PROCESS SELECTION SYSTEM

Project Report

SUPERVISOR: Ms. Gunjan Rani

SUBMITTED BY

Sambit Basu 19001570034

Sarvesh Sharma 19001570036

Prajwal Soni 19001570025

Course – B.Sc.(Hons.) Computer Science

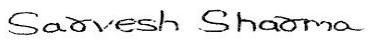
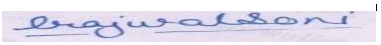
2021

Department of Computer Science

ACHARYA NARENDRA DEV COLLEGE

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher Ms. Gunjan Rani who gave us the golden opportunity to do this wonderful project on the topic Process Selection System, which also helped us in doing a lot of Research and we came to know about so many new things. Secondly we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

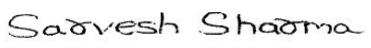
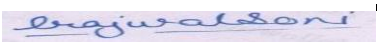

Sarvesh Sharma Prajwal Soni Sambit Basu

ACHARYA NARENDRA DEV COLLEGE

(University of Delhi)

CERTIFICATE

This is to certify that Sarvesh Sharma, Prajwal Soni and Sambit Basu of B.Sc.(Hons.) Computer Science of 4th semester have successfully completed their project work on building the software for Process Model Selection System under the guidance of Ms. Gunjan Rani.


Sarvesh Sharma Prajwal Soni Sambit Basu

Supervisor

Ms. Gunjan Rani

Contents

Chapter 1

PROBLEM STATEMENT

We see that many software developers face some problems and decision issues regarding selecting a good process model which takes up a lot of unnecessary time.

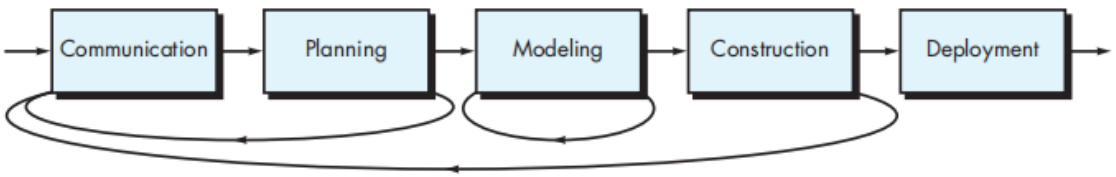
The goal of our software is to suggest a proper and appropriate process model by understanding their requirements for their project.

Chapter 2

PROCESS MODEL

.

We have decided to go ahead with the iterative process model for our software because we have done a bit of planning for our project and we have made a skeleton structure about our idea. We have also started coding around that idea. After sometime we will face some problems and we will cater to that and slowly like that our project will take shape. And finally, we will be able to deploy our project.



Iterative Process Model

Chapter 3

Requirement Analysis & Modeling

3.1 DFD

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

3.1.1 Context Level DFD

It is also known as a 0-level diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

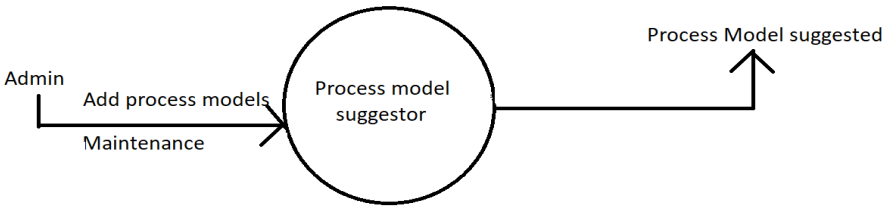


Figure 3.1: Context Level DFD

3.2.2 Level 1 DFD

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub-processes.

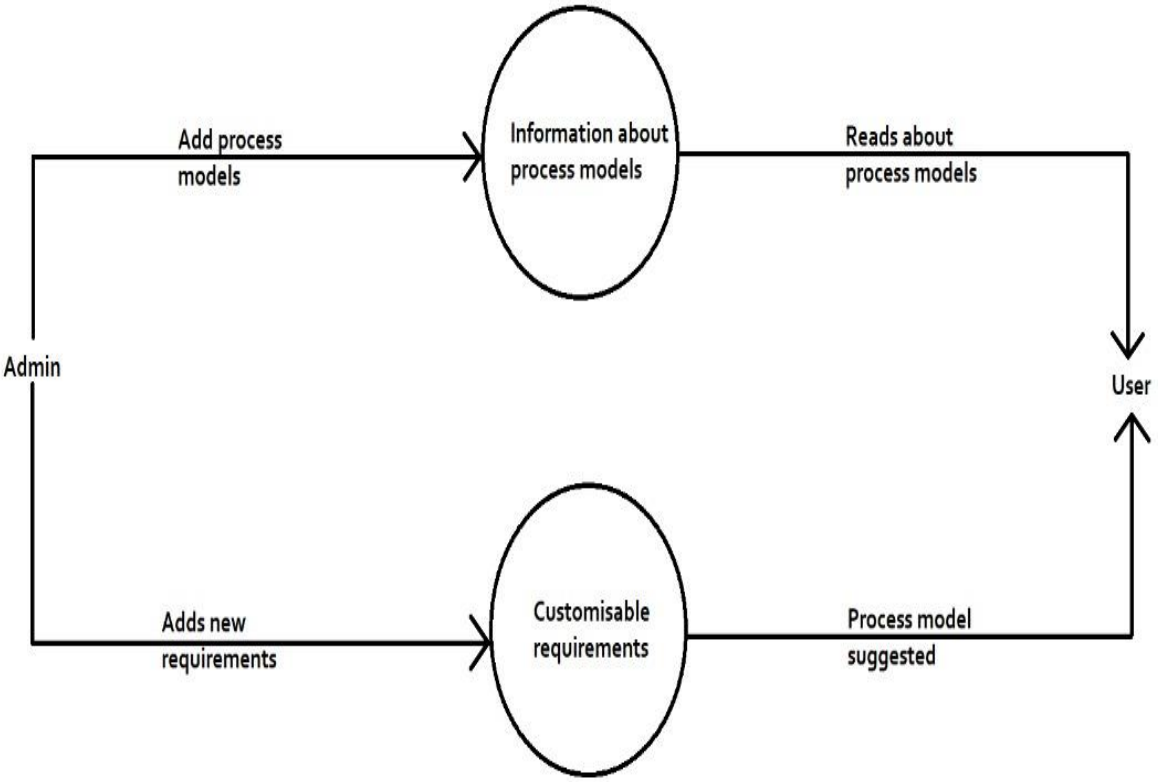


Figure 3.2: Level 1 DFD

3.2 Data Dictionary

A data dictionary is a file or a set of files that includes a database's metadata. The data dictionary holds records about other objects in the database, such as data ownership, data relationships to other objects, and other data. The data dictionary is an essential component of any relational database.

3.3. Use Case Diagrams

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity.

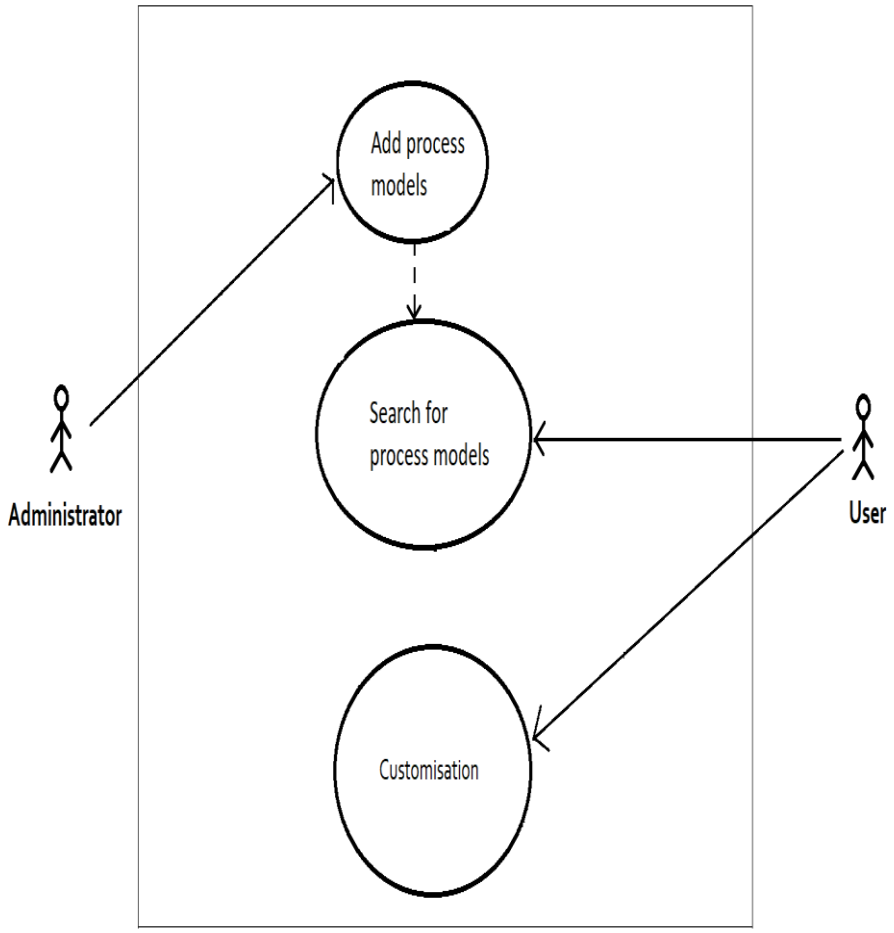


Figure 3.3: Use Case Diagram

3.4 Sequence Diagrams

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

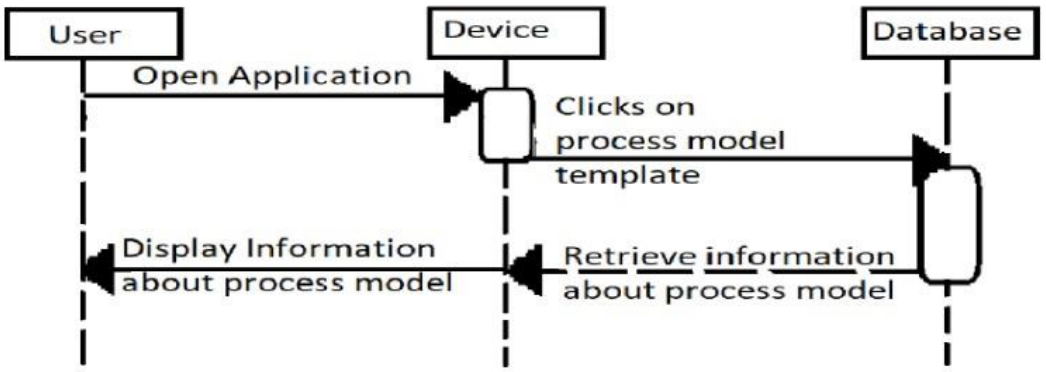


Figure 3.3: Sequence Diagram

Chapter 4

SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification (SRS) Format as name suggests, is complete specification and description of requirements of software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-requirements depending upon type of requirement. The interaction between different customers and contractor is done because its necessary to fully understand needs of customers.

4.1 Overall Description

The purpose of the Software Requirements Specification document system under development namely Process Selection Model. The intended user of this document includes the developers such as requirement team, requirement analyst, design team and supervisor of developing software.

4.1.1 Product Functions

Our software, Process Selection System helps Software developers and designers to decide upon an appropriate process model according to their requirements for their upcoming software.

4.1.2 User Characteristics

This is the most interesting part of our software. The users of our software will be software designers themselves. They will benefit greatly from our software as the hectic task of deciding a process model for their upcoming software is greatly reduced. Our users, that is, software designers will then create a software based on our suggested process for their client.

Our users, that is, software developers should have the basic knowledge about software process models and can plan how to design a software.

4.1.3 General Constraints

The user should have an android compatible device. The version of the android device should be Kitkat or later.

Blind users can’t use this application.

4.1.4 Assumptions and Dependencies

We are assuming that our users will have proper knowledge of how to develop a software. He/she should also have knowledge about process models.

We are also assuming that the user will use an android compatible device for using our software, the device having android version Kitkat or later.

4.2 External Interface Requirements

The member has to register using a form provided on the application. The user can input data with the help of the virtual keyboard or tapping on the screen of the smartphone wherever necessary. The package provides pull down menus from which the user can select and links and icons to navigate among the process models.

4.2.1 User Interface

Our software requires no such external UI.

4.2.2 Hardware Interface

- At least 256 MB RAM
- At least 10 MB freed hard disk space

4.2.3 Software Interface

Our software requires no such external

4.3 Functional requirements

4.3.1 FR1 Login Requirement

The user should enter a username and a password to login into our application. If the user is using our app for the first time, then he/she should register himself/herself on our application.

4.4 Performance Requirements

4.4.1 Performance Requirement 1

The application is portable.

4.5 Design Constraints

1. Software Language: All coding will be done in Java(Android).
2. The software can support a maximum of 500 members.

Chapter 5

Estimations

5.1 Function Points

A Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size. They are widely accepted as an industry standard for functional sizing.

The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

5.2 Efforts

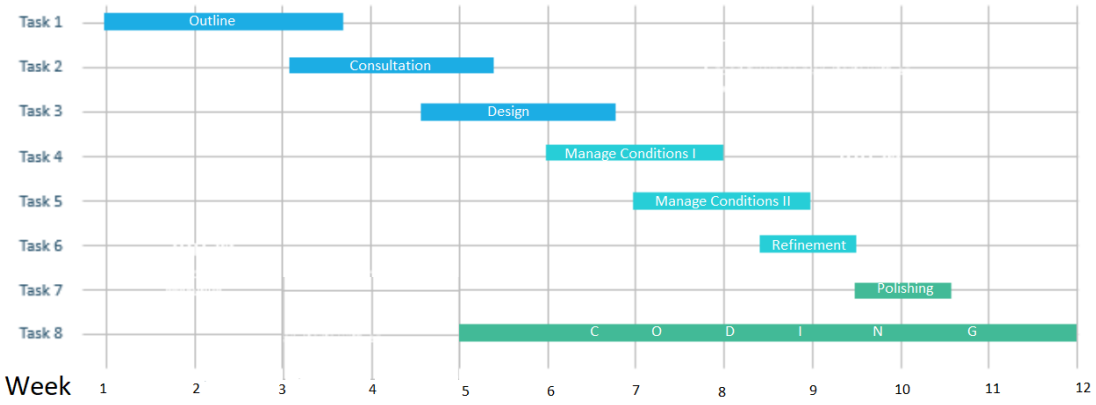
In software engineering effort is used to denote measure of use of workforce and is defined as total time that takes members of a development team to perform a given task. It is usually expressed in units such as man-day, man-month, man-year.

Chapter 6

Scheduling

Project scheduling in software engineering is a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what time frame. A project schedule is a document collecting all the work needed to deliver the project on time.

Gantt Chart of the Project Schedule



Task 1. Outline

Task 2. Consultation

Task 3. Design

Task 4. Manage conditions I

Task 5. Manage conditions II

Task 6. Refinement

Task 7. Polishing

Task 8. Coding

Task 9. Testing

Chapter 7

Risk Management

A software project can be concerned with a large variety of risks. In order to be adept to systematically identify the significant risks which might affect a software project, it is essential to classify risks into different classes. The project manager can then check which risks from each class are relevant to the project.

The risks involved are :-

1. The user does not find the documentation about the process model satisfactory.

2. The user does not find the provided customization criteria enough or valid for their requirements.

3. The app might crash if the user does not have the specified hardware or software requirements.

Chapter 8

DESIGN

Software design is the process of envisioning and defining software solutions to one or more sets of problems. One of the main components of software design is the software requirements analysis (SRA). SRA is a part of the software development process that lists specifications used in software engineering.

8.1 Structured Chart

Structure Chart represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated. Modules at top level called modules at low level. Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.

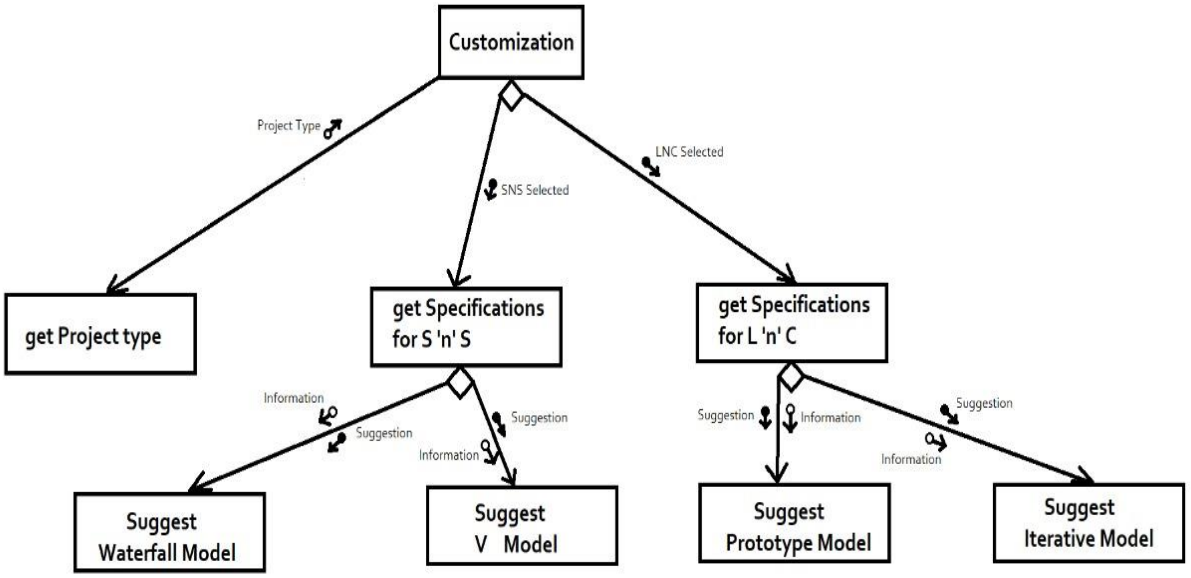


Figure 8.1: Structured Chart

8.2 Pseudo Code

In computer science, pseudocode is a plain language description of the steps in an algorithm or another system. Pseudocode often uses structural conventions of a normal programming language, but is intended for human reading rather than machine reading.

Chapter 9

CODING

9.1 Code Snippet 1 (Main Activity)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    ImageButton  iv,iv2;
    TextView tv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        iv=(ImageButton) findViewById(R.id.imageButton);
        tv=(TextView) findViewById(R.id.textView);
        iv2=(ImageButton) findViewById(R.id.IB2);
    }
```

```
    public void kou(View v)
    {
        Intent intent= new Intent(this, Processmodels.class);
        startActivity(intent);
    }

    public void req(View v)
    {
        Intent intent= new Intent(this, Requirements.class);
        startActivity(intent);

    }
}
```

9.2 Code Snippet 2 (Process Models Activity)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class Processmodels extends AppCompatActivity {

    ImageButton ib2,ib3,ib4,ib5;
    TextView v;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_processmodels);

        ib2=(ImageButton) findViewById(R.id.ib2);
        ib3=(ImageButton) findViewById(R.id.ib3);
        ib4=(ImageButton) findViewById(R.id.ib4);
        ib5=(ImageButton) findViewById(R.id.ib5);
        v=findViewById(R.id.textView5);
    }

    public void mai(View view)
    {
        Intent intent=new Intent(this,Veemodel.class);
        startActivity(intent);
    }

    public void prot(View b)
    {
        Intent intne=new Intent(this,Prototypemodel.class);
        startActivity(intne);
    }
```

```
    public void wer(View r)
    {
        Intent intent=new Intent(this,Waterfallmodel.class);
        startActivity(intent);
    }

    public void poi(View v)
    {
        Intent intent=new Intent(this,Iterativemodel.class);
        startActivity(intent);
    }
}
```

9.3 Code Snippet 3 (Customization Activity)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

public class Requirements extends AppCompatActivity {

   RadioGroup rg;
   RadioButton rb1,rb2;
   TextView tv1,tv2;
   Button b;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_requirements);

      rg=(RadioGroup) findViewById(R.id.radioGroup);
      tv1=(TextView) findViewById(R.id.textView2);
      tv2=(TextView) findViewById(R.id.textView3);
      rb1=(RadioButton) findViewById(R.id.radbut1);
      rb2=(RadioButton) findViewById(R.id.radbut2);
      b=findViewById(R.id.button2);


      b.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {

                  if (rb1.isChecked()) {
                      Intent in = new Intent(Requirements.this, sns.class);
                      startActivity(in);
```

```
                }
                else if (rb2.isChecked())
                {
                   Intent in2= new Intent(Requirements.this, lnc.class);
                   startActivity(in2);
                }
                else
                   Toast.makeText(getApplicationContext(),"Please select one
option",Toast.LENGTH_SHORT).show();

            }


        });
    }
}
```

9.4 Code Snippet 4 (Waterfall Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;

public class Waterfallmodel extends AppCompatActivity {

    Spinner spinner;
    TextView textView;
    ImageView df;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_waterfallmodel);

        spinner=findViewById(R.id.spinner);
```

```
        textView=findViewById(R.id.text);
        df=findViewById(R.id.was);


        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                if(spinner.getSelectedItemId()==0)
                {
                    textView.setText(R.string.waterfall_definition);
                }
                else if (spinner.getSelectedItemId()==1)
                {
                    textView.setText(R.string.waterfall_advantages);
                }
                else if(spinner.getSelectedItemId()==2)
                {
                    textView.setText(R.string.waterfall_disadvantages);
                }
            }


            @Override
            public void onNothingSelected(AdapterView<?> parent) {


            }
        });
    }
}
```

9.5 Code Snippet 5 (V Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Spinner;
import android.widget.TextView;

public class Veemodel extends AppCompatActivity {

    Spinner spinner;
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_veemodel);

        spinner=findViewById(R.id.spinner);
        textView=findViewById(R.id.text);
```

```
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                if(spinner.getSelectedItemId()==0)
                    textView.setText(R.string.vmodel_definition);
                else if(spinner.getSelectedItemId()==1)
                    textView.setText(R.string.vmodel_advantages);
                else if(spinner.getSelectedItemId()==2)
                    textView.setText(R.string.vmodel_disadvantages);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

    }
}
```

9.6 Code Snippet 6 (Prototype Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Spinner;
import android.widget.TextView;

public class Prototypemodel extends AppCompatActivity {

    Spinner spinner;
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prototypemodel);

        spinner=findViewById(R.id.spinner);
        textView=findViewById(R.id.text);
```

```
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                if(spinner.getSelectedItemId()==0)
                    textView.setText(R.string.protmodel_definition);
                else if(spinner.getSelectedItemId()==1)
                    textView.setText(R.string.protmodel_advantages);
                else if(spinner.getSelectedItemId()==2)
                    textView.setText(R.string.protmodel_disadvantages);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });
    }
}
```

9.7 Code Snippet 7 (Iterative Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Spinner;
import android.widget.TextView;

public class Iterativemodel extends AppCompatActivity {

    Spinner spinner;
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_iterativemodel);

        spinner=findViewById(R.id.spinner);
        textView=findViewById(R.id.text);

        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {
            if(spinner.getSelectedItemId()==0)
                textView.setText(R.string.iterative_definition);
            else if(spinner.getSelectedItemId()==1)
                textView.setText(R.string.iterative_advantages);
            else if (spinner.getSelectedItemId()==2)
                textView.setText(R.string.iterative_disadvantages);
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });
}
}
```

9.8 Code Snippet 8 (Short & Simple)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;

public class sns extends AppCompatActivity {

    TextView gg1;
    CheckBox kk1,kk2,kk3,kk4,kk5;
    Button b;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sns);
```

```
    gg1=findViewById(R.id.qw2);
    kk1=findViewById(R.id.er);
    kk2=findViewById(R.id.er2);
    kk3=findViewById(R.id.er3);
    kk4=findViewById(R.id.er4);
    kk5=findViewById(R.id.er5);
    b=findViewById(R.id.bu);


  }
  public void Any(View view)
  {
        if (kk2.isChecked() || kk1.isChecked()) //Process measurer OR
Debugging and Testing
        {
          Intent nt= new Intent(this, Vmodel.class);
          startActivity(nt);
        }
        else if (kk3.isChecked() || kk4.isChecked() || kk5.isChecked()) //Easy
to manage OR Static technology OR Completion of phases one by one
        {
          Intent intent= new Intent(this, Wmodel.class);
          startActivity(intent);
        }
        else
          Toast.makeText(getApplicationContext(),"Please select at least 1
option",Toast.LENGTH_SHORT).show();
  }
}
```

9.9 Code Snippet 9 (Long & Complex)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;

public class lnc extends AppCompatActivity {
    TextView tv1;
    CheckBox cb1,cb2,cb3,cb4,cb5,cb6;
    Button b;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lnc);

        tv1 = findViewById(R.id.textView2);
        cb1 = findViewById(R.id.checkBox);
        cb2 = findViewById(R.id.checkBox2);
```

```
        cb3 = findViewById(R.id.checkBox3);
        cb4 = findViewById(R.id.checkBox4);
        cb5 = findViewById(R.id.checkBox5);
        cb6 = findViewById(R.id.checkBox6);
        b = findViewById(R.id.button);


    }
    public void mnb(View v)
    {
            if (cb1.isChecked() || cb2.isChecked() || cb4.isChecked() || cb5.isChecked())
//Debugging & Testing OR Process measurer OR Risk analysis OR Clear
communication
            {
                Intent nt = new Intent(this, Iteramodel.class);
                startActivity(nt);
            } else if (cb3.isChecked() || cb6.isChecked()) //Dynamic technology OR OOP
related
            {
                Intent nt = new Intent(this, Protomodel.class);
                startActivity(nt);
            }
            else
                Toast.makeText(getApplicationContext(),"Please select at least 1
option",Toast.LENGTH_SHORT).show();
    }
}
```

9.10 Code Snippet 10 (Suggesting Waterfall Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class Wmodel extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_wmodel);

        TextView tv=findViewById(R.id.t);
        ImageButton im=findViewById(R.id.ib2);

        String x="We suggest <b>Waterfall model</b> according to your
requirements";
        tv.setText(Html.fromHtml(x));

    }
```

```
    public void dg(View v)
    {
        Intent ser=new Intent(this,Waterfallmodel.class);
        startActivity(ser);
    }

}
```

9.11 Code Snippet 11 (Suggesting V Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class Vmodel extends AppCompatActivity {

    TextView tv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_vmodel);

        tv=findViewById(R.id.textView4);
        ImageButton imageButton=findViewById(R.id.ib5);

        String x="We suggest <b>V-model</b> according to your requirements";
        tv.setText(Html.fromHtml(x));
```

```
    }

    public void dfg(View v)
    {
        Intent intent=new Intent(this,Veemodel.class);
        startActivity(intent);
    }
}
```

9.12 Code Snippet 12 (Suggesting Prototype Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class Protomodel extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_protomodel);

        ImageButton im=findViewById(R.id.ib4);
        TextView t=findViewById(R.id.tau);

        String x="We suggest <b>Prototype model</b> according to your
requirements";
        t.setText(Html.fromHtml(x));
    }
```

```
    public void fun(View b)
    {
        Intent inty=new Intent(this,Prototypemodel.class);
        startActivity(inty);
    }
}
```

9.13 Code Snippet 13 (Suggesting Iterative Model)

```
package com.example.processselectionsystem;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class Iteramodel extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_iteramodel);

        ImageButton im=findViewById(R.id.ib3);
        TextView t=findViewById(R.id.tex);

        String x="We suggest <b>Iterative model</b> according to your
requirements";
        t.setText(Html.fromHtml(x));
    }
```

```
    public void nuf(View r)
    {
        Intent uyt=new Intent(this,Iterativemodel.class);
        startActivity(uyt);
    }
}
```

Chapter 10

TESTING

White Box Testing

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

Black Box Testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

10.1 Test Case Design

Test Case	Level of program	Debugging & Testing	Process Measurer	Complete Phases one by one	Static Technology	Easy To Manage	Expected Output
1.	Short & Simple	Selected	Selected	Selected	Selected	Selected	V-Model
2.	Short & Simple	Selected	Selected	-	Selected	-	V-Model

3.	Short & Simple	Selected	-	Selected	-	-	V-Model
4.	Short & Simple	Selected	-	-	-	Selected	V-Model
5.	Short & Simple	Selected	Selected	-	-	Selected	V-Model
6.	Short & Simple	-	-	Selected	-	-	Waterfall Model
7.	Short & Simple	-	-	-	Selected	Selected	Waterfall Model
8.	Short & Simple	-	-	Selected	Selected		Waterfall Model
9.	Short & Simple	-	-	-	-	Selected	Waterfall Model
10.	Short & Simple	-	-	Selected	Selected	Selected	Waterfall Model

Test Case	Level Of your program	Debugging & Testing	Risk Analysis	Dynamic Technology	Clear Communication	OOP Related	Process Measurer	Expected Output
1.	Long & Complex	Selected	Selected	Selected	Selected	Selected	Selected	Iterative

2.	Long & Complex	Selected	Selected	-	-	-	-	Iterative
3.	Long & Complex	Selected	-	Selected	Selected	-	-	Iterative
4.	Long & Complex	Selected	Selected		Selected	-	-	Iterative
5.	Long & Complex	-	Selected	Selected	Selected	-	-	Iterative
6.	Long & Complex	Selected	-	Selected	-	-	-	Iterative
7.	Long & Complex	-	Selected	Selected	Selected	-	-	Iterative
8.	Long & Complex	Selected	-	Selected	-	Selected	-	Iterative
9.	Long & Complex	-	-	-	-	Selected	Selected	Iterative
10.	Long & Complex	-	-	-	-	Selected	-	Prototype Model

10.2 Flow Graph

A Control Flow Graph (CFG) is the graphical representation of control flow or computation during the execution of programs or applications. Control flow graph is process oriented. It shows all the paths that can be traversed during a program execution. It is a directed graph. Edges in CFG portray control flow paths and the nodes in CFG portray basic blocks.

10.3 Basis Path Set

Basis Path Testing is a white-box testing technique based on the control structure of a program or a module. Using this structure, a control flow graph is prepared and the various possible paths present in the graph are executed as a part of testing. Therefore, by definition,

Basis path testing is a technique of selecting the paths in the control flow graph, that provide a basis set of execution paths through the program or module.

Since this testing is based on the control structure of the program, it requires complete knowledge of the program's structure. To design test cases using this technique, four steps are followed :

1. Construct the Control Flow Graph
2. Compute the Cyclomatic Complexity of the Graph
3. Identify the Independent Paths
4. Design Test cases from Independent Paths

10.4 Cyclomatic Complexity

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

Chapter 11

References

Geeksforgeeks.com

Javatpoint.com

Tuttorialspoint.com

Stackoverflow.com

Bibliography

Books

Software Engineering by KK Aggarwal and Yogesh

Websites

GeeksforGeeks - https://www.geeksforgeeks.org