SOEN-6611
Software Measurement
Project Deliverable - I
Team - A

Team Members:
Nazanin AhmadyShahpourAbady(40086734)
Priyanka Anantha Padmanabhan(40040066)
Maryam Ansari Sadrabadi(40021574)
Rahul Reddy Ayyapaneni(40071075)
Mitra Azari Sissi(40021574)
Divyeshkumar Balar(40062267)
Nikitha Papani(40070806)

Guided by:
Dr. Pankaj Kamthan
Samia Hilal

June 15, 2018

Contents

List of Figures

List of Tables

1 Introduction

The Process of Software Measurement has been integrated with Software Engineering and depends on the global environment characteristics such as improvement, control, structuring, estimation, and analysis. Software Measurement is like a computation in any other behavior, it will obey the value of measurement if it obtains a wide range of **Acceptance and Validity**. This is the extremely beneficial effect on the principles of measurement in the subject. The empirical study of software measurement will emphasize the advantages and disadvantages of metrics that have been used and also validate the work of metric. The context related to the software development will act as an important role in the risk management because the metrics which are integrated into a predictive model will give predictive warning messages to the potential risks. To predict a software measurement process, different models, approaches, and standards are required.

2 Assumptions

- It is assumed that customer survey includes all necessary questions required for GQM for proposed metric set.
- It is also assumed that required past artifacts of products are available(Project documentation, Organization source code).

3 Objective

The main objective of this Software Measurement project is to provide **DESCRIPTIVE STATISTICS**[1] generation of random numbers. The DESCRIPTIVE STATISTICS will acknowledge the coefficients in a descriptive manner and illustrates the available data set. The required calculations such as mean, median and mode will obtain the measurement of center of distribution and retrieve the maximum number, minimum number and standard deviation will include the **measure of variability**. The source code of this project has been developed in R language.

4 GOAL-QUESTION-METRIC

The Software Measurement considered should be meaningful because almost in every software project time is the major resource to make the effective measurement. It requires a meaningful methodology which is called GQM (Goal Question Metric) approach[2] which is completely based on the goal. This Goal based approach will be the deciding factor for choosing data on which the measurement has to performed and the reason behind the measurement. GQM reveals the structure in the hierarchy, in which goals are finalized in the initial stages and the relationship between the goal to questions and questions to metrics are one-to-many. After deciding on the goal, relevant questions and related metrics will be generated. The goal should provide a statement for an achievement with some measurements.

5 GOAL

Customer Service Manager wants to **Improve the Customer Satisfaction** of the product by **analyzing** obtained data from the DESCRIPTIVE STATISTICS on recent customer feedback.

Purpose: To analyze the customer survey/feedback, in order to improve the overall customer satisfaction.
Perspective: Examine the usability from the view-point of the user and examine reliability and maintainability from the view-point of service manager.
Environment: In the context of the maintainance and quality assurance phase from the life cycle.

Table 1: GQM Definition

5.1 SUB-GOAL 1: Usability

Question 1: What is the estimated number of defects in the software?

Question 2: The Defect removal efficiency of the product?

Metric 1:Defect removal efficiency[10]

$$DRE = \frac{E}{(E + D)}$$

where,
E= Errors before product delivery
D= Errors from end user after product delivery

Question 3: How effective is the software in completing a given task?

Metric 2: Effectiveness[10][11]

$$Effectiveness = \frac{Numberoftaskscompletedsuccessfully}{Totalnumberoftasksundertaken} X100$$

Question 4: How efficient is our software in terms of task-time?

Metric 3: Time Based Efficiency[12]

$$TimeBasedEfficiency = \frac{\sum_{i=1}^{R} \sum_{j=1}^{N} \frac{n_i j}{t_i j}}{NR}$$

N = The total number of tasks (goals)
R = The number of users
Nij = The result of task i by user j; if the user successfully completes the task, then Nij = 1, if not, then Nij = 0.
Tij = The time spent by user j to complete task i. If the task is not successfully completed,

then time is measured till the moment the user quits the task.

Question 5: What is the satisfaction level of users of the software after a certain task?

Metric 4: Customer Satisfaction[11]

$$CSat = \frac{SumofallScore}{NumberofRespondents} X10$$

Question 6: What is the error rate of certain task performed by users who have accessed the software?

Metric 5: Net Promoter Score[11]

$$NetPromoterScore = \frac{Promoter - Detractor}{TotalRespondents}$$

Question 7: What is the extent that customer is able to utilize the existing features in the software?

Metric 6: Product Utilization Factor

"Analyzed and Designed theoretically by DivyeshKumar Balar and Rahul Reddy Ayyapaneni"

$$ProductUtilizationFactor = \frac{AverageNumberofFeatureutilizedbythecustomers}{Totalnumberofavailablefeaturesinthesoftwares}$$

5.2 SUB-GOAL 2 : Maintainability

Question 8: Is your code Simple?

Question 9: Is code structured well?

Metric 7: Customer Satisfaction Control Flow Complexity = E - N + 2
Where,
E= total number of arcs
N= total number of nodes

Metric 8: LOD - Lack of documentation[10]

$$LOD(c) = 1 - \frac{|D(c)|}{|T(c)|}$$

Where,
$D(c) = \{e \in T(c) | e.hasRDoc == true\}$
$T(C) = Succ(c, containsMethod) \cup c$
Where, $c \in Scope^{LOD}$

Question 10: What is level of abstraction of source code?

Metric 9: DIT - Depth of Inheritance[9]

$$DIT(c) = Max(dist(c, P(c)))$$

Where,
$P(c) = Pred^*(c, Inheritance^{DIT})$ Set of classes,c inherits directly or indirectly

5.3 SUB-GOAL 4: Reliability

Question 11: What is the Product(Application/Software) crash rate?

Metric 10: Application Crash Rate[13]

$$AverageCrashRate = \frac{TotalNumberofFailure}{TotalNumberofAttempt}$$

Question 12: What is Systems **dependability**?

Metric 11: Dependability[14]

$$Dependability = 1 - Probabilityoffailure$$

Where, Probability of failure is **mean crash rate**(ACR).

Question 13: Is your system **fault tolerant**?

Metric 12: Instability[8]

$$Instability(I) = \frac{Ce}{Ce + Ca}$$

Where,

$$Ce = EfferentCoupling\ Ca = AfferentCoupling$$

Question 14: Is your code **consistent** (In logical order)?

Metric 13: LCOM - Lack of Cohesion[8]

$$LCOM^* = \frac{\frac{1}{a}[\sum_{i=1}^{a} \mu(A_i)] - m}{1 - m}$$

Where,
a is number of attributes and m is number of methods

6 Use case Model

A Use case diagram is an **extension of UML** it is similar in behavior to UML, which are standard indications for modeling real-world objects and systems. Use case diagrams will standardize the functionality of the system based on actors, who are people or objects handling under predefined responsibilities in a system and relevant Use cases. The physical overview of use case is a combination of actions, services, and functions that the system has to perform. Basically, Use case diagram illustrates a sequence in a combination of actions that results in an **outcome of measurable** value for an object and it is obtained as a horizontal ellipse.
The Use case diagram consists of the following major components:
1. It describes the **relationship** between the actors and Use cases.
2. The Use cases, which are the **major roles of the actors** within the system.
3. The actors try to **involve individually** within the system and act according to their respective roles.
4. The margin defines the interest of the system related to the environment.

Our Use case diagram consists of **two actors** namely customer and customer service manager. The use case present for the customer is getData and the use case for customer service manager is sample data. With the help of these two use cases Descriptive Statistics has been generated which included Max.val, Min.val, Mean.val, Median.val, Mode.val, Standard Deviation.

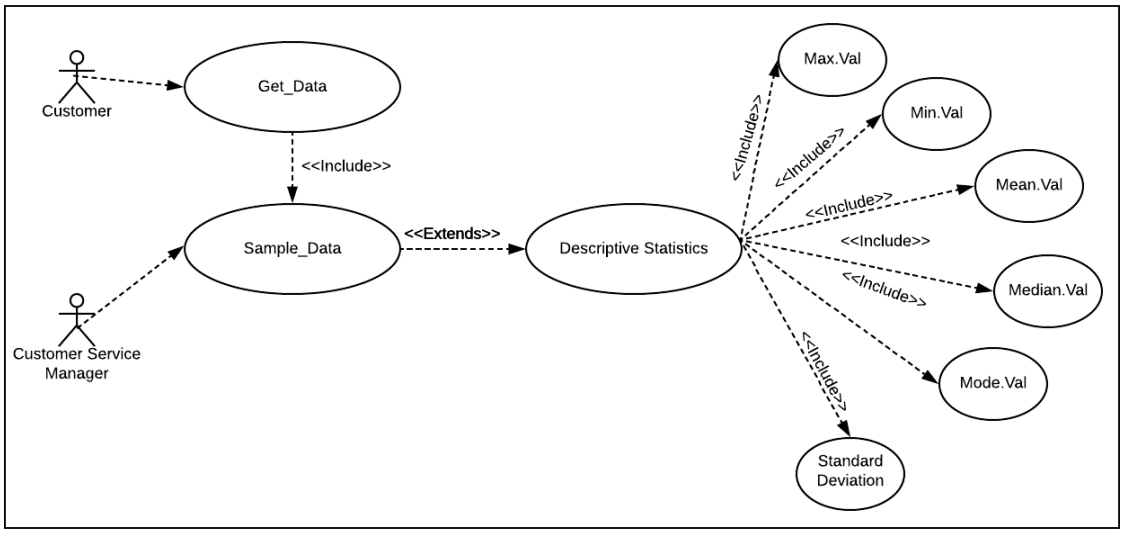

Figure 1: Use Case Approach

Use Case Model:	On demand DESCRIPTIVE STATISTICS(DS)
Brief Description:	The Input values will be selected randomly from the online customer satisfaction survey
Actor:	Customer Service Manager, Customer
PreConditions:	Do Customer Survey
PostConditions:	Save the result of DS into database for future use.
Trigger:	Customer Service Manager(CSM) will request to run the DS.
Program Scenario:	1.User performs survey and feed data to organizations database 2.Systems request the required quantity input from database and retrive in random order. 3.CSM will trigger the DS to run on that sample data. 4.developed R based system obtain results for Maximum, Minimum, Mean, Median, Mode,Arithmetic Mean and Standard Deviation and other mathematical expressions. 5.System Display statistical data for respected measurements.

Table 2: Use Case Description

7 Project Effort Estimation

7.1 Use case Points (UCP) Approach:

The UCP approach will **estimate the cost and effort** for a Software Measurement project based on the prediction values of the size of the software system.[3] The process of the estimation for effort for a software project is **much complex** in the initial stages of the software development life cycle where the details of the specific requirements are not clearly defined. There are multiple enhanced optimization techniques which are available to improve the accuracy in effort of a software project. The number of use case points in a software project is totally based on the complexity of the use cases available in the system and also based on the complexity of the actor in the system.

The UCP approach will be determined from four the following variables:

1.Technical Complexity Factor (TCF)
2.Environment Complexity Factor (ECF)
3.Unadjusted Use Case Points (UUCP)
4.Productivity Factor (PF)

From the output values of the above variables, the Effort estimation of a software system will result. The equation of the effort estimation is given by:

$$EffortEstimation(E) = UCP * PF = UUCP * TCF * ECF * PF \tag{1}$$

Where,
UCP = Use Case Points
PF = Productivity Factor
UUCP = Unadjusted Use Case Points
TCF = Technical Complexity Factor
ECF = Environment Complexity Factor

7.1.1 Technical Complexity Factor (TCF)

To make the effort estimation reliable, we should consider the technical factors, representing relative weights of the factors. There are 13 technical factors for the estimation of use case points approach. Technical complexity factors are designated with a value from 0 to 5 with respect to perceived complexity. The below values of the perceived complexity denotes:

0 - Irrelevant
3 - Average
5 - Strong Influence

These 13 technical factors will contribute the complexity of a software project according to the given weights in the below table:

Special user training facilities are required. The Calculated Factor is the combination of weight of the Technical complexity factor and Perceived complexity.

Factor	Description	Weight	Percieved complexity	Calculated Factor
T1	Distributed System	2	5	10
T2	Response time or throughput performance objectives	1	4	4
T3	End user efficiency	1	2	2
T4	Complex internal processing	1	3	3
T5	Code must be reusable	1	4	4
T6	Easy to install	0.5	5	2.5
T7	Easy to use	0.5	3	1.5
T8	Portable	2	2	4
T9	Easy to change	1	5	5
T10	Concurrent	1	1	1
T11	Includes special security objectives	1	2	2
T12	Provides direct access for third parties	1	4	4
T13	Special user training facilities are required	1	3	3

Table 3: TCF Calculation

Calculated Factor = Weight * Perceived Complexity

Then the Total Factor = 45

$$TCF = C1 + (C2 * TotalFactor) \tag{2}$$

Since the actor is Complex actor, we should consider assign the maximum values for the constants C1 and C2

Then C1 = 0.6 and C2 = 0.01

TCF = .60 + (0.01 * Total Factor)
= .60 + (0.01 * 45)
= .60 + 0.45
= 1.05
Hence The Technical Complexity Factor **TCF = 1.05**

7.1.2 Environment Complexity Factor (ECF)

Environment complexity factor will estimate the **impact on the productivity** on **multiple environmental variable** factors which are existing in a software project.[3] There are eight environment complexity factor and representing with weight factors.

Environmental Factor	Description	Weight	Percieved complexity	Calculated Factor
E1	Familiarity with UML	1.5	4	6
E2	Application Experience	0.5	2	1
E3	Object Oriented Experience	1	3	7
E4	Lead analyst capability	0.5	2	1
E5	Motivation	1	5	5
E6	Stable Requirements	2	4	8
E7	Part-time workers	-1	1	-1
E8	Difficult Programming language	2	1	1

Table 4: ECF Calculation

The Environment Total Factor = 28
Since for ECF C1 = 1.4 and C2 = -0.03
ECF = C1 + (-0.03 * Total Factor)
= 1.4 + (-0.03 * 28)
= 1.4 - 0.84
= 0.56
ECF = 0.56

7.1.3 Unadjusted Use case Point (UUCP)

Unadjusted Use Case is the combination of two basic variables.
UUCW = **U**nadjusted **U**se **C**ase **W**eight which consists total number of steps in use case process. UAW = **U**nadjusted **A**ctor **W**eight which is derived on the basis the complexity of the use case actors.

7.1.4 Unadjusted Actor Weight (UAW)

The UAW is measured on the basis of number of actors available in the software system and combining the result with weight of the particular actor.

Actor Type	Use Case Type	Weight	No of Use cases	Results
Simple	Simple	1	0	0
Average	Average	2	1	2
Complex	Complex	3	0	0

Total UAW = 2

7.1.5 Unadjusted Use case Weight (UUCW)

The Unadjusted use case weight is estimated based on number of use cases available in the software system per each category and also by combining the value of use case along with actor

weight.

Use Case Type	Description Weight	Weight	No of Use cases	Results
UC1	Simple Use Case	5	0	0
UC2	Average Use Case	10	1	10
UC3	Complex Use Case	15	0	0

Then the Total **UUCW = 10**
Hence calculate the Unadjusted Use case points
UUCP = UUCW + UAW = 10 + 2 = **12**.

7.1.6 Productivity Factor (PF)

The productivity is defined as the ratio of the number of man hours per use case scenario based on the previous software projects.
There will be a default value for the productivity factor is **20**.

Hence the Use case points UCP = TCF * ECF * UUCP = 1.05 * 0.56 * 12 = 7.056
Then from the value of **UCP = 7.056**, calculate Effort.
Effort Estimation = UCP * PF = 7.056 * 20 = **141.12 Person-Hours**

7.2 COCOMO 81 (COCOMO I)

The COCOMO 81 which is also called constructive estimation cost model is a fundamental method of **cost evaluation** of a software projects.[3] The software projects are classified into three classes:

1.Organic Projects
2.Semi-Detached
3.Embedded

7.2.1 Organic Projects

The Organic mode projects are **comparatively smaller and simpler software** projects. This is because these simple software projects contain a small team with good application experience. The organic projects look similar to the previously developed projects which require less amount of innovation.

7.2.2 Semi-Detached

The semi-detached projects are **intermediate in size and complexity** in the context of the software projects in which the teams have mixed experience levels must meet a mix of unaltered and less than immutable requirements.

7.2.3 Embedded

This is a type of software project which is developed under **very strong requirements** and a set of transparent hardware and software constraints.

The given equation for the basic COCOMO calculation is:

$$Effort = a(S)^b * F$$

a,b = coefficients
S = Size of the system
F = Adjustment Factor = 1

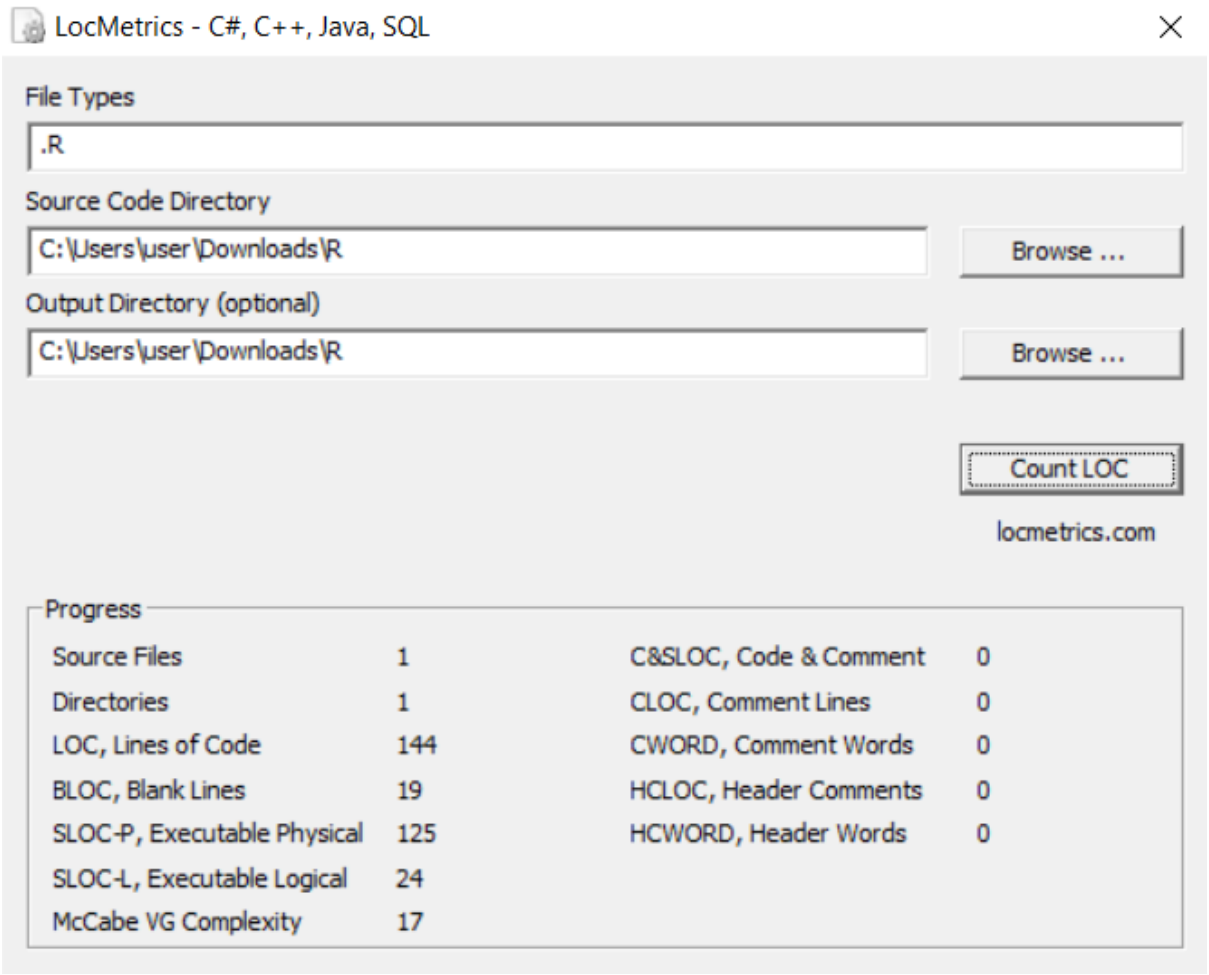

Figure 2: Lines of Code From LOCMetrics

The below is the tabular information regarding coefficients of COCOMO

Development Mode	**a**	**b**	**c**	**d**
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

$$\textbf{Effort} = \frac{a(S)^b * F}{} = \frac{3.0(0.179)^{1.12} * 1}{} = \textbf{0.436 Person-Month}$$

Lets consider **1 Person-Month = 176 Person-Hours**

Because the user will be working **22 days in a month**, **8 hours per day**.

Then Effort **E** = .436 * 176 = **77 Person-Hours**

This has been concluded from the above two different evaluations of effort estimation with Use case point approach and COCOMO calculation. The difference between Use case points approach and basic **COCOMO calculation** is **45 Percentage**. The use case point approach indicated 141 person-hours which are more predictive and relevant. COCOMO has given the result of 77 person-hours, which is less than use case point approach.
It is a better way to go with an **overestimation** since the development mode is semi-detached, which is a mixed level of experiences as well as with rigid requirements.

8 Cyclomatic Complexity Number

The concept of the Cyclomatic complexity indicates the level of complexity of the source code. The measurement of Cyclomatic complexity revolves around two variables.[6]

Nodes = n
Edges = e
The given equation for cyclomatic complexity is:

$$V(G) = e - n + 2 \tag{3}$$

Through this calculation, we can identify for the features like, whether the source code of the program is compatible with testing or not, is it able to understand the logic that exists in the source code and whether the source code is reliable. There are multiple cyclomatic complexity levels according to the risk management.

1 - 10 = Low - Simple Control flow graph

11 - 20 = Moderate - Complex control flow graph

21 - 50 = High - Complex, but manageable control flow graph

Greater Than 50 = Very Strong - Very Complex, unmanageable control flow graph.
The cyclomatic complexity of the DESCRIPTIVE STATISTICS are provided below along with control flow graphs.

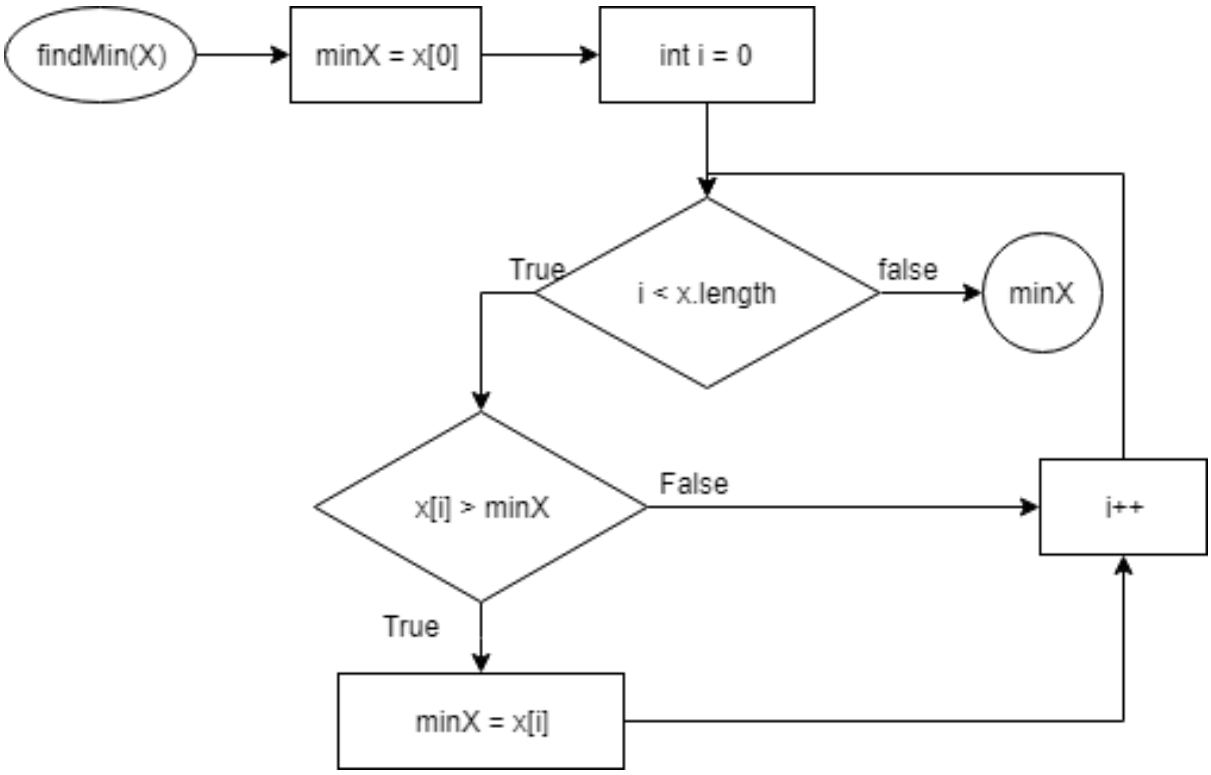

Figure 3: Control flow graph for Minimum Number

The Cyclomatic complexity = e - n + 2 = 9 - 8 + 2 = 3

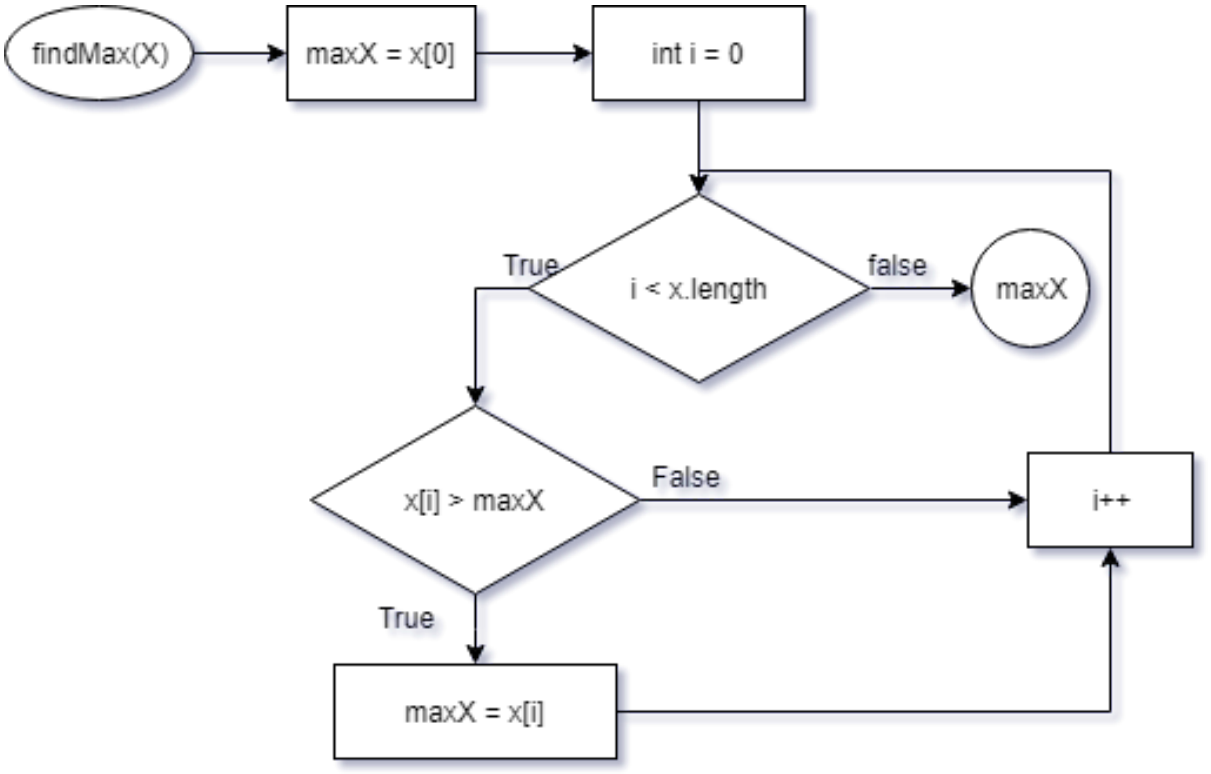

Figure 4: Control flow graph for Maximum Number

The cyclomatic complexity = e - n + 2 = 9 - 8 + 2 = 3

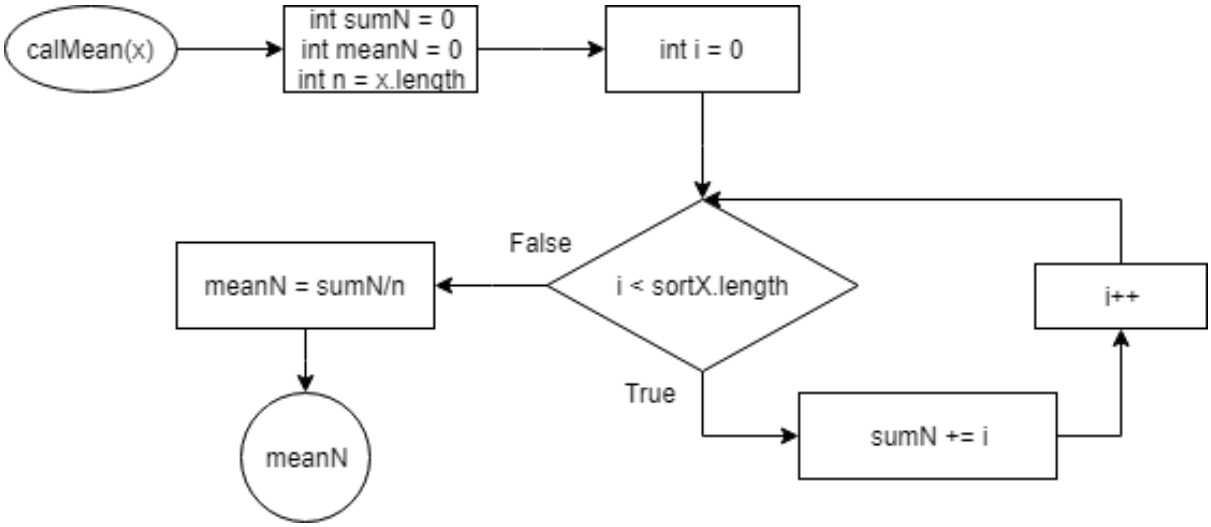

Figure 5: Control flow graph for Mean

The Cyclomatic complexity = e - n + 2 = 8 - 8 + 2 = 2

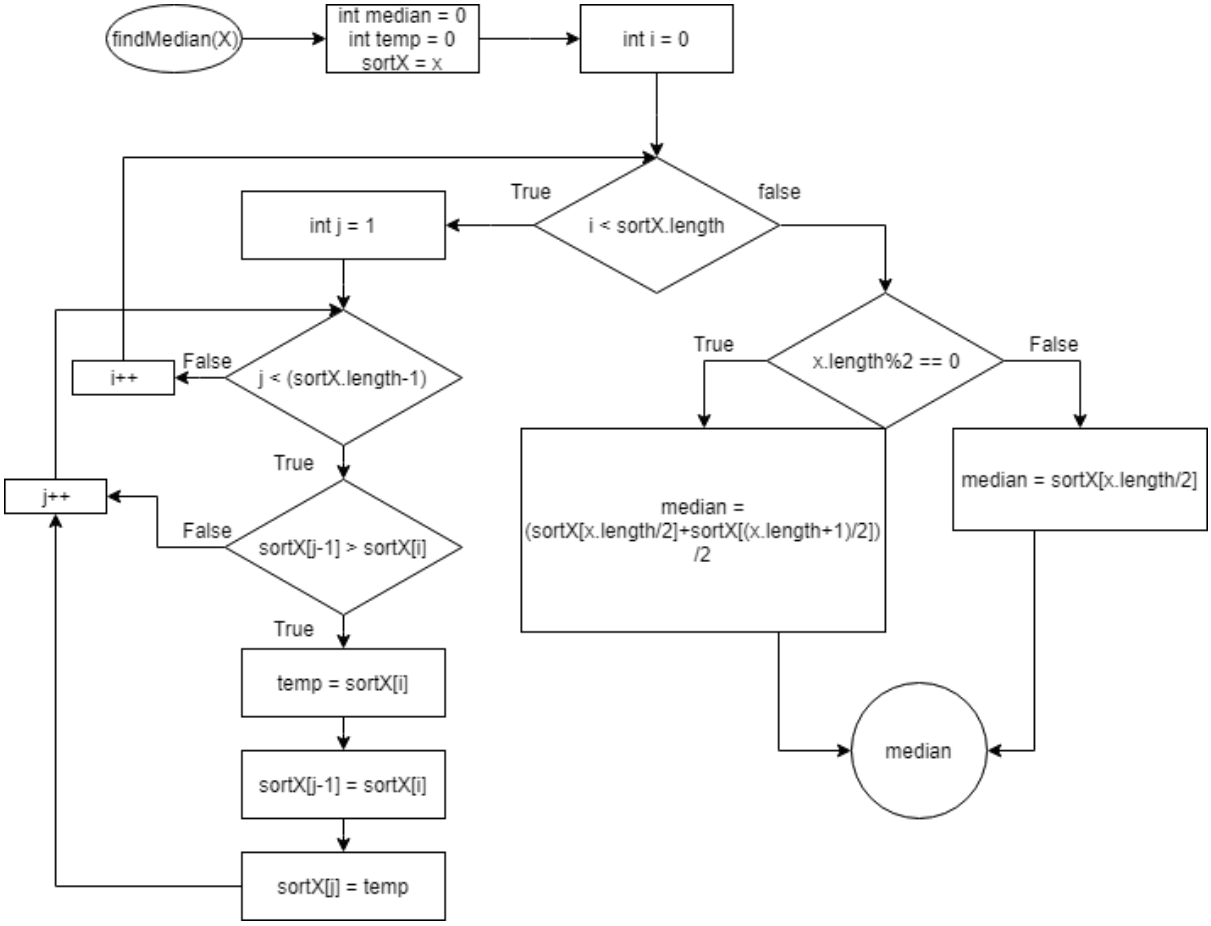

Figure 6: Control flow graph for Median

The Cyclomatic complexity = e - n + 2 = 19 - 16 + 2 = 5

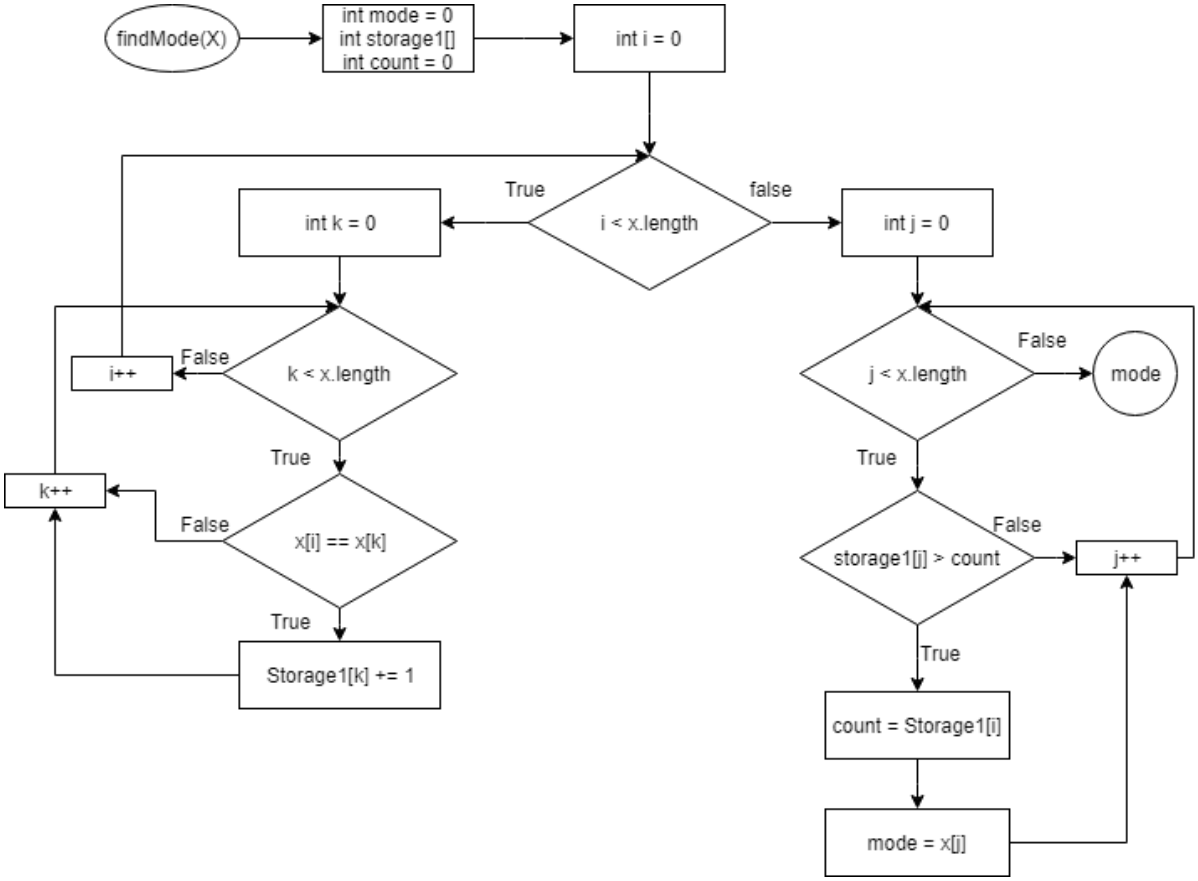

Figure 7: Control flow graph for Mode

The Cyclomatic complexity = e - n + 2 = 21 - 17 + 2 = 6

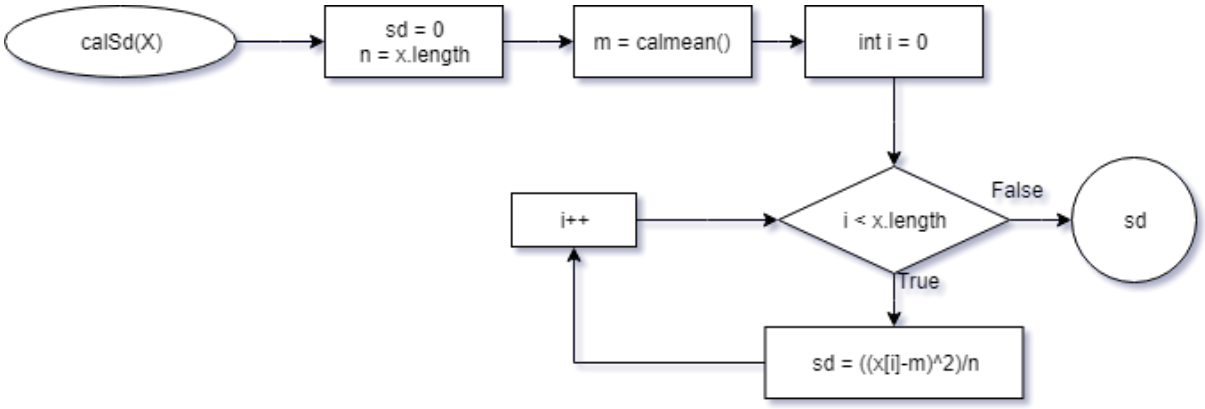

Figure 8: Control Flow Graph for Standard Deviation

The Cyclomatic Complexity = e - n + 2 = 8 - 8 + 2 = 2

9 Object-Oriented Metrics WMC, CF and LCOM*

The Object-Oriented Metrics Paradigm for assessing the system has become widely used in order to produce high-quality results. For measuring the quality of Object-Oriented Software, Chidamber and Kemerer software

Method Name	Cyclometric Complexity
getMinimum()	3
getMaximum()	3
getRandomNumber()	2
getMean()	2
getMedian()	5
getMode()	6
getStandard Deviation()	2

Table 5: WMC Values for each method

metrics suite was developed in the year 1994. C and K initially developed six metrics which were widely accepted by the Software Development Community. Weighted Methods per Class (WMC) and Lack of Cohesion in methods (LCOM*) are the metrics of that suite. Coupling Factor (CF) is one of the six metrics of the Metrics for Object Oriented Design (MOOD Suite) given by Fernando Brito and Rogerio Carpuca in 1994.[9]

9.1 Weighted Methods per Class (WMC)

WMC calculates the sum of **complexities of all methods** in a class. It is used to estimate the time and effort needed to develop and maintain a class.

$$WMC = \sum_{i=1}^{k} C_i$$

where, i= i^{th} complexity of the method.
k= total number of methods of class.

The total value of **WMC = 23**

9.2 Lack of Cohesion Methods($LCOM^*$)

LCOM is used for measuring the correlation between a method and the local instance variables of a class.

$$LCOM^* = \frac{\frac{1}{a}[\sum_{i=1}^{a} \mu(A_i)] - m}{1 - m}$$

where,
a is number of attributes and m is number of methods
Total attribute access = 18
Number of methods = 7
Number of attributes = 13
$LCOM^*$ **= 0.93**
Most of the methods had access to attributes which is an indicator of low coupling. Low $LCOM^*$ is a positive to quality attributes like analyzability, modifiability and testability.

9.3 Coupling Factor (CF)

CF is used for measuring the coupling between classes excluding the coupling due to Inheritance. It can be used to measure the size of the relationship between two classes. It can be computed as the ratio of of the possible

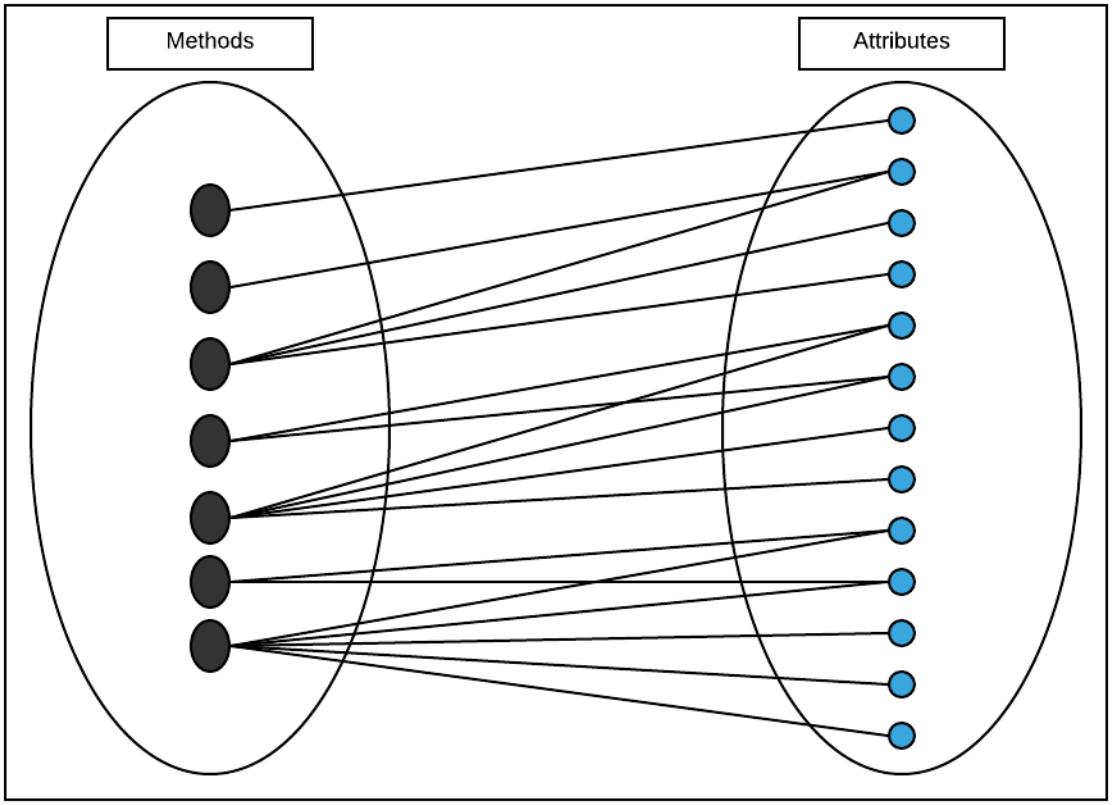

Figure 9: Scatter Plot of WMC and Rank

number of couplings in the software to the actual number of couplings not imputable to Inheritance.

$$CF = \frac{\sum_{i=1}^{n}(\sum_{j=1}^{n} IsClient(C_i, C_j))}{n^2 - n}$$

where,
Numerator is the actual number of couplings and denominator is the maximum possible couplings.
CF = $\frac{3}{42}$
CF = 0.07
Low Coupling Factor is a **positive to maintainability and testability** but it is a **negative to reusability**.

10 Physical SLOC and Logical SLOC

The major classifications of source lines of code are Physical SLOC and Logical SLOC. The definition will be different for both the measurements.[5]

10.1 Physical SLOC

It is actually the measure of the number of lines in the text format available in a particular program of the source code and it will take commented lines and blank lines into consideration.

10.2 Logical SLOC

In this case, it will measure only the statements or conditions which are going to be executed from the source code of a program. The statements or conditions depend on the programming language, for example: In C

program, every statement will be delimited with (;). Hence count is based on terminators available in the code. If we compare both Physical and Logical SLOC, Physical SLOC is very easier to describe and measure the style convention according to the language. The measurement of logical SLOC is irrelevant to formatting and styling conventions.

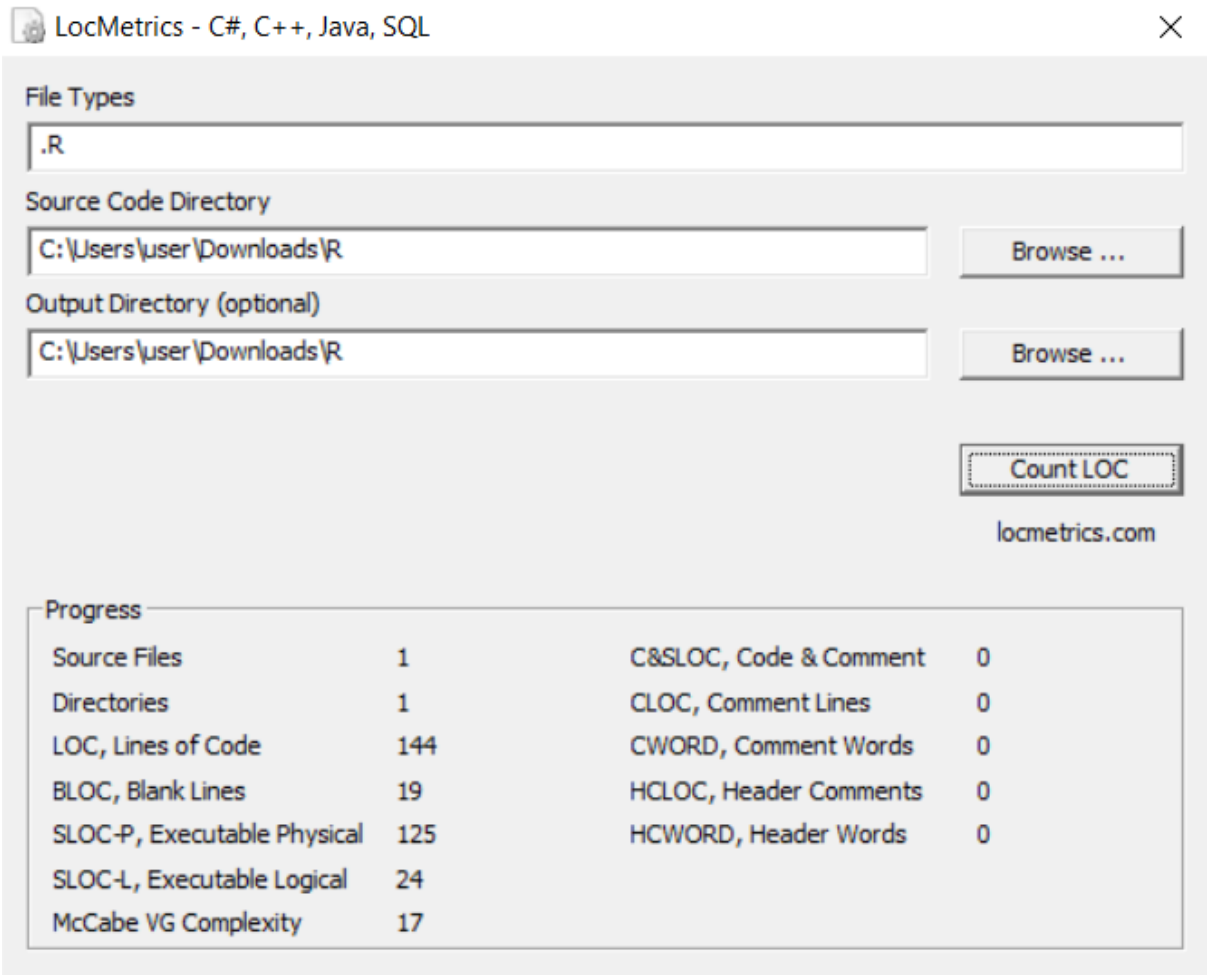

Figure 10: SLOC using LocMetrics

The **Logical SLOC = 24** using a tool called **LocMetrics**.

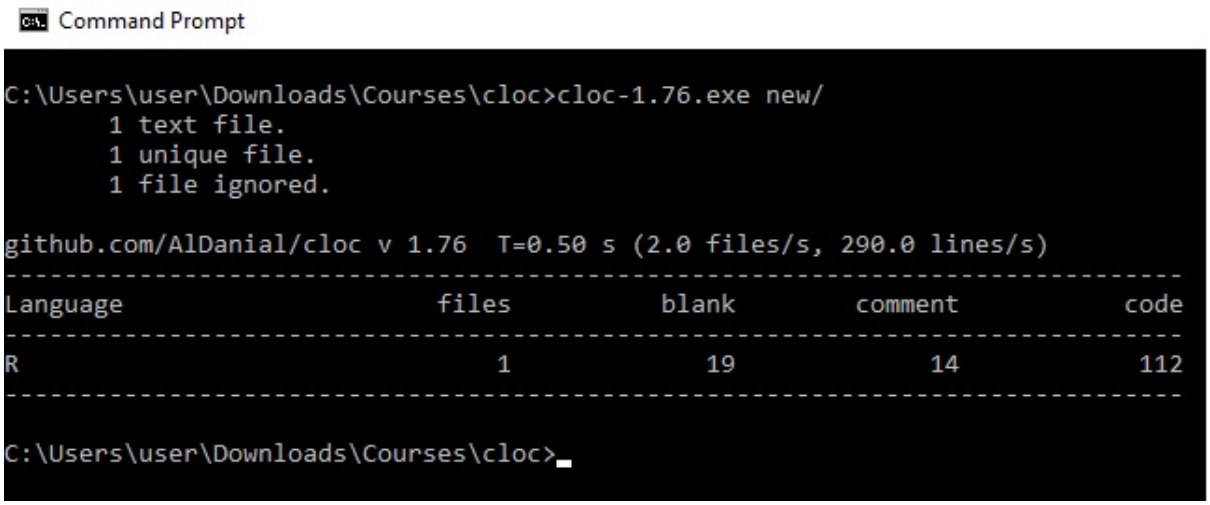

Figure 11: SLOC using CLOC

After analyzing our source code using two tools namely, LocMetrics and CLOC whose results are in Figure 9 and Figure 10 respectively, it is shown that both the tools have some drawbacks. Eventhough CLOC computes the SLOC value appropriately, it cannot compute Physical SLOC and Logical SLOC separately.LocMetrics is good comparatively as it can distinguish between physical and logical SLOC but it cannot identify the comments in our source code.
We have analyzed our source code manually and the results are as follows

Function	LOC	SLOC-P	SLOC-L	MVG
findMin ¡- function(x)	8	8	2	2
findMax ¡- function(x)	8	8	2	2
calmode ¡- function(x	26	26	6	7
sortdata ¡- function(x)	12	12	7	3
findmedian ¡- function(x)	17	17	3	1
calmean ¡- function(x)	9	9	2	1
calsd ¡- function(x)	10	10	2	1

Table 6: LOC Values for each function

11 Correlation Coefficient

Scatter Plot provides greater understanding between two variables. In order to understand the association between variable, We plotted historical data in Scatter Plot using Microsoft Excel. It helped us to understand and analyze precisely.[8]

WMC(x_i)	**Rank**(x_i)	**SLOC**(y_i)	**Rank**(y_i)	d_i	d_i^2
1	1	31	1	0	0
50	7	184	8	-1	-1
32	5	90	4	1	1
51	8	137	6	1	1
245	10	1128	10	0	0
34	6	142	7	-1	1
73	9	549	9	0	0
19	2	63	2	0	0
31	4	108	5	-1	1
21	3	86	3	0	0
23		24			

Table 7: Historic Data

First, we are analyzing the association of WMC and SLOC of our historic data from Table 6. From Figure 12, it is evident that SLOC and WMC have a positive association. With the increase in value of

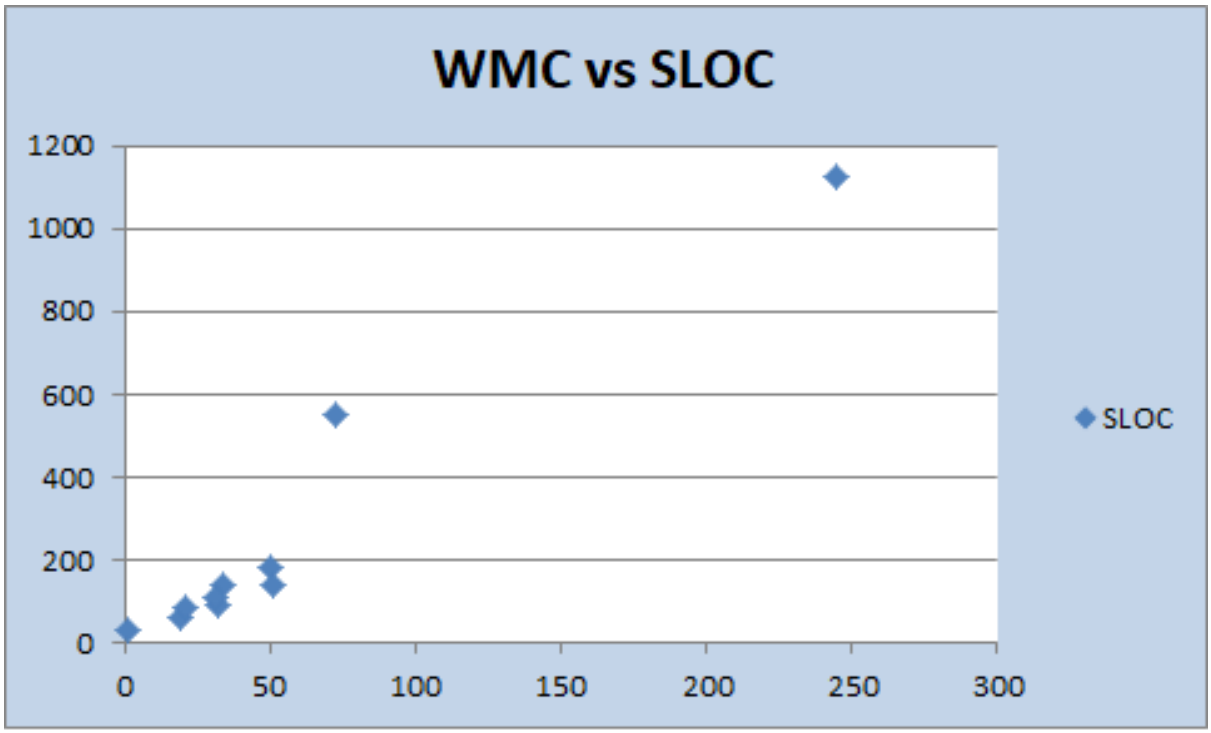

Figure 12: Scatter Plot of WMC and SLOC

SLOC, there will be a corresponding increase in the value of WMC.

11.1 Correlation Coefficient

From the historical data, we have ranks for WMC and SLOC. Now, we can determine ranks of our project using our SLOC and WMC value, which in turn helps us to find the correlation coefficient.

11.1.1 WMC vs Rank

We plotted the Scatter plot for WMC and Rank to find the rank value for our project. In Figure 13, we are

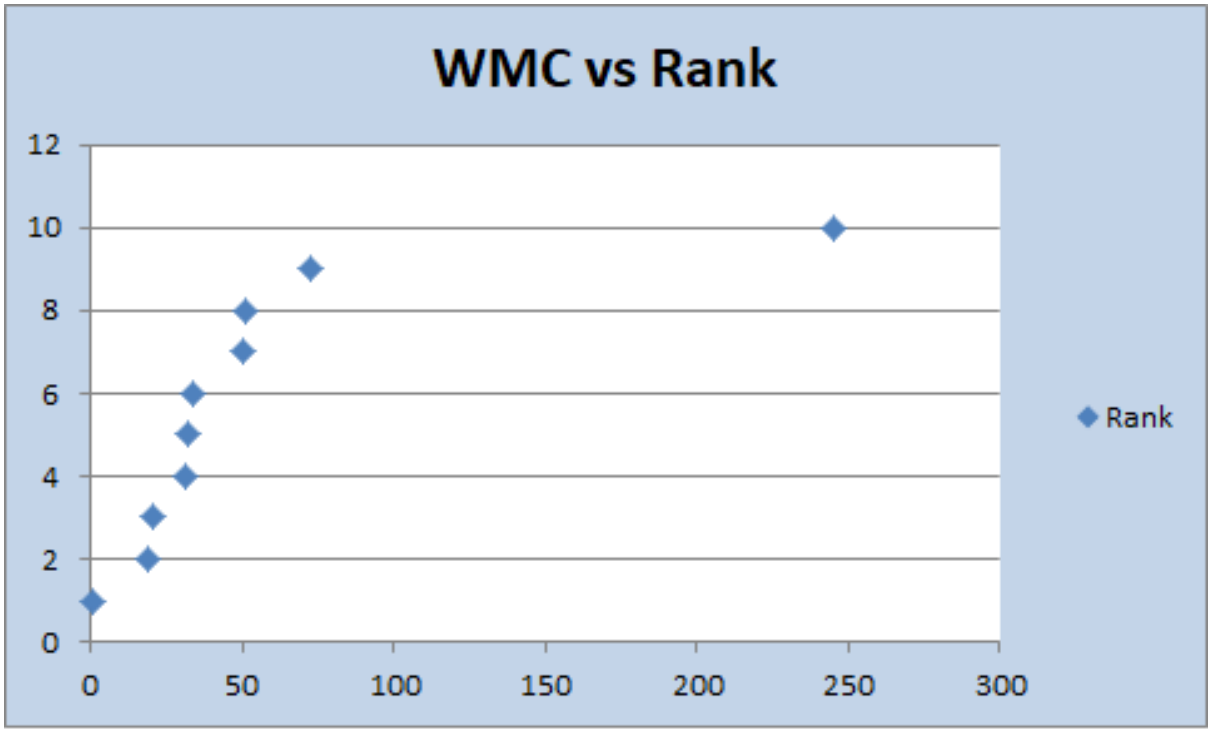

Figure 13: Scatter Plot of WMC and Rank

comparing WMC and Rank. So, we projected a line across the x-axis where x = 23 on this graph. We got an intersection at y-axis where y = 3.5. Rank for our project is 3.5.

11.1.2 SLOC vs Rank

In Figure 13, we are comparing SLOC and Rank. We projected a line across the x-axis where x = 24 on this graph. We got an intersection at y-axis where y = 0.7. Rank is 0.7.

11.1.3 Finding Correlation Coefficient

To find the correlation coefficient value, we are using Spearman's Rank Correlation Coefficient as our values are not normally distributed. The data is ranked in ascending order.[7]

$$r_s = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n^3 - n}$$

After substituting our values from Table 7 in equation, we obtain our correlation coefficient value.

$$r_s = 1 - \frac{134.64}{990} r_s = 0.997$$

Our correlation coefficient value is close to 1. Hence, it indicates a strong positive correlation. This data can be stored and used for future projects.

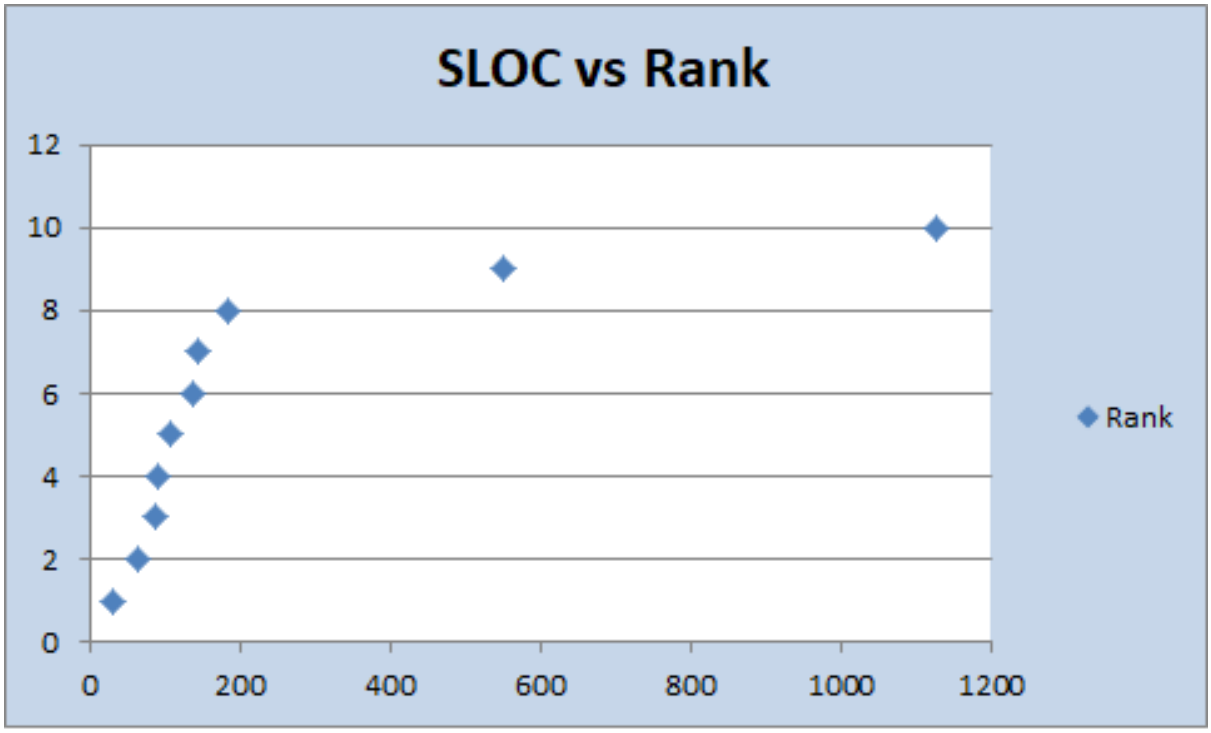

Figure 14: Scatter Plot of SLOC and Rank

WMC(x_i)	**Rank**(x_i)	**SLOC**(y_i)	**Rank**(y_i)	d_i	d_i^2
1	1	31	1	0	0
50	7	184	8	-1	1
32	5	90	4	1	1
51	8	137	6	2	4
245	10	1128	10	0	0
34	6	142	7	-1	1
73	9	549	9	0	0
19	2	63	2	0	0
31	4	108	5	-1	1
21	3	86	3	0	0
23	4.5	24	0.7	3.8	14.44

Table 8: Our Project Data

12 Conclusion

By analyzing the data which is generated randomly, we were able to deduce and compare measurements processed on the metrics which can be used to improve the customer satisfaction. We also realized the significance of having historical data to derive future estimation as well as deciding the threshold value, which can be helpful to satisfy our business goal.

13 Acknowledgement

The inclusion of an image in this document and the usage of mathematical equations related to metrics are for academic purpose, and its use is hereby acknowledged.

14 Contribution Distribution

14.1 Project Work distribution

1. Goal finalization, decision making and brainstorming.
    - Total Contribution of each member.
2. UseCase Modeling of part A
    - Rahul Reddy Ayyapaneni, Divyeshkumar Balar
3. Cost Estimation using COCOMO & UCP
    - Rahul Reddy Ayyapaneni, Priyanka Anantha Padmanabhan
4. R Programming, testing, documentation
    - Maryam Ansari Sadrabadi, Rahul Reddy Ayyapaneni, Priyanka Anantha Padmanabhan
5. Cyclometic Complexity measurement, Flow Chart making
    - Divyeshkumar Balar, Rahul Reddy Ayyapaneni
6. Calculation of WMC, CF and LCOM* for source code
    - Nikitha Papani, Divyeshkumar Balar, Mitra Azari Sissi, Rahul Reddy Ayyapaneni
7. Physical and Logical SLOC Calculation
    - Rahul Reddy Ayyapaneni
8. Scatter Plot analysis on correlation
    - Priyanka Anantha Padmanabhan, Nazanin AhmadyShahpourAbady

14.2 Documentation

1. LaTex Documentation
    - Divyeshkumar Balar, Rahul Reddy Ayyapaneni, Nikitha Papani, Priyanka Anantha Padmanabhan
2. validation & verification
    - Divyeshkumar Balar, Rahul Reddy Ayyapaneni, Nikitha Papani, Priyanka Anantha Padmanabhan

15 Glossary

A
Acknowledge acceptance or admitting something
Accuracy - the degree to which the result of a measurement, calculation, or specification conforms to the correct value or a standard
Actor - The actor can be a human or other external system
Append- add (something) as an attachment or supplement.
Analyze- discover or reveal (something) through detailed examination
Assigned- designate or set (something) aside for a specific purpose.
Average- a number expressing the central or typical value in a set of data, in particular the mode, median, or (most commonly) the mean, which is calculated by dividing the sum of the values in the set by their number

B
Beneficial - favorable

C
Complex- consisting of many different and connected parts.
Concurrent- existing, happening, or done at the same time
Computed- calculate
COCOMO - Constructive Cost Model.
Cohesion forming an united whole
Commercial- making or intended to make a profit.
Correlation- a mutual relationship or connection between two or more things.
Coupling- the pairing of two items.
Coefficient- a numerical or constant quantity placed before and multiplying the variable in an algebraic

D
Dataframe - a collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer.
Defect- imperfection
Dependability measure of systems availability, reliability
Descriptive - describe
Display- performance
Distributed systems- many independent computers linked by a network.
DRE Defect Removal Efficiency
DS - Descriptive Statistics

E
ECF Environment Complexity Factor
Estimated- roughly calculate or judge the value
Effort- a vigorous or determined attempt
Evaluate- form an idea of the amount, number, or value of; assess.
Embedded-Implant.
Executed- perform
Edges - Line connecting two nodes.

G
GQM - Goal, Question, Metrics.

I
Inspection - careful examination
Instability - low stability

L
LOD Lack of Documentation
LCOM Lack of Cohesion

M
Mean - In mathematics and statistics, the arithmetic mean, or simply the mean or average when the context is clear, is the sum of a collection of numbers divided by the number of numbers in the collection.
Median - The middle number.
Mode - The number which appears most often in a set of numbers.

O
OOD - Object oriented design.
Organic - denoting a relation between elements of something such that they fit together harmoniously as necessary parts of a whole.

P
Productivity - Productivity is an average measure of the efficiency of production.
Performance - the action or process of carrying out or accomplishing an action, task, or function
Portable - able to be easily carried or moved, especially because of being a lighter and smaller version than usual

R
Reusability - In computer science and software engineering, reusability is the use of existing assets in some form within the software product development process.

S

Semi-detached - partially separate.
Standard Deviation- In statistics, the standard deviation (SD, also represented by the Greek letter sigma or s) is a measure that is used to quantify the amount of variation or dispersion of a set of data values
System - the programs and other operating information used by a computer that considered.
Significant - sufficiently great or important to be worthy of attention.
Scenarios - a postulated sequence or development of events.

T
TCF Technical Complexity Factor

U
UML -Use case modeling.
UCP - Use case point.
UUCP - Unadjusted use case points.
UAW - Unadjusted use case actor weight.

V
Variability- not consistent or having a fixed pattern.
Variance- In probability theory and statistics, variance measures how far a set of numbers are spread out. A variance of zero indicates that all the values are identical.
Variables- an element, feature, or factor that is liable to vary or change

References

[1] P. Kamthan, "INTRODUCTION TO SOFTWARE MEASUREMENT". https://users.encs.concordia.ca/kamthan/courses/soen6611/software measurement introduction.pdf.

[2] P. Kamthan, "INTRODUCTION TO FRAMEWORKS FOR SOFTWARE MEASUREMENT". https://users.encs.concordia.ca/kamthan/courses/soen6611/software measurement frameworks introduction.pdf

[3] P. Kamthan, "INTRODUCTION TO SOFTWARE PROJECT COST ESTIMATION". https://users.encs.concordia.ca/kamthan/courses/soen6611/software project cost estimation introduction.pdf

[4] P. Kamthan, "THE ESTIMATION OF EFFORT BASED ON USE CASES". https://users.encs.concordia.ca/kamthan/courses/soen6611/effort estimation use cases.pdf

[5] P. Kamthan, "LENGTH OF SOURCE CODE". https://users.encs.concordia.ca/kamthan/courses/soen6611/source code length.pdf

[6] P. Kamthan, "CONTROL FLOW STRUCTURE OF SOURCE CODE". https://users.encs.concordia.ca/kamthan/courses/soen6611/source code control flow structure.pdf

[7] P. Kamthan, "INTRODUCTION TO ANALYSIS OF SOFTWARE MEASUREMENT DATA". https://users.encs.concordia.ca/kamthan/courses/soen6611/software measurement data analysis.pdf

[8] P. Kamthan, "METRICS FOR PACKAGES IN OBJECT-ORIENTED DESIGN ". https://users.encs.concordia.ca/kamthan/courses/soen6611/ood metrics packages.pdf

[9] P. Kamthan, "METRICS FOR CLASSES IN OBJECT-ORIENTED DESIGN".https://users.encs.concordia.ca/kamthan/courses/soen6611/ood metrics classes.pdf

[10] "Software Quality Metrics". http://www.arisa.se/compendium/node88.html/.

[11] CSAT VS. NPS VS. CES: YOUR GUIDE TO CUSTOMER SUPPORT SCORES https://www.taskus.com/blog/csat-vs-nps-vs-ces-guide-customer-support-scores/.

[12] F. Eric, H. Morton, H. Kasper "Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?" https://dl.acm.org/citation.cfm?id=332455

[13] "16 metrics to ensure MobileAppSuccess". https://www.appdynamics.com/media/uploaded-files/1432066155/white-paper-16-metrics-every-mobile-team-should-monitor.pdf.

[14] P. Ramachandran, S. Adve et. all"Metrics for Lifetime Reliability"