ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

Predict the number of remaining faults during inspection using actual inspection data, where as Stranded predict which files will contain the most faults in the next release

K.Rama Krishna[1], B.Kavitha Laxmi[2], S.Prathibha Bharathi[3], A.Radhika[4]

Assistant Professor, Department of Computer Science Engineering and technology[1,2,3,4]

Abstract: The project consists of an efficient Currently Many researchers have addressed this important problem with varying end goals and have proposed valuation techniques to compute the total number of faults. A group of researchers focuses on finding error-prone modules base on the size of the module. Briand et al. Predict the number of remaining faults during inspection using actual inspection data, where as Strand et al predict which files will contain the most faults in the next release. Zhang and Mockus use data collected from previous projects to estimate the number of faults in a new project. However, these data sets are not always available or, even if they are, may lead to inaccurate estimates. For example, Zhang and Mockus use a naïve method base only on the size of the product to select similar projects while ignoring many other critical factors such as project type, complexity, etc. Another alternative that appears to produce very accurate estimates is base on the use of Bayesian Belief Networks (BBNs) .However, these techniques require the use of additional information, such as expert knowledge and empirical data, that are not necessarily collected by most software development companies. Software reliability growth models (SRGMs) are also used to estimate the total number of faults to measure software reliability. Although they can be used to indicate the status of the testing process, some have slow convergence while others have limited application as they may require more input data or initial values that are selected by experts.

keywords: network of nodes, Data Sanitization, Denial of Service, Bayesian Belief Networks, data computation, Error Valuation, test fault data, Software metrics, empirical data.

I.INTRODUCTION

Software metrics are crucial for characterizing the development status of a software product. Well-defined metrics can help to address many issues, such as cost, resource planning (people, equipment such as test beds, etc.), and product release schedules. Metrics have been proposed for many phases of the software development lifecycle, including requirements, design, and testing. In this paper, the focus is on characterizing the status of the software testing effort using a single key metric: the estimated number of faults in a software product. The availability of this estimate allows a test manager to improve his planning, monitoring, and controlling activities; this provides a more efficient testing process. Also, since, in many companies, system testing is one of the last phases (if not the last), the time to release can be better assessed; the estimated remaining faults can be used to predict the required level of customer support. Ideally, a fault valuation technique has several important characteristics. First, the technique should be accurate as decisions base on inaccurate estimates can be time consuming and costly to correct. However, most estimators can achieve high accuracy as more and more data becomes available and the process nears completion. By that time, the estimates are of little, if any, use. Therefore, a second important characteristic is that accurate estimates need to be available as early as possible during the system testing phase. The faster the estimate converges to the actual value (i.e., the lower its latency), the more valuable the result is to a test manager. Third, the technique should be generally applicable in different software testing processes and on different kinds of software products. The inputs to the process should be commonly available and should not require extensive expertise in an underlying formalism. In this case, the same technique can be widely reused, both within and among software development companies, reducing training costs, the need for additional tool support, etc. Many researchers have addressed this important problem with varying end goals and have proposed valuation techniques to compute the total number of faults.

A group of researchers focuses on finding error-prone modules base on the size of the module. Briand et al. predict the number of remaining faults during inspection using actual inspection data, whereas Ostrand predict which files will contain the most faults in the next release. Zhang and Mockus use data collected from previous projects to estimate the number of faults in a new project. However, these data sets are not always available or, even if they are, may lead to inaccurate estimates. For example, Zhang and Mockus use a naïve method base only on the size of the product to select similar projects while ignoring many other critical factors such as project type, complexity, etc. Another alternative that appears to

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

produce very accurate estimates is base on the use of Bayesian Belief Networks (BBNs) .

Valuation of Faults base on Fault Decay Model (ED3M), is a novel approach proposed here which has been rigorously validated using case studies, simulated data sets, and data sets from the literature. Base on this validation work, the ED3M approach has been shown to produce accurate final estimates with a convergence rate that meets or improves upon closely related, well-known techniques. The only input is the fault data; the ED3M approach is fully automated .

Although the ED3M approach has yielded promising results, there are fault prediction issues that are not addressed by it. For example, system test managers would benefit from obtaining a prediction of the faults to be found in ST well before the testing begins, ideally in the requirements or design phase. This could be used to improve the plan for developing the test cases. The ED3M approach, which requires test fault data as the input, cannot be used for this. Alternate approaches which rely on different input data (e.g., historical project data and expert knowledge) could be selected to accomplish this. However, in general, these data are not available at most companies.

A second issue is that test managers may prefer to obtain the predictions for the number of faults on a feature-by-feature basis, rather than for the whole system. Although the ED3M approach could be used for this, the number of sample points for each feature may be too small to allow for accurate predictions. As before, additional information could be used to achieve such valuations, but this is beyond the scope of this paper. Third, the performance of the ED3M approach is affected when the data diverge from the underlying assumption of an exponential decay behavior.

II.SYSTEM OVERVIEW

EXISTING SYSTEM:

Several researchers have investigated the behavior of fault density base on module size. One group of researchers has found that larger modules have lower fault density. Two of the reasons provided for their findings are the smaller number of links between modules and that larger modules are developed with more care. The second group has suggested that there is an optimal module size for which the fault density is minimal. Their results have shown that fault density depicts a U-shaped behavior against module size. Still others have reported that smaller modules enjoy lower fault density, exploiting the famous divide and conquer rule. Another line of studies has been base on the HAIDER ET AL.: VALUATION OF FAULTS BASE ON FAULT DECLINE MODEL . Convergence statistics, collected from the simulation of 100 data sets generated from the Triple-Linear behavior, of the estimator with (A)10 percent tolerance, (b)20 percent tolerance, and (c) 30 percent tolerance. Convergence statistics, collected from the simulation of 100 data sets generated from the Multi exponential behavior, of the estimator with: 10 percent tolerance, (b) 20 percent tolerance, and (c) 30 percent tolerance. use of design metrics to predict fault-prone modules. Briand et al. have studied the degree of accuracy of capture-recapture models, proposed by biologists, to predict the number of remaining faults during inspection using actual inspection data. They have also studied the impact of the number of inspectors and the total number of faults on the accuracy of the estimators base on relevant capture models. Ostrand et al. Bell et al. have developed a model to predict which files will contain the most faults in the next release base on the structure of each file, as well as fault and modification history from the previous release.

PROPOSED SYSTEM:

Many researchers have addressed this important problem with varying end goals and have proposed valuation techniques to compute the total number of faults. A group of researchers focuses on finding error-prone modules base on the size of the module. Briand et al. Predict the number of remaining faults during inspection using actual inspection data, whereas Ostrand et al predict which files will contain the most faults in the next release. Zhang and Mockus use data collected from previous projects to estimate the number of faults in a new project. However, these data sets are not always available or, even if they are, may lead to inaccurate estimates. For example, Zhang and Mockus use a naïve method base only on the size of the product to select similar projects while ignoring many other critical factors such as project type, complexity, etc. Another alternative that appears to produce very accurate estimates is base on the use of Bayesian Belief Networks (BBNs) .However, these techniques require the use of additional information, such as expert knowledge and empirical data, that are not necessarily collected by most software development companies. Software reliability growth models (SRGMs) are also used to estimate the total number of faults to measure software reliability. Although they can be used to indicate the status of the testing process, some have slow convergence while others have limited application as they may require more input data or initial values that are selected by experts.

III.THE WORKING PRINCIPLE

SOFTWAREREQUIREMENTS SPECIFICATION:

3.1. Scope:

The goal of the project is estimate the faults in a software product. The availability of this estimate allows a test manager to improve his planning, monitoring, and controlling activities; this provides a more efficient testing process. Estimators can achieve high accuracy as more and more data becomes available and the process nears completion.

3.2 Project Features:

This project is used to remove the faults from the c# programs by checking the programs for compiler errors, manual errors, faults and other bugs.

3.3 User characteristics:

This project is mainly designed for the software testing personnel and mainly professional in CSharp programming.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

3.4 Constraints:
The main constraint of the project is that the project can be used to find the faults only on the c# programs.
3.5 Dependencies:
The project is mainly dependent on CLR – Common Language Runtime libraries.
3.6.The Overall Description:
An accurate prediction of the number of faults in a software product during system testing contributes not only to the management of the system testing process but also to the valuation of the product's required maintenance. Here, a new approach, called Valuation of Faults base on Fault Decay Model (ED3M) is presented that computes an estimate of the total number of faults in an ongoing testing process. ED3M is base on valuation theory. Unlike many existing approaches, the technique presented here does not depend on historical data from previous projects or any assumptions about the requirements and/or testers' productivity. It is a completely automated approach that relies only on the data collected during an ongoing testing process.
This is a key advantage of the ED3M approach as it makes it widely applicable in different testing environments. Here, the ED3M approach has been evaluated using five data sets from large industrial projects and two data sets from the literature. In addition, a performance analysis has been conducted using simulated data sets to explore its behavior using different models for the input data. The results are very promising; they indicate the ED3M approach provides accurate estimates with as fast or better convergence time in comparison to well-known alternative techniques, while only using fault data as the input.
Software metrics are crucial for characterizing the development status of a software product. Well-defined metrics can help to address many issues, such as cost, resource planning (people, equipment such as test beds, etc.), and product release schedules. Metrics have been proposed for many phases of the software development lifecycle, including requirements, design, and testing. In this paper, the focus is on characterizing the status of the software testing effort using a single key metric: the estimated number of faults in a software product. The availability of this estimate allows a test manager to improve his planning, monitoring, and controlling activities; this provides a more efficient testing process. Also, since, in many companies, system testing is one of the last phases (if not the last), the time to release can be better assessed; the estimated remaining faults can be used to predict the required level of customer support. Ideally, a fault valuation technique has several important characteristics. First, the technique should be accurate as decisions base on inaccurate estimates can be time consuming and costly to correct. However, most estimators can achieve high accuracy as more and more data becomes available and the process nears completion.
By that time, the estimates are of little, if any, use. Therefore, a second important characteristic is that accurate estimates need to be available as early as possible during the system testing phase. The faster the estimate converges to the actual value (i.e., the lower its latency), the more valuable the result is to a test manager. Third, the technique should be generally applicable in different software testing processes and on different kinds of software products.
The inputs to the process should be commonly available and should not require extensive expertise in an underlying formalism. In this case, the same technique can be widely reused, both within and among software development companies, reducing training costs, the need for additional tool support, etc. Many researchers have addressed this important problem with varying end goals and have proposed valuation techniques to compute the total number of faults. A group of researchers focuses on finding error-prone modules base on the size of the module. Briand et al. predict the number of remaining faults during inspection using actual inspection data, whereas Ostrand . Predict which files will contain the most faults in the next release. Zhang and Mockus use data collected from previous projects to estimate the number of faults in a new project. However, these data sets are not always available or, even if they are, may lead to inaccurate estimates. For example, Zhang and Mockus use a naïve method base only on the size of the product to select similar projects while ignoring many other critical factors such as project type, complexity, etc. Another alternative that appears to produce very accurate estimates is base on the use of Bayesian Belief Networks (BBNs) .
Valuation of Faults base on Fault Decay Model (ED3M), is a novel approach proposed here which has been rigorously validated using case studies, simulated data sets, and data sets from the literature. Base on this validation work, the ED3M approach has been shown to produce accurate final estimates with a convergence rate that meets or improves upon closely related, well-known techniques. The only input is the fault data; the ED3M approach is fully automated.
Although the ED3M approach has yielded promising results, there are fault prediction issues that are not addressed by it. For example, system test managers would benefit from obtaining a prediction of the faults to be found in ST well before the testing begins, ideally in the requirements or design phase. This could be used to improve the plan for developing the test cases. The ED3M approach, which requires test fault data as the input, cannot be used for this. Alternate approaches which rely on different input data (e.g., historical project data and expert knowledge) could be selected to accomplish this. However, in general, these data are not available at most companies.
A second issue is that test managers may prefer to obtain the predictions for the number of faults on a feature-by-feature basis, rather than for the whole system. Although the ED3M approach could be used for this, the number of sample points for each feature may be too small
to allow for accurate predictions. As before, additional information could be used to achieve such valuations, but this is beyond the scope of this paper. Third, the performance of the ED3M approach is affected when the data diverge from the underlying assumption of an exponential decay behavior.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

FEASIBILITY STUDY:
The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY

TECHNICAL FEASIBILITY

SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY:
This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.
TECHNICAL FEASIBILITY:
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.
SOCIAL FEASIBILITY:
The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

IV.IMPLEMENTATION OF SYSTEM

METHODOLOGY :
Definitions:
Error:A mistake made by a member of the software team.
Fault:
The section of code or documentation, which must be changed in order to correct a fault.
Failure:
A situation in which the software fails to execute as intended.
Problem Report:
Usually documentation that a failure has occurred during testing or use.
May also be used to document faults found in inspections and reviews.
Fault -"An instance in which a requirement is not satisfied. Here it must be recognized that a requirement is any agreed upon commitment. It is not only the recognizable external product requirement, but can include internal development requirements…".
Software Faults:

Software Fault: Any flaw or imperfection in software work product or software process.
Software work product is any artifact created a part of the software process.
Modules:
1) Login
2) Browse
3) Error Valuation
4) Error Correction
5) Report.
Module Description:
1) Login:
The Valid user enter into login to send data to available network systems, if the user doesn't register it will move to new user creation form. In this Module Collecting the general user details and store database for future references. It having Name, Password, Confirm Password, and Email address.
2) Browse :
The user select the already created project given as input. we have to select the (.exe) file of the project from debug folder. using (.exe) file we retrieve a class name, method name, Parameter name and namespace. we going to apply a refactor and merging techniques in selected class name.
3) Error Valuation :
In this module we computes an estimate of the faults in an ongoing testing process. This could be used to improve the plan for developing the test cases.
System testing is one of the last phases (if not the last), the time to release can be better assessed; the estimated remaining faults can be used to predict the required level of customer support.
4) Error Correction :
Error correction techniques are used to improve the estimates and, consequently, reduce the convergence time. After error has been estimated we going to correct the error in specified line number and files.
It is used to correct the current estimate; the corrected value is the Output.
5) Report :
In this module process report has been generated for specified users. In report form specified error corrected filename, username ,date and status of the project display in report form. The information stored in admin database.
Software process is a set of activities, methods, practices and transformations that people use to develop and maintain software work products.

V.EXPERIMENTAL RESULTS

DESIGN

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

User Model View

This view represents the system from the users perspective.

The analysis representation describes a usage scenario from the end-users perspective.

Structural model view

This model view models the static structures.

- Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

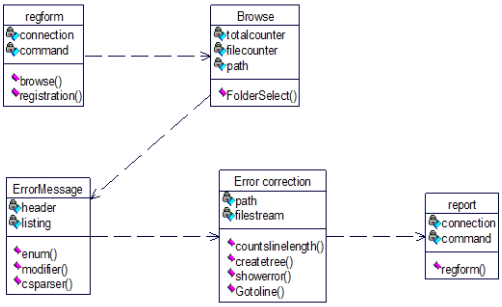
- Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

- Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

Class Diagram:



VI. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

6.1 Coding login :

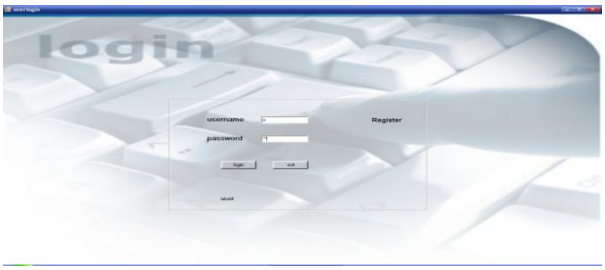
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Data.SqlClient;


namespace CodeTreeView
{
public partial class regform : Form
{
SqlConnection con;
SqlCommand cmd;
SqlCommand cmd1;
SqlDataReader dr;
public regform()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
con = new SqlConnection("server=.;database=valuation;user id=sa;password=");
con.Open();
cmd =new SqlCommand ("select * from login where username='"+username .Text +"' AND password='"+password .Text +"'",con );
cmd1 = new SqlCommand("select * from rreport", con);
cmd1.CommandText = "insert into rreport(userreport)values(@userreport)";
cmd1.Parameters.Add("@userreport", SqlDbType.VarChar, 20).Value = username.Text;
cmd1.ExecuteNonQuery();
dr = cmd.ExecuteReader();
if (dr.Read())
{
browse bbb = new browse();
bbb.Show();
}
else
{
label4.Text = "please enter a correct username and password";
}
}
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
registration rrr = new registration();
rrr.Show();
}
private void button2_Click(object sender, EventArgs e)
{
Application.Exit();
}
}
}
```

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

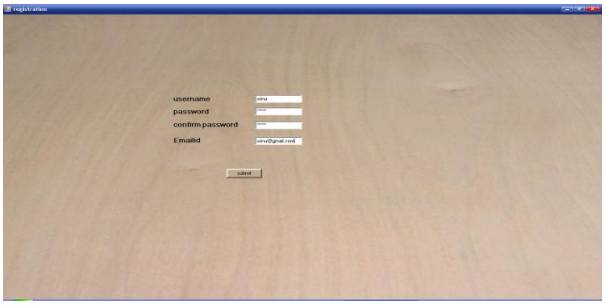
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

VII. Screen shots

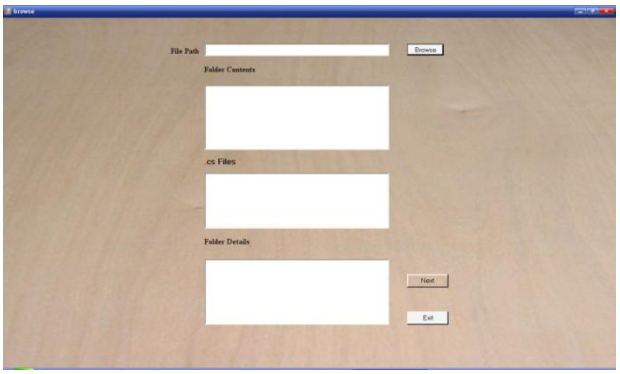


Screenshot Login:

Screenshot New Registration



Screenshot browse:



Screenshot Error Report:

VIII. Conclusion

This project provides efficiency in testing the software product in an accurate manner. By working on this paper

we can find many testing products and also prediction of faults in a software product. By working on this project I have implemented software which can find errors, faults, and bugs.The software has been developed in such a way that it will find the error that are in the C Sharp files and will able to correct them from the software only.The software also generates the error report of how many errors are occurring in the programming. Thus we can also analyze the throughput of the programmers. This is designed to find the compiler errors.The efficiency of the software is also high, as it is finding all the compiler errors and faults and other things. There is cent percent efficiency achieved.Since the software is providing the exact results, we can correctly estimate the faults using the above model.

IX.FUTURE ENHANCEMENT

Project is to be enhanced to test types of programming files.Project is used to test for even the exe files.Multi file and linked file support is to be implemented.Analysis of the error files for future comparison.Multi platform compatibility.Program adapting to recurring errors.Automated error correction technique.The compatibility for other programming languages source files is to be implemented.

REFERENCES

[1] Microsoft Visual Basic.NET Programmer's CookBook:- MATTHEW MACDONALD (Tata McGrawHill Edition)
[2]. Grey Buczek, .NET developers guide 2002, Prentice-Hall India.
[3] .Benolt Marchal, VB.NET by example 2003 – TataMcGraw- Hill. System Analysis & Design – Alenis Leon.
[4]. Integral approach to software engineering – Pankaj Jalole.
[5] N.E. Fenton and M. Neil, "A Critique of Software Fault Prediction Models," IEEE Trans. Software Eng., vol. 25, no. 5,pp. 675-689, Sept.-Oct. 1999.
[6] Mays, R., et al., Experiences with Fault Prevention, IBM Systems Journal, January 1990.
[7] P. Zhang, A. Mockus, and D. Weiss, "Understanding and Predicting Effort in Software Projects," Proc. 25th Int'l Conf.Software Eng., pp. 274-284, 200
[8] J.W. Cangussu, R.M. Karcich, R.A. DeCarlo, and A.P. Mathur, "Software Release Control Using Fault Base Quality Valuation,"Proc. 15th Int'l Symp. Software Reliability Eng., Nov. 2004.
[9] C. Bai, K.-Y. Cai, and T.Y. Chen, "An Efficient Fault Valuation Method for Software Fault Curves," Proc. 27th Ann. Int'l Computer Software and Applications Conf., Nov. 2003.
[10] I. Burnstein, A. Homyen, T. Suwanassart, G. Saxena, and R. Grom, "A Testing Maturity Model for Software Test Process Assessment and Improvement," Software Quality Professional, vol. 1, Sept. 1999.

BIOGRAPHIES

Mr.K.Ramakrisha, presently working as an assistant professor in computer science engineering and technology department, samara university, samara ,Ethiopia .He received the master of technology degree in VNR Vignana Jyothi Institute of Engineering and Technology- Jawaharlal Nehru Technological University Hyderabad, India in 2010. He received the bachelor of technology degree in The Vazir Sultan College of Engineering And technology, kakatiya university

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

,Warangal, India. He Has 6+ Years Teaching Experience, His Research Interests Include mobile ad-hoc networks , Data Mining, Information Security, Software Testing, mobile communication and cloud computing.

B.Kavitha Laxmi, she is currently working as an Assistant Professor, Department of Computer Science &Engineering in HITAM, Hyderabad, R.R.Dist, Telangana, India. she received the master of technology degree in VNR Vignana Jyothi Institute of Engineering and Technology-Jawaharlal Nehru Technological University Hyderabad, India in 2010. She Has 5+ Years Teaching Experience. Research Interests Include mobile ad-hoc networks , Data Mining ,Web Technologies Cloud Computing and data warehouse.

S. Prathibha Bharathi, Graduated computer science and Engineering From Jawaharlal Nehru technological University, Hyderabad, Andhra Pradesh, India , And M.Tech In Software Engineering,VNR Vignana Jyothi Institute of Engineering and Technology- --Jawaharalal Nehru technological University Hyderabad ,A.P,India In 2010.She Is Working Presently As Assistant Professor MLRIT , Hyderabad, She Has 7 Years Experience. Research Interests Wireless sensor networks, network security, data mining, image processing.

A.Radhika, Graduated Information Technology and Engineering, Andhra Pradesh, India. And M.Tech In Computer Science &Engineering-Jawaharalal Nehru Technological University Hyderabad , A.P,India She Is Working Presently As Assistant Professor-HITAM , Hyderabad, She Has 7 Years Experience. Research Interests Network Security ,Computer Organization ,Data warehousing.