Name: Arka Das

MCA 2nd year 1st Semester

Roll number: 002210503046

Session: 2022 - 2024

software engineering lab report

Jadavpur University

**Assignment- 1**

**Question: 1**

Write a menu driven program (Using COCOMO I)

**Part: (i) Basic COCOMO**

**Problem Statement:**

1. Determine the Effort to develop of a Software product
2. Determine the Duration to develop of a Software product
3. Determine the Number of People engaged to develop of a Software product.

Input of your program is Lines of Code and Effort Adjustment Factor.

Also determine the type of project (i) Organic, (ii) Semi-detached and (iii) Embedded.

**Source Code:**

#include <stdio.h>

#include <math.h>

double Organic(double kloc, double eaf);

double Semi\_Detached(double kloc, double eaf);

double Embedded(double kloc, double eaf);

int main() {

double kloc, eaf;

printf("Enter KLOC: ");

scanf("%lf", &kloc);

printf("Enter Effort Adjustment Factor: ");

scanf("%lf", &eaf);

printf("\_\_\_\_\_\_\_For Organic\_\_\_\_\_\_\_\_\n");

double effort = Organic(kloc, eaf);

double time = 2.5 \* pow(effort, 0.38);

double person = effort / time;

printf("Effort required: %.0lf person-months\n", round(effort));

printf("Time required: %.0lf months\n", round(time));

printf("Person required: %d\n", (int)(person + 0.5));

printf("\_\_\_\_\_\_\_For Semi\_Detached\_\_\_\_\_\_\_\_\n");

effort = Semi\_Detached(kloc, eaf);

time = 2.5 \* pow(effort, 0.35);

person = effort / time;

printf("Effort required: %.0lf person-months\n", round(effort));

printf("Time required: %.0lf months\n", round(time));

printf("Person required: %d\n", (int)(person + 0.5));

printf("\_\_\_\_\_\_\_For Embedded\_\_\_\_\_\_\_\_\n");

effort = Embedded(kloc, eaf);

time = 2.5 \* pow(effort, 0.32);

person = effort / time;

printf("Effort required: %.0lf person-months\n", round(effort));

printf("Time required: %.0lf months\n", round(time));

printf("Person required: %d\n", (int)(person + 0.5));

return 0;

}

double Organic(double kloc, double eaf) {

double a = 2.4;

double b = 1.05;

double c = 2.5;

double d = 0.38;

double E = a \* pow(kloc, b) \* eaf;

double time = c \* pow(E, d);

double person\_required = E / time;

return E;

}

double Semi\_Detached(double kloc, double eaf) {

double a = 3.0;

double b = 1.12;

double c = 2.5;

double d = 0.35;

double E = a \* pow(kloc, b) \* eaf;

double time = c \* pow(E, d);

double person\_required = E / time;

return E;

}

double Embedded(double kloc, double eaf) {

double a = 3.6;

double b = 1.20;

double c = 2.5;

double d = 0.32;

double E = a \* pow(kloc, b) \* eaf;

double time = c \* pow(E, d);

double person\_required = E / time;

return E;

}

**Output:**

Enter KLOC: 25

\_\_\_\_\_\_\_For Organic\_\_\_\_\_\_\_\_

Effort required: 70 person-months

Time required: 13 months

Person required: 6

\_\_\_\_\_\_\_For Semi\_Detached\_\_\_\_\_\_\_\_

Effort required: 110 person-months

Time required: 13 months

Person required: 9

\_\_\_\_\_\_\_For Embedded\_\_\_\_\_\_\_\_

Effort required: 171 person-months

Time required: 13 months

Person required: 13

**Part: (ii) Intermediate COCOMO**

**Problem Statement:**

1. Determine the Effort to develop a Software product.
2. Determine the Time to develop a Software Product.
3. Determine the Number of People engaged to develop of a Software product.
4. The value of EAF

The inputs of your program are the Lines of Code, and a set of “cost drivers” having values for its 15 attributes.

Also determine the type of project (i) Organic, (ii) Semi-detached and (iii) Embedded.

**Source Code:**

#include <stdio.h>

#include <math.h>

double Organic(double kloc, double eaf);

double Semi\_Detached(double kloc, double eaf);

double Embedded(double kloc, double eaf);

int main() {

double kloc, eaf = 1.0;

double costDrivers[15];

printf("Enter KLOC: ");

scanf("%lf", &kloc);

printf("Enter values of 15 cost dirvers(space separated): \n");

for(int i=0;i<15;i++)

scanf("%lf", &costDrivers[i]);

// calculate the effort adjustment factor

for(int i=0;i<15;i++)

eaf \*= costDrivers[i];

printf("\_\_\_\_\_\_\_For Organic\_\_\_\_\_\_\_\_\n");

double effort = Organic(kloc, eaf);

double time = 2.5 \* pow(effort, 0.38);

double person = effort / time;

printf("Effort required: %.0lf person-months\n", round(effort));

printf("Time required: %.0lf months\n", round(time));

printf("Person required: %d\n", (int)(person + 0.5));

printf("\_\_\_\_\_\_\_For Semi\_Detached\_\_\_\_\_\_\_\_\n");

effort = Semi\_Detached(kloc, eaf);

time = 2.5 \* pow(effort, 0.35);

person = effort / time;

printf("Effort required: %.0lf person-months\n", round(effort));

printf("Time required: %.0lf months\n", round(time));

printf("Person required: %d\n", (int)(person + 0.5));

printf("\_\_\_\_\_\_\_For Embedded\_\_\_\_\_\_\_\_\n");

effort = Embedded(kloc, eaf);

time = 2.5 \* pow(effort, 0.32);

person = effort / time;

printf("Effort required: %.0lf person-months\n", round(effort));

printf("Time required: %.0lf months\n", round(time));

printf("Person required: %d\n", (int)(person + 0.5));

return 0;

}

double Organic(double kloc, double eaf) {

double a = 3.2;

double b = 1.05;

double c = 2.5;

double d = 0.38;

double E = a \* pow(kloc, b) \* eaf;

double time = c \* pow(E, d);

double person\_required = E / time;

return E;

}

double Semi\_Detached(double kloc, double eaf) {

double a = 3.0;

double b = 1.12;

double c = 2.5;

double d = 0.35;

double E = a \* pow(kloc, b) \* eaf;

double time = c \* pow(E, d);

double person\_required = E / time;

return E;

}

double Embedded(double kloc, double eaf) {

double a = 2.8;

double b = 1.20;

double c = 2.5;

double d = 0.32;

double E = a \* pow(kloc, b) \* eaf;

double time = c \* pow(E, d);

double person\_required = E / time;

return E;

}

**Output:**

Enter KLOC: 25

Enter values of 15 cost dirvers(space separated):

1.15 1.08 1.15 1.11 1.06 1.15 1.07 0.86 0.91 0.86 0.90 0.95 0.91 0.91 1.10

\_\_\_\_\_\_\_For Organic\_\_\_\_\_\_\_\_

Effort required: 102 person-months

Time required: 14 months

Person required: 7

\_\_\_\_\_\_\_For Semi\_Detached\_\_\_\_\_\_\_\_

Effort required: 120 person-months

Time required: 13 months

Person required: 9

\_\_\_\_\_\_\_For Embedded\_\_\_\_\_\_\_\_

Effort required: 144 person-months

Time required: 12 months

Person required: 12

**Question: 2**

**Problem Statement:**

Write a menu driven program using COCOMO II with the following sub-models

1. Application Composition Model
   1. Determine the Object Point to develop a Software product.
   2. Determine the New Object Point to develop a Software product.
   3. Determine the Effort to develop a Software product.
   4. Determine the Total Project Cost if Labour Rate is $1500.

Assume there are 14 screens, 6 reports, 10 components & 90% reuses. Productivity is 25.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Complexity factor | Count | Simple | Average | Complex |
| Screens | 14 | 1 | 2 | 3 |
| Reports | 6 | 2 | 5 | 8 |
| Components | 10 | 1 | 1 | 10 |

1. Early Design Stage Model
   1. Determine the Effort to develop a Software product.
   2. Determine the Number of People engaged to develop a Software product, if duration of development of the software is 5 years.
2. Post Architecture Stage Model
   1. Determine the Effort to develop a Software product.
   2. Determine the Number of People engaged to develop a Software product, if duration of development of the software is 5 years.

EXAMPLE TABLE FOR Cost Driver

COST

DRIVER Very-Low Low Normal High Very-High

Cd1 1.15

Cd2 1.3

Cd3 1.1

Cd4 1.16

Cd5 1.4

Cd6 0.85

Cd7 0.7

Cd8 1.21

Cd9 0.86

Cd10 0.9

Cd11 1

Cd12 1

Cd13 0.8

Cd14 1

Cd15 0.83

Cd16 0.85

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct Cost\_Driver

{

int stage; // very low -> 0 // low -> 1 // Normal -> 2 // High -> 3 // Very High->4

float value;

} CD;

void Simple(float screen, float reports, float components, float reusability, float productivity);

void Medium(float screen, float reports, float components, float reusability, float productivity);

void Complex(float screen, float reports, float components, float reusability, float productivity);

void EarlyDesignModel(int KLOC, float prodCostDrivers)

{

printf("Early Stage Design Model\n");

float effort = KLOC \* prodCostDrivers;

int time = 5; // year

float people = ceil((effort / (time \* 12)));

printf("Effort : %f\n", effort);

printf("People Needed : %f\n", people);

}

void PostArchitectureModel(int KLOC, float prodCostDrivers)

{

printf("Post Architecture Stage Model\n");

float a, b;

printf("1. Organic\n2. Semi-Detached\n3. Embedded\n");

int ch;

scanf("%d", &ch);

switch (ch)

{

case 1:

a = 2.4;

break;

case 2:

a = 3.0;

break;

case 3:

a = 3.6;

break;

default:

a = 3.6; /// default embedded

}

b = 1.10; // let

float effort = a \* pow(KLOC, b) \* prodCostDrivers;

int time = 5; // year

float people = ceil((effort / (time \* 12)));

printf("Effort : %f\n", effort);

printf("People Needed : %f\n", people);

}

int main()

{

float size;

float screen, reports, components, reusability, productivity;

int NoCostDrivers;

float prodCostDrivers = 1.0;

printf("Enter the size of the project in KLOC: ");

scanf("%f", &size);

printf("Enter number of Cost Drivers : ");

scanf("%d", &NoCostDrivers);

CD cDrivers[NoCostDrivers];

printf("Enter Cost Drivers\n");

printf("As Stage value (space separated):\n");

for (int i = 0; i < NoCostDrivers; i++)

{

scanf("%d %f", &cDrivers[i].stage, &cDrivers[i].value);

prodCostDrivers \*= cDrivers[i].value;

}

printf("\nThe type of project is simple.\n");

printf("Enter the number of screens, reports, components, reusability, productivity:\n");

scanf("%f %f %f %f %f", &screen, &reports, &components, &reusability, &productivity);

printf("\n");

if (size <= 30) // Simple Project : <= 30KLOC

Simple(screen, reports, components, reusability, productivity);

else if (size <= 60) // Medium Project : <= 60KLOC

Medium(screen, reports, components, reusability, productivity);

else // Complex Project : > 60KLOC

Complex(screen, reports, components, reusability, productivity);

printf("\n");

EarlyDesignModel(size, prodCostDrivers);

printf("\n");

PostArchitectureModel(size, prodCostDrivers);

return 0;

}

void Simple(float screen, float reports, float components, float reusability, float productivity)

{

float time;

float op, nop, effort, cost;

int choice;

printf("\nThe type of project is simple.\n");

printf("Enter the number of screens, reports, components, reusability, productivity:\n");

scanf("%f %f %f %f %f", &screen, &reports, &components, &reusability,

&productivity);

op = (screen \* 1 + reports \* 2 + components \* 1);

nop = op \* (100 - reusability) / 100;

effort = nop / productivity;

cost = 1500 \* (float)(nop / productivity);

printf("Object Points: %.2f\n", op);

printf("New Object Points: %.2f\n", nop);

printf("Effort (in person-months): %.2f\n", effort);

printf("Total Cost required: %.0f\n", cost);

}

void Medium(float screen, float reports, float components, float reusability, float productivity)

{

float op, nop, effort, cost;

op = (screen \* 2 + reports \* 5 + components \* 1);

nop = op \* (100 - reusability) / 100;

effort = nop / productivity;

cost = 1500 \* (float)(nop / productivity);

printf("Object Points: %.2f\n", op);

printf("New Object Points: %.2f\n", nop);

printf("Effort (in person-months): %.2f\n", effort);

printf("Total Cost required: %.0f\n", cost);

}

void Complex(float screen, float reports, float components, float reusability, float productivity)

{

float op, nop, effort, cost;

op = (screen \* 3 + reports \* 8 + components \* 10);

nop = op \* (100 - reusability) / 100;

effort = nop \* productivity;

cost = 1500 \* (float)(nop / productivity);

printf("Object Points: %.2f\n", op);

printf("New Object Points: %.2f\n", nop);

printf("Effort (in person-months): %.2f\n", effort);

printf("Total Cost required: %.0f\n", cost);

}

**Output:**

**Case 1:**

Enter the size of the project in KLOC: 60

Enter number of Cost Drivers : 16

Enter Cost Drivers

As Stage value (space separated):

3 1.15 4 1.3 3 1.1 4 1.6 4 1.4 1 0.85 1 0.7 3 1.21 3 0.86 3 0.9 2 1 2 1 1 0.8 2 1 4 0.83 0 0.85

The type of project is simple.

Enter the number of screens, reports, components, reusability, productivity:

14 6 10 90 25

Object Points: 68.00

New Object Points: 6.80

Effort (in person-months): 0.27

Total Cost required: 408

Early Stage Design Model

Effort : 69.512589

People Needed : 2.000000

Post Architecture Stage Model

1. Organic

2. Semi-Detached

3. Embedded

3

Effort : 376.860931

People Needed : 7.000000

**Case 2:**

Enter the size of the project in KLOC: 40

Enter number of Cost Drivers : 16

Enter Cost Drivers

As Stage value (space separated):

3 1.15 4 1.3 3 1.1 4 1.6 4 1.4 1 0.85 1 0.7 3 1.21 3 0.86 3 0.9 2 1 2 1 1 0.8 2 1 4 0.83 0 0.85

The type of project is simple.

Enter the number of screens, reports, components, reusability, productivity:

14 6 10 90 25

Object Points: 68.00

New Object Points: 6.80

Effort (in person-months): 0.27

Total Cost required: 408

Early Stage Design Model

Effort : 46.341724

People Needed : 1.000000

Post Architecture Stage Model

1. Organic

2. Semi-Detached

3. Embedded

1

Effort : 160.838303

People Needed : 3.000000

**Assignment- 2**

**Question: 1**

**Problem Statement:**

Failure data for 172 hypothetical electronic components are given in the accompanying table. Compute and sketch the following quantities:

1. The hazard function, z(t)
2. The density function, f(t)
3. The cumulative distribution function, F(t)
4. The reliability function, R(t)

Failure data for 172 hypothetical electronic components

|  |  |
| --- | --- |
| Time interval, h | Failures in the interval |
| 0 – 1000 | 59 |
| 1001 – 2000 | 24 |
| 2001 – 3000 | 29 |
| 3001 – 4000 | 30 |
| 4001 – 5000 | 17 |
| 5001 – 6000 | 13 |
| **Total** | 172 |

**Source Code:**

#include <stdio.h>

#include <math.h>

#define SIZE 6

int main() {

double time\_intervals[SIZE] = {1000, 2000, 3000, 4000, 5000, 6000};

int failures\_in\_interval[SIZE] = {59, 24, 29, 30, 17, 13};

int val = 172;

int survivors[SIZE];

int cumulative\_failures[SIZE] = {0};

double reliability[SIZE] = {0.0};

double cumulative\_fail\_distri[SIZE] = {0.0};

double failure\_density\_func[SIZE] = {0.0};

double hazard\_rate[SIZE] = {0.0};

// Calculate survivors

for(int i = 0; i < SIZE; i++) {

survivors[i] = val - cumulative\_failures[i];

}

// Calculate cumulative failures and reliability

for(int i = 0; i < SIZE; i++) {

cumulative\_failures[i] = i > 0 ? cumulative\_failures[i-1] + failures\_in\_interval[i] : failures\_in\_interval[i];

reliability[i] = (double)survivors[i] / val;

cumulative\_fail\_distri[i] = (double)cumulative\_failures[i] / val;

}

// Calculate density function and hazard rate

for(int i = 0; i < SIZE - 1; i++) {

failure\_density\_func[i] = (double)(cumulative\_failures[i + 1] - cumulative\_failures[i]) / (time\_intervals[i] \* val);

hazard\_rate[i] = (double)(cumulative\_failures[i + 1] - cumulative\_failures[i]) / (time\_intervals[i] \* survivors[i]);

}

// Print the results

printf("\n t \t\t R(t) \t F(t) \tf(t) \tz(t)\n");

printf("----------------------------------------------------------\n");

for(int i = 0; i < SIZE - 1; i++) {

printf("%f\t%.3f\t%.3f\t%.5f\t%.5f\n", time\_intervals[i], reliability[i], cumulative\_fail\_distri[i], failure\_density\_func[i], hazard\_rate[i]);

}

return 0;

}

**Output:**

t R(t) F(t) f(t) z(t)

----------------------------------------------------------

1000.000000 1.000 0.343 0.00014 0.00014

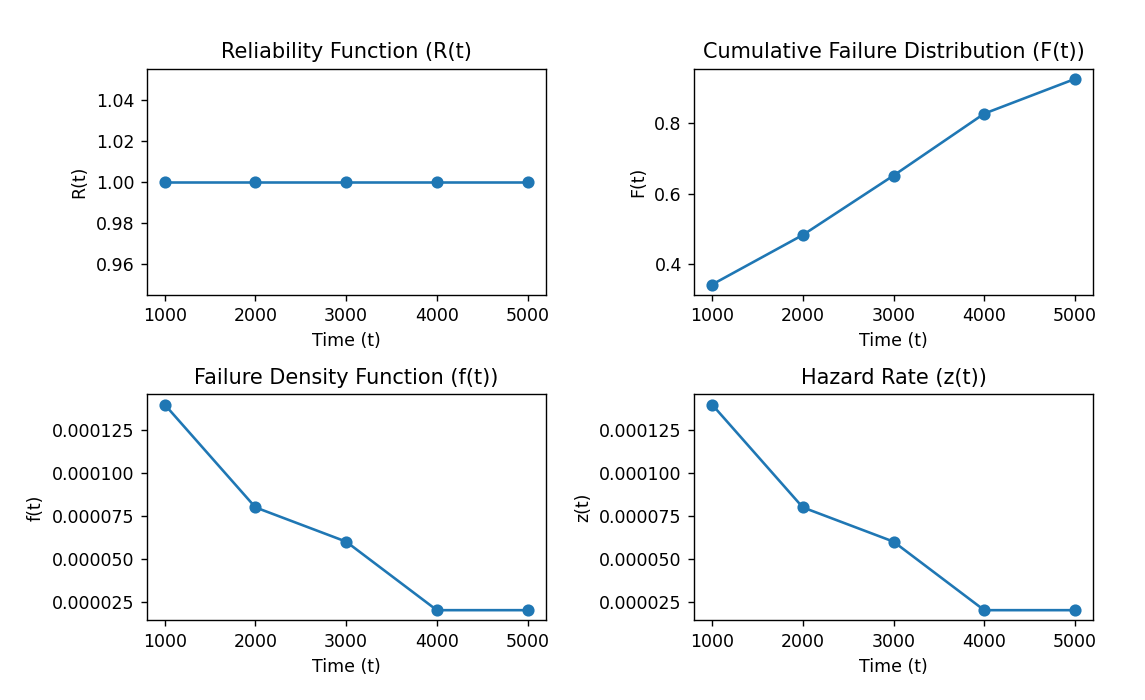
2000.000000 1.000 0.483 0.00008 0.00008

3000.000000 1.000 0.651 0.00006 0.00006

4000.000000 1.000 0.826 0.00002 0.00002

5000.000000 1.000 0.924 0.00002 0.00002

**Plots:**



**Question: 2**

**Problem Statement:**

Determine the reliabilities, of the following cases, of a electronic component which is operated by a software program.

1. CASE –I : If the hazard function is constant, λ, where λ = 0.3 and t = 3 hour, time to failure of the software.
2. CASE – II : If the hazard function is linearly increasing, kt, where k = 0.9 and t =5 hour, time to failure of the software.
3. CASE – III : If the hazard function is Weibul distribution, where m = 0.5, k = 0.9 and t = 5 hour, time to failure of the software.

Each of the above case plot a two dimensional graph reliability vs. time to failure ( 1h, 2h, 3h, 4h, 5h, 6h, 7h, 8h, 9h, 10h )

**Source Code:**

#include <stdio.h>

#include <math.h>

#define SIZE 10

int main() {

double t[SIZE], R1[SIZE], R2[SIZE], R3[SIZE];

double lambda = 0.3;

double k = 0.9;

double m = 0.5;

// Time to failure

for(int i = 0; i < SIZE; i++) {

t[i] = i + 1;

}

// Case I

for(int i = 0; i < SIZE; i++) {

R1[i] = exp(lambda \* t[i]);

}

// Case II

for(int i = 0; i < SIZE; i++) {

R2[i] = exp(-0.5 \* k \* pow(t[i], 2));

}

// Case III

for(int i = 0; i < SIZE; i++) {

R3[i] = exp(-pow(t[i]/k, m));

}

// Print the results

printf("Time to Failure (hours) | Case I: Constant Hazard | Case II: Linearly Increasing Hazard | Case III: Weibull Distribution\n");

for(int i = 0; i < SIZE; i++) {

printf("%-23.2f | %-23.2f | %-33.2f | %-30.2f\n", t[i], R1[i], R2[i], R3[i]);

}

return 0;

}

**Output:**

Time to Failure (hours) | Constant Hazard | Linearly Increasing Hazard | Weibull Distribution

1.00 | 1.35 | 0.64 | 0.35

2.00 | 1.82 | 0.17 | 0.23

3.00 | 2.46 | 0.02 | 0.16

4.00 | 3.32 | 0.00 | 0.12

5.00 | 4.48 | 0.00 | 0.09

6.00 | 6.05 | 0.00 | 0.08

7.00 | 8.17 | 0.00 | 0.06

8.00 | 11.02 | 0.00 | 0.05

9.00 | 14.88 | 0.00 | 0.04

10.00 | 20.09 | 0.00 | 0.04

**Plots:**

