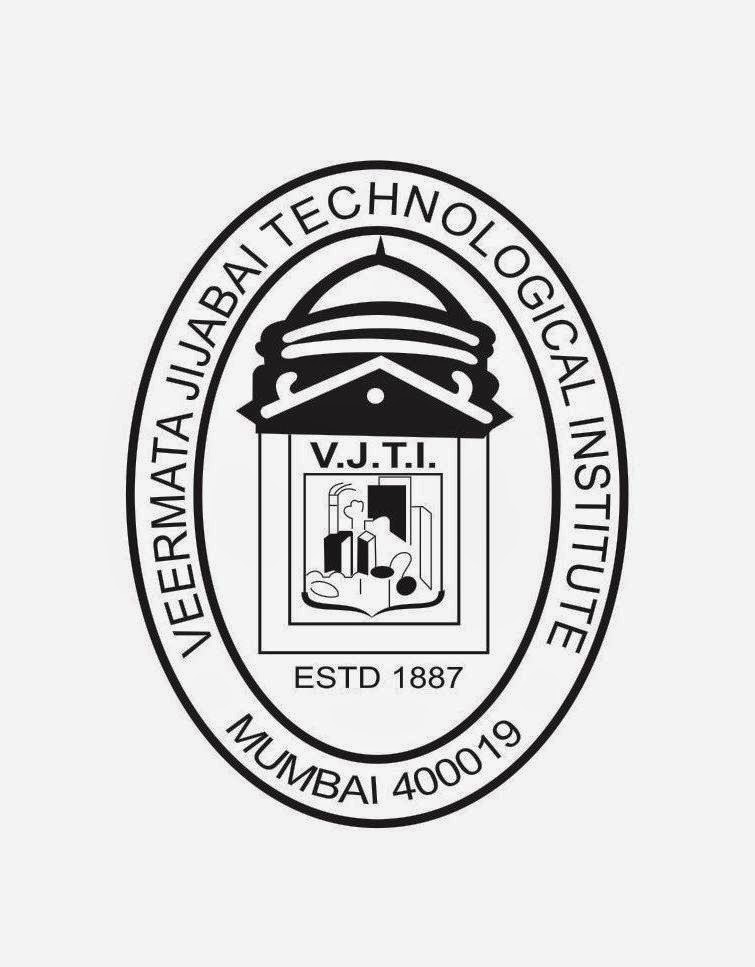
**NETWORK USABILITY MONITOR**



Under the guidance of

**Prof. P. M. Chawan**

**CERTIFICATE**

**STATEMENT BY CANDIDATE**

**TABLE OF CONTENTS**

1. Problem statement
   1. Introduction
   2. Background
      1. Previous attempt
      2. Current scenario
   3. Elaboration
2. Scope
   1. Features
   2. Constraints
   3. Performance requirements
      1. Response time
      2. Workload
      3. Scalability
      4. Platform
3. Resources
   1. Human resources
   2. Reusable software resources
4. Project Estimation
   1. LOC Based Estimation
   2. Function Point Based Estimation
   3. COCOMO Model Based Estimation
5. Risk Analysis and Management
   1. Risk Identification
   2. Risk Estimation
   3. Risk Mitigation

**PROBLEM STATEMENT**

* 1. **Introduction:**

To design a network monitoring system

* 1. **Background:**
     1. **Previous attempt:**

It would be indecorous to state that the previous attempts made to design an all in one network monitor were fruitful, as they have found little practical usage today. They are mostly used by service providers, companies owning huge networks, etc. and find no use in local and small networks.

* + 1. **Current scenario:**

The existing network monitoring softwares are too complex and difficult to use for people from a non-networking or a non-computer background. These softwares use many technical terms which are jargon for computer and networking scientists. Several ways to measure network performance have been developed so far; however, performance management is still difficult because of lack of effective tools to evaluate network system usability. Also, the networking commands used in a Windows or Linux terminal have numerous uses and their output is static, i.e. we need to run them continuously for a time dependent analysis. We can state the inadequacies of this system as follows: -

1. Difficulty in understanding technical jargon for people from a non-networking background
2. Static outputs from Windows Command Prompt or Linux Terminal which compel users to continuously run them in order to view the change in statistics with respect to time
3. Lack of effective tools to evaluate network system usability
4. Problems are faced by the network users and administrators to determine as to what exactly is the fault and the way it should be dealt with; as an example, if a network gives warns us about limited access, the DNS doesn’t reply the host with the IP address and hence, the communication breaks, however, the actual reason behind the problem is often ignored.
   1. **Elaboration:**

Our aim is to design a network monitoring tool for any client system which can be useful in any environment by a network administrator; be it home, office, an educational institute or a computer laboratory. The goal is to develop a new performance evaluation tool for network system usability. In our approach, we measure network system usability and performance through the mechanisms installed in the clients. The purpose of the system is to make clear the behaviour of the client applications which allows us to measure system performance for client usability and perform analytics on the data, thus generating useful outputs for the user. Some of the aspects of our implementation are: -

1. Obtaining crucial data for the network and its parameters such as the access delay, processing time, etc.
2. The tool would run on a Windows or a Linux operating system on any simple PC.
3. The software would be standalone and freely distributable, which would mean that it could be shared on other devices without any chaotic methods.
4. The data would be represented graphically which would provide an attractive GUI for users.

**PROJECT SCOPE**

**2.1 Features:**

This particular project is expected to have the following features:

* An easy to use graphical user interface (GUI) for system administrators. The system administrator won’t be required to type in commands in the terminal directly. Our easy-to-use GUI will provide an excellent interface using which a person need only click on particular buttons and he/she will receive all the necessary information in an easy to interpret manner.
* Real time graph based monitoring of network resources to help them identify any bottlenecks in the network. This will be accomplished by continuous packet transfer in the background. Based on the received data, graphs will plot to provide the necessary analytics of the network.
* Efficient tools to evaluate network system usability. Network commands to be executed indirectly in the terminal through our application, providing a layer of abstraction from background details.
* Easy detection of network problems
* Notifications to be sent to the system administrators through emails or text messages about any network issues
* Effective solutions to solve the identified problems

**2.2 Constraints:** Although the project will be advancement on the previous attempts, there are some limitations of the application. They are mentioned below:

* Notifications about network problems cannot be sent via text messages if Telnet protocol is not functional or is not installed.
* Performance enhancement can be achieved only on certain network configurations.
* Solutions to problems might require restarting the entire system, which is real time problem solutions are difficult to achieve.
* Only the total data flow in the network can be monitored and not the data flow through each node.
* Bandwidth of the network will affect the performance of the application.
* Network congestion might result in delayed monitoring and analysis of the usability features.

**2.3 Performance requirements:**

Following are the four categories of performance measures of our application:

* + 1. **Response Time**: It is the amount of time required to load the application completely. This involves continuous packet transfer in the background to enable real time monitoring and graph based analysis. As such, a response time of about 2-3 seconds can be estimated under minimal requirement conditions. Variations in the response time can be caused due to heavy server payload in times of peak usage. A variation of 1-2 seconds is expected under such conditions. If there is no network connection, then the application will not start.
    2. **Workload**:

|  |  |  |
| --- | --- | --- |
| Scenario | Daily total | Time (in seconds) |
| Authentication | 200 | 5 |
| Graph analysis | 170 | 5,10,15(based on options) |
| Delay calculation | 150 | Varying |
| Notification system | 10 | 2 |
| Performance management | 120 | 10 |

* + 1. **Scalability**:

This application is meant to be used for large networks as well as small private networks. Ideal usage is in a college local area network (LAN) or other such institutions. As distance between nodes and server will affect the performance of the application, huge networks viz. wide area networks (WAN) or metropolitan area networks (MAN) might work with reduced performance and efficiency. Also, the bandwidth of the network will affect the performance.

The application will work with about 200 active nodes in the network and active servers. A peak usage of 400 active nodes can also be dealt with.

* + 1. **Platform**:

The application will be developed in Java. Therefore it is platform-independent. Although, the minimum requirements of 10MB hard disk space to store the application and 512MB of RAM for its execution. Software requirements include the pre-installation of Java Runtime Environment (JRE) of any current available version present on the Oracle website. A working network connection is required and a valid authentication for the system administrator.

**RESOURCES**

A part of software planning task is estimation of the resources required to accomplish the software development effort.

Following are the resources required in software development:-

1. **Human Resources**

**Developers**

A developer is the person who researches, designs, develops and tests the software. A software developer also takes part in developing the software UI and programming the software logic.

In this project, a group of 5 members (Vishal, Varun, Saif, Rohit, Dharmin) are the developers performing the above tasks.

* 1. **Reusable Software Resources**

Component-based software engineering (CBSE) emphasizes reusability—that is, the creation and reuse of software building blocks. Such building blocks, often called components, must be catalogued for easy reference, standardized for easy application, and validated for easy integration.

**Off-the-shelf components**

Existing software that can be acquired from a third party or that has been developed internally for a past project. COTS (commercial off-the-shelf) components are purchased from a third party, are ready for use on the current project, and have been fully validated.

**Full-experience components**

Existing specifications, designs, code, or test data developed for past projects that are similar to the software to be built for the current project. Members of the current software team have had full experience in the application area represented by these components. Therefore, modifications required for full-experience components will be relatively low-risk.

**Partial-experience components**

Existing specifications, designs, code, or test data developed for past projects that are related to the software to be built for the current project but will require substantial modification. Members of the current software team have only limited experience in the application area represented by these components. Therefore, modifications required for partial-experience components have a fair degree of risk.

**New components**

Software components that must be built by the software team specifically for the needs of the current project.

* 1. **System Requirements**

**Development Environment**

The environment that supports the software project, often called the software engineering environment (SEE), incorporates hardware and software. Hardware provides a platform that supports the tools (software) required to produce the work products that are an outcome of good software engineering practice.

**Hardware Requirements:**

* Random access memory: 512 MB
* Hard disk space: 10 MB
* At least one active network connection

**Software Requirements:**

* Operating System: Windows XP/Vista/7/8/8.1/10, Linux
* Administrative rights to access network parameters
* Pre-installed Java Runtime Environment

**Project Estimation**

**4.1 LOC Based Estimation**

LOC (Lines of Code), usually referring to non-commentary lines, meaning pure whitespace and lines containing only comments are not included in the metric. The

number of lines of program code is a wonderful metric. It's so easy to measure and almost impossible to interpret. There are many different ways to count lines (e.g., with or without comments, counting statements rather than lines, or counting lines in automatically formatted code)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function | Optimistic | Most Likely | Pessimistic | Estimated LOC |
| Graphical User Interface (GUI) | 800 | 1000 | 1200 | 1000 |
| Graph plotting and analysis | 400 | 500 | 800 | 500 |
| Network Monitoring by packet transfer | 500 | 700 | 900 | 700 |
| Performance measurement and enhancement | 600 | 800 | 1000 | 800 |
| Notification system | 300 | 500 | 700 | 500 |
| Authentication | 100 | 200 | 300 | 200 |
| TOTAL |  |  |  | 3700 |

Estimated LOC= ( Optimistic+(4\* Most Likely)+Pessimistic)/6

1. GUI= (800+4000+1200)/6 = 1000
2. Graph plotting = (400+2000+700)/6 = 500
3. Network monitoring = (500+2800+900)/6 = 700
4. Performance tools = (600+3200+1000)/6 = 800
5. Notification system = (300+2000+700)/6 = 500
6. Authentication= (100+800+300)/6 = 200

Total= 3700

A historical data indicates that organizational average productivity for system of this kind is 1000 LOC/per month. Labour rate $8000 per month based on this cost per line of code $10.

Total estimated project cost is 3700 \* $ 10 = $37000.

Months = 3700/1000 =3 months and 21 days

**4.2 Function point based estimation**

FP = count total ×[0.65 + 0.01 ×( Σfi)]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Measurement Parameter | count | simple | average | complex | F.P |
| Number of user inquiries | 3 | 3 | 4 | 5 | 12 |
| Number of user output | 3 | 4 | 5 | 7 | 12 |
| Interface file type | 1 | 7 | 10 | 15 | 7 |
| External inquiry type | 2 | 3 | 4 | 6 | 6 |
| Number of external interfaces | 1 | 5 | 7 | 10 | 5 |
|  |  |  |  |  | Total=42 |

Then we must consider 14 “complexity adjustment values” rated on a scale of 0 – 5

(Step 2of FP Estimation):

• No influence 0

• Incidental 1

• Moderate 2

• Average 3

• Significant 4

• Essential 5

|  |  |
| --- | --- |
| FP Estimation Factor | Ratings |
| 1. Reliable backup and recovery | 0 |
| 2. Data communications | 5 |
| 3. Distributed processing functions | 2 |
| 4. Performance critical? | 4 |
| 5. Will the system run on existing heavily utilised operational environment? | 5 |
| 6. On/line data entry? | 1 |
| 7. On/line data entry over multiple screens or operations | 0 |
| 8. Are the master files updated on/line? | 0 |
| 9. Are the inputs, outputs files or enquiries complex? | 3 |
| 10. Is the internal processing complex? | 4 |
| 11. Is the code designed to be reusable? | 4 |
| 12. Are conversion and installation included in the design? | 1 |
| 13. Is the system designed for multiple installations in different organisations? | 5 |
| 14. Is the application designed to facilitate change and ease of use by the user? | 2 |

Total of Complexity adjustment factors = 36

* Total of FPs = 42
* Sum of all Fi’s = 36
* FP = 42 x (0.65 + 0.01 x 36) = 42.42

For systems of this type we have 7 FP/PM.

Labour rate is $8000 per person month,

Cost per FP is approximately $1000.

Project cost is 42\*$1000 = $42000

**4.3 COCOMO Model based Estimation**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software Projects | ab | bb | cb | db |
| 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 |

This project is semi-detached type. Effort is calculated as follows:

E=ab(KLOC)bb

  =3.0 (3.7)1.12

  =13 person-months

D=cb(E)db

   =2.5 (13)0.35

   =6.125 Months

P=E/D

  =2.12 Persons

**5. Risk Analysis and Management**

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. A risk is a potential problem that might happen or not. But regardless of its outcome we must identify it, assess its probability of occurrence, estimate its impact and establish a contingency plan in order to reduce its impact on the software engineering process. The following are the steps involved

in risk analysis:

**5.1 Risk Identification:**

Risk identification is a systematic attempt to specify threats to the project plan. By identifying known and predictable risks, the first step towards avoiding them can be taken whenever possible and control them when necessary.

There are two distinct types of risks for each of the categories

(1) Generic risks are potential threats to every software product.

(2) Product/specific risks – any special characteristics of the product that may threaten the project plan.

One method for identifying risks is to create a risk item checklist. The checklist can be used for risk identification and focuses on some subset of known and predictable risks.

**Generic Risks:**

Risks associated with the use of new technology and thus lack

of project experience.

Project budget could get exceeded.

Delay in the completion of the project due to changes in the project.

More funding required than projected due to non-availability of certain

component.

Compromise on the quality of the software due to tightening of

deadlines

**Product Specific Risks:**

If the packet filter is slow in filtering packets, the kernel may start dropping packets when its internal buffer gets filled up. As a result, packets containing data meeting the user specified criteria may not be captured by the packet filter.

The monitoring system may also be malfunctioning, causing the system to stop sending error messages.

**5.2 Risk Projection or Risk Estimation**:

Risk projection, also called risk estimation, attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur.

Four risk projection activities include:

(1) Establish a scale that reflects the perceived likelihood of a risk

(2) Delineate the consequences of the risk

(3) Estimate the impact of the risk on the project and the product and

(4) Note the overall accuracy of the risk projection so that there will be no misunderstandings.

The risks that may occur are:

 Project Delay

 System Failure

 Misuse of System

 Crossing the Projected Budget

**5.3 Risk Mitigation, Monitoring and Management:**

In order to avoid memory lapses and mistakes, the goal should be that it is enough to add new devices to a single location, from where all monitoring systems retrieve their information. Usually it is remembered to remove devices from monitoring at the latest when a notification is received that the device cannot be reached.

All of the risk analysis activities presented to this point has a single goal—to develop a strategy for dealing with risk.