SENG 637
PROJECT ON
ONBOARD DIAGNOSTICS SYSTEM

SUBMITTED TO:
DR. BEHROZ FAR

SUBMITTED BY:
SUCHINA PARIHAR
ID: 10168164
Email: suchina.parihar2@ucalgary.ca

CONTENTS

INTRODUCTION

OBD-II stands for On-Board Diagnostics, II generation. It is a set of documents issued by SAE and ISO, which describe the interchange of digital information between on-board emission-related Electronic Control Units (ECUs) of road vehicles and an OBD-II scan tool. OBD-II also commonly refers to the physical on-board diagnostic system of a vehicle, which consists of an ECU (or multiple ECUs), Malfunction Indicator Light(MIL), Diagnostic Link Connector (DLC), and the wiring that connect the different elements[1] Onboard Diagnostics, OBD, technology benefits motorists, technicians and the environment by monitoring a vehicles performance every time it is driven, identifying performance and emissions problems immediately and providing technicians with information to help them quickly and accurately diagnose and repair malfunctions[2]
This project is made to design and develop a hardware and software solution for collecting, storing, and Transmitting data from the OBDII of a car, to a backend server; and providing the remote users access to the collected data through a web application. The Project ensures that the designed software matches the reliability standards. The system boundary is defined and the functionality of the system is tested on the basis of reliability and stability. Firstly, the system architecture is designed as shown in Figure.1

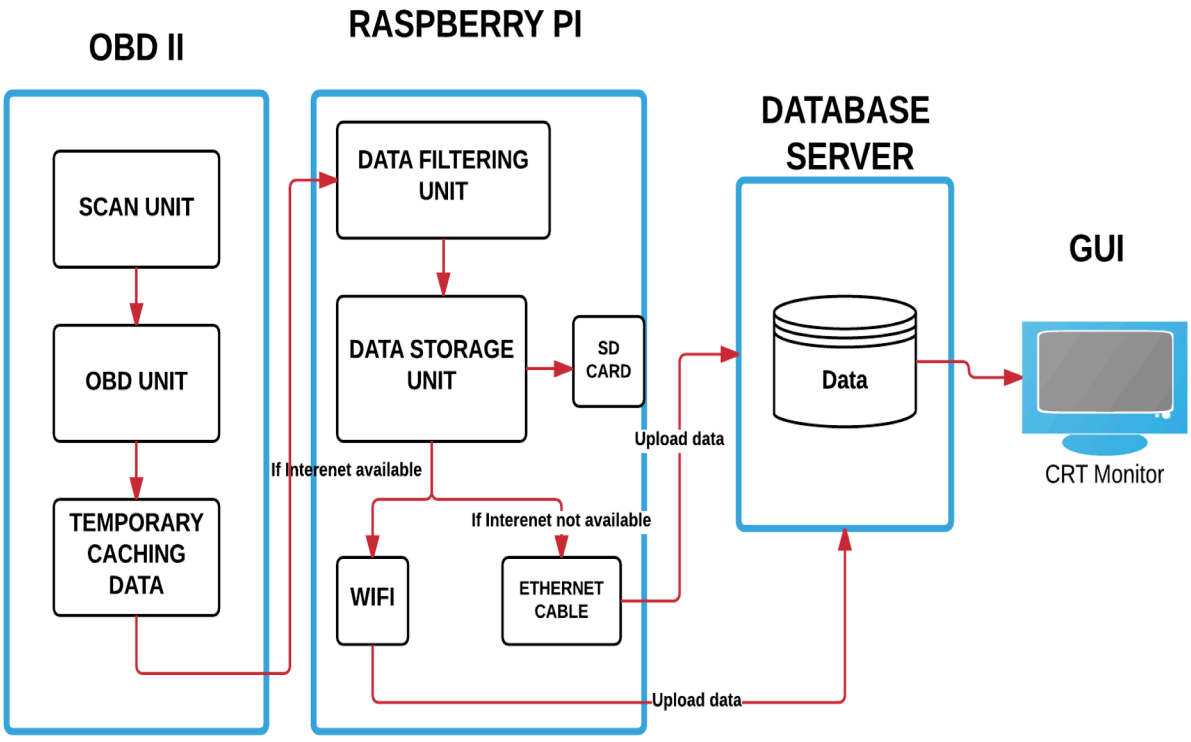


Figure 1. SYSTEM ARCHITECTURE

From Figure 1, the first module of the diagram is the OBDII module, which consists of 2 sub units. The SCAN UNIT scans the vehicle data (DTC codes) and sends it to the OBD unit for diagnostics. The OBD unit decodes the DTC codes and saves the data temporarily in TEMPORARY CACHING UNIT. This module is basically the data acquisition module where the data is gathered and decoded and saved. From this module the data is transferred to Raspberry Pi module. The Raspberry Pi has 2 sub units, one for FILTERING the data and the other for STORING the data. The data is stored in the SD Card in Raspberry Pi and is uploaded to the DATABASE server through WIFI, if internet is available or ETHERNET cable otherwise. The data can be retrieved from the DATABASE server by a USER through WEB APPLICATION. The USER can see the vehicle data through a GRAPHICAL USER INTERFACE. A few mock up figures are shown below just so to get an idea how our Web Application might look.

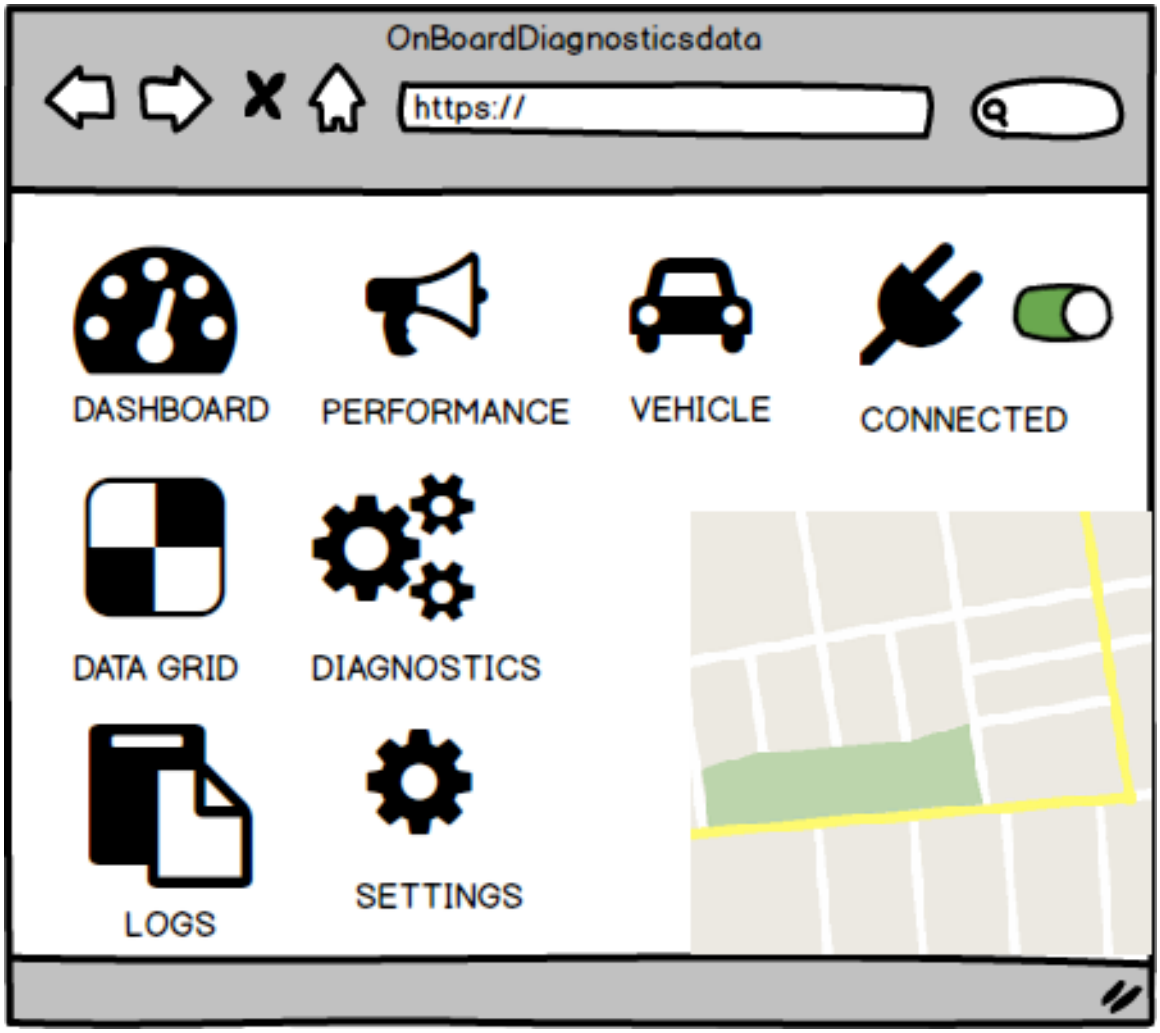


Figure 1a: MOCK UP OF THE WEB APPLICATION

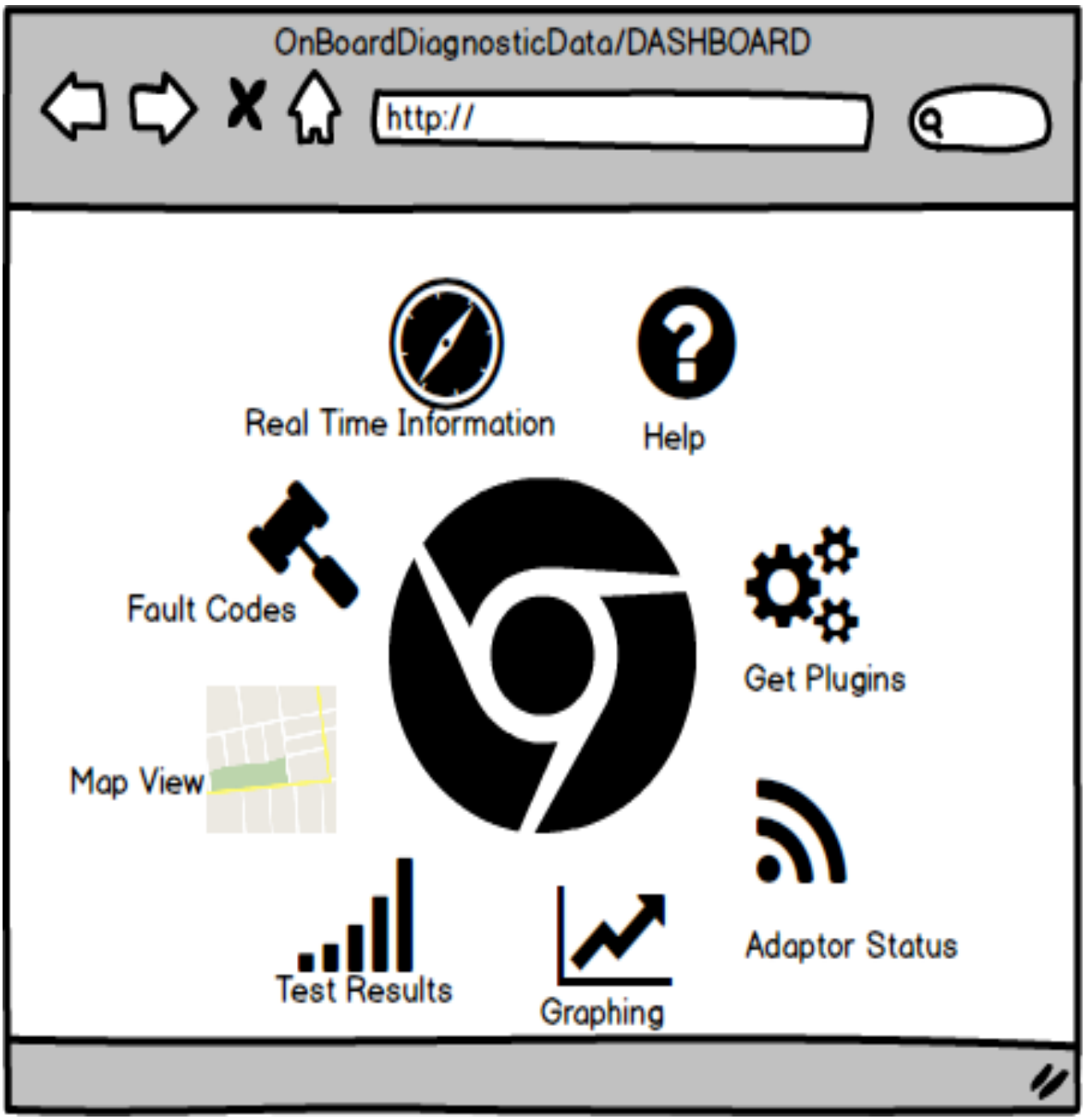


Figure 1b: MOCK UP OF THE WEB APPLICATION/DASHBOARD PAGE

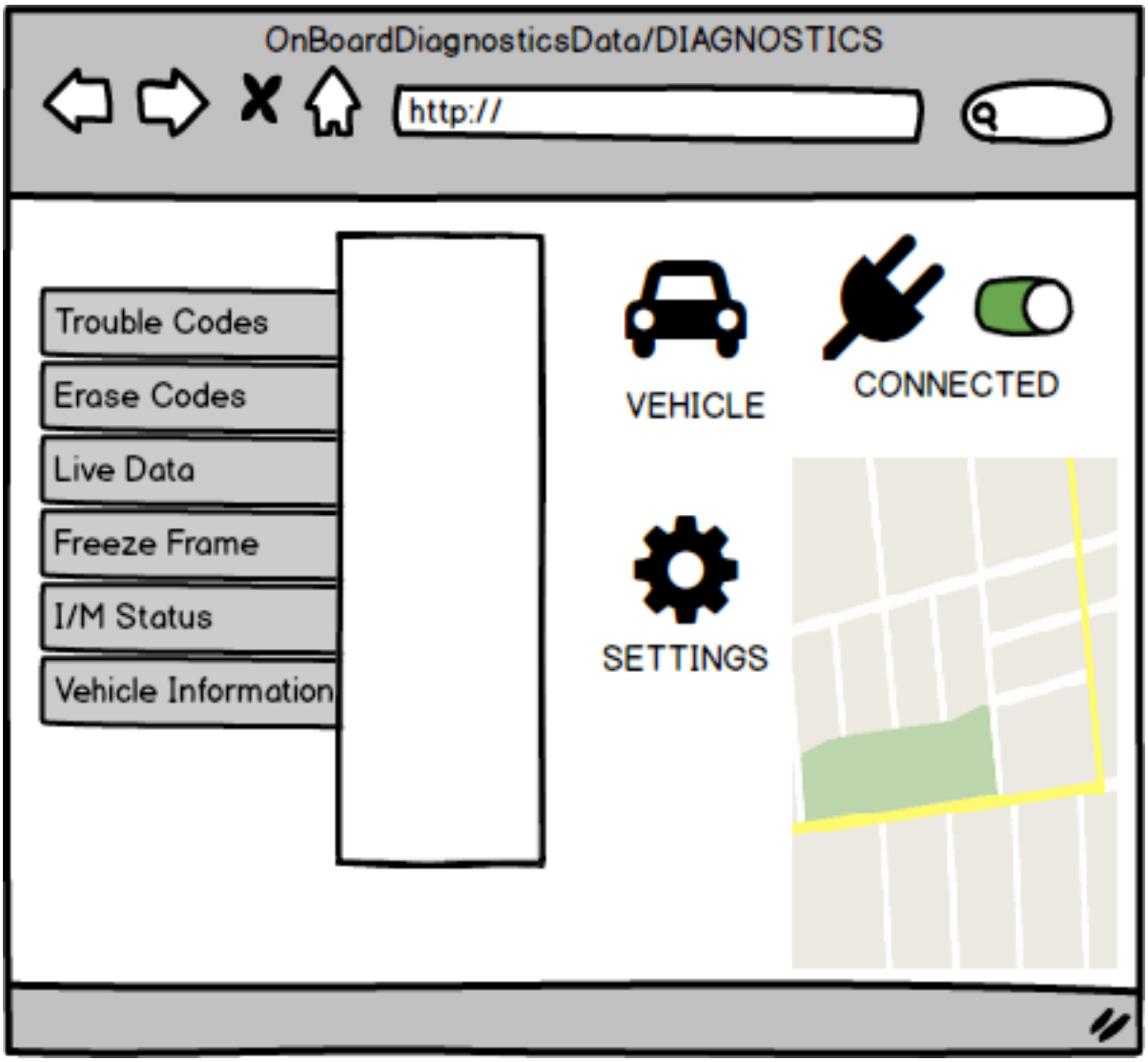
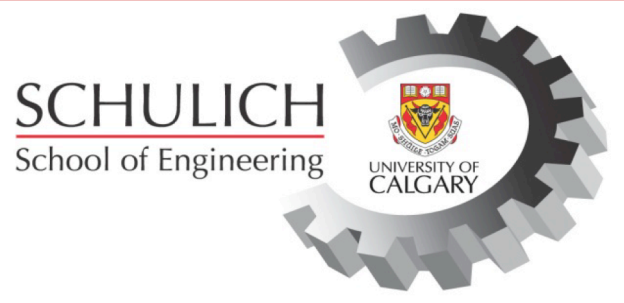


Figure 1c: MOCK UP OF THE WEB APPLICATION/DIAGNOSTICS PAGE



SENG 637
PROJECT ON
ONBOARD DIAGNOSTICS SYSTEM

ASSIGNMENT 1
SOFTWARE PROJECT SIZE

SUBMITTED BY:
SUCHINA PARIHAR
ID: 10168164
EMAIL:
suchina.parihar2@ucalgary.ca

EXECUTIVE SUMMARY

In this ASSIGNMENT, Project size will be calculated in terms of Function Points. This report focuses on the steps of how to calculate the Size of the Project. Firstly, the System boundary is defined. This helps to know the System in a better way. Secondly, the System is divided into different modules so as to understand the details of the system and calculate the size of different modules. In the end, the sizes of all the modules are combined together to get the size of the whole system. In the Step 1, to calculate the size of each module, we have to count External inputs(EI), External outputs(EO), External inquires(EQ), Internal Logic Files(ILF), External Interface Files(EIF) of that module. Then the weights are assigned to them and they are combined together and called Unadjusted Function Points (UFP).

An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system [3]. An external output (EO) is an elementary process that sends data or control information outside the application boundary The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information. The processing logic must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILFs, or alter the behavior of the system [3]. An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary [3]. An ILF is a user-identifiable group of logically related data or control information maintained within the boundary of the application [3]. An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application [3].

In the Step 2, Value Added Factors(VAF) are assigned to the modules. There are 14 Value Adjustment Factors and we have to value them from 0 to 5 for our system modules. The Value Adjustment Factors are given below:

1.Data Communications
2.Distributed Data Processing
3.Performance
4.Heavily Used Configuration
5.On-line Data Entry
6.End-User Efficiency
7.On-line Update
8.Complex Processing
9.Reusability
10.Installation Ease
11.Operational Ease
12.Multiple Sites
13.Facilitate Change
14. Transaction Rate

In Step 3, FUNCTION POINTS are calculated by this formula: FP = UFP * VAF and the Function Points of all modules are added together.

PROJECT SIZE

The System is divided into 9 modules:
MODULE 1: SCAN UNIT
MODULE 2: OBD UNIT
MODULE 3: TEMPORARY CACHING DATA UNIT
MODULE 4: FILTERING UNIT
MODULE 5: STORAGE UNIT
MODULE 6: WIFI UNIT
MODULE 7: ETHERNET UNIT
MODULE 8: DATABASE UNIT
MODULE 9: USER INTERFACE UNIT

SCAN UNIT

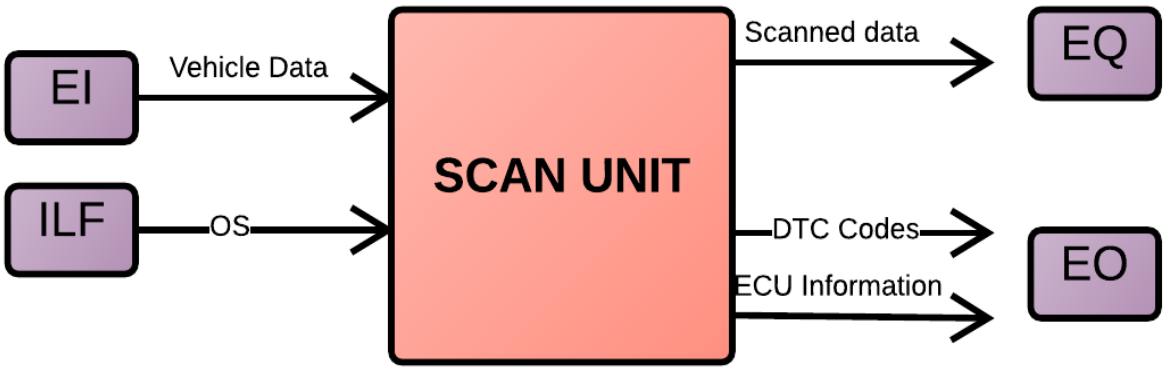


Figure 2. SCAN UNIT

UNADJUSTED AND UNWEIGHTED FP:

#EI = 1
#EO = 2
#EQ = 1
#ILF = 1

UNADJUSTED FP (UFP):

3EI + 4EO + 3EQ + 7ILF = 21

VAF:

0.65 + 0.05 + 0.04 + 0.05 + 0.02 + 0.04 + 0.01 + 0.03 + 0.05 = 0.94

FP: 19.74

OBD UNIT

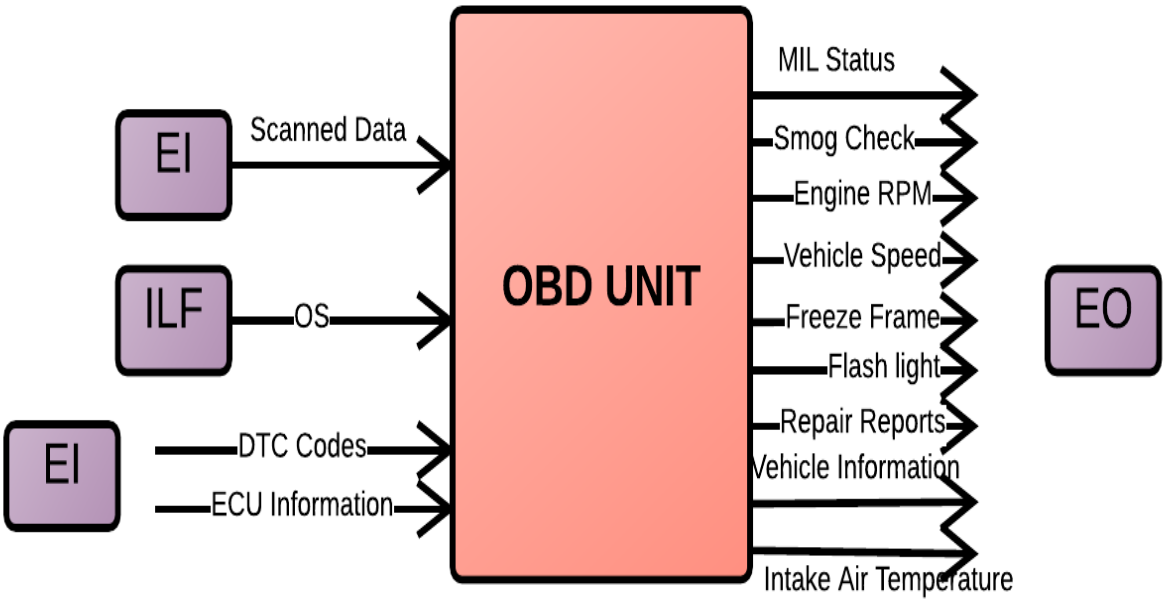


Figure 3: OBD UNIT

UNADJUSTED AND UNWEIGHTED FP:

#EI = 3
#EO = 10
#ILF = 1

UNADJUSTED FP(UFP):

56

VAF:

1.01

FP: 56.56

TEMPORARY CACHING UNIT

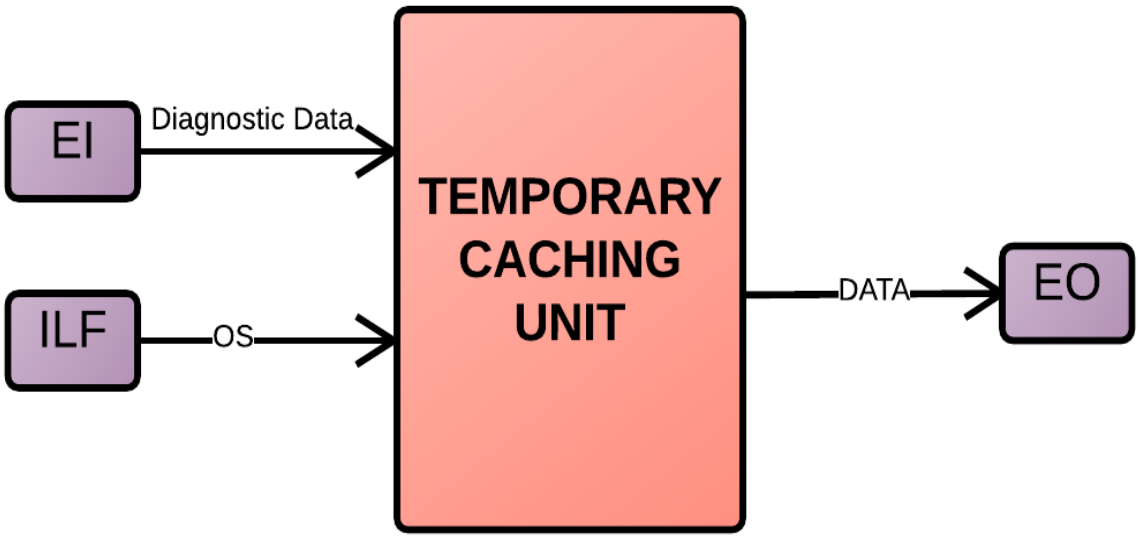


Figure 4. TEMPORARY CACHING UNIT

UNADJUSTED AND UNWEIGHTED FP:

#EI = 2
#EO = 1
#ILF = 1

UNADJUSTED FP(UFP):

17

VAF:

1

FP: 17

DATA FILTERING UNIT

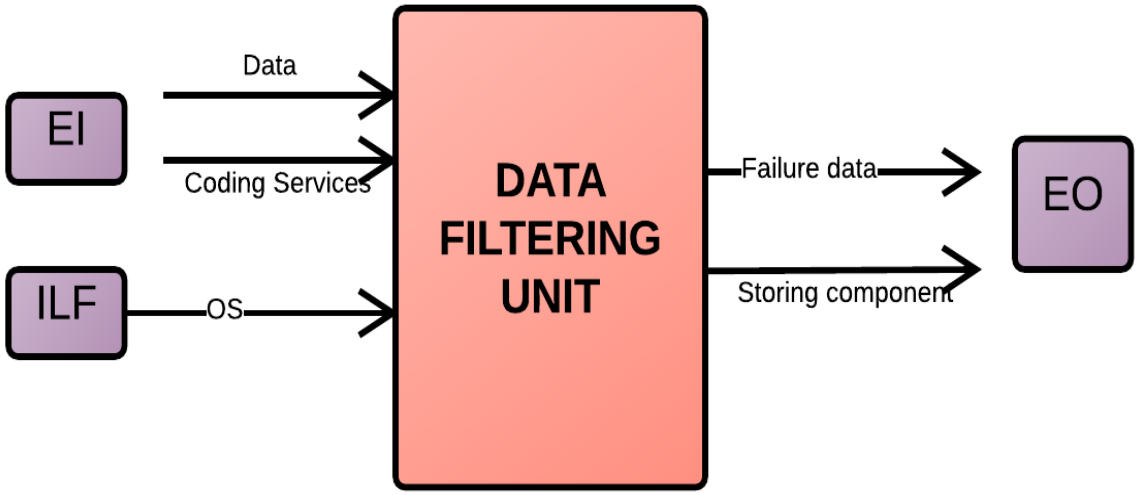


Figure 5. DATA FILTERING UNIT

UNADJUSTED AND UNWEIGHTED FP:

#EI = 2
#EO = 2
#ILF = 1

UNADJUSTED FP(UFP):

21

VAF:

1.02

FP: 21.42

DATA STORAGE UNIT

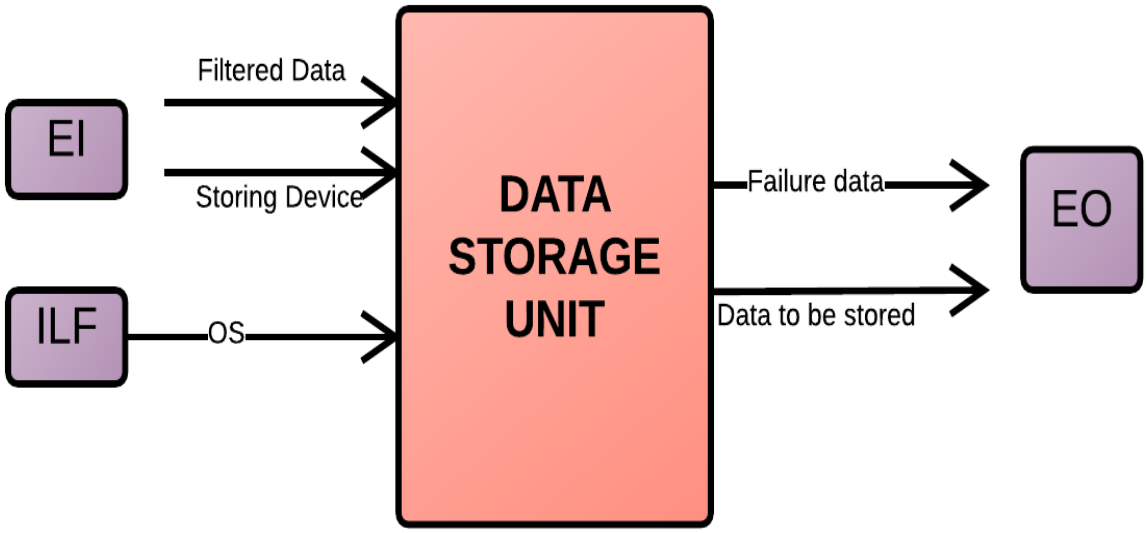


Figure 6. DATA STORAGE UNIT

UNADJUSTED AND UNWEIGHTED FP:

#EI = 2
#EO = 2
#ILF = 1

UNADJUSTED FP(UFP):

21

VAF:

1.02

FP: 21.42

WIFI UNIT

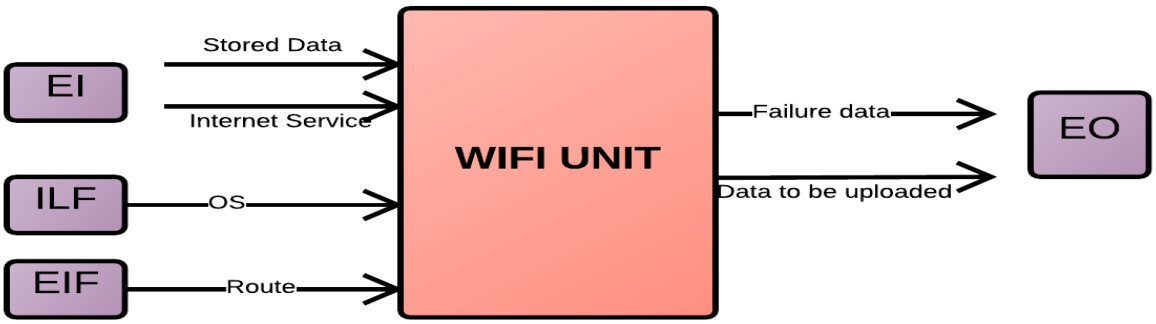


Figure 7. WIFI UNIT

UNADJUSTED AND UNWEIGHTED FP:
#EI = 2 #EO = 2 #ILF = 1 #EIF = 1

UNADJUSTED FP(UFP): 26
VAF: 1.1
FP: 28.6

ETHERNET UNIT

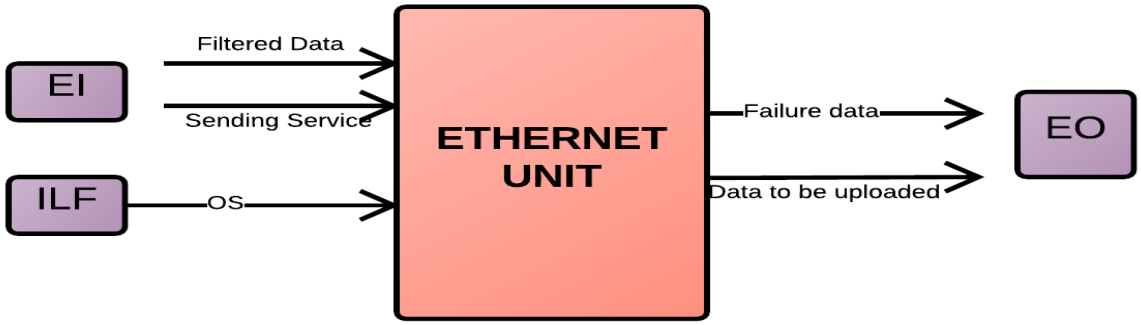


Figure 8. ETHERNET UNIT

UNADJUSTED AND UNWEIGHTED FP:
#EI = 2 #EO = 2 #ILF = 1

UNADJUSTED FP(UFP): 21
VAF: 1.21
FP: 25.41

DATABASE UNIT

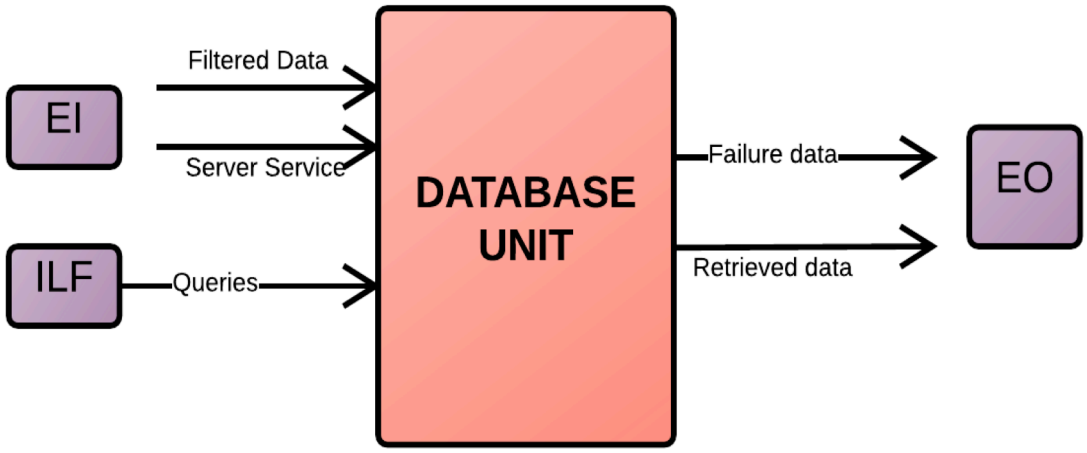


Figure 9. DATABASE UNIT

UNADJUSTED AND UNWEIGHTED FP:
#EI = 2 #EO = 1 #ILF = 1

UNADJUSTED FP(UFP): 17

VAF: 0.92

FP: 15.64

WEB APPLICATION UNIT

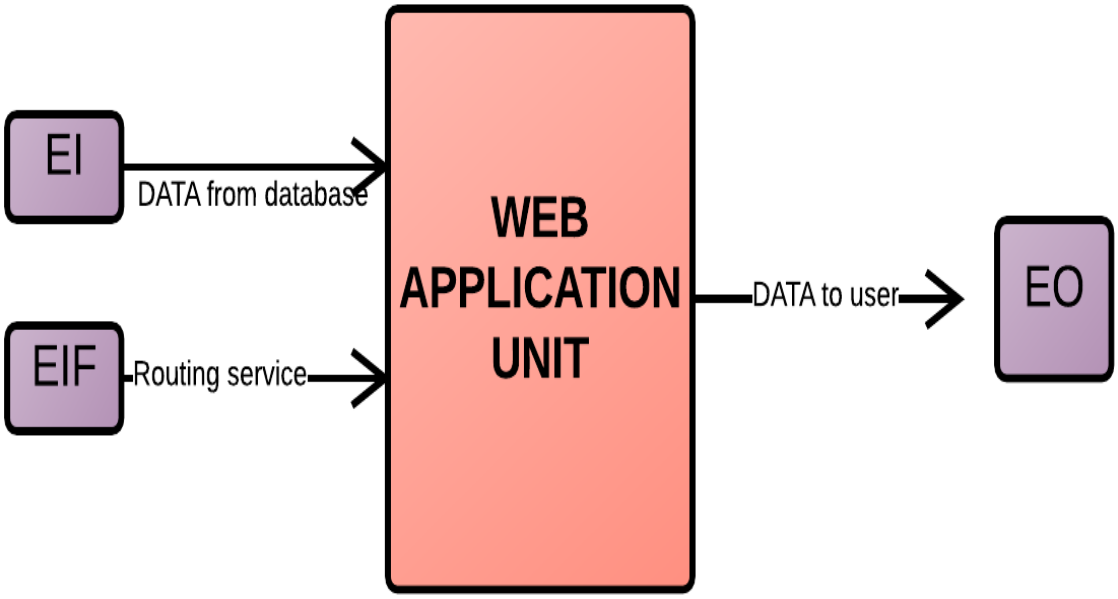


Figure 10. WEB APPLICATION UNIT

UNADJUSTED AND UNWEIGHTED FP:
#EI = 1 #EO = 1 #ILF = 1

UNADJUSTED FP(UFP): 12

VAF: 1.11

FP: 13.32

TOTAL FUNCTION POINTS

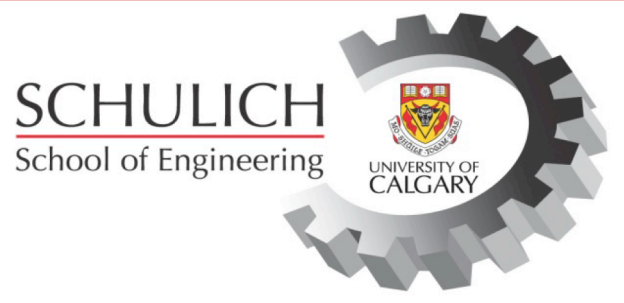
Total function points = 219.11

CONCLUSION

In this Assignment, Project size is calculated. There are various means to calculate a project size, through Function Points, Object Points, Feature Points etc. The function point count is a measure of project size from a functionality point of view. In my Project, the total Function Points are 219.11. Most of the complexity of the project is a result of the need to support multiple, simultaneous online transactions in real time. This complexity is taken into account in the VAF, which has a value slightly above 1. Assuming that the majority of the project will be implemented in JAVA, the project is expected to have over 6800 lines of code. We can deduce from this data that the project is relatively small and does not require significant effort by industry standards. It seems that the significance of function points is rather abstract. Unless one has access to a significant amount of relevant project data from which to derive relationships, it is difficult to convert this function point result to a concrete measurement of effort. During the process of function point analysis, it was evident that in order to accurately measure the function point count, the project specification needs to be well defined. Otherwise, the inaccuracy in the specification will deteriorate the correctness of the function point count, making it irrelevant. As a consequence, a significant amount of time was spent creating the project specification before beginning the function point count. It is clear, however, that this remains an issue in real-world projects, where requirements, even if well defined, are constantly changing. Overall, the assignment provided a good opportunity to understand the complexity of function point count from a more practical aspect. However, it would be interesting to be able to place the project in context of similar projects to give the final result a more concrete meaning. Function point analysis is primarily meant to be useful in understanding the size of the project and the effort involved. However, it was also quite useful in developing a greater understanding of the scope of the project. It made me break the system into smaller and manageable-sized components, to identify varying levels of datasets and how these interact inside and outside the system, and it helped developers to consider usability and performance aspects of project complexity- considerations that could potentially improve the final product if noted before the design stage.

REFERENCES

[1] http://www.onboarddiagnostics.com/download/u581.pdf
[2] http://www.nyc.gov/html/tlc/downloads/pdf/intro_to_obd2.pdf
[3] B. H. Far "Seng 637" Software Reliability & software quality Chapter 2 [online] Department of Electrical & Computer Engineering, University of Calgary



SENG 637
PROJECT ON
ONBOARD DIAGNOSTICS SYSTEM

ASSIGNMENT 2
SOFTWARE PROJECT TIME&EFFORT

SUBMITTED BY:
SUCHINA PARIHAR
ID: 10168164
EMAIL:
suchina.parihar2@ucalgary.ca

EXECUTIVE SUMMARY

In this ASSIGNMENT, Project effort, time and cost are to be calculated. For this purpose we are using COCOMO II model. COCOMO means COST CONSTRUCTIVE MODEL. Software cost estimation is the process of predicting the amount of effort required to build a software system and time to develop it. Models provide mathematical algorithms to compute cost as a function of a number of variables such as size (using function points) and/or complexity (using cyclomatic complexity, etc.)[4].

COCOMO II includes three-stage series of models:

- The earliest phases will generally involve prototyping, using the ***Application Composition*** model capabilities. [4]
- The next phases will generally involve exploration of architectural alternatives or incremental development strategies. To support these activities, COCOMO II provides an early estimation model called the ***Early Design model***. [4]
- Once the project is ready to develop and sustain a fielded system, it should have a life-cycle architecture, which provides more accurate information on cost driver inputs, and enables more accurate cost estimates. To support this stage, COCOMO II provides the ***Post-Architecture model***. [4]

We are using the Early Design Model and Post Architecture Model for our Project. For COCOMO II model, we use **Cost Drivers** for Early Design Model and **Scale Factors** for Post Architecture Model.

	Cost Driver	***Description***
1	**RCPX**	Product reliability and complexity
2	**RUSE**	Required reuse
3	**PDIF**	Platform difficulty
4	**PERS**	Personnel capability
5	**PREX**	Personnel experience
6	**FCIL**	Facilities
7	**SCED**	Schedule

TABLE 1. COST DRIVERS for Early Design Model

EXECUTIVE SUMMARY

Scale Factor SF		*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
Precedentedness Degree to which system is new and past experience applies	**PREC**	6.20	4.96	3.72	2.48	1.24	0.00
Development/Flexibility Need to conform to specified requirements	**FLEX**	5.07	4.05	3.04	2.03	1.01	0.00
Architecture/Risk Resolution Degree of design thoroughness and risk elimination	**RESL**	7.07	5.65	4.24	2.83	1.41	0.00
Team Cohesion Need to synchronize stakeholders and minimize conflict	**TEAM**	5.48	4.38	3.29	2.19	1.10	0.00
Process Maturity SEI CMM process maturity rating	**PMAT**	7.80	6.24	4.68	3.12	1.56	0.00

TABLE 2. SCALE FACTORS for Post architecture Model

COCOMO II RESULTS(EARLY DESIGN)

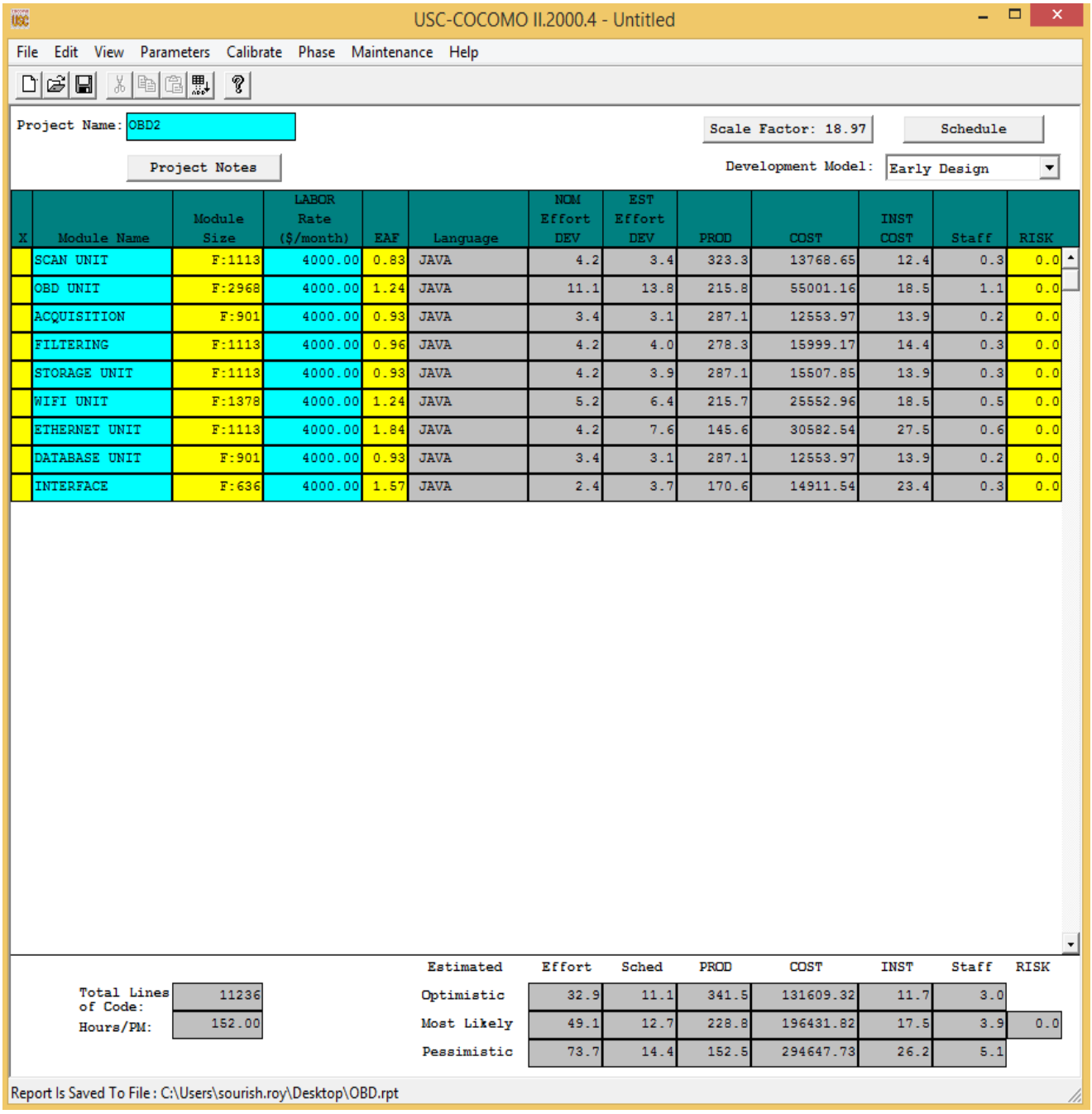


Figure 11. COCOMO II Results for EARLY DESIGN Model

COCOMO II RESULTS(EARLY DESIGN)

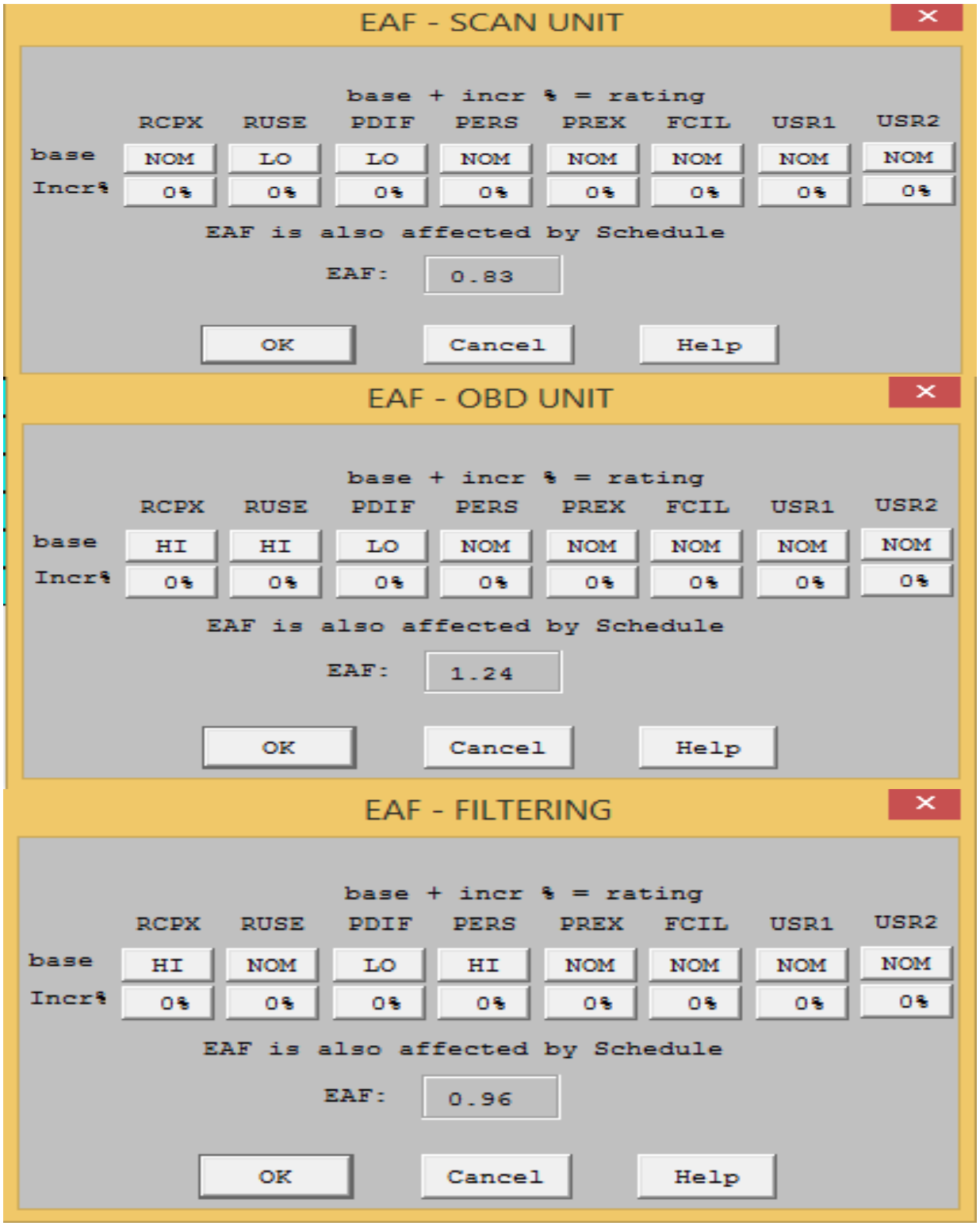


Figure 12. EAF for EARLY DESIGN Model

COCOMO II RESULTS(EARLY DESIGN)

Figure 13. EAF for EARLY DESIGN Model

COCOMO II RESULTS(EARLY DESIGN)

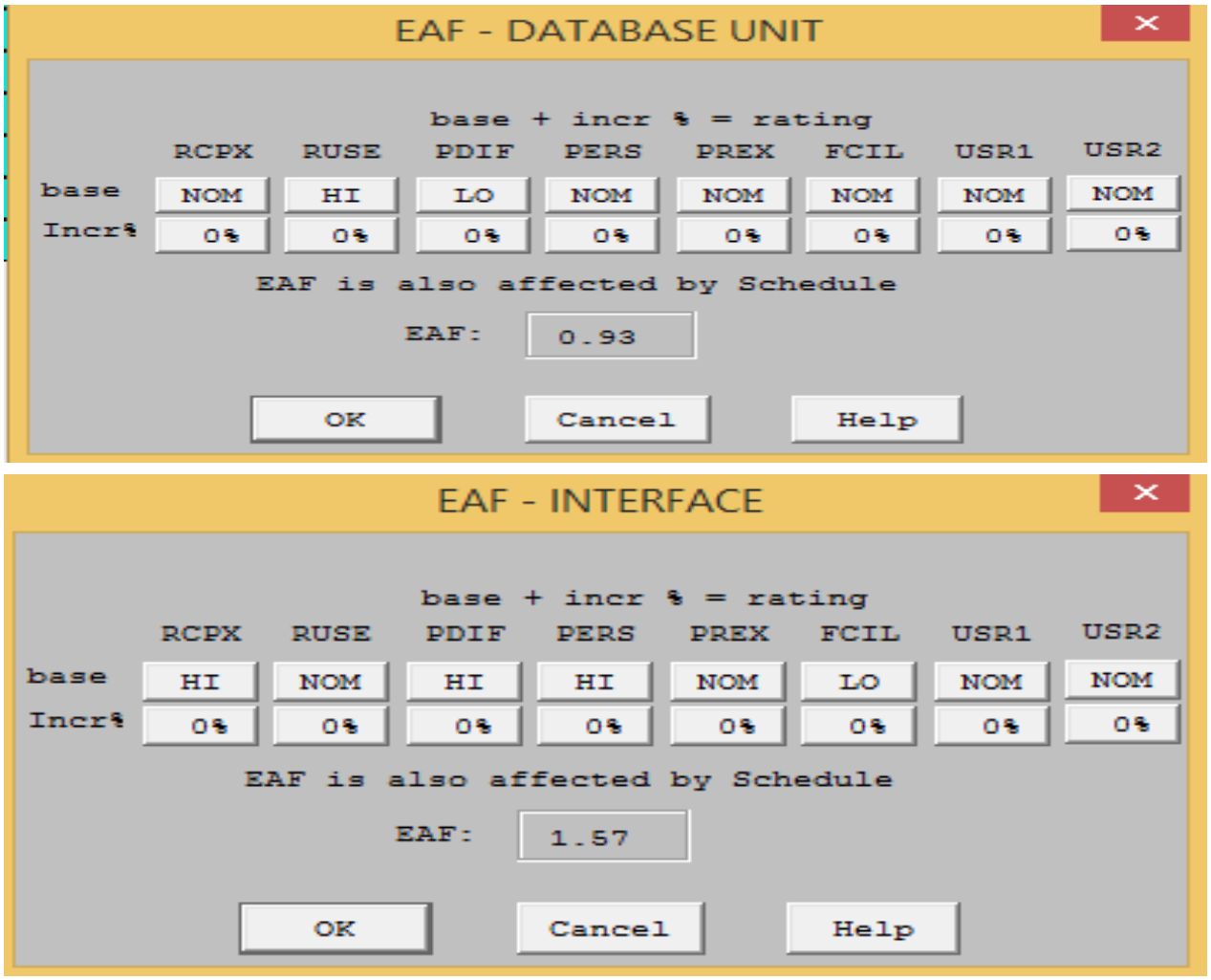


Figure 14. EAF for EARLY DESIGN Model

COCOMO II RESULTS (POST ARCHITECTURE)

Project Name: obd_post

Scale Factor | Schedule

Development Model: Post Architecture

X	Module Name	Module Size	LABOR Rate ($/month)	EAF	Language	NOM Effort DEV	EST Effort DEV	PROD	COST	INST COST	Staff	RISK
	SCAN UNIT	F:1113	4000.00	1.13	JAVA	4.2	4.7	235.7	18891.29	17.0	0.4	1.7
	OBD UNIT	F:2968	4000.00	1.09	JAVA	11.1	12.2	244.2	48617.53	16.4	0.9	1.0
	ACQUISITION UNIT	F:901	4000.00	1.19	JAVA	3.4	4.0	224.4	16057.59	17.8	0.3	1.7
	FILTERING	F:1113	4000.00	1.13	JAVA	4.2	4.7	235.7	18891.29	17.0	0.4	1.7
	STORAGE UNIT	F:1113	4000.00	1.47	JAVA	4.2	6.1	181.3	24558.67	22.1	0.5	1.7
	WIFI UNIT	F:1378	4000.00	1.26	JAVA	5.2	6.5	212.3	25962.02	18.8	0.5	1.7
	ETHERNET UNIT	F:1113	4000.00	1.66	JAVA	4.2	6.9	160.6	27715.28	24.9	0.5	1.7
	DATABASE UNIT	F:901	4000.00	1.63	JAVA	3.4	5.5	163.6	22032.41	24.5	0.4	1.7
	INTERFACE	F:636	4000.00	1.68	JAVA	2.4	4.0	158.9	16014.41	25.2	0.3	1.7

Total Lines of Code: 11236

Estimated	Effort	Sched	PROD	COST	INST	Staff	RISK
Optimistic	43.7	12.2	256.8	174992.40	15.6	3.6	
Most Likely	54.7	13.1	205.5	218740.50	19.5	4.2	14.5
Pessimistic	68.4	14.1	164.4	273425.62	24.3	4.9	

Figure 15. COCOMO II Results for POST ARCHITECTURE Model

COCOMO II RESULTS (POST ARCHITECTURE)

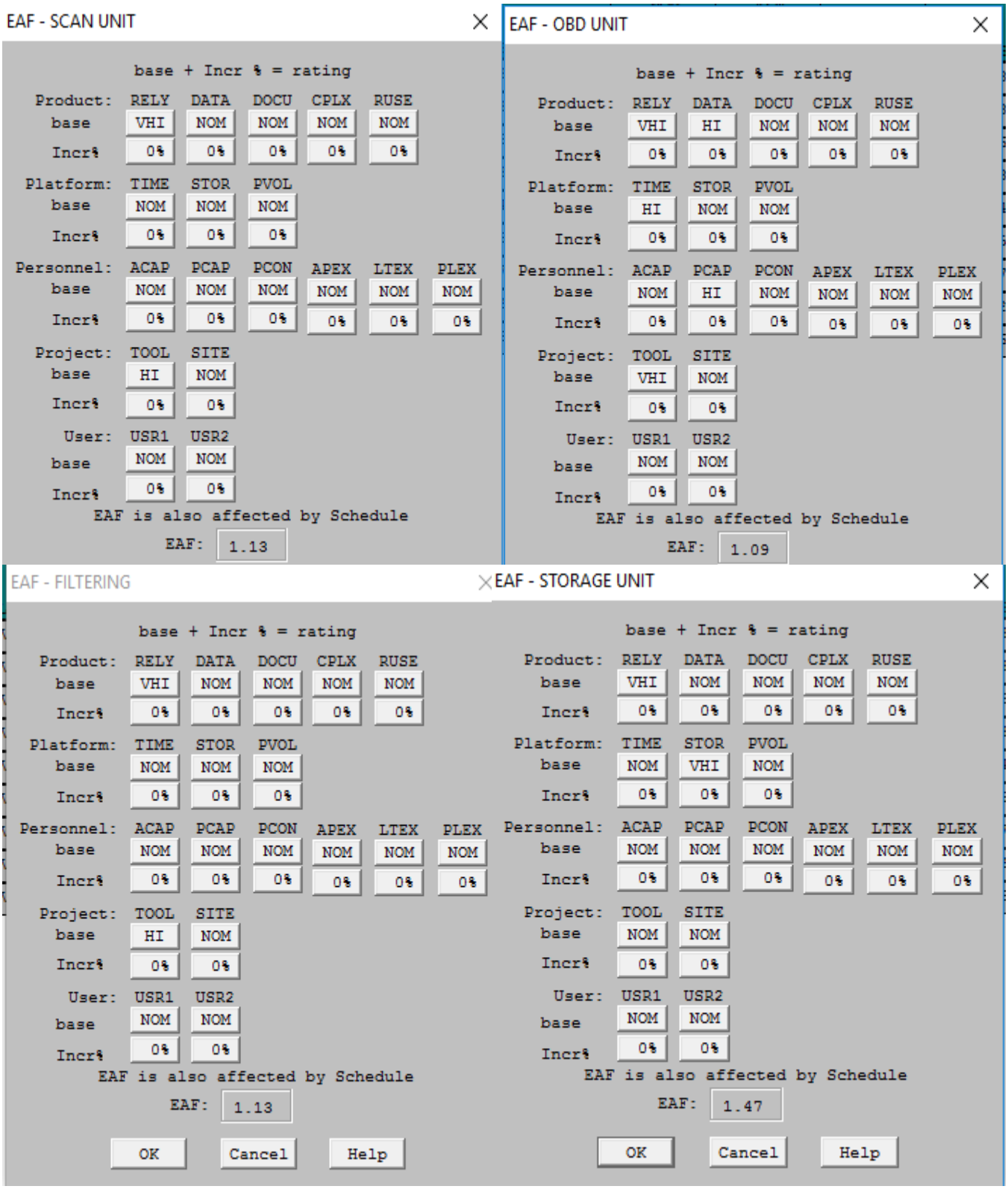


Figure 16. EAF for POST ARCHITECTURE Model

COCOMO II RESULTS (POST ARCHITECTURE)

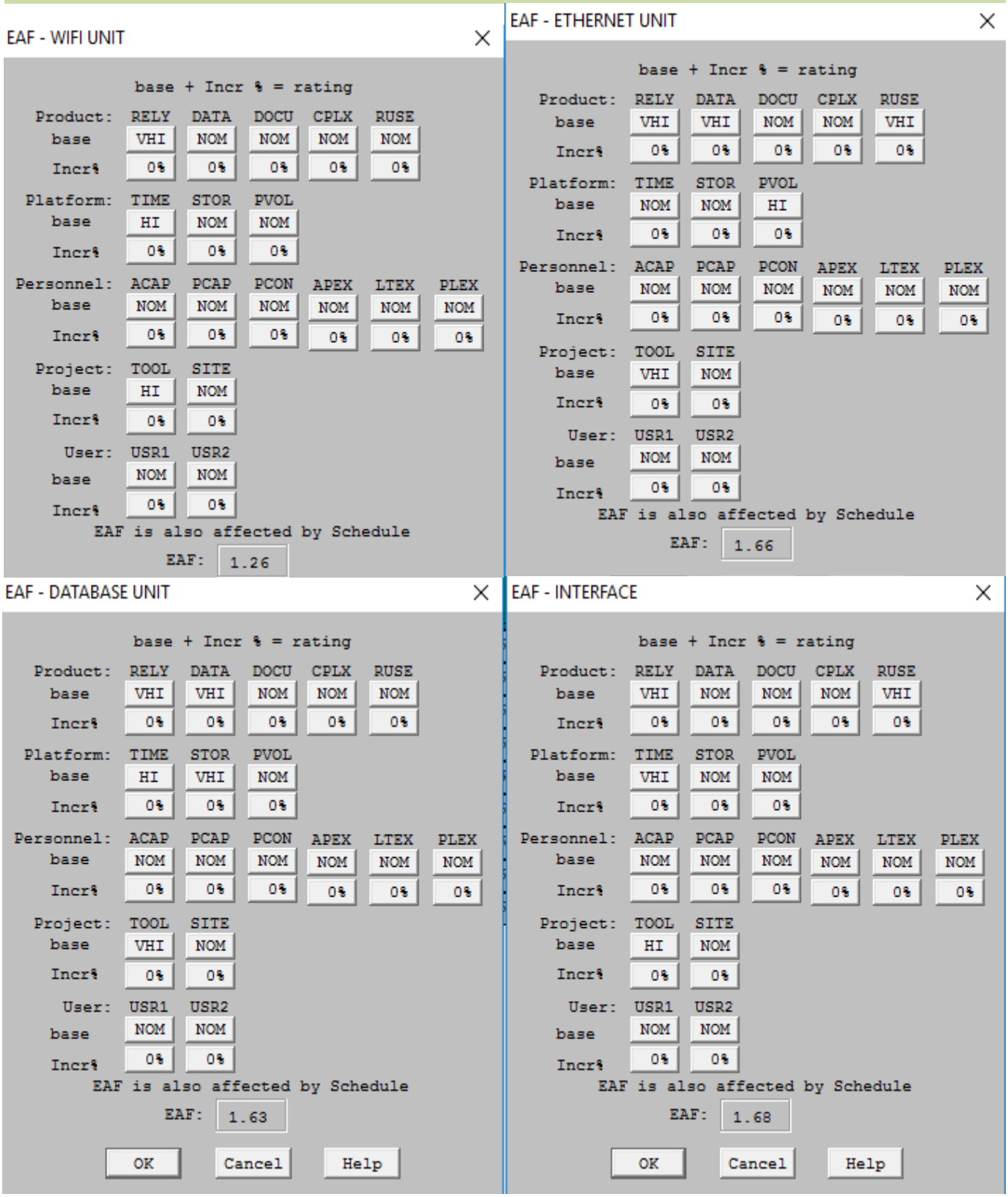


Figure 17. EAF for POST ARCHITECTURE Model

COMPARISON OF RESULTS

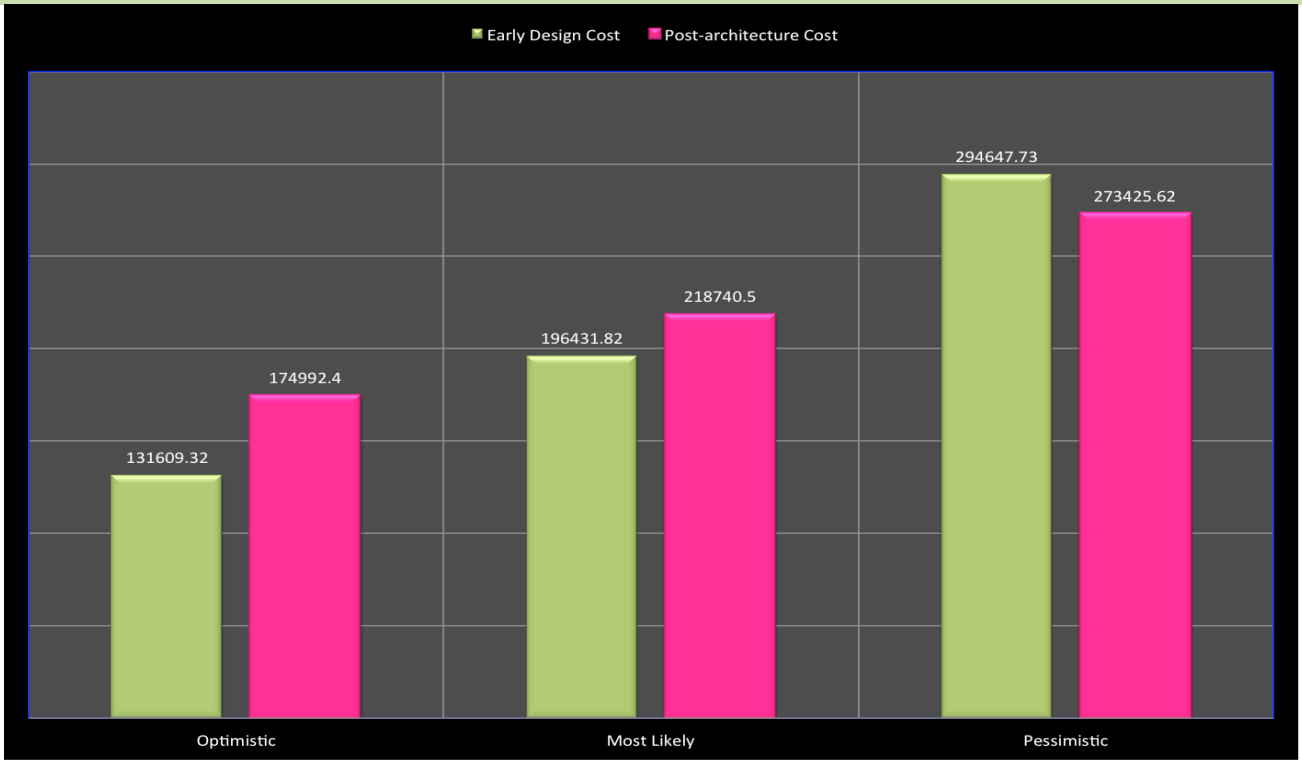


Figure 18. COMPARISON of Early Design COST & Post Architecture COST

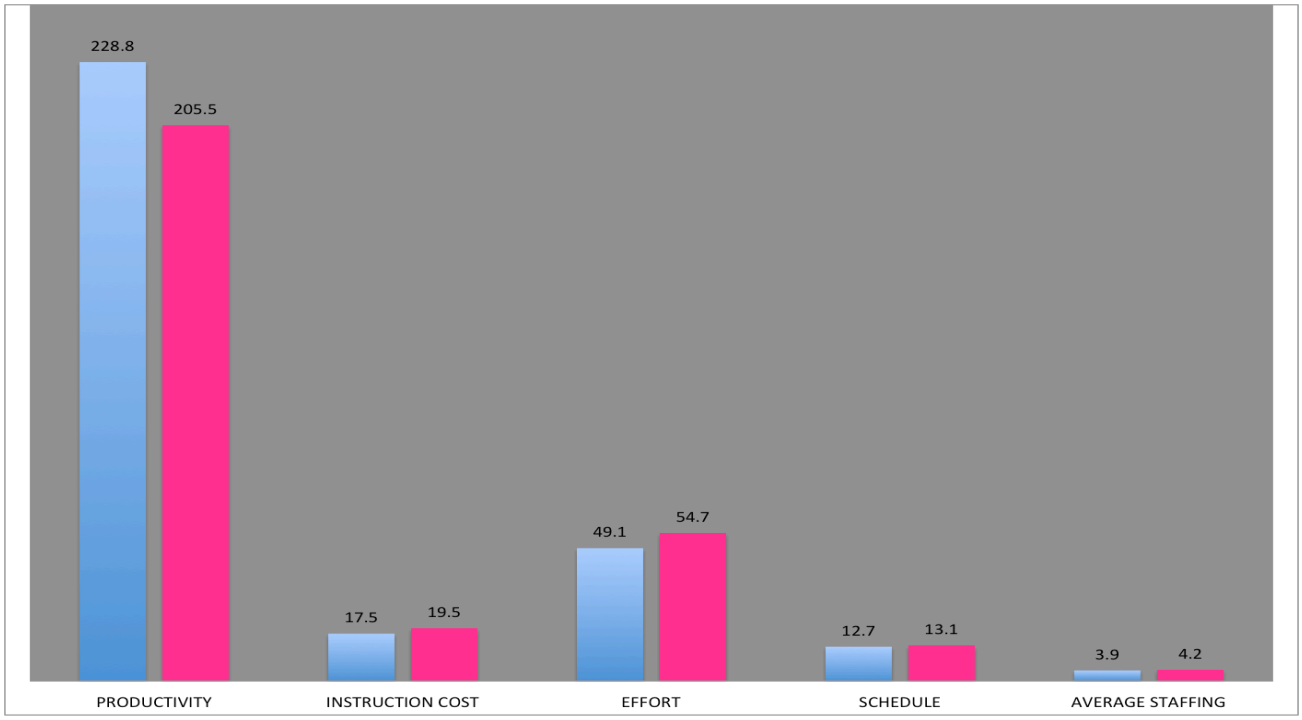


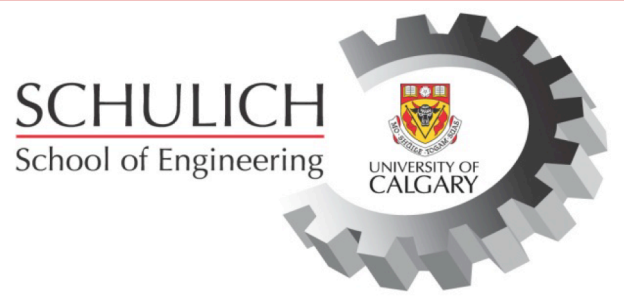
Figure 19. COMPARISON of Early Design(in blue) & Post Architecture(in pink) RESULTS

CONCLUSION

The COCOMO II model was very useful in calculating the size and effort of the Project.
The Figures 11 and 15 summarizes the results of the COCOMO II model for Early Design and Post Architecture. COCOMO II offers 3 phases of estimation. But in my case I used the model for only 2 phases. Each phase has different effort estimation results. In each phase we broke the system into more refines components and try to investigate additional factors. As we advanced in the phases more accuracy is required. Despite the fact that function point was used to estimate the size in assignment 1 and the two phases of assignment 2, the results are within 20 -25% tolerance hence it's acceptable. COCOMO II approach is therefore more accurate when the module is performed in parallel by different developers. It also demonstrates the importance of well-defined modules in reducing the effort required to complete the project. It also offers a comprehensive way to consider many different operating conditions before starting a project, and it can help us to see how certain decisions may affect the project effort and cost. In general the COCOMO II tool was very useful in automating the effort estimation.

REFERENCES

[1] http://www.onboarddiagnostics.com/download/u581.pdf
[2] http://www.nyc.gov/html/tlc/downloads/pdf/intro_to_obd2.pdf
[3] B. H. Far "Seng 637" Software Reliability & software quality Chapter 2 [online] Department of Electrical & Computer Engineering, University of Calgary
[4] B. H. Far "Seng 637" Software Reliability & software quality Chapter 3 [online] Department of Electrical & Computer Engineering, University of Calgary



SENG 637
PROJECT ON
ONBOARD DIAGNOSTICS SYSTEM

ASSIGNMENT 3
DEFINING NECESSARY RELIABILITY

SUBMITTED BY:
SUCHINA PARIHAR
ID: 10168164
EMAIL:
suchina.parihar2@ucalgary.ca

EXECUTIVE SUMMARY

In this ASSIGNMENT, we will define the necessary Reliability for our Project. For the Software Reliability Engineering process, the very first step is to define the necessary reliability of the system. Thus, for the necessary reliability, we need to define the risk as Necessary Reliability depends on the RISK. Higher Risk software requires higher reliability. Risk is a combination of occurrence of an abnormal event or failure and the consequences of that event or failure to a system's operators, users, or environment. A risk can range from catastrophic to negligible. Risks are also categorized according to the likelihood of occurrence [4]

Therefore, to define the Necessary reliability, firstly we need to define the Failure Severity Classes for the different subsystems as shown in Table 6,7,8,9. Failures usually differ by their impact on the system. A failure Severity Class (FSC) is a set of failures that have the same per-failure impact on users using a failure classification criteria. Common classification criteria: cost, system capability, human life and environment. Now, secondly we need to set a Failure Intensity Objective (FIO) for those subsystems as shown in Table 10. Failure intensity objective (FIO) reflects an estimation of the "bugs" allowed to be remained in the product at the release time. FIO is an alternative way of expressing reliability. For our project, Failure Intensity is given in terms of number of Failures per 100,000 drive cycles. Thirdly, we will find the Product Acquired Failure Intensity Objective as shown in Table 11,12,13,14. Then we will find the Developed Software Failure Intensity Objective. Lastly, we will engineer strategies to meet the software failure intensity objective like Fault Removal, Fault Prevention and Fault Tolerance strategies.

TIME UNIT

Choosing a time unit would have caused issues if we want to take into account that the system can be measured at any random time because in our Project, the system cannot be measured at anytime but can only be diagnosed if it completes a whole drive cycle. DRIVE CYCLE means the time from the start of the engine to when it stops. Taking drive cycles as unit allows us to take into account the use of the system. As the cost of our Project is more than 100,000, therefore by RULE OF THUMB, we will take into account 100,000 DRIVE CYCLES. Thus our unit of measurement is 100,000 DRIVE CYCLES

SUBSYSTEM 1: SCAN UNIT

	DEFINITION	DESCRIPTION
1	Scanner not working properly	Faulty scanner code
2	Device won't boot scanner	Incompatible scanner
3	Data is missing	Scanner lack some functionalities
4	Scanner is very slow	Need refactoring of code

TABLE 3. FAILURE DATA

SUBSYSTEM 2: ACQUISITION UNIT

	DEFINITION	DESCRIPTION
1	Device won't boot OS	Either incompatible softwares/platform dependent programming language etc
2	Errors when reading /writing data	Faulty code
3	Slow read/write data to SD card	Need to increase the transfer capacity or refactor the code.

TABLE 4. FAILURE DATA

SUBSYSTEM 3: DATABASE UNIT

	DEFINITION	DESCRIPTION
1	Unable to retrieve or store data	Database connectivity is lost/not there/code is not working/etc
2	Produce or store erroneous results	Wrong queries/ queries mixup
3	Processing is very slow	Data is too large /difficulty in reading data/data is not filtered/ etc
4	Database connection not established	Need connectivity

TABLE 5. FAILURE DATA

SUBSYSTEM 4: USER INTERFACE UNIT

	DEFINITION	DESCRIPTION
1	Unable to retrieve or store data	Database connectivity is lost/not there/code is not working/etc
2	Unable to process data	Wrong queries/ queries mixup
3	Unable to display data	Need connectivity/ data in wrong or incompatible format
4	Wrong data displayed	Wrong queries/ queries mixup

TABLE 6. FAILURE DATA

FAILURE CLASSES

	FAILURE	FAILURE CLASS (COST)	FAILURE CLASS (System Capability)
F1	Scanner OS freezes	2	1
F2	Scanner does not boot	2	1
F3	Scanner has connectivity issues with RPi	3	2
F4	Scanner is slow in processing vehicle data codes	3	3
F5	Scanner does not read some data codes	3	2

TABLE 7. FAILURE CLASSES OF SCAN UNIT

	FAILURE	FAILURE CLASS (COST)	FAILURE CLASS (System Capability)
F1	RPi OS freezes	3	1
F2	RPi does not boot	2	1
F3	RPi has connectivity issues with OBD2	2	2
F4	SD Card corruption	2	1
F5	RPi does not filter data	3	2

TABLE 8. FAILURE CLASSES OF ACQUISITION UNIT

FAILURE CLASSES

	FAILURE	FAILURE CLASS (COST)	FAILURE CLASS (System Capability)
F1	Connection lost between database and Data Acquisition unit	2	1
F2	Connection lost between database and User Interface unit	2	1
F3	Database unable to store OBD II data	2	1
F4	Database unable to encode the stored OBD II data	2	1
F5	Database read/write attempts fail due to high packet error rates	2	3
F6	Database OS crashes	1	1
F7	Database files become corrupt	1	1

TABLE 9. FAILURE CLASSES OF DATABASE UNIT

	FAILURE	FAILURE CLASS (COST)	FAILURE CLASS (System Capability)
F1	Application looses network access	3	1
F2	Application is unable to connect to database	3	1
F3	Application hangs	2	1
F4	Application incorrectly shows data	3	3
F5	Application crashes and restarts	2	1
F6	Application takes too long to do an operation	3	2
F7	Incorrectly decode OBDII data	3	3

TABLE 10. FAILURE CLASSES OF USER INTERFACE UNIT

FAILURE INTENSITY OBJECTIVE (FIO)

WHOLE PROJECT	1 FAILURE/100,000 DRIVE CYCLES
DATA ACQUISITION SUBSYSTEM	0.30 FAILURES/100,000 DRIVE CYCLES
DATABASE SUBSYSTEM	0.20 FAILURES/100,000 DRIVE CYCLES
USER INTERFACE SUBSYSTEM	0.25 FAILURES/100,000 DRIVE CYCLES
DATA SCAN SUBSYSTEM	0.25 FAILURES/100,000 DRIVE CYCLES

TABLE 11. FAILURE INTENSITY OBJECTIVES

PRODUCT ACQUIRED FAILURE INTENSITY

COMPONENT	FAILURE RATE	UNIT
RASPBIAN OS	0.0760	FAILURES/100,000 DRIVE CYCLES

TABLE 12. FAILURE RATES OF DATA SCAN ACQUIRED COMPONENTS

PRODUCT ACQUIRED FIO = 0.0760

COMPONENT	FAILURE RATE	UNIT
RASPBIAN OS	0.0760	FAILURES/100,000 DRIVE CYCLES
RASPBERRY Pi HARDWARE	0.05779	FAILURES/100,000 DRIVE CYCLES

TABLE 13. FAILURE RATES OF DATA ACQUISITION ACQUIRED COMPONENTS

PRODUCT ACQUIRED FIO = 0.0760 + 0.05779 = 0.1338

COMPONENT	FAILURE RATE	UNIT
LINUX OS	0.0760	FAILURES/100,000 DRIVE CYCLES
DATABASE HARDWARE	0.00760	FAILURES/100,000 DRIVE CYCLES

TABLE 14. FAILURE RATES OF DATABASE ACQUIRED COMPONENTS

PRODUCT ACQUIRED FIO = 0.0760 + 0.00760 = 0.0836

COMPONENT	FAILURE RATE	UNIT
WEB OS	0.0760	FAILURES/100,000 DRIVE CYCLES
WEB APPLICATION	0.00760	FAILURES/100,000 DRIVE CYCLES

TABLE 15. FAILURE RATES OF USER INTERFACE ACQUIRED COMPONENTS

PRODUCT ACQUIRED FIO = 0.0760 + 0.00760 = 0.0836

PRODUCT ACQUIRED FAILIURE INTENSITY = 0.377 failures/100,000 drive cycles

PRODUCT DEVELOPED SOFTWARE FIO

DATA SCAN UNIT:

PRODUCT DEVELOPED SOFTWARE FIO = (TOTAL DATA SCAN UNIT FIO) – (DATA SCAN PRODUCT ACQUIRED FIO) = **0.25 – 0.0760 = 0.174**

DATA ACQUISITION UNIT:

PRODUCT DEVELOPED SOFTWARE FIO = **0.30 – 0.1338 =0.1662**

DATABASE UNIT:

PRODUCT DEVELOPED SOFTWARE FIO = **0.2 – 0.0836 = 0.1164**

USER INTERFACE UNIT:

PRODUCT DEVELOPED SOFTWARE FIO = **0.25 – 0.083 = 0.167**

TOTAL PRODUCT DEVELOPED S/W FIO = 0.6236 failures/100,000 drive cycles

FAULT PREVENTION (20%)

1. REQUIREMENT REVIEWS (15%)
2. DESIGN REVIEWS (5%)

FAULT REMOVAL (55%)

1. CODE INSPECTION (15%),
2. BETTER TECHNIQUES (5%)
3. UNIT TEST (25%)
4. SYSTEM TEST (10%)

FAULT TOLERANCE (25%)

1. REDUNDANT DESIGN (10%)
2. SEQUENTIAL DESIGN (20%)

Fault Tolerance techniques are shown in Figure 21. Fault Tolerance is costly but because this project is having so many single points of failures, therefore Fault Tolerance becomes necessary. Thus, from Figure 21, we can see that as SCAN UNIT is the very first point of connection between OBD II system and the vehicle, it needs a redundant Software algorithm in case of emergency when the original algorithm stops working for some reason. Similarly, in Raspberry Pi, we need a spare SD Card which can be used in case of faulty SD Card or when SD Card gets full. Moreover, in FILTERING UNIT, we put a redundant software algorithm in case the original algorithm fails to work. In Database, we can keep redundant data tables for a single vehicle as its always good to keep back up of data.

FAULT TOLERANCE

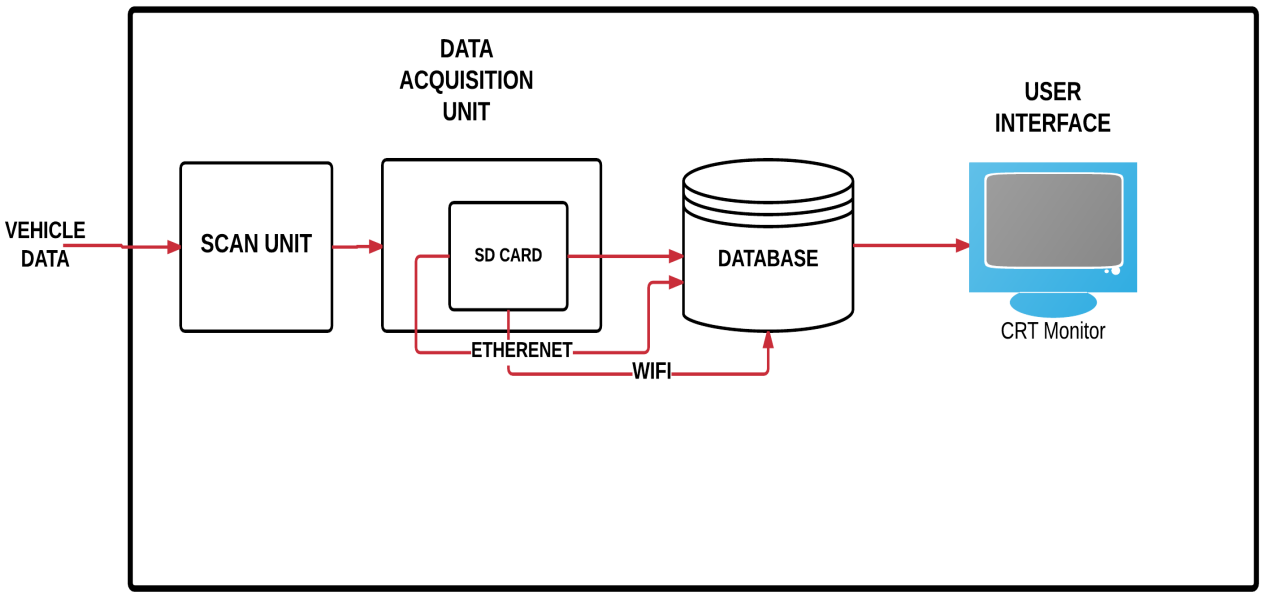


Figure 20. Block diagram of system

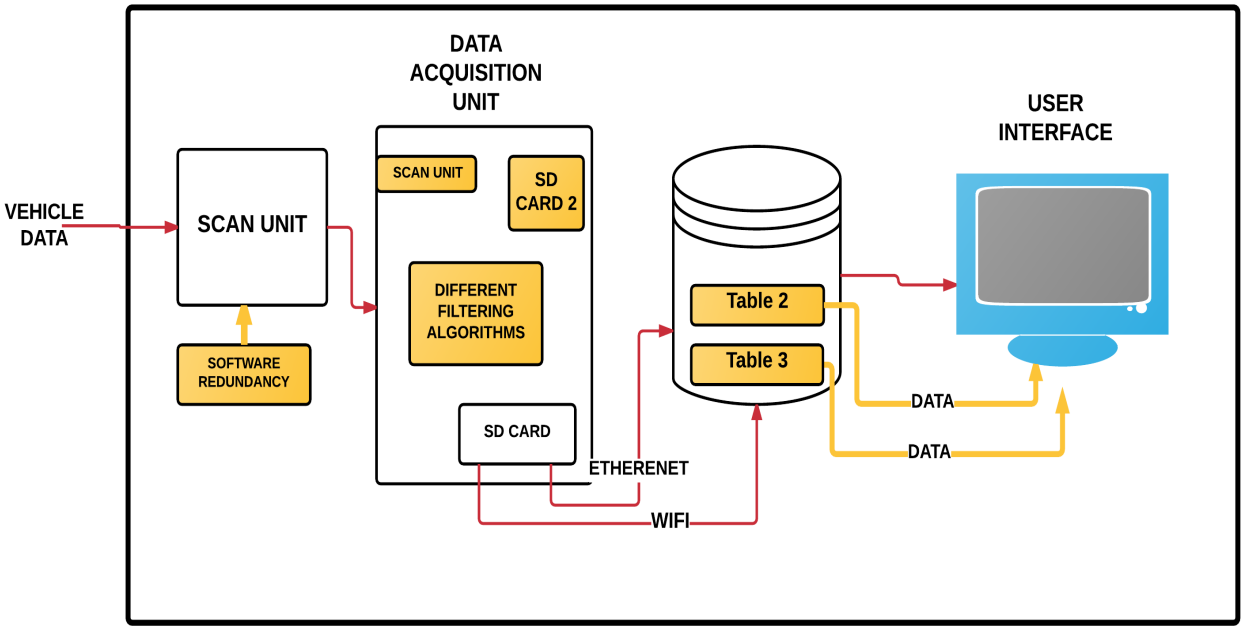


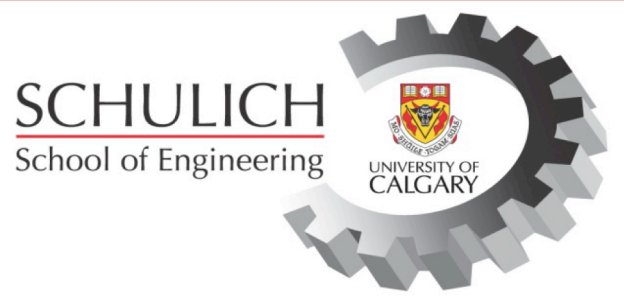
Figure 21. Block diagram of system after applying Fault Tolerance techniques

CONCLUSION

The very first step of the Software Reliability Engineering is to define the Necessary Reliability. Necessary Reliability means to define a threshold upto which our system can accept failures. This step helps us to better understand the failure capacity of our system. It helps us to define a Failure Intensity Objective and Failure Severity Classes for our system. Failures usually differ by their impact on the system. Failure Severity Class (FSC) is a set of failures that have the same per-failure impact on users using a failure classification criteria. Failure intensity is usually given in terms of number of failure per unit of time but for my Project it is given in number of failures per 100,000 drive cycles. Total PRODUCT DEVELOPED S/W FIO for my Project is 0.6236 failures/100,000 drive cycles. I have used certain Fault Removal Techniques in this assignment which are 55%. Fault Prevention techniques are 20%. I have included Fault Tolerance Techniques like Redundancy and Sequential approach. I have given more weightage to Fault Tolerance than Fault Prevention because my Project has so many single points of failure and to deal with them I need to inculcate redundancy. Therefore, in this Assignment, I have defines Failure Severity classes, Failure Intensity Objective, Fault Removal Techniques, Fault Prevention Techniques and Fault Tolerance Techniques for the whole Project.

REFERENCES

[1] http://www.onboarddiagnostics.com/download/u581.pdf
[2] http://www.nyc.gov/html/tlc/downloads/pdf/intro_to_obd2.pdf
[3] B. H. Far "Seng 637" Software Reliability & software quality Chapter 2 [online] Department of Electrical & Computer Engineering, University of Calgary
[4] B. H. Far "Seng 637" Software Reliability & software quality Chapter 3, 7 [online] Department of Electrical & Computer Engineering, University of Calgary



SENG 637
PROJECT ON
ONBOARD DIAGNOSTICS SYSTEM

ASSIGNMENT 4
OPERATIONAL PROFILES

SUBMITTED BY:
SUCHINA PARIHAR
ID: 10168164
EMAIL:
suchina.parihar2@ucalgary.ca

EXECUTIVE SUMMARY

Operational profile is the set of operations (operation names and frequencies) and their probabilities of occurrences. Operational mode is a distinct pattern of system use and/or set of environmental conditions that may need separate testing due to likelihood of stimulating different failures. System operational profile must be developed for all of its important operational modes [3]. The operational modes in this project are classified based on functionality i.e. User mode functions and System mode functions. The Consumer will be the users, who will be using the application for viewing the diagnostic reports of their vehicle, viewing log details, viewing past data, viewing error reports. The System mode will access the application for updating reports, check error profiles, scan new data or save data in database etc. The operational modes of the developed application can also be classified on the basis of authorization etc. Every user accessing the system must be using it for a specified function to be performed. Users require different functionalities from the application, which during the design phase are referred to as operations. The difference between the functionality and operation is that functionality is developed according to customer requirements (what the user can do) and the operations are design implementations of those functions in the development phase. Therefore each of the functionality can become one or a series of operations. When operations are combined with statistics of their occurrences, it is referred to as an operational profile [3]

OPERATIONAL MODES

Operational mode is a distinct pattern of system use that may need separate testing due to likelihood of stimulating different failures. The same operation may appear in different operational mode with different probabilities. **System operational profile** must be developed for all of its important operational modes [3]. Various Operational Modes for my system are in Table below

FACTORS	MODES	OUTCOMES
FUNCTIONALITY	System Mode and User Mode	Upload data, Scan data, View Log details, Filter data, View Diagnostics data, Store data etc
AUTHORIZATION	Client Mode and Administrator Mode	Update data, Create/Modify/delete data, view data etc
COMMUNICATION LINKS	OBD interface, Dtabase Interface and Web Interface	Link ready, Connection available, Send data over the link, Link broken etc
TIME	Realtime data and Previous Log data	During Drive Cycle, After Drive Cycle

TABLE 16. OPERATIONAL MODES

OPERATION MODE	OPERATION INITIATORS
SYSTEM MODE	USERS, SYSTEM VARIABLES, FUNCTIONS, ADMINISTRATOR
USER MODE	USERS

TABLE 17. OPERATION INITIATORS

OPERATIONAL PROFILE

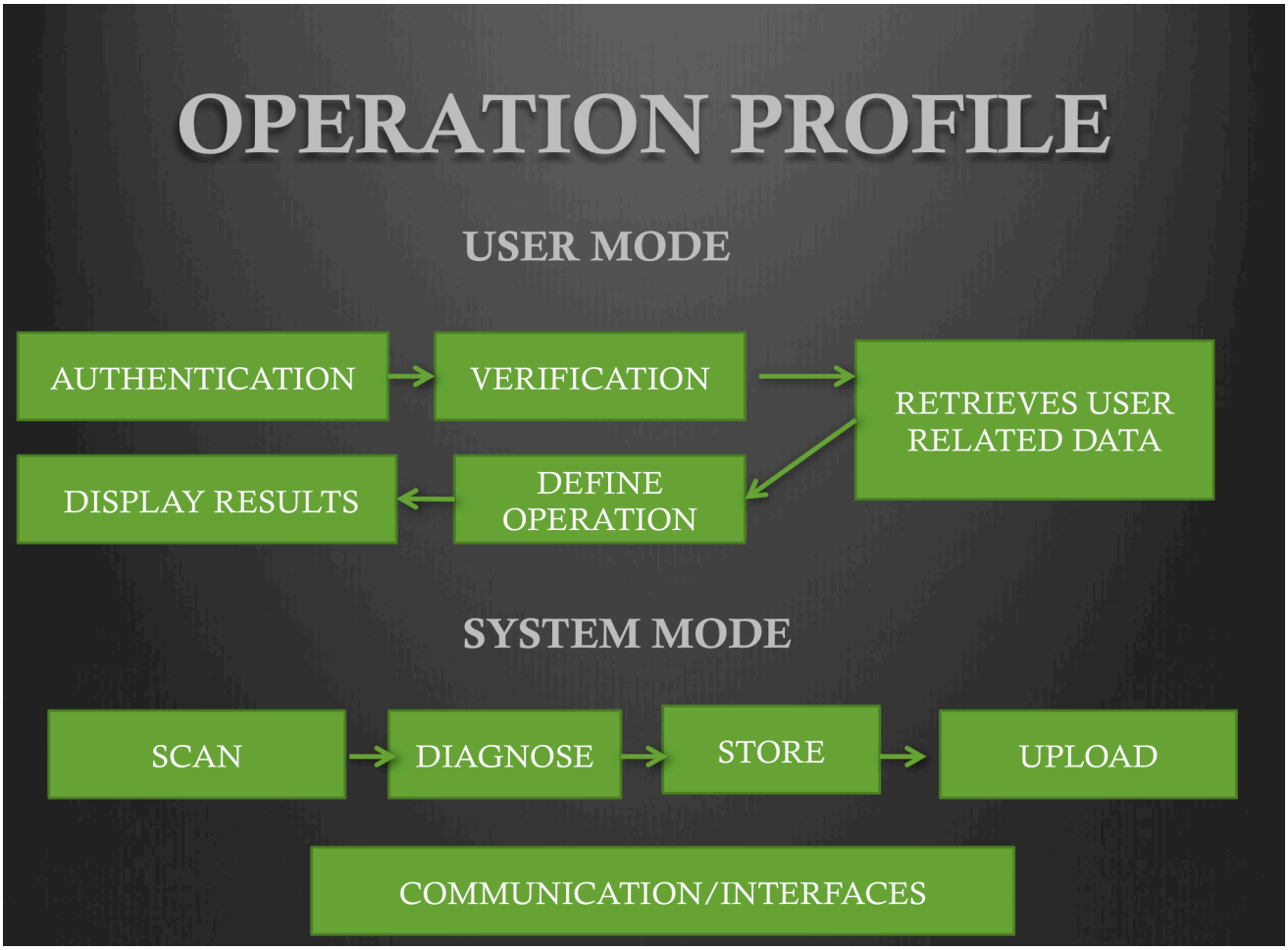


Figure 22. OPERATION PROFILE OF USER AND SYSTEM MODE

OPERATIONAL PROFILE

	SYSTEM MODE OPERATIONS	OCCURRENCE RATE	OCCURRENCE PROBABILITY
1	Scan vehicle data	30/100	0.3
2	Send vehicle scanned data to OBDII	30/100	0.3
3	Send enhanced DTC codes to OBDII	2/100	0.02
4	Filter diagnostic data	10/100	0.1
5	Store the diagnostic data in the SD Card	10/100	0.1
6	Upload data file into the database	7/100	0.07
7	SD card almost full warning	3/100	0.03
8	Delete the failure data	3/100	0.03
9	Retrieve data	5/100	0.05

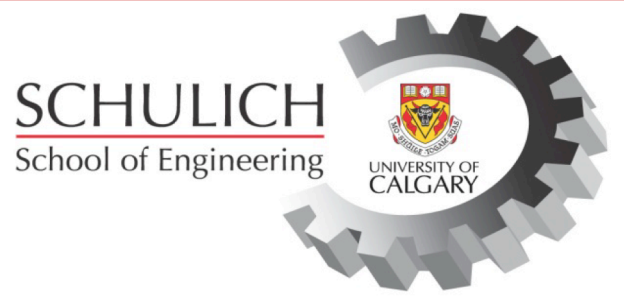
TABLE 18. SYSTEM MODE OPERATIONAL PROFILE AND OCCURRENCE

CONCLUSION

The Operational profile developed in this assignment is a quantitative characterization of how a system will be used and is essential in software reliability analysis. The Operational Profiles are made with the help of Use Case diagrams or Activity diagrams where one scenario makes an operational profile.The operational profiles show how to increase productivity and reliability and speed development by allocating development and test resources to functions on the basis of use. The occurrence probabilities estimated in this assignment are going to be used for allocating test cases and test time. The analysis clearly indicate that the User Access mode is the most frequently used mode and majority of the tests will be allocated to this mode. In estimating the occurrence rate, professional experience of people in the field is taken into account. The exceptional flows are also kept in mind and are also included in the profile to formally include them in the testing phase. The language selection is not included into profile development because the language selection is itself not an executional operation rather it being just an output GUI display because all the operations at the background are the same. The overall operational profile of the application is expected to give a good way forward in the testing phase of this project.

REFERENCES

[1] http://www.onboarddiagnostics.com/download/u581.pdf
[2] http://www.nyc.gov/html/tlc/downloads/pdf/intro_to_obd2.pdf
[3] B. H. Far "Seng 637" Software Reliability & software quality Chapter 2,10 [online] Department of Electrical & Computer Engineering, University of Calgary
[4] B. H. Far "Seng 637" Software Reliability & software quality Chapter 3, 7,11,12,13 [online] Department of Electrical & Computer Engineering, University of Calgary



SENG 637
PROJECT ON
ONBOARD DIAGNOSTICS SYSTEM

ASSIGNMENT 5
TESTING

SUBMITTED BY:
SUCHINA PARIHAR
ID: 10168164
EMAIL:
suchina.parihar2@ucalgary.ca

EXECUTIVE SUMMARY

In this ASSIGNMENT, I will discuss the process of TESTING which is the next step in Software Reliability Engineering. In this process I will identify how many test cases we need and how much time I have to spend on testing the product. Finally I will define the test cases and develop the test suite. Estimation for number of test cases will be established first. Test cases will be allocated to sub systems and associated parts to be tested. Test cases will then be assigned to the developed product. Test profile for one of the operational modes will be presented as well. The report will than enlist the hours devoted to feature, regression (if needed), and load test for the product. The hours of load (integration) test among the operational modes will be calculated. Based on the data sets provided by the Dr. Far, tests will be run on the program. Finally STRAT tool will be used to verify the actual reliability growth. The Reliability Demo Chart will help us to understand whether our project can be released into the market or it needs more testing.

TEST CASES ESTIMATION

Test estimation is affected by two factors:

1. **Time: (available time * available staff) / (average time to prepare a test case)**
2. **Cost: (available budget) / (average preparation cost per test case)**

In my case, time criterion is used to estimate the number of test cases required for the project. The project is to be developed in four month time (2880 hours), and one group member will be working on it on an average of 40 hours/week. This brings to total of 640 hours being invested in the development of the system. There is only one members in the group for the development of the project and she will be spending 30% of the time in the testing of the system, we can say that the total available staff for test execution is 0.30 * 1 = 0.30 persons. According to the statistics, testing of a single-component console application takes about 20% of its development time. Testing of a two-component console application takes 20-30% of its development time, an application with GUI - 30-35%, a distributed application with GUI - 35-50%. [5] Time for the development of test case depends on the complexity of the test plan but on an average, developing of one test case takes 1 hour.

Test cases = (available time * available staff) / (average time to prepare a test case)
Test cases = (640 * 0.30)/1 = 192 test cases

TEST CASES ALLOCATION

For this Project, I will allocate the maximum number of test cases to the developed project. As most of the testing is necessary only when the product is developed, therefore I have allocated 80% of the test cases to the developed software project. The Operating System is not a very important part of my system and the chances of its failure are very less, therefore, I have allocated only 1% of the test cases to it. The Developing Project, means the parts under development needs testing as well. I have allocated 19% of the test cases to the project under development. The Browser has not been allocated any test cases as it is assumed to be a properly test acquired component. Overall, 153 test cases are allocated to Developed Product, 37 test cases are allocated to the Product under development and 2 test cases are allocated to Operating system.

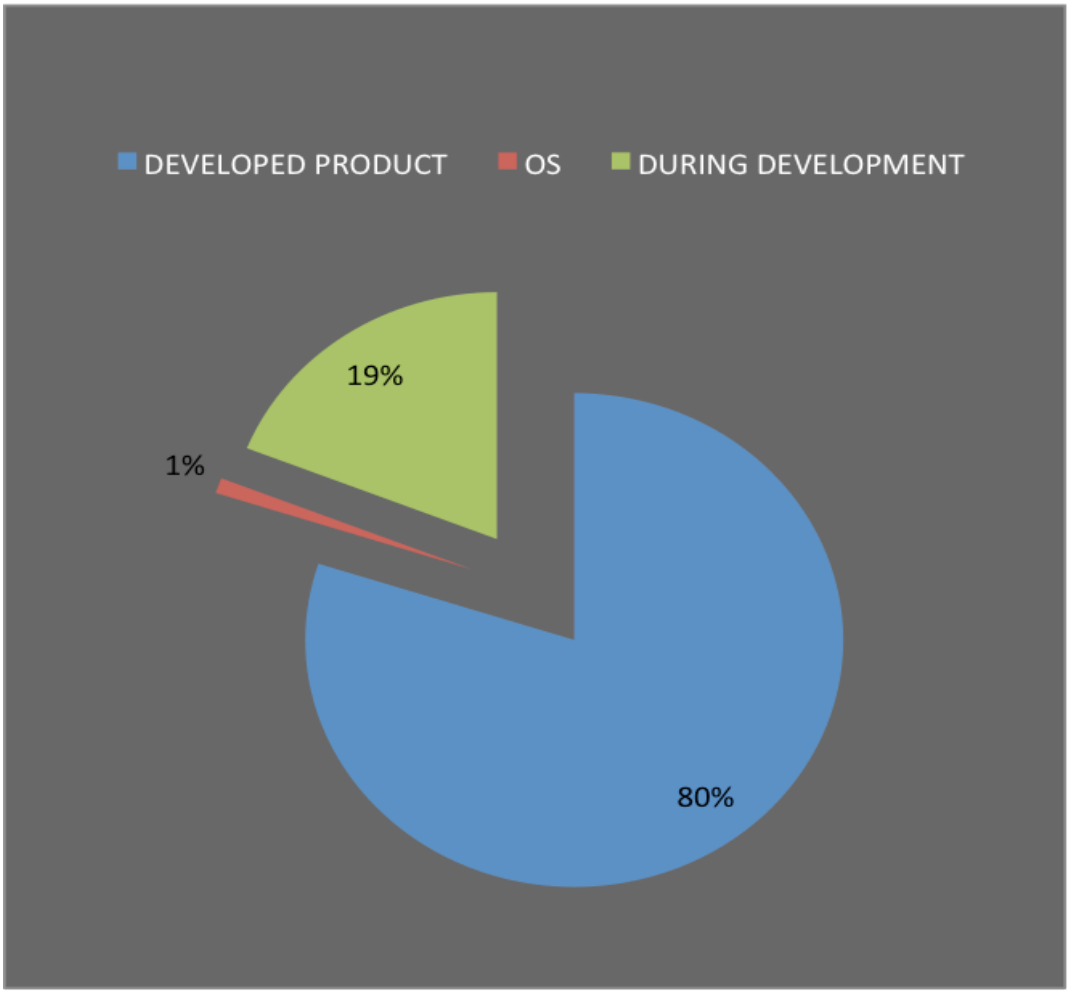


Figure 23. Pie Chart showing Test Cases Allocation

TEST CASES ALLOCATION FOR DEVELOPED OPERATIONS

For this Project, I have allocated tests to developed operations based on the occurrence probabilities and the risk severity of these operations which is calculated in Assignment 3 and 4. In this ASSIGNMENT, I have used system defined operational mode i.e. " DIAGNOSE" to create the test cases. The operational profile of this mode is shown in the Table below. Some of the operations that are mentioned are critical for the system. Critical operations are those for which failure may result in great financial harm or seriously damage the capability of the system. For these critical operations, 2-4 test cases should be allocated to them. In my project, I have allocated 2 test cases for every critical operation. Infrequent operations are those that occur so rarely that they would not be assigned any test cases based on their occurrence probability. But for my project, I have allocated 1 test case for every infrequent new operation based on the threshold occurrence probability. Critical Operations are in red color and infrequent operations are in yellow color.

	SYSTEM MODE "DIAGNOSE" OPERATIONS	OCCURRENCE RATE (USER REQUESTS)	OCCURRENCE PROBABILITY	TEST CASES
1	Diagnostic test failed	4/100	0.04	1
2	Send enhanced DTC codes to OBDII	2/100	0.02	2
3	Send vehicle scanned data to OBDII	20/100	0.20	37
4	Store the diagnostic data in the SD Card	20/100	0.20	37
5	Filter diagnostic data	20/100	0.20	37
6	Upload data file into the database	15/100	0.15	29
7	View logging details	8/100	0.08	6
8	SD card almost full warning	4/100	0.04	2
9	Delete the failure data	2/100	0.02	1
10	Request test results	5/100	0.05	1

TABLE 19. Test Cases Allocation Of "DIAGNOSE" Operations

TEST CASES ALLOCATION FOR DEVELOPED OPERATIONS

	SYSTEM MODE "DIAGNOSE" OPERATIONS	OCCURRENCE RATE (USER REQUESTS)	OCCURRENCE PROBABILITY	TEST CASES
1	Diagnostic test failed	4/100	0.04	1
2	Send enhanced DTC codes to OBDII	2/100	0.02	2
3	Send vehicle scanned data to OBDII	20/100	0.20	37
4	Store the diagnostic data in the SD Card	20/100	0.20	37
5	Filter diagnostic data	20/100	0.20	37
6	Upload data file into the database	15/100	0.15	29
7	View logging details	8/100	0.08	6
8	SD card almost full warning	4/100	0.04	2
9	Delete the failure data	2/100	0.02	1
10	Request test results	5/100	0.05	1

TABLE 20. Test Cases Allocation Of "DIAGNOSE" Operations

TEST PROFILE FOR SYSTEM OPERATIONAL MODE

	SYSTEM MODE "DIAGNOSE" OPERATIONS	OCCURRENCE PROBABILITY INITIAL	OCCURRENCE PROBABILITY ADJUSTED	OCCURRENCE PROBABILITY FINAL
1	Diagnostic test failed	4.25%	4.3%	4%
2	Send enhanced DTC codes to OBDII	2.1%	1.3%	1.3%
3	Send vehicle scanned data to OBDII	25.2%	25%	25%
4	Store the diagnostic data in the SD Card	20.56%	20.6%	20%
5	Filter diagnostic data	22.9%	23%	22.4%
6	Upload data file into the database	15.01%	15.1%	15%
7	View logging details	8.128%	8.2%	8%
8	SD card almost full warning	4%	1.3%	1.3%
9	Delete the failure data	2.61%	2.6%	2%
10	Request test results	5.69%	5.7%	1%

TABLE 21. Test Profile for "DIAGNOSE" Operations

TEST CASE DOCUMENTATION

TEST CASE ID: OBD-t1	AUTHOR: SUCHINA	LAST UPDATED BY: SUCHINA
PARENT USE CASE ID: OBD	ORIGINAL DATE: 16/04/2017	LAST UPDATED ON: 18/04/2017
TEST OBJECTIVE: VIEW LOGGING DETAILS		

ITEM NO	TEST CONDITION	OPERATOR ACTION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL	COMMENTS
1	User Mode	Click logging details button	Valid VIN	User logins into his/her account and see the log details	P	N/C
2	User Mode	Click logging details button	Valid Password	User logins into his/her account and see the log details	P	N/C
3	User Mode	Click logging details button	Invalid VIN	User is denied login	F	N/C
4	User Mode	Click logging details button	Invalid Password	User is denied login	F	N/C

TABLE 22. TEST CASE documentation for " VIEW LOGGING DETAILS" operation

TEST CASE DOCUMENTATION

TEST CASE ID: OBD-t2	AUTHOR: SUCHINA	LAST UPDATED BY: SUCHINA
PARENT USE CASE ID: OBD	ORIGINAL DATE: 16/04/2017	LAST UPDATED ON: 18/04/2017
TEST OBJECTIVE: REQUEST TEST RESULTS		

ITEM NO	TEST CONDITION	OPERATOR ACTION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL	COMMENTS
1	User Mode	Click TEST RESULTS button	Valid TEST number	User can view test results	P	N/C
2	User Mode	Click TEST RESULTS button	Valid TEST data	User can run the test and see the results	P	N/C
3	User Mode	Click TEST RESULTS button	Invalid TEST number	User is denied	F	N/C
4	User Mode	Click TEST RESULTS button	Invalid TEST data	User is denied to run a test	F	N/C
5	User Mode	Click TEST RESULTS button	TEST complete	User can see the results	P	
6	User Mode	Click TEST RESULTS button	TEST incomplete	User cannot see the results	F	User can view the results once the test is complete

TABLE 23. TEST CASE documentation for " VIEW TEST RESULTS" operation

TEST CASE DOCUMENTATION

TEST CASE ID: OBD-t3	AUTHOR: SUCHINA	LAST UPDATED BY: SUCHINA
PARENT USE CASE ID: OBD	ORIGINAL DATE: 16/04/2017	LAST UPDATED ON: 18/04/2017
TEST OBJECTIVE: Upload data into database		

ITEM NO	TEST CONDITION	OPERATOR ACTION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL	COMMENTS
1	System Mode	Update database	Drive cycle complete	Uploads data into the database	P	N/C
2	System Mode	Update database	Drive cycle incomplete	Data upload failed	F	N/C
3	System Mode	Update database	Database link active	Uploads data into the database	P	N/C
4	System Mode	Update database	Database link inactive	Data upload failed	F	N/C

TABLE 24. TEST CASE documentation for “Upload data into database” operation

TEST TIME ALLOCATION

For this Project, the development time is 4 months (2880 hours) and 1 group member is working on it on an average of 40 hours/weeks. This means 640 hours to be invested in development of the system. I also calculated that 0.3 person of testing staff is available. So, for the testing phase, a total of 192 person hours are available. Now, it takes 40 minutes per test case for the preparation of a test case which equals 128 hours.
Thus, **Testing Time is equal to 64 hours**

SUBSYSTEM	APPROXIMATION TIME ALLOCATION	TIME ALLOCATION HOURS
DEVELOPED PRODUCT	80%	51
DURING DEVELOPMENT	19%	12
OS	1%	1

TABLE 25. Test Time Allocation

HOURS FOR FEATURE, REGRESSION and LOAD TESTS

As mentioned in [4],

- for all new test cases: Allocate time to Feature Test (first release)
- for all new test cases: Allocate time to Regression test (subsequent releases)

In this way, testing all critical new operations will be guaranteed. The remaining time goes to Load test. Since my Project is a new product, all test cases are new and test time will be allocated for Feature and Regression Tests and the remaining time will be allocated to Load Test. We know that each Feature test case will require 10 seconds of time in Feature Testing. I have a total of 640 hours and Regression Testing is done every 8 hours. Hence it is executed about 80 times in Regression tests.

SUBSYSTEM	NEW TEST CASES	TEST TIME (HOURS)	FEATURE TEST TIME	REGRESSION TEST TIME	LOAD TEST TIME
DEVELOPED	153	51	0.425	25.5	25.07
DURING DEVELOPMENT	37	12	0.102	6.16	5.738
OS	2	1	0	0	1

TABLE 26. Hours for FEATURE, REGRESSION & LOAD Tests

LOAD TEST TIME ALLOCATION

In this Project the System mode and the User mode occurrence ratio is 60:40. I have set this ratio keeping in mind that System mode operations are initiated by User as well as System itself. Thus, overall they occur more times than User mode operations.

SUBSYSTEM	LOAD TEST TIME	SYSTEM MODE	USER MODE
DEVELOPED	25.07	15.04	10.03
DURING DEVELOPMENT	5.738	3.44	2.29
OS	1	0.60	0.40

TABLE 27: Load Test Time Allocation per Operation mode

TEST DATA

Failure Number	Time Between Failures (min)	Failure Description	Failure Severity
1	10	Failure to power on when a car is powered on	Class 1
2	25	Failure to Acquire OBDII Data from a vehicle	Class 1
3	15	Failure to Acquire OBDII Data from a vehicle	Class 1
4	30	Failure to Acquire OBDII Data from a vehicle	Class 1
5	35	Failure to filter OBDII data of unimportant messages	Class 4
6	30	Failure to filter OBDII data of unimportant messages	Class 4
7	50	Failure to store Data to the RPi's SD Card	Class 4
8	60	Failure to detect WiFi availability	Class 2

TABLE 28: Test Data for TRIP MODE

TEST DATA

Failure Number	Time Between Failures (min)	Failure Description	Failure Severity
1	1	Failure to interpret coordinates with timestamps into a mapping API	Class 1
2	5	Failure to show information about individual vehicles	Class 2
3	5	Failure to interpret coordinates with timestamps into a mapping API	Class 2
4	10	Failure to show path of trip on map	Class 3
5	15	Failure to display total area of the map covered by trips	Class 3
6	10	Failure to display total distance travelled	Class 3
7	20	Failure to authenticate users	Class 1
8	20	Failure to authenticate users	Class 1
9	25	Failure to list driver trips with information of each trip	Class 2
10	30	Failure to list driver trips with information of each trip	Class 2
11	35	Failure to display total area of the map covered by trips	Class 3

TABLE 29: Test Data for DRIVER MODE

RESULTS

CASRE tool was used to analyze the test data and to run the tests. CASRE tool takes .dat file as input. The data was tested for read as an input file by CASRE® and the following steps were followed:

1. Parameter estimation method was selected.
2. Selected and ran the model(s)
3. Viewed and interpreted model results based on
   a) Goodness of fit test
   b) Model ranking
   c) Display results of prediction based on plots

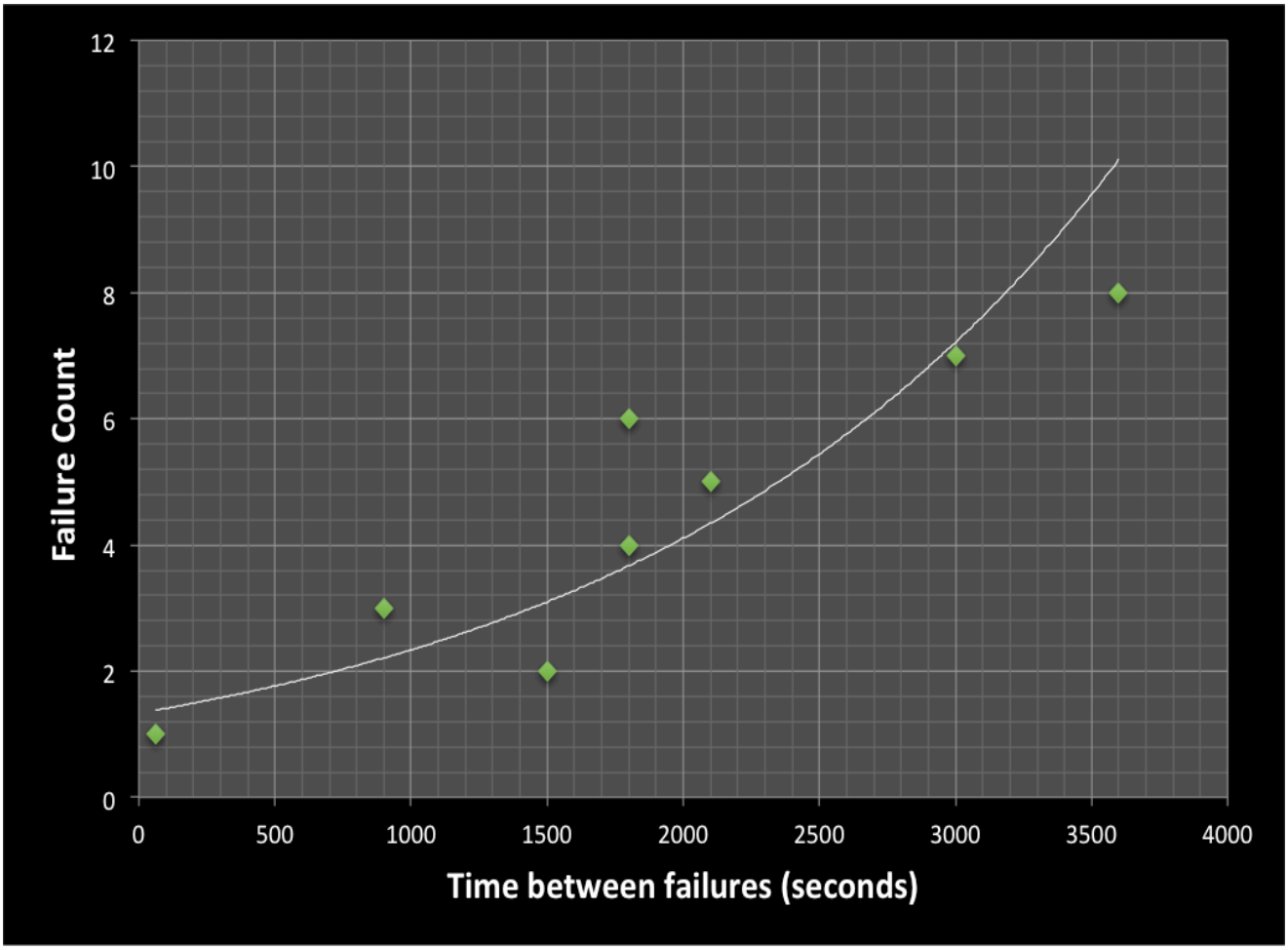


FIGURE 24. Time between failures graph for Trip mode

RESULTS

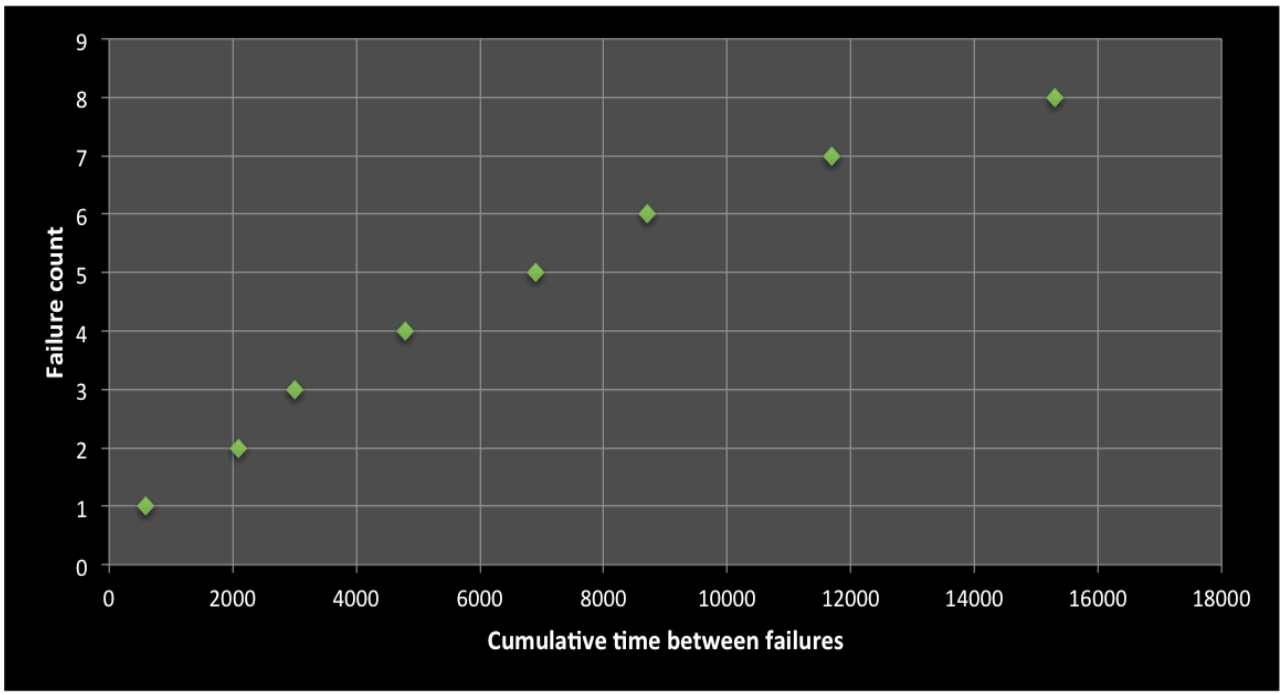


FIGURE 25. CUMULATIVE TIME between failures graph for Trip mode

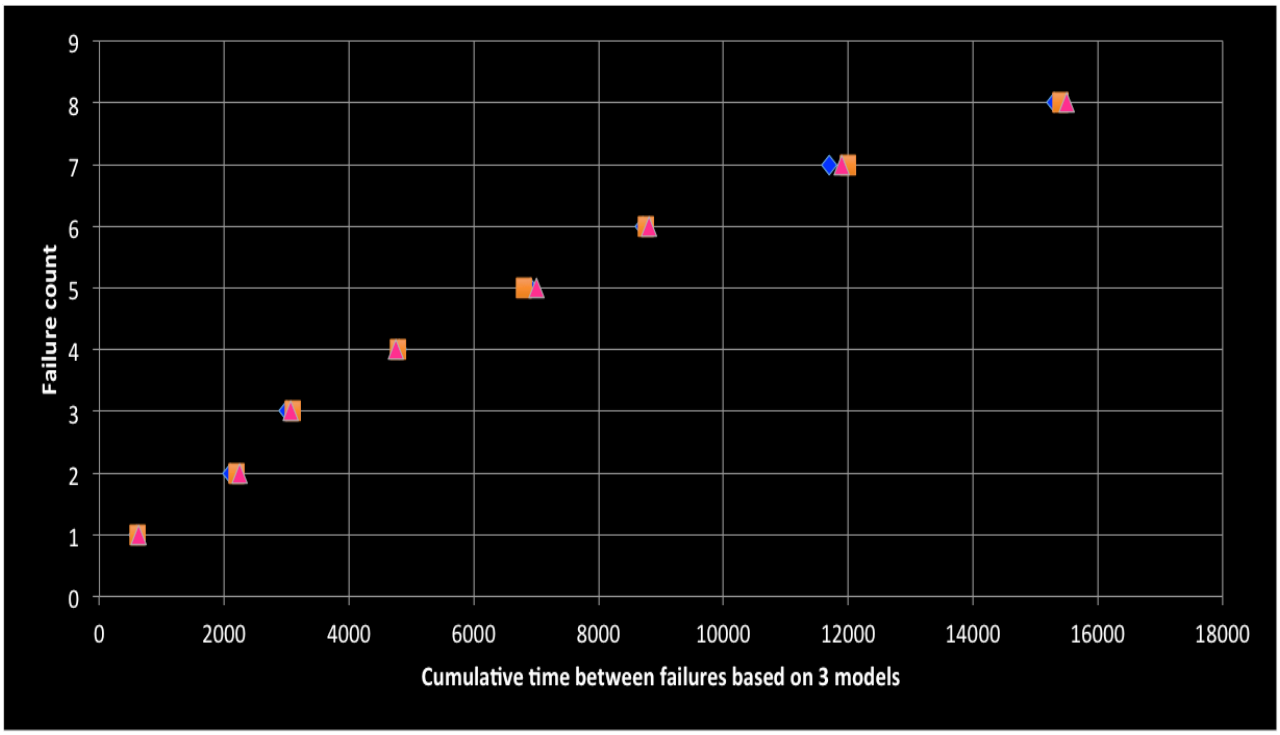


FIGURE 26. CUMULATIVE TIME between failures graph based on 3 models

RESULTS

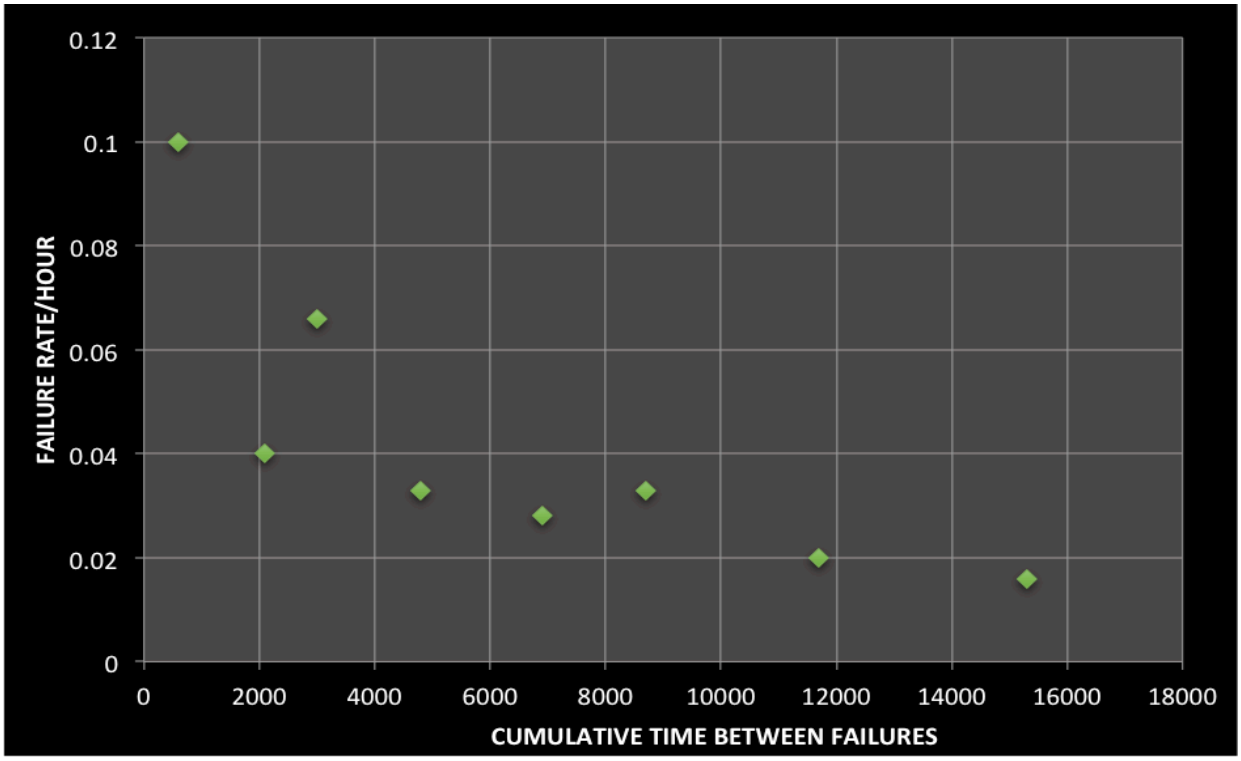


FIGURE 27. FAILURE INTENSITY GRAPH

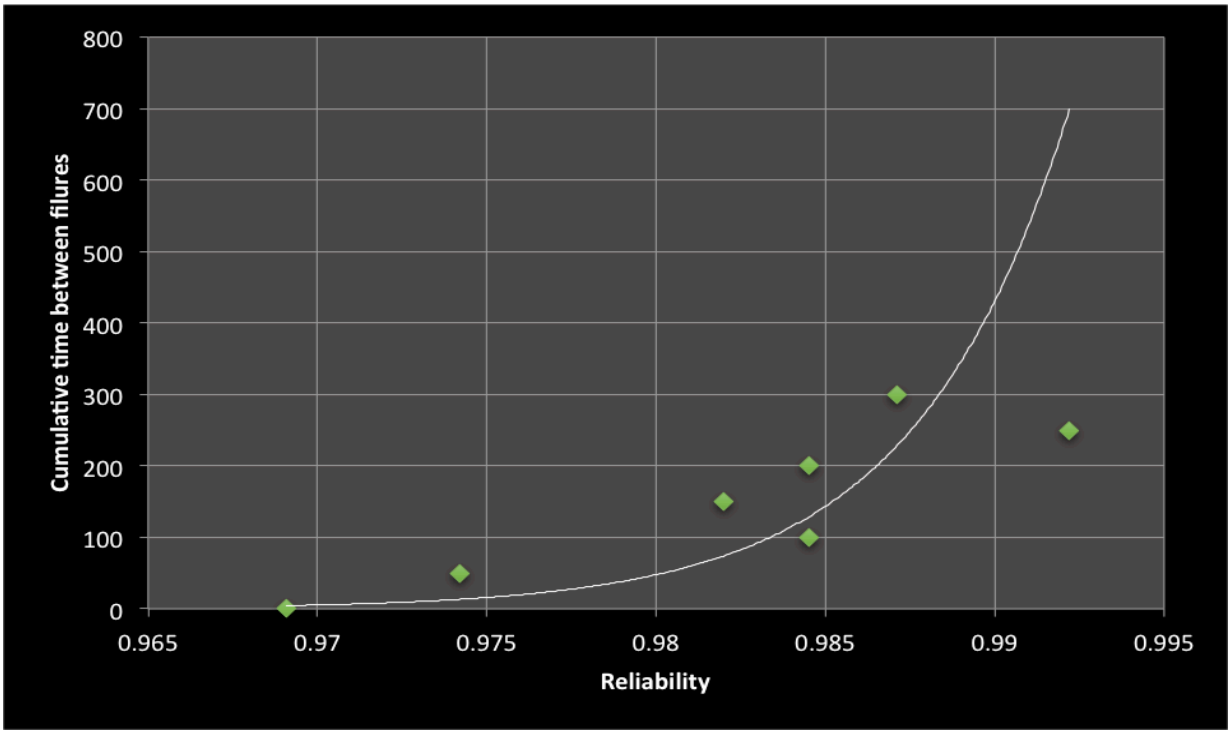


FIGURE 28. RELIABILITY GRAPH

RESULTS

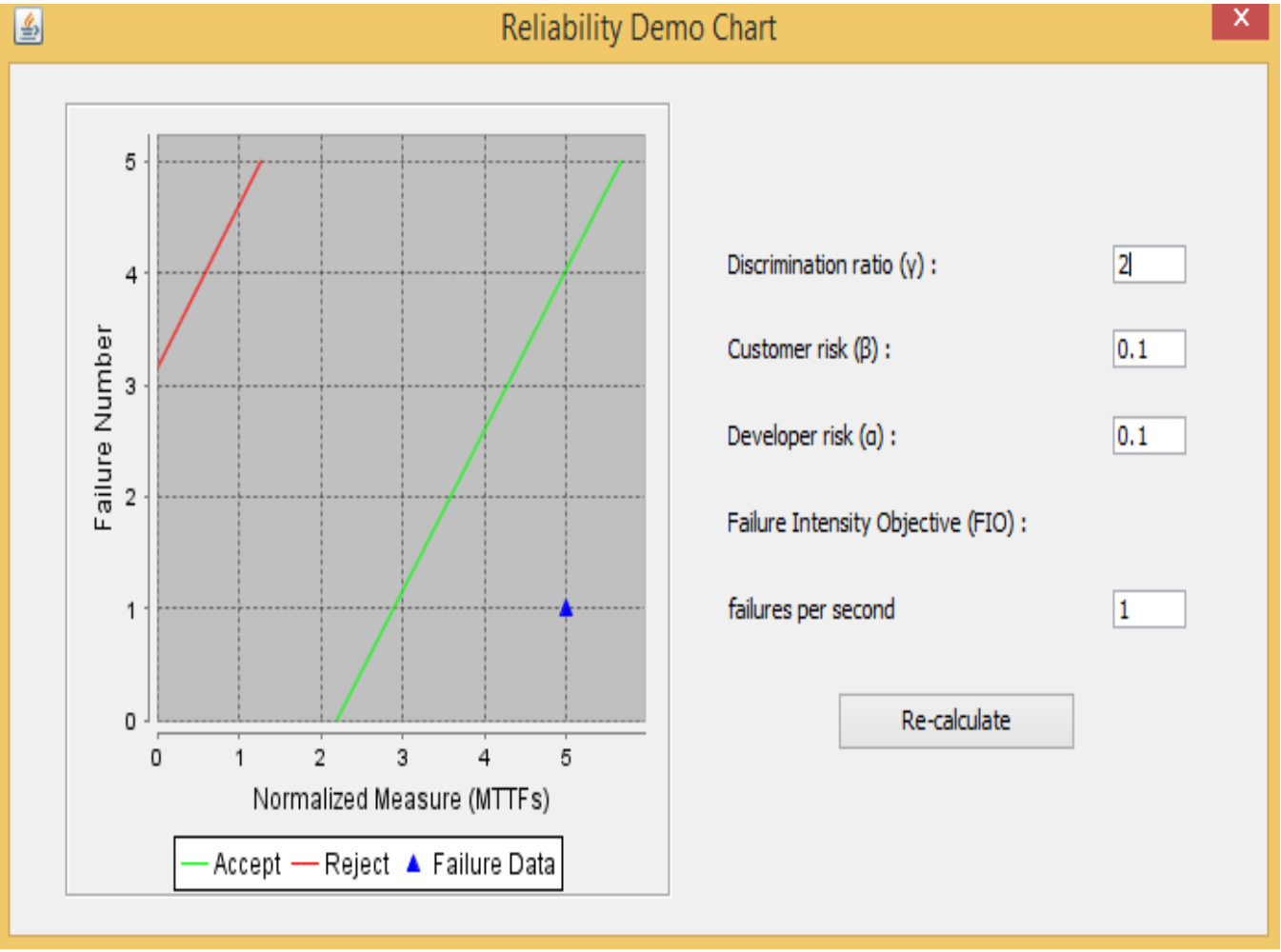


FIGURE 29. RELIABILITY DEMO CHART

ANALYSIS

Figure 25 shows the Cumulative time between failure and it is evident that as we approach towards the end of the execution time period, the time between failures is increasing which shows improvement in reliability of the product.

For the Project, 3 models were selected, i.e. Geometric, Jelinski-Moranda and Linear Littlewood–Verrall. From the Casre tool analysis,the data sets are shown to fit Jelinski–Moranda Model the best.

Figure 26 shows the prediction of time between failures based on the selected three models, and all three models suggest an increase of time between failures, Jelinski –Moranda shows are markably large time between failures for the data set.

Figure 27 shows the failure intensity curve, which is evidently decreasing showing an increase in reliability.

Figure 29 shows reliability demo Chart with **Risk assumptions** of Discrimination ratio (γ) is 2, Consumer risk (β) is 10% and Supplier risk (α) is 10%.

CONCLUSION & RELEASE RECOMMENDATION

As we have calculated in Assignment # 3, the FIO for our project will be 1 failure per 100,000 drive cycles. Based on the results that I have analysed earlier in this section, I see an increase of reliability as I approach the end of the test execution time from Figure 28. Considering the results from Figure 27, we can see that the failure intensity is showing a decreasing trend. Hence I can say that I have achieved the targeted value of the failure intensity that was defined for the project earlier in Assignment 3. The trend obtained by analysing the Failure Intensity graph shows that the reliability is increasing and failure intensity is decreasing. I am confident to say that, if the testing is done for the whole allocated test execution time of Hours, I will have much lower failure intensity and a higher reliability value. I have made a number of assumptions throughout the development of this system. I have recognized that if each drive cycle is not completed in the assumed time, reliability and failure intensity will not reach the targeted value. Hence we say that the reliability of my project is highly dependent upon the time in which each drive cycle is completed, i.e. it is dependent on number of drive cycles per second. Even from Reliability Demo Chart in Figure 29, we can see that our Certification test result lies in the Acceptable region. This means that The project has got a Pass certificate and it is ready for release. Based on the discussion presented above, I proudly recommend the release of the OBD II Application. I hope that this application will improve vehicle diagnostics and help them figure out more failure.

REFERENCES

[1] http://www.onboarddiagnostics.com/download/u581.pdf
[2] http://www.nyc.gov/html/tlc/downloads/pdf/intro_to_obd2.pdf
[3] B. H. Far "Seng 637" Software Reliability & software quality Chapter 2 [online] Department of Electrical & Computer Engineering, University of Calgary
[4] B. H. Far "Seng 637" Software Reliability & software quality Chapter 3, 7,11,12,13 [online] Department of Electrical & Computer Engineering, University of Calgary
[5] https://en.wikipedia.org/wiki/Reliability_engineering