**Project Report (Paper-410)**

B. Sc (H) Computer Science (IV Semester)

Area: Software Engineering

**WiFi Penetration Testing**



**Ram Lal Anand College**,

Benito Juarez Rd, South Campus, New Delhi

University of Delhi

**Submitted By:** **Under Supervision**

Anuj Saraswat Ms. Shikha Verma

Saurav Shah

Md Tauseef Alam

T. kuki

Gaurav Narang

**CERTIFICATE**

This is to certify that the project entitled, “W” has been done by:-

**Anuj Saraswat,** **Saurav Shah,** **Md Tauseef Alam,** **T. Kuki** and **Gaurav Narang** of Bachelor of Computer Science (Hons.) during semester IV from **Ram Lal Anand College, University of Delhi** under the supervision of **Ms. Shikha Verma**.

**Ms. Shikha Verma**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

The Project was jointly undertaken by **Anuj Saraswat,** **Saurav Shah,** **Md Tauseef Alam,** **T. Kuki** and **Gaurav Narang** as their 4th Semester Software Engineering Project, under the able guidance and supervision of **Ms. Shikha Verma**. Our primary thanks goes to her, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It’s our privilege to acknowledge our deepest sense of gratitude to her for her inspiration which has helped us immensely. We are extremely grateful to her for her unstilted support and encouragement in the preparation of this project.



**Apr 2016**

**Mar 2016**

**Feb 2016**

**Jan 2016**

*Fig:-. Timeline chart*

**PROJECT MANAGEMENT**

Project management involves the planning, monitoring, and control of the people, process, and events that occur as software evolves from a preliminary concept to an operational implementation. Project managers plan, monitor, and control the work of a team of software engineers. Effective software project management focuses on the four P‟s: people, product, process, and project.

**1. Function Points**

Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since, “functionality” cannot be measured directly; it must be derived indirectly using other direct measures. Function points are computed by completing the table . Five information domain characteristics are determined and counts are provided in the appropriate table location. Information domain values are defined in the following manner:

**Number of user inputs**: Each user input that provides distinct application oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.

**Number of user outputs**: Each user output that provides application oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

**Number of user inquiries**: An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.

**Number of files**: Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.

**Number of external interfaces**: All machine readable interfaces (e.g., data files on storage media) that are used to transmit information to another system are counted.

**Table:-** **Weighting factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Measurement factors** | **Count** | **Simple** | **Average** | **Complex** |  |
| Number of user inputs | 2 | 3 | 4 | 6 | 4\*2 = 8 |
| Number of user output | 2 | 4 | 5 | 7 | 5\*2 = 10 |
| Number of user inquiries | 1 | 3 | 4 | 6 | 1\*3 = 3 |
| Number of internal logical files | 1 | 7 | 10 | 15 | 1\*10 = 10 |
| Number of external interface files | 0 | 5 | 7 | 10 | 0\*7 = 0 |
| Count total | | | | | 31 |

The *Fi* (*i* = 1 to 14) are "Complexity Adjustment Values" based on responses to the

following questions:

|  |
| --- |
| 1 |
| 3 |
| 4 |
| 2 |
| 3 |
| 0 |
| 0 |
| 0 |
| 1 |
| 5 |
| 2 |
| 3 |
| 4 |
| 1 |

**1.** Does the system require reliable backup and recovery?

**2.** Are data communications required?

**3.** Are there distributed processing functions?

**4.** Is performance critical?

**5.** Will the system run in an existing, heavily utilized operational environment?

**6.** Does the system require on-line data entry?

**7.** Does the on-line data entry require the input transaction to be built over multiple?

**8.** Are the master files updated on-line?

**9.** Are the inputs, outputs, files, or inquiries complex?

**10.** Is the internal processing complex?

**11.** Is the code designed to be reusable?

**12.** Are conversion and installation included in the design?

**13.** Is the system designed for multiple installations in different organizations?

**14.** Is the application designed to facilitate change and ease of use by the user?

Once these data have been collected, a complexity value is associated with each count. Organizations that use function point methods develop criteria for determining whether a particular entry is simple, average, or complex. To compute function points (FP), the following relationship is used:

**FP** = count total \*[0.65 + 0.01 \*Σ (Fi)]

= 31 \* ( 0.65 + 0.01 \* 29 )

= 31 \* 0.94

= 29 (approx.)

Where count total is the sum of all Fi entries obtained from Figure.

**2.ESTIMATING EFFORTS**

Barry Boehm introduced a hierarchy of software estimation models bearing the name COCOMO, for COnstructive COst MOdel. The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry. The COCOMO II application composition model uses object points.

The object point is an indirect software measure that is computed using counts of the no. of screens (user interface), reports and components likely to be required to build the application. Each object instance is classified into one of three complexity levels using criteria suggested by Boehm.

Once complexity is determined, the number of screens, reports, and components are weighted. The object point count is then determined by multiplying the original number of object instances by the weighting factor in and summing to obtain a total object point count. When component-based development or general software reuse is to be applied, the percent of reuse (%reuse) is estimated and the object point count is adjusted:

***NOP = (object points) x [(100 -%reuse)/100],***

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, “productivity rate” must be derived.

***PROD = NOP/person-month***

Table 3.1 presents the productivity rate for different levels of developer experience and development environment maturity. Once the productivity rate has been determined, an estimate of project effort can be derived as

***Estimated effort = NOP/PROD***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object type** | **No. of objects** | **Complexity Weight** | | | **Count** |
| Simple | Medium | Difficult |
| Screen | 4 | 1(2) | 2(2) | 3(1) | 7 |
| Report | 2 | 2(2) | 5(1) | 8(0) | 9 |
| 3GL component | 2 |  |  | 10(2) | 20 |
| Object points sum | | | | | 36 |

Table 3.2 Estimating object points

Data used in estimating effort are:

(1) Object points is 36(taken from table 3.2)

(2) Estimated reuse is 36%

(3) Prod is 13 (average value taken)

**NOP = Object points \*[(100-reuse%)/100]**

= 36\*[(100-35)/100]

= 24

**ESTIMATED EFFORT = NOP/PROD**

= 24/13

= 2 person-months

Hence estimated effort of the project is 2 person-months.

**3. ESTIMATING SCHEDULE**

Putnam and Myers suggest a set of equations derived from the software equation. Minimum development time is defined as

in months for > 6 months

Since project’s time period is less than 6 months, the above equation cannot be applied.

An estimation model of the form:

*E* = (equation 1)

where *E* = effort in person-months or person-years

*t* = project duration in months or years

*B* = “special skills factor”

*P* = “productivity parameter”

Calculating development time for project, using equation 1 and effort calculated in section 3.2

2 =

t4 =0.06

Hence estimated schedule of the project is 0.06 months.