SOFTWARE ENGINEERING LAB

II. B. Tech II Semester

N. Swaroopa
Asst. Professor

2024-2025

LEAD EXPERIMENTS

1.To Estimate the effort, development time, and cost of a software project using the COCOMO model and analyze its applicability in real –world scenarios ?

2. Estimation of effort using FP estimation for online examination system

1.To Estimate the effort, development time, and cost of a software project using the COCOMO model and analyze its applicability in real –world scenarios ?

COCOMO Model :

COCOMO (Constructive Cost Model) is an empirical software cost estimation model developed by Barry Boehm in the 1980s. It helps project managers estimate the **effort (person-months)**, **development time**, and **cost** based on the size of the software measured in **KLOC** (thousands of lines of code).

COCOMO Basic Model Formula

The basic COCOMO model is defined as:

- **Effort (E)** = $a \times (KLOC)^b$
- **Development Time (D)** = $c \times (Effort)^d$
  Where:
- a, b, c, d are constants depending on the project type:
  - **Organic**: Small teams with good experience and stable requirements
  - **Semi-Detached**: Intermediate
  - **Embedded**: Complex, tight constraints

Project Type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Example Estimation Using COCOMO

Assume a project of **50 KLOC**, categorized as *Semi-Detached*.

Step 1: Effort Estimation

- Effort = 3.0 × (50)^1.12 ≈ 3.0 × 86.4 = **259.2 person-months**

Step 2: Development Time Estimation

- Time = 2.5 × (259.2)^0.35 ≈ 2.5 × 8.9 = **22.3 months**

Step 3: Cost Estimation
If the cost per person-month = ₹100,000
Total Cost = 259.2 × ₹100,000 = **₹2.59 Crores**

Applicability in Real-World Scenarios

Advantages

- **Simple and easy to use** for early project estimates.
- Useful in **planning budgets**, staffing, and schedules.
- Gives a **quantitative basis** for software effort estimation.

Limitations

- Depends heavily on **accurate size estimation (KLOC)**, which is difficult early on.
- Not suitable for **modern agile or iterative models** where requirements evolve.
- May not handle **component reuse, GUI-based systems, or AI-based systems** well.

Real-World Use

- Used in **defense, aerospace, and legacy enterprise systems** where traditional waterfall models are still relevant.
- Helps in **benchmarking historical project data**.
- Often integrated in **estimation tools** in modified forms (e.g., COCOMO II for modern development).

2. To Estimation of effort using FP estimation for online examination system?

FP estimation for online examination system

1. Functional Components of the System

The Online Examination System includes features such as:
- Student and Admin login
- Registering students and exams
- Question paper management
- Conducting exams
- Viewing results and reports

These are mapped to the five Function Point (FP) categories:
EI - External Inputs
EO - External Outputs
EQ - External Inquiries
ILF - Internal Logical Files
EIF - External Interface Files

2. Function Point Count Table

Function Type	Function Description	Count	Complexity	Weight	Total FP
EI	Login, Register, Create Exam, Submit Answers	6	Average	4	24
EO	View Results, Reports	3	Average	5	15
EQ	Fetch Student Info, Exam List, Exam Details	3	Simple	3	9
ILF	Students DB, Questions DB, Results DB	3	Average	10	30
EIF	Authentication System, University DB	2	Simple	5	10

Unadjusted Function Points (UFP) = 24 + 15 + 9 + 30 + 10 = 88

3. Value Adjustment Factor (VAF)

Based on 14 General System Characteristics (GSCs), assume a medium influence score:
Total Degree of Influence (DI) = 35
VAF = 0.65 + (0.01 × DI) = 0.65 + 0.35 = 1.00

4. Adjusted Function Points (AFP)

AFP = UFP × VAF = 88 × 1.00 = 88

5. Effort and Cost Estimation

Productivity Rate = 10 hours per FP
Total Effort = 88 × 10 = 880 hours
Person-Months (assuming 160 hours/month) = 880 ÷ 160 = 5.5 PM
Cost per Person-Month = ₹1,00,000
Estimated Project Cost = 5.5 × ₹1,00,000 = ₹5.5 Lakhs

Database Management Systems Lab

II. B. Tech II Semester

M. Soujanya
Asst. Professor

2024-2025

LEAD EXPERIMENTS

1) Create views to insert, delete and update tuples.

2) Create trigger. the trigger as to execute on inserting a table and create a backup data.

1)Create views to insert, delete and update tuples

Aim: To implement operations on relations using PL/SQL Views.

Definition: A view is, in essence, a virtual table. It does not physically exist. Rather, it is created by a query joining one or more tables.

Create view employee dept-view as select t1.emp_name,t2.dept_name
From employee as t1 left join department as t2
Insert into employee(‘fname’,’lname’,’ssn’,dno)
values(‘robert’,’hitler’,9673553574,2);
Delete from employee
where fname=‘robert’;
Update employee
SET fname=’brown’,lname=’rob’,ssn=9680377384,Dno=2;

Result: Views in PL/SQL language are performed.

2). Create trigger. the trigger as to execute on inserting a table and create a backup data

Aim: To implement operations on relations using PL/SQL trigger

Definition: a **trigger** is a SQL procedure that initiates an action (i.e., fires an action) when an event (INSERT, DELETE or UPDATE) occurs. Since**triggers** are event-driven specialized procedures, they are stored in and managed by the **DBMS**.

```
CREATE TRIGGER PRINT_SALARY_CHANG IS
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (new.empno>0)
DECLARE
Sal-diff number
BEGIN
Sal-diff=new Sal-old sal;
Dbms_output.put(“old salary://:old sal”);
Dbms_output.put(“new salary://:new sal”);
END;
/
```

Result: A triggers in PL/SQL language are performed.

Computer Networks Lab

III. B. Tech II Semester

D. Siva Raja Kumar
Asst. Professor

2024-2025

LEAD EXPERIMENTS

1) Create views to insert, delete and update tuples.

2) Create trigger. the trigger as to execute on inserting a table and create a backup data.

1) Implement the Stop-and-Wait protocol for data transmission.

Aim: To implement the Stop-and-Wait protocol for reliable data transmission over an unreliable communication channel.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>  // for sleep()

// Simulated sender function
void sender(int frames[], int n) {
   int i = 0, ack;
   while (i < n) {
      printf("Sender: Sending frame %d\n", frames[i]);

      // Simulate frame transmission delay
      sleep(1);

      // Ask if the frame was received
      printf("Receiver: Was frame %d received? (1 for Yes, 0 for No): ", frames[i]);
      scanf("%d", &ack);

      if (ack == 1) {
         printf("Sender: Acknowledgment received for frame %d\n\n", frames[i]);
         i++;  // Go to next frame
      } else {
         printf("Sender: No ACK received for frame %d. Resending...\n\n", frames[i]);
      }

      // Simulate time delay before next transmission
      sleep(1);
   }
   printf("All frames sent successfully.\n");
}

int main() {
   int n;
```

```
    printf("Enter the number of frames to send: ");
    scanf("%d", &n);

    int frames[n];
    printf("Enter %d frame data (integers):\n", n);
    for (int i = 0; i < n; i++) {
        printf("Frame %d: ", i);
        scanf("%d", &frames[i]);
    }

    printf("\n--- Starting Stop-and-Wait Protocol ---\n\n");
    sender(frames, n);
    return 0;
}
```

Result:

```
Enter the number of frames to send: 3
Enter 3 frame data (integers):
Frame 0: 10
Frame 1: 20
Frame 2: 30

--- Starting Stop-and-Wait Protocol ---

Sender: Sending frame 10
Receiver: Was frame 10 received? (1 for Yes, 0 for No): 1
Sender: Acknowledgment received for frame 10

Sender: Sending frame 20
Receiver: Was frame 20 received? (1 for Yes, 0 for No): 0
Sender: No ACK received for frame 20. Resending...

Sender: Sending frame 20
Receiver: Was frame 20 received? (1 for Yes, 0 for No): 1
Sender: Acknowledgment received for frame 20
```

2) To study about the working of basic networking commands

Aim: To study the working of basic networking commands (ping, ip, netstat, and traceroute) by executing them from a C program.

Program:

```
#include <stdio.h>
#include <stdlib.h>

void show_menu() {
   printf("\n--- Basic Networking Commands ---\n");
   printf("1. Ping a host\n");
   printf("2. Show IP configuration\n");
   printf("3. Show network statistics\n");
   printf("4. Traceroute to a host\n");
   printf("5. Exit\n");
   printf("Enter your choice: ");
}

int main() {
   int choice;
   char host[100];

   while (1) {
      show_menu();
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            printf("Enter host to ping: ");
            scanf("%s", host);
            printf("Pinging %s...\n", host);
            system((char []){"ping -c 4 "});  // Need to concatenate string
            char cmd1[150];
            snprintf(cmd1, sizeof(cmd1), "ping -c 4 %s", host);
            system(cmd1);
            break;
```

```
case 2:
            printf("Showing IP configuration...\n");
            system("ip addr show");  // For modern systems. Use "ifconfig" on older systems.
            break;

        case 3:
            printf("Showing network statistics...\n");
            system("netstat -tuln");
            break;

        case 4:
            printf("Enter host for traceroute: ");
            scanf("%s", host);
            char cmd2[150];
            snprintf(cmd2, sizeof(cmd2), "traceroute %s", host);
            system(cmd2);
            break;

        case 5:
            printf("Exiting program.\n");
            exit(0);

        default:
            printf("Invalid choice. Try again.\n");
      }
   }

   return 0;
}
```

Result:

--- Basic Networking Commands ---

1. Ping a host
2. Show IP configuration
3. Show network statistics
4. Traceroute to a host
5. Exit

Enter your choice: 1
Enter host to ping: google.com
Pinging google.com...

PING google.com (142.250.77.14) 56(84) bytes of data.
64 bytes from del03s30-in-f14.1e100.net (142.250.77.14): icmp_seq=1 ttl=115 time=13.5 ms
64 bytes from del03s30-in-f14.1e100.net (142.250.77.14): icmp_seq=2 ttl=115 time=13.4 ms
64 bytes from del03s30-in-f14.1e100.net (142.250.77.14): icmp_seq=3 ttl=115 time=13.6 ms
64 bytes from del03s30-in-f14.1e100.net (142.250.77.14): icmp_seq=4 ttl=115 time=13.3 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 13.301/13.462/13.643/0.127 ms

Basic Networking Commands

1. Ping a host
2. Show IP configuration
3. Show network statistics
4. Traceroute to a host
5. Exit

Enter your choice: 2
Showing IP configuration...

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 ...
   inet 127.0.0.1/8 scope host lo
   ...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
   inet 192.168.0.104/24 brd 192.168.0.255 scope global dynamic eth0
   ...

Basic Networking Commands

1. Ping a host
2. Show IP configuration
3. Show network statistics
4. Traceroute to a host
5. Exit

Enter your choice: 3
Showing network statistics...

Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:68 0.0.0.0:*

Basic Networking Commands

1. Ping a host
2. Show IP configuration
3. Show network statistics
4. Traceroute to a host
5. Exit

Enter your choice: 4
Enter host for traceroute: google.com
traceroute to google.com (142.250.77.14), 30 hops max

1 192.168.0.1 (192.168.0.1) 1.123 ms 0.823 ms 0.733 ms
2 100.67.128.1 (100.67.128.1) 4.552 ms 4.481 ms 4.428 ms
3 ...
...
30

Basic Networking Commands

1. Ping a host
2. Show IP configuration
3. Show network statistics
4. Traceroute to a host
5. Exit

Enter your choice: 5
Exiting program.

Machine Learning Lab

III. B. Tech II Semester

Dr. A. Amarajyothi
Assoc. Professor

2024-2025

LEAD EXPERIMENTS

1) Recognize the handwritten digits using k-nearest neighbor algorithm.

2) Implement Gassian Naïve Bayes using sk learn.

1) Recognize the handwritten digits using k-nearest neighbor algorithm.

Aim: To create a model capable of accurately identifying handwritten digits using K-nearest neighbors algorithm.

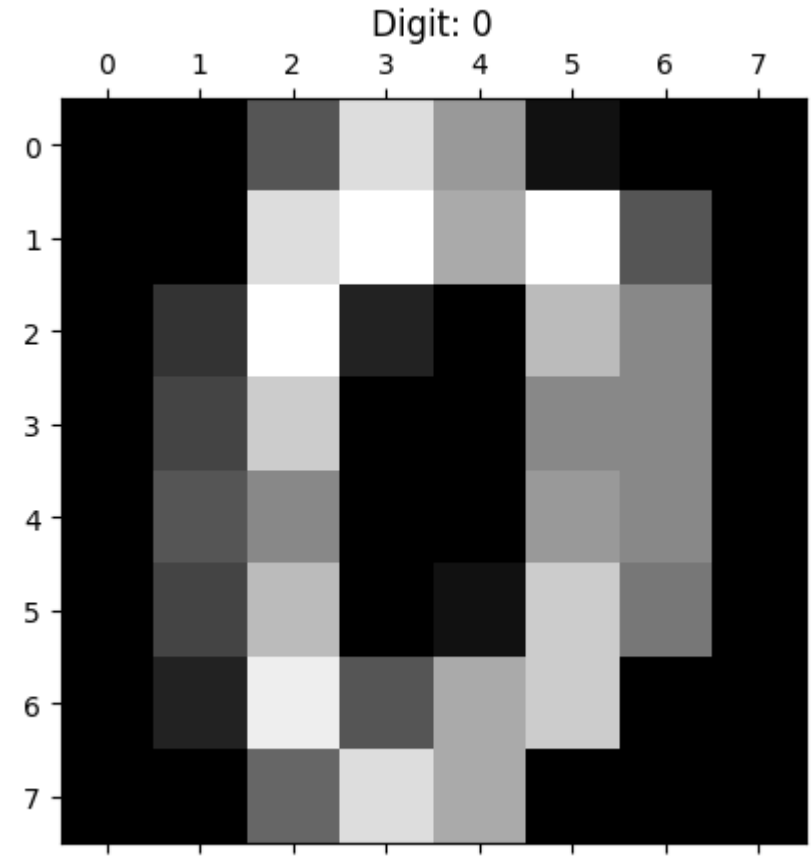
Definition: Handwritten digit recognition is the process of training a machine to identify and classify handwritten numerical digits (0-9) from images. For this, use K-nearest neighbors algorithm using **sklearn.datasets**.

Program:

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# 1. Load the dataset
digits = load_digits()
X, y = digits.data, digits.target  # 1797 samples, 64 features (8x8
images flattened)

# Optional: visualize some digits
plt.gray()
plt.matshow(digits.images[0])
plt.title(f"Digit: {digits.target[0]}")
plt.show()

# 2. Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 3. Initialize and train KNN model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# 4. Predict and evaluate
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred)
```

Result:

```
<Figure size 640x480 with 0 Axes>
```




```
Accuracy: 0.9833333333333333
```

2) Implement Gassian Naïve Bayes using sk learn.

Aim: To implement Gaussian Naive Bayes using **Scikit-learn.**

Definition: Gaussian Naive Bayes is a classification algorithm. It is a variant of the Naive Bayes algorithm specifically designed for continuous features, assuming that the likelihood of features given a class follows a Gaussian (normal) distribution. This algorithm is available within the **sklearn.naive_bayes** module in **Scikit-learn**. It assumes that features are conditionally independent given the class label, and that each feature's distribution within each class is Gaussian. It is well-suited for datasets with continuous numerical features.

Program:

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
digits = load_digits()
X, y = digits.data, digits.target  # X shape: (1797, 64), y shape:
(1797,)

# Split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train the model
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions
y_pred = gnb.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Result:

```
Accuracy: 0.8472222222222222
```

NATURAL LANGUAGE PROCESSING LAB

III. B. Tech II Semester

B. Revathi
Asst. Professor

2024-2025

LEAD EXPERIMENTS

1) Create a python program to prepare Multilingual transcription seeking inputs of words.
2) Write a python program to generate parse tree.

1) Create a python program to prepare Multilingual transcription seeking inputs of words.

Aim: The **aim** of the Multilingual Transcription program is:

To accept one or more words as input from the user and generate their translations (transcriptions) in multiple selected languages using an automated translation service.

Definition: The **Multilingual Transcription Program** is a **Python application** designed to take one or more words as input from the user and generate their translations in multiple languages. It uses the googletrans library (a Python interface for Google Translate) to perform real-time, automated translation of words into user-specified languages.

Program Code:

```
from googletrans import Translator

def get_language_codes():
    print("\nEnter the language codes (comma-separated) you want the transcription in.")
    print("Examples: en=English, es=Spanish, fr=French, de=German, hi=Hindi, zh-cn=Chinese Simplified")
    codes = input("Enter language codes (e.g., en,fr,es): ").strip()
    return [code.strip() for code in codes.split(',') if code.strip()]

def main():
    print("=== Multilingual Transcription Program ===\n")

    # Input words
    words_input = input("Enter the words (comma-separated): ")
    words = [word.strip() for word in words_input.split(',') if word.strip()]

    if not words:
        print("No words entered. Exiting.")
        return

    # Get language codes
    lang_codes = get_language_codes()
    if not lang_codes:
        print("No language codes entered. Exiting.")
        return

    translator = Translator()
```

```
    # Perform translation
    print("\n=== Transcriptions ===")
    for word in words:
        print(f"\nOriginal Word: {word}")
        for lang in lang_codes:
            try:
                translated = translator.translate(word, dest=lang)
                print(f"  [{lang}] {translated.text}")
            except Exception as e:
                print(f"  Error translating to {lang}: {e}")

if __name__ == "__main__":
    main()
```

 Example Usage:

```
csharp
Copy
Edit
Enter the words (comma-separated): hello, world
Enter language codes (e.g., en,fr,es): fr,es,de,hi
```

Result:

```
=== Transcriptions ===

Original Word: hello
  [fr] bonjour
  [es] hola
  [de] hallo
  [hi] नमस्ते

Original Word: world
  [fr] monde
  [es] mundo
  [de] welt
  [hi] दुनिया
```

2). Write a python program to generate parse tree.

Aim: To analyze the grammatical structure of a given English sentence and generate its syntactic parse tree using natural language processing techniques.

Definition: The **Parse Tree Generator** is a **Python program** that takes an English sentence as input and analyzes its grammatical structure by generating a **syntactic parse tree**. It uses **Natural Language Processing (NLP)** techniques such as **tokenization**, **part-of-speech (POS) tagging**, and **chunking** with defined grammar rules to identify phrases like noun phrases (NP), verb phrases (VP), and prepositional phrases (PP).

Program Code:

```
import nltk
from nltk import pos_tag, word_tokenize
from nltk.tree import Tree
from nltk.chunk import RegexpParser

def generate_parse_tree(sentence):
    # Tokenize and POS tag
    tokens = word_tokenize(sentence)
    tagged = pos_tag(tokens)

    # Define a simple grammar for noun and verb phrases
    grammar = r"""
      NP: {<DT>?<JJ>*<NN.*>}   # Noun phrase
      VP: {<VB.*><NP|PP>*}     # Verb phrase
      PP: {<IN><NP>}           # Prepositional phrase
    """

    # Create a parser with the defined grammar
    parser = RegexpParser(grammar)
    tree = parser.parse(tagged)

    return tree

def main():
    print("=== Parse Tree Generator ===")
    sentence = input("Enter a sentence: ")
    tree = generate_parse_tree(sentence)
```

```
    print("\nGenerated Parse Tree:")
    print(tree)

    # Display the tree graphically
    tree.pretty_print()

if __name__ == "__main__":
    main()
```

Result:

Input:

css
CopyEdit
Enter a sentence: The quick brown fox jumps over the lazy dog

Output:
A printed structure and a graphical tree (in console, via ASCII):

swift
CopyEdit

```
            (S
  (NP The/JJ quick/JJ brown/JJ fox/NN)
  (VP jumps/VBZ (PP over/IN (NP the/DT lazy/JJ dog/NN))))
```