interface. On this interface, the user can view the process description, check my knowledge base for inconsistency, connect to a client and exchange data.

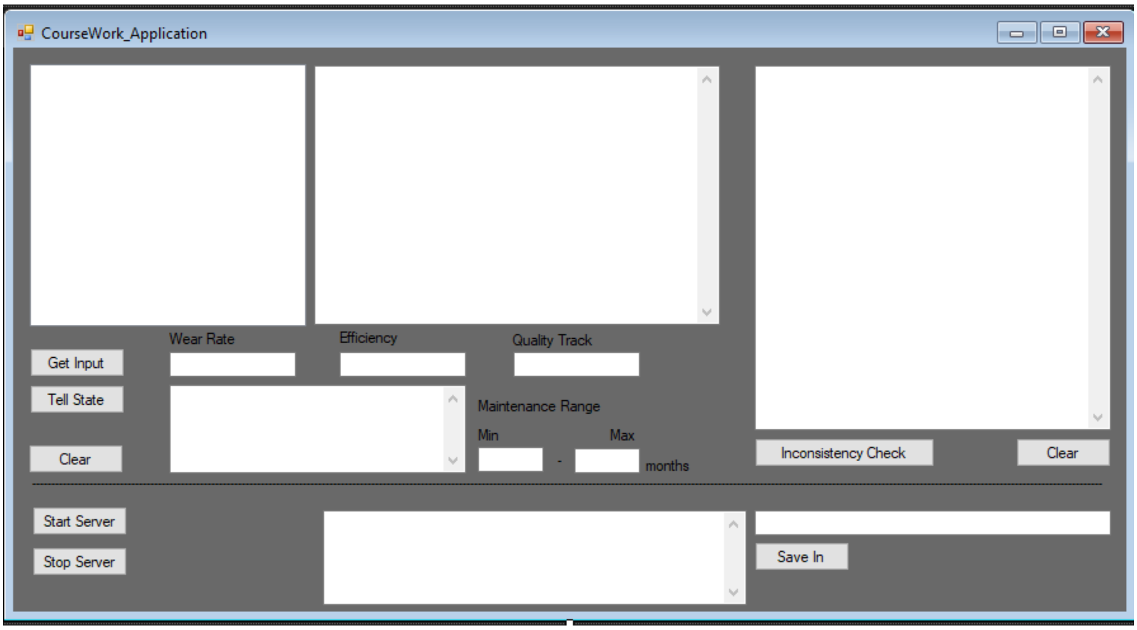


*Fig. 2.6. User Interface*

## Database management system

In this project, my sensor inputs are stored in my database. My sensor reading are stored in a csv file. To use this file, my algorithm uses this code

```
using (var rd = new StreamReader("KBE.csv"))
            {
                var splits = rd.ReadLine().Split(',');
                csv.Add(splits[0].Trim(), new List<double>());
                csv.Add(splits[1].Trim(), new List<double>());
                csv.Add(splits[2].Trim(), new List<double>());
                csv.Add(splits[3].Trim(), new List<double>());
                csv.Add(splits[4].Trim(), new List<double>());
                csv.Add(splits[5].Trim(), new List<double>());
                while (!rd.EndOfStream)
                {
                    splits = rd.ReadLine().Split(',');
                    csv["Daily Production"].Add(Double.Parse(splits[0]));
                    csv["Speed"].Add(Double.Parse(splits[1]));
                    csv["Pressure"].Add(Double.Parse(splits[2]));
                    csv["Sliding Distance"].Add(Double.Parse(splits[3]));
                    csv["Total Production"].Add(Double.Parse(splits[4]));
                    csv["Bad Production"].Add(Double.Parse(splits[5]));
                }
            }
```

Having created a dictionary called csv, I add the values of my database as list values using their heading as the key.

## Information Measurement System:

In my project, my measuring system consists of sensors at the infeed and discharge of the assets. These values are stored in the database and used to calculate the performance indicators.

# Unit 3: APPLICATION DEVELOPMENT

Developing the application was a gradual procedure starting from designing the IDEF0 diagram to writing the code. The process of application development is given below;

## 1. Problem Identification

This is the stage where I made a proposal and selection from different task this project. I chose this project because of my experience on a production line which highlight the importance of maintenance.

## 2. Designing the IDEF0 Diagram

After making the decision on task to work on, I began researching for process involved to accomplish this project. I made an extensive search and, in the end, I came up with the IDEFO diagrams shown in pages 5 – 8

## 3. Making the Knowledge Base

After having a detailed idea and visual representation of my idea in the IDEF0, I started writing my knowledge base. Firstly, I defined my prefixes like shown below;

```
@prefix ind:<URN:inds:>.
@prefix prop:<URN:props:>.
@prefix classes:<URN:class:>.
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
@prefix xml:<http://www.w3.org/2001/XMLSchema#integer>.
```

The main body of my knowledge base consisted of triples that defined the underlying factors in my project. Shown below is a representation of my main process

```
ind:PreventiveMaintenanceScheduling           a                   classes:MProcess;
                              rdfs:label              "Preventive Maintenance Scheduling";

                              prop:hasSubProcess ind:CollectionofData, ind:PerformanceDiagnosis,
ind:PerformanceAssessment, ind:PerformancePrediction;
                              prop:hasKPI                   ind:Productivity, ind:Utilization,
ind:CustomerSatisfaction, ind:InteractionLevelInventory, ind:ProcessCapability;
                              prop:hasInput                     ind:RequirementsandCompliance,
ind:EngineeringModelandDiagnosisTools, ind:ExpertSystem;
                              prop:hasOutput                    ind:ProactiveMaintenanceDecision,
ind:CurrentState.
```

This above illustration shows my main process. A complete triple is made up of a subject, predicate and object. In this example, my main process serves as the subject. All the predicates and objects listed describe the main process. This action was taken for all my process. My inputs, outputs and classes were also defined.

## 4. Process Description in C#:

In my application, there is always an interface where I keep my user updated on the processes involved in this maintenance system. This makes my client have a better understanding of the whole process. Every process has a detailed description of its activities with factors such as input, output and their performance indicators.

Developing this module, I used the tree-view approach on my interface. In the way, my client can click and view depending on choice. The following are steps u=involved in developing this

- I created a new graph called Uzy and a Notification3Parser called passe.

```
Graph Uzy = new Graph();
Notation3Parser passe = new Notation3Parser();
```

- I loaded my knowledge base into my system as a graph using this parsing command.

```
passe.Load(Uzy, @"Knowledge_base.n3");
```

- I sent a Sparql query from my algorithm to the knowledge base to return the name of my Main process.

```
string query_1 = @"

                prefix ind:<URN:inds:>
                prefix classes:<URN:class:>
                prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>
                    SELECT ?pr ?label
                       {
                              ?pr a classes:MProcess;
                                                      rdfs:label ?label.
                                         }";
       SparqlResultSet Nice1 = (SparqlResultSet)Uzy.ExecuteQuery(query_1);
```

- Using the main process as my root node, I queried it for sub processes.

```
string query_2 = "Prefix ind:<URN:inds:>" +
                "prefix classes:<URN:class:>" +
                "prefix prop:<URN:props:>" +
                "prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                "SELECT ?label ?subprocess " +
                "WHERE{ " +
                "<" + nodeUrn + "> prop:hasSubProcess ?subprocess." +
                "?subprocess rdfs:label ?label . }";

           passe.Load(Uzy, @"D:\Study material\SPBPU\Sem 2\KNOWLEDGE ENGINEERING AND
KNOWLEDGE MANAGEMENT\kb.n3");

           SparqlResultSet Nice2 = (SparqlResultSet)Uzy.ExecuteQuery(query_2);
```

- I created a loop to query the subprocesses until there is no subprocess anymore.

```
private bool GetNode(TreeNode treeNode, string nodeUrn)
```

```
        {
            string[] subLabels = new string[0];
            string[] subUrns = new string[0];
            List<string> subLabelsList = subLabels.ToList();
            List<string> subNodesList = subUrns.ToList();

            querySubNodes(nodeUrn, subLabelsList, subNodesList);

            subLabels = subLabelsList.ToArray();
            subUrns = subNodesList.ToArray();
            if (subLabels.Length == 1 && subLabels[0].Equals(""))
            {
                return false;
            }
            for (int i = 0; i < subLabels.Length; i++)
            {
                TreeNode currentNode = treeNode.Nodes.Add(subLabels[i]);
                currentNode.Tag = subUrns[i];
                GetNode(treeNode.Nodes[i], subUrns[i]);
            }
            return true;
        }
```

## 5. Deriving my Performance indicators using Formula

For an effective data handling process, I decided to write my formula codes in python. This is partly because of its dynamism. These formulas help to calculate the performance indicators using the values gotten from the database and predictions are made using the values gotten from these formulas. I incorporated these formulas into my C# code using IronPython.

I sent a query to the knowledge base which returned with the name of the of the formula to be used for each indicator. This is how the query looked like;

```
string GetStateQuery(string State) //this function calls the query
        {
            string query_1 =
                            "prefix ind:<URN:inds:>" +
                            "prefix prop:<URN:props:>" +
                            "prefix classes:<URN:class:>" +
                            "prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                    "select ?Inlabel ?label ?File " +
                        "where {" +
                             "<" + State + "> prop:hasFormular ?formular;" +
                             " rdfs:label ?label ." +
                             "?formular prop:hasInvar ?Invar;" +
                             " prop:usesFile ?File ." +
                             " ?Invar rdfs:label ?Inlabel.}";
            return query_1;
        }
```

### Concept mapping of formula

A conceptual model is a representation of a system, made of the composition of concepts which are used to help people know, understand, or simulate a subject the model represents. The conceptual model for each formula I shown below.

## Efficiency

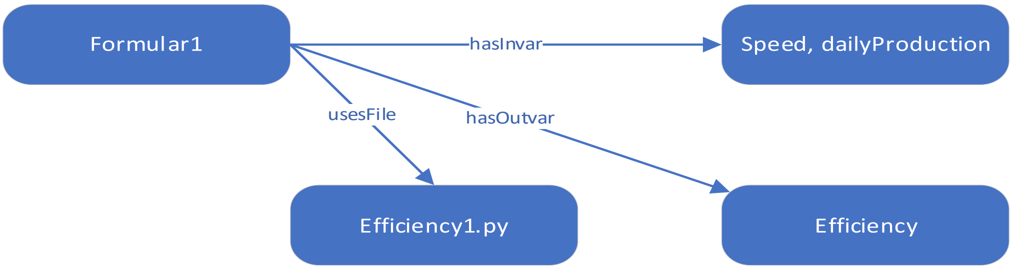


*Fig. 3.1. Conceptual Mapping for Formula1*

To calculate the Efficiency of my asset;

$$Max\ Prod = \frac{Speed}{12};$$

$$HourlyProd = \frac{DailyProd}{24}$$

$$Efficiency = (\frac{HourlyProd}{MaxProd}) \times 100$$

## Wear Rate

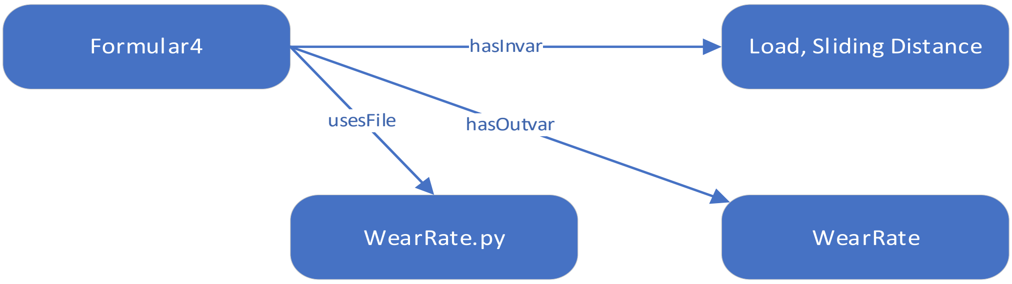


*Fig. 3.2. Conceptual Mapping for Formula3*

To calculate the wear Rate;

$$Wear\ Rate = k \times Load \times SlidingDistance$$

$$where\ k = wear\ coefficient = 1.7 \times 10^{-5}$$

## Quality Rating

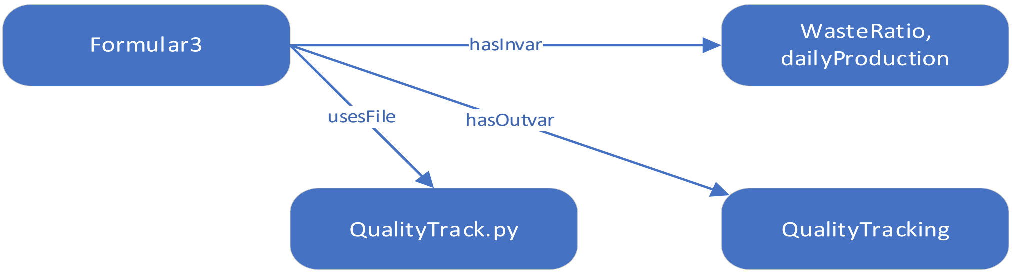


*Fig. 3.3. Conceptual Mapping for Formula3*

To calculate the Quality Rating;

$$Quality\ Rating = 1 - (\frac{BadProduct}{TotalProduct} \times 100)$$

## 6. Creating and Adapting my Artificial Neural network

After creating my formula, I planned a neural network using the limits of my performance indicators saved in the knowledge base. The query sent to the Knowledge base is shown below;

```
string GetStatQuery(string State) //this function calls the query
        {
            string query_1 =
                            "prefix ind:<URN:inds:>" +
                            "prefix prop:<URN:props:>" +
                            "prefix classes:<URN:class:>" +
                            "prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                    "select ?usage ?min ?max " +
                        "where {" +
                             "<" + State + "> prop:isCaused ?kpiInter ." +
                             "?kpiInter prop:hasUsage ?usage;" +
                             " prop:hasMin ?min;" +
                             " prop:hasMax ?max .}";
            return query_1;
        }
```

When this query is sent, it returns the limit values of the performance indicators. With these values, we implement the network shown on page. The limits for my performance indicators are shown below

Table 3.1. KPI Interval

| | KPI | Critical3(Low Priority) | Critical2(Medium Priority) | Critical1(High Priority) |
|---|---|---|---|---|
| 1 | Wear Rate | 1-2 | 2-5 | Above 5 |
| 2 | Efficiency | 86-94 | 70-87 | Below 71 |
| 3 | Quality Rating | 91-97 | 83-92 | Below 84 |

## 7. Inconsistency Check

This check helps us to monitor the knowledge base and identify inconsistencies. This is done by sending a query from my C# code to the knowledge base. To achieve this, I made a rule with triples that states the fact that I need to look out for in my knowledge base. These rules have my classes, datatype properties and object properties. An example is shown below

```
classes:MProcess              a                   owl:Class;
                  rdfs:subClassOf            classes:Process;
                  owl:oneOf         ind:PreventiveMaintenanceScheduling.
classes:MKPI                  a                   owl:Class.
```

| | | |
|---|---|---|
| classes:InitialInput | a | owl:Class. |
| classes:FinalOutput | a | owl:Class. |
| prop:hasInput | a | owl:ObjectProperty. |
| prop:hasOutput | a | owl:ObjectProperty. |
| prop:hasResult | a | owl:ObjectProperty. |
| prop:hasMin | a | owl:DatatypeProperty. |
| prop:hasUnit | a | owl:DatatypeProperty. |

Queries are sent to the Rules from the algorithm. The queries follow nine owl rule formats. An example is shown below

classes:NNVarKPI owl:disjointWith classes:NVarKPI.

This rule states that no individual should belong to classes NNVarKPI and NVarKPI. Its query is shown below;

```
public string SLRuleQuery1(string State) //this function calls the query
        {
            string query_1 =
                                "prefix ind:<URN:inds:>" +
                                "prefix prop:<URN:props:>" +
                                "prefix classes:<URN:class:>" +
                                "prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                                "prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
+
                                "prefix owl:<http://www.w3.org/2002/07/owl#>" +
                        "select ?termA ?termB " +
                            "where {" +
                                "?termA <" + State + "> ?termB .}";
            return query_1;
        }
```

In this query, “State” replaces “owl:disjointWith”. This query returns classes:NNVarKPI and classes:NVarKPI. This next query is sent to the main knowledge base to look out for these terms.

```
public string disjointWithRule(string obj1, string obj2) //this function calls the
query
        {
            string query_5 =
                                "prefix ind:<URN:inds:>" +
                                "prefix prop:<URN:props:>" +
                                "prefix classes:<URN:class:>" +
                                "prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                                "prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
+
                                "prefix owl:<http://www.w3.org/2002/07/owl#>" +
                        "select ?value " +
                            "where {" +
                                 "?value a <" + obj1 + ">;" +
                                 " a <" + obj2 + ">. }";
            return query_5;
        }
```

This looks into the knowledge base for inconsistencies.

## 8. Creating a Transfer protocol for Data Transfer

I implemented TCP model for my transfer protocol. Starting a server is done with a button on the user interface. In my algorithm, this server once started waits for a connection with a client. Once a connection is set, data can be transferred. This data can be saved and form a new formula or a model.

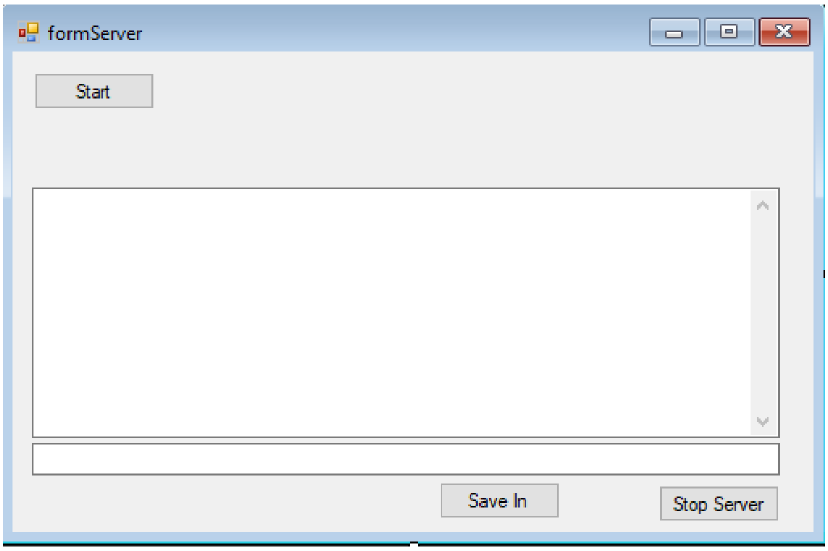


*Fig. 3.4. User Interface for Data Transfer*

## Output Data

I used Windows form to display my output. On this interface, various buttons help perform different functions. These functions have been described above.

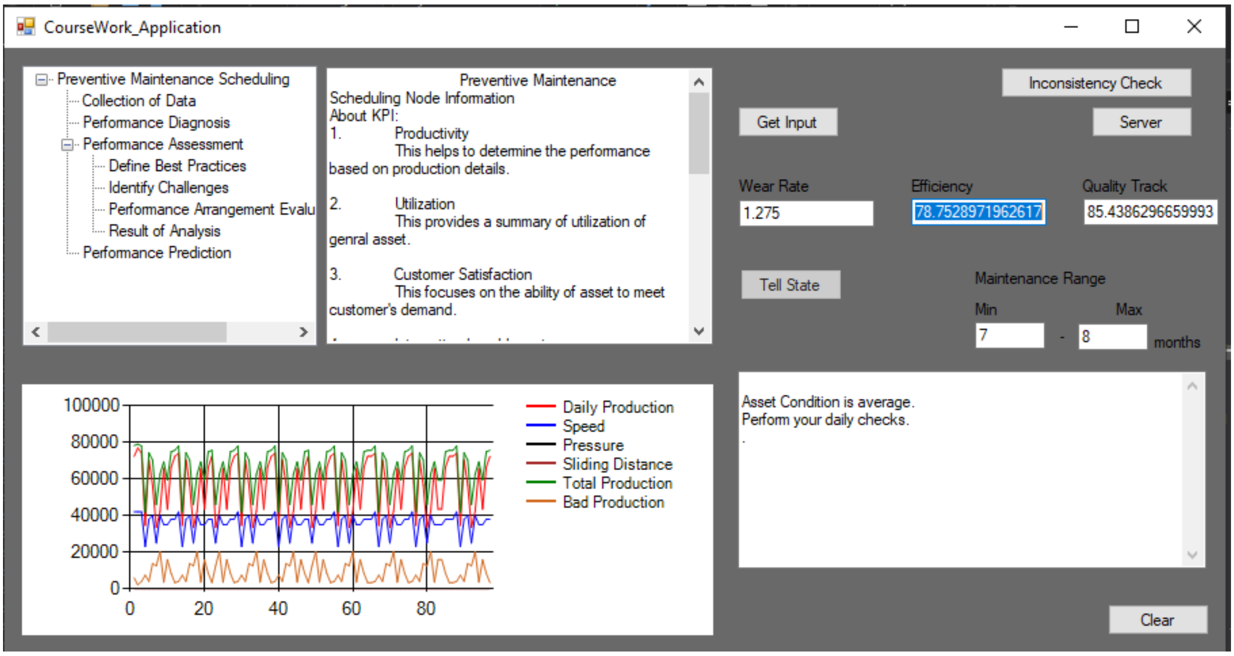


*Fig. 3.5. Output from my Application*

# CONCLUSION

Developing this application exposed me to many criticalities and factors considered in a factory setting while making plan for a predictive maintenance implementation. For different asset, the factors vary largely. I managed to extract three of the most important factors and use them in this project.

Having a proper check on these factors implement this algorithm and make a prediction on the best time to carry out maintenance practices on the machine. This helps to reduce the time spent in carrying out a preventive or corrective maintenance and also minimize the level of breakdown occurrence. A minimization of these two major effects increases production time. Having a knowledge of these numerous factors will help me in implementation of my Master's Thesis.

In general, my application coding skills have been greatly improved through this project. I acknowledge the help of my Professor who was a consistent help in all stages of this project development. I have gained a deeper and more valuable understanding of the basic in software development especially in C#.

A challenge I'm still having I in my notebook capability. It freezes anytime I turn on the data exchange server but the algorithm work fine on debug.