SURESH GYAN VIHAR UNIVERSITY
The first research oriented University of state

Certificate

This certifies that the dissertation entitled

"Evaluation Of Changeability Indicator In Component Based Software System"

Is submitted by

Madhu Sudan Yadav

SGVU091083093

M.Tech (SE) *in the year* ***2014*** *in partial fulfillment of*

Degree in Master of Technology in *Computer Science & Engineering*

Suresh Gyan Vihar University, Jaipur

Dr Savita Shivani
Assistant Professor (IT Department)

Date:
Place: Jaipur, Rajasthan

Candidate's Declaration

I hereby declare that the work, which is being presented in the dissertation, entitled **"Evaluation Of Changeability Indicator In Component Based Software System"**

in partial fulfillment for the award of Degree of "Master of Technology" in Department of Computer Science & Engineering with Specialization in Computer Science & Engineering, and submitted to the Department of Computer Science & Engineering, Suresh Gyan Vihar University is a record of my own investigations carried under the Guidance of **Dr. Savita Shivani**, Department of Information Technology, Suresh Gyan Vihar University, India.

I have not submitted the matter presented in this Dissertation anywhere for the award of any other Degree**.**

(Name and Signature of Candidate)

MADHU SUDAN YADAV

Software Engineering

Enrollment No: SGVU091083093

Counter Singed By:

Dr. Savita Shivani

Supervisor

Assistant Professor, SGVU, Jaipur

Details of Candidate, Supervisor(s) and Examiners

Name of Candidate: Madhu Sudan Yadav

Department Of Study: M. Tech. (Software Engineering)

Enrolment No.: SGVU091083093

Dissertation Title: "Evaluation Of Changeability Indicator In Component Based Software System"

Supervisor (s) and Examiners Recommended **(with Office Address including Contact Numbers, email ID)**	
Supervisor **Dr. Savita Shivani**	
Examiner	

Signature with Date

(Head of Department)

Acknowledgements

I would like to express my sincere appreciation to Department of Computer Science & Engineering Suresh Gyan Vihar University; this study would not have been accomplished without their guidance. My sincerest gratitude goes to **Dr. Savita Shivani**, Assistant Professor and M.Tech Coordinator, Computer Science & Engineering, Suresh Gyan Vihar University, Jaipur and who guided me with her dedicated attention, expertise and knowledge throughout the process of this dissertation.

I thank her for her valuable guidance, her commitment, timely advice and constant support at each & every step in completion of the study. Her scientific and analytical approaches to new problems, wide knowledge and discerning remarks really helped me at every need of this work. It was due to her immense keenness and continuous attention that this study could take place a final picture.

I express my sincere thanks to **Mr. Dinesh Goyal,** Vice Principal, Suresh Gyan Vihar University, Jaipur, for his encouragement.

I also thanks to my friends for helping me during this work.

MADHU SUDAN YADAV

Abstract

The maintaining of software system is a major cost concern. The maintaining of a software system depends on how the changes made to it. The maintainability of a system depending on the folw of software, its design pattern and CBSS. In Maintainability phase of a sotware system there are 4 parts, like analyzing, testing, stability, and changes made to it. In some side areas, these systems emerged very rapidly. There are many companies which purchase software instead of developing it .These companies do not have any interst in the testing of the system but wants to like a smoothness in the flow of the system during changes. This article mainly focuses on the single part as changeability.

Component based software system is used very widely in the software system. Researchers find that component based software system are best for change and improvement etc .The assessment of the changeability of software systems is most important for the large buying systems.

Changeability can be defined as a measure of impact on changes made to a component on the rest of the system. Changeability is one of the characteristics of maintainability .Software changeability is associated with refactoring which makes code simpler and easier to maintain(enable all programmers to improve their code).Factors that affect changeability include coupling between the modules, lack of code comments ,naming of functions and variables. Basically, ”changeabilty” is the ability of a product or software to be able to change the structure of the program. It is the rate the product allows the modification to its components.

In this work changeability based cost estimation is done. Initially we take four components, these components are evaluated based on the coupling, cohesion and Interface metrix. Next some changes are made to the existing components and than again these components are evaluated. Now, on the basis of these two evaluations some conclusion is made for changeability cost.

TABLE OF CONTENTS

LIST OF TABLES

Table Number: **Page Number**

LIST OF FIGURES

Chapter-1

Introduction

1.1 Overview

The maintaining of software system is a major cost concern. The maintaining of a software system depends on how the changes made to it. The maintainability of a system depending on the folw of software, its design pattern and CBSS. In Maintainability phase of a sotware system there are 4 parts, like analyzing, testing, stability, and changes mad eto it. In some side areas, these systems emerged very rapidly . There are many companies which purchase software instead of developing it .These companies do not have any interst in the testing of the system but wants to like a smoothness in the flow of the system during changes. This article mainly focuses on the single part as changeability. An idea of assising changeablity to check the effects by changes made to it. Earlier, systems were developed by using structured approach, which was very successful, but only for simple applications. Then came the object-oriented (OO) approach, which is based upon encapsulation, inheritance, and polymorphism[15]

At the time of developing of software there are many reasons for changes. These effects are depend on some unique components. Hence, it is most important to assess the effects of these changes on the different components individually and also find out these changes on the group of changes. During software evolution a number of changes are applied to the system. This can be done due to many things like improvement, correct result etc. Different area of the software shows different sensitivity towards these changes. After identifying which part of the software is more sensitive to the changes is very helpful. If we know which area of the software is more sensitive than an appropriate actions are taken.[16]

Component based software system is used very widely in the software system. Researchers find that component based software system are best for change and improvement etc .The assessment of the changeability of software systems is most important for the large buying systems.[17]

Developing systems in this environment needs how the system provides real time values to the market. The ambiguous meaning of the system provides an inability in searching values.[15][18]

Changeability in other means, the ability to meet the requirements autonomously in the near future .If the developed system is robust or changeable than the speed of execution of the task is also high.[19]

1.2 Change

Softwares are like organic things that change internally and externally with multiple environmental business reasons, for usage of software to continue, and for it to remain non obsolute it needs to remain constant state of change to remain in sync with the real life. Software becomes best and worse after changes made to it. If we made any change to the existing system, it is very expensive and difficult but it helps a lot in the near future. Hence, it is very important to identify the changes and made fix them for better future decisions.[1]

Transition of a system to an altered state(if there are functional,structural or interfacing changes in the software) over time can be defined as change. The system that requires no change is impossible to produce. If the software system at time k and k+1 is not same, this means that the system has gone under some changes.When a software is released and used, as time passes many new things are needed so for the market needs the system is gone for changes. To made the errors fix that are found and increase the speed and some other parts of the software may need to be modified, this modification is said to be change. All about these means that a system must undergo changes for the need of the market.[2]

Software applications are not only single components systems, they are made up by collection of components in a single system. These components can be arranged to developed a software in such a way that the code written once can be reused many times by doing some changes in the code , which can reduce the development time ,cost and effort. Many times these changes produce new errors which degrades the quality of the system. There the rate at which these changes is applied to the system is governed by companies priorly. [2]

The depth of the work about these things is carried out by Lahman in the year 1985 . After this study, they giving laws of changeability and also known as Lahman's laws. He said that these

laws are very helpful and broadly used. The researchers before giving these laws, they analysed a number of broad systems.

The **first law** tells that the maintenance of a software system is a not ended process. When the time passes the environment in which the system works is changed and it needs some new features resulting in modification of the existing system. When the improved system is re-launched in the environment it produces more changes, so the process recycles again.

The **second law** tells that of a system is changed its structure is degraed. To avoid this thing we use only preventive improvement in which the system improves without changing its functionality.

The changes made to the software system follows a proper approach. The new improved software contains features of the customer needed. The basic properties of change that are recorded by a simplest change management system include ownership(who made change),object that is changed (product,subsystems,module,file,set of lines) and time when the change was performed.[1]

A change process can be divided into three parts : (1) change agent, (2) the method used in the change and (3) effect of this change on the system. The change agent is the cause for which this change is required. The aim of change agent is the intention for which the change is fixed. The method of change tells about the process to reach at the final stage from the initial stage in terms of cost, overhead etc. The effect due to change is tells about what is difference b/w the initial stage and the final stage of the system. So overall we use these three events in performing a change to the software system.

1.3 Need to measure change

Measuring changes is important for the following reasons:

1.3.1 The change provides the managers to understand the design flow of the system and used by different users.

1.3.2 Metices can also help programmers in improving their developing skills and programming skills to provide best results.

1.3.3 The change can also provides help to the researchers in understanding the evolution of the software system.

When we want to measure the changes then the changes can be measure before the tme and after the tme and then compute the measurements of both. Metrices are needed that can measure change object and that provides meaningful values. We can define change metrices as a metrics that provides the different b/w the initial stage and final stage.[3]

1.4 How can we measure change ?

The change request from the market or users is to identify how the system is undergone changes or how much cost it needs in applying this change and overhead concern.[2] If these changes are accepted by the developer than a new plan for the system is developed. At the time of the planning , these accepted changes are considered. Now a final decision is made for these changes to applying in the new release.[2] These new changes are modified to the existing system a new release of the existing one is released. Hence a new version of the existing software system is released.

To automate the change management we have to built a collection of tools which we refer to as a "soft change". The change measure provides a new information infrastructure for managers as well as for future research on large software systems.The new insights generated by using measures of software change underline the importance of studying changes to source code.

Measures on the change can be of 4 classes :

(i) Size measures
(ii) Duration measure
(iii) Complexity measure
(iv) Expertise measure

Size measures-It includes the number of lines of codes that are to be added or deleted and the line of codes which are touched in the process of the change .The change size is also measured by number of sub changes.

Duration measures-Indicate the temporal interval spanned by the change.

Complexity measures-change complexity or interaction include total number of files or componens that are to be touched during this change or how many organistions or programmers ie developers are needed.

Expertise measures-change expertise measures are based on average expertise of developers performing the change.[6]

1.5 To measure impact of change-

However change is very difficult to track and measure its impact on the software process needs a framework that works to get us influence on all aspects of software design ,where and how it is influencing, what is its implications on the system and consequence of its influence.

By collecting data before the change can be made to the software, during the changes made to the software and after the change made to the software that you implement. The process of measuring of changes should be as to measure the simple measure instead of the complex measures. Change impact analysis would be gained measurable attention towards these challenges of condition. The system software community feels the need of recognizing these sequences of changes. The analysation of these changes is also important. In the software life cycle process (slcp), dependencies are more difficult or complex as many softwares has millions of lines of code. The effect of change analysis on the software system make the process more fiddle and provides good measure estimation. [6]

1.6 Software projects gets affected due to change

The changing software system can be made difficult by complexity and conformity. When we change only a single part of the software it results an error in other part or provides undesired results, all this needs many more changes in the newly developed system for maximum efficiency. Also, a designer typically will need to make trade offs between two or more characteristics when designing the system. Let considering a highly modularized code and this code is usually easy to maintain such that it has a better changeability characteristic, but may not perform as well. On a similar vein a normalized database may not perform as well as a non

normalized database. These issues needs to be identified due to which informed design decisions can be made[10][2].

1.7 Change can lead to software erosion

Inspite of many advantages there are many cases where frequent changes may lead to software application failure leading to software erosion .This may be due to lack of experience for reuse, lack of documentation, tools and methodology for reuse or may be due to conflicts .Therefore, it is very important to study and identify parameters are related to the changeability of software components[6]

1.8 Changeabilty

Changeability can be defined as a measure of impact on changes made to a component on the rest of the system. Changeability is one of the characteristics of maintainability .Software changeability is associated with refactoring which makes code simpler and easier to maintain(enable all programmers to improve their code).Factors that affect changeability include coupling between the modules, lack of code comments ,naming of functions and variables. Basically, ”changeabilty” is the ability of a product or software to be able to change the structure of the program. It is the rate the product allows the modification to its components.

A system can be defined in the form of group of parameters that describes functional and non functional features. Robustness is a characteristic that becomes “constant” inspite of changes done to the system. Scalability is the characteristic that maintains the levels of measures. Changeability is related with the members with the parameters of the group. The scalability can also be defined as the scaling of the number of satellites in a constellation and number of users in the system. One of the most studied Source Code issue is the cloning in the software. These Clones are the duplicate codes in the software program. Since, there is not any agreement or declaration on the harmfulness of clones. Changeability is the ease with which a source code entity is modified or changed in its source code. This is analysed through the metrics calculated from the history of changes of the methods that are used. The effect of clones on the modifiability of methods is measured by comparing the metrics of methods that contain clones to those that do not.

1.9 Component Based Software Testing

Now a days ,component based techniques are gradually increases in managing the growing of software complexity in software system. By the use of these technologies, developers develop large system by integrating small sytem of code specific.

Component based software engineering provides different types of advantages in the softaware systems. But it can also complicates the task used for developing software system.For eg. By the use of components software evaluation becomes difficult and there is also a problem in validating the software. These problems occurs because of the deficiency of the information about the cpmponents to the application developers utilizing these components. Developers are also reluctant in giving source code due to many resons. Without having source code, informations which are used to validating software and maintaining software, like dependence information, can't be calculated. And if there is an available of source code for these components, computing some information may also require analysis tools or complex analysis that are not available for component based user. Atlast , some information which engineers might want to use, like specification, can't be easily calculated by the source code alone.

Software engineering task based on component applications are supported by component meta data. Component metadata are the metadata of components or metamethods, which are associated with these components that are used to calculating thse metadata. Generally, metadata consist of abstract source code representation, different forms of static data, complexity metrics, control dependency with some dynamic information.

Chapter-2

Related Work

Lustman, Keller, R.K.; published a work in 2001, changeability indicator in object-oriented paradizm typically cohesion, on Software Maintenance & Re-engineering, in 5th European conference on , volume, no., pp.39,46.

The assessment these changes in the software system is of a big concern for the fast moving system for the clients that purchases large system like telecom sector etc. First we see that there is a solution to this problem is that we analyses the dependency of the software system about changeability and the design pattern, by analyzing the only properties that are used for the changeability in this design. In the era of component based software system, experiments had played to show the coupling among the components or classes. Yet, cohesion class in comparison to the changeability of the software system had not been analyzed. In the article presented, there is a result to show whether cohesion is related with the changeability or not. The metrix for cohesion like LCC and LCOM metrix were used to measure changeability and a model for changeability called change impact model was used. The information gathered from this, when we analyzed three system of individual size shows no relation among them. When manual analization is performed for components shows a weak cohesion that the metrics used not collects all the facts. Here, we find that the metrics like LCC and LCOM will not be used in the case of changeability indicator.

Hebig, Gabrysiak and Giese, H., in 2012 researched on a topic named, to find the process for MDE – patterns and handle the risk of changeability, in the year 2012, International Conference on , volume, no., pp.38,47, 2-3 June 2012

There is a factor out of many technical factor that posses an effect on the changebility is Model Driven Engineering, where there is many model and multitude techniques that are mainly used to investigate the final system. The most important problem in software change is to change with minimum cost in iterative and incremental development of software system. Therefore the effective software change is done by the use of Model Driven Engineering techniques. Yet, here at this time there is currently no process available that detect and handle these Model Driven Engineering process of changeability. In this article we increment the beforehand-by introducing

a process for the modeling to indicate the risk for the changeability. In additional, 4 researchers were also published paper for this envisioned pattern in detail. Next, we introduced techniques to handle the risks that are associated with these patterns.

Ajrnal Chaumun, Kabaili, Keller, and R.K.; Saint completed a research in the year 2000 on a topic named, Design properties and changeability in OO software, rengineering. Proceedings of the 4th European , vol., no., pp.45,54, Feb 2000. The assessment these changes in the software system is of a big concern for the fast moving system for the clients that purchases large system like telecom sector etc. First we see that there is a solution to this problem is that we analyses the dependency of the software system about changeability and the design pattern, by analyzing the only properties that are used for the changeability in this design. In this article, we developed a model for the changes in the software and impact of changes to ths system and it is used in the C++ langauage. Next, we find a group of 9 object-oriented(OO) metrix, 4 out of these are especially used for the detection of the changeability in the system software. Both of these things are used for the 3 tests of the industrial sized systems. This above experimental provides a greater relation with the system, changes or changeability and by accessing a class or component from the other class or component. While, there is no observations that supports to the hypothesis that there is no influence on the depth of the inheritance tree. Next, the these observations confirmed that the inheritance tree is limited used in the systems of industrial size.

According to Ajrnal Chaumun, Kabaili and R.K, Growth in costs of maintenance have become an important concern for those who are developers and users of software systems. The ability to change is an important feature of maintainability usually in those areas which often require changes in softwares. Present research is based on assumption that influence of high-level design on maintainability is passed on to changeability and is tested for quality features. The strategies adopted to evauate the changing ability of an object-oriented (OO) system to assess the influence of modifications made to the groups of the system. The definition of change impact model is basically conceptual and this model is usually mapped on the C++ language. Practicality of the change impact model on the large industrial software systems can be evaluated by the tests involving the impact of a change which are basically performed on the telecome

systems. The outcomes show that this software can very easily imbibe such sort of changes and such conventional OO design metrics which are selected very carefully can be used as precursors of changeability.

According to Lozano, A., A, Before we elaborate on the different approaches used for the study, Let us first explicitly distinguish "methodology and methods". "Methodology" is something broader than method. Such methodologies, which can be defined as major ways of proceeding to answer a question, or major ways of approaching the subject matter, come in two assorted and mutually exclusionary "materialist or idealist" varieties.

There are three main types of research designs: qualitative, quantitative and mixed methods. These three approaches appear discrete but they are not. According to Newman & Benz, qualitative and quantitative should not be viewed as polar opposites or dichotomies but they represent different ends on a continuum. Mixed methods research resides in the middle of the continuum because it incorporates elements of both qualitative and quantitative approaches.

According toLozano, A. & Wermelinger, M., Qualitative research is a means of exploring and understanding the meaning individuals or groups ascribe to a social or human problem. The process of research involves emerging questions and procedures, data typically collected in the participants setting, data analysis inductively from particulars to general themes, and the researcher making interpretations of the meaning of the data. Those who engage in this form of inquiry support a way of looking at research that honors an inductive style, a focus on individual meaning, and the importance of rendering the complexity of a situation.

The approach adopted by qualitative researchers tends to be inductive which means that they develop a theory or look for a pattern of meaning on the basis of the data that they have collected. This involves a move from the specific to the general and is sometimes called a bottom-up approach.

According Ma Zhe, From the point of view of time, research is either as one-time research or longitudinal research. In the former case the research is confined to a single time-period; this type of research is also called cross-sectional. Cross-sectional research generally involves a large

number of samples as in this case data is collected only once from subjects to study how a certain phenomenon cuts across a group. However, in case of longitudinal research a small set of subjects are observed over a long period of time and the objective of this type of research is to observe the behaviour of subjects over a period of time.

D'Amorim, F., Borba, P, shows that Research can also be categorized as Fundamental or Applied. While Applied research aims at finding a solution for an immediate problem facing society or an institution, Fundamental research is concerned with formulation of a theory. To elaborate these types further, whereas basic research is directed towards finding information that has a broad base of applications and thus, adds to the already existing organized body of scientific knowledge, gathering knowledge for knowledge's sake is termed fundamental research. Research on nature of language, syntactic features of a language or word formation rules in a language are examples of fundamental research. Examples of Applied research, on the other hand, are offering solutions to language disorders, language learning problems, improving language teaching methods, etc. Even the current research is applied in nature as it aims at offering a more robust method of speaker identification.

Ross, A.M. proposed that the research is about making an inquiry into the nature of voice, resonance and few other features of human voice that distinguishes an individual from the other. Beyond this general enquiry, the research is aimed at offering a robust method of analysis that could be exploited for speaker identification. Figuring out acoustic parameters of voice that should be investigated and how they should be studied are some of the objectives of the current research. Thus, the research is designed to offer some solutions to practitioners in forensic labs on the one hand and on the other discusses in general the nature of human voice, which makes every individual unique. On the basis of these research goals, the current research can be called both Applied and Fundamental. The researcher relies on analytical techniques for analysis of the data collected for the research. In most parts, the researcher has based her judgement on available data and produced results from analysis of it. The research has also required looking into a large database, which has numbers and values of different types, for example, the values of amplitude and pitch of individual vowels of all the subjects. Because of the nature of data and

approach to evaluating the data, the current research can be categorized as both Analytical and Quantitative.

According to Duro, Moreira, F.; Rogado, Once the research type is established, it's easier to plan the research, but there are too many things that a researcher will need to do to accomplish any research. Every research involves planning for steps beginning with writing the objective and scope of the research to the last stage of deciding the style of reporting or presentation. There are many other stages of research about which a researcher must think before taking a deep plunge. In this book Research Methodology has laid down 10 questions which must be answered by the researcher for carrying out any research. Research design is thus a conceptual structure within which research is conducted; it constitutes the blueprint for the collection, measurement and analysis of data. The following section, thus, discusses the answers to the above questions in order to prepare a blueprint for the current research.
Forensic Phonetics and its application for speaker identification, the research methods employed for previous research in this area have been reviewed. After a close review of the existing research work and the work of practitioners in forensic labs, a suitable research design has been constructed.

In forensic speaker identification, one of the most common tasks is to compare a question sample with the suspect sample. A question sample is the voice sample of an offender of a crime. On the basis of this sample, a suspect sample is obtained from the suspects of the crime. Obtaining suspect sample is a tricky job. First, the investigator carefully listens to the question sample and notes down the peculiarities observed in the offender's language. It can be a word, a sound, pitch level or any other feature of voice which may be peculiar. The investigator then prepares a script which includes the most commonly used words in the question sample which carry the peculiarities. This script is then provided to the suspects and they are made to read out that script. While they read it out, their voices are recorded.

Shu-Ching Chen; Gulati, S.; Huang, X.; Leroy and Chengjun Zhan claims that Most labs use the above method for forensic speaker identification. The researcher of the present study was fortunate enough to work with the Central Forensic Science Lab (CFSL), New Delhi. She got an

opportunity to learn their method of forensic speaker identification. She also assisted the investigators at CFSL in solving a few criminal cases. The details of these cannot be discussed here because of the confidentiality issues. But, their method of speaker identification will be discussed here.

The investigators at CFSL also follow almost the same method as discussed above. But, there are cases where the script of the suspect sample is the exact copy of what has been said in the question sample. This is not a very good method of obtaining a suspect sample because the question sample may have statements which prove him guilty of an act. A suspect would hesitate to utter the same sentence. Most of the time, suspects refuse to give a sample because they feel that the investigator is asking him/her to utter that sentence to trap him. For example, if in a question sample the offender has asked for bribe, the suspect will hesitate to repeat that sentence. It is so, because the suspect is not aware about the process of speaker identification and he/she may think that police will try to accuse him/her by using his/her voice sample as an evidence in the court. Due to this fear, the suspect may also try to disguise his/her voice while giving a voice sample.

Lately, the CFSL team has started obtaining voice samples of the suspects in their spontaneous speech. After the samples are collected, they are analyzed on the basis of a number of auditory features.

According to, Paulisch, F., There are several methods of primary data collection, namely observation method, interview method, through questionnaires or schedules, etc. However, since the same data is to be collected twice, the observation method is not suitable for the current research. It is essential that the subjects are informed of the data collection needs and they read out the given text twice.

Though the research is done in the domain of forensic speaker identification, there is no crime scene, no question sample and, as a result, no suspect sample here. The purpose of this research work is to show the efficiency of the voice, resonance and manner features in speaker identification and not to prove if any suspect in any case is guilty or not. Therefore, the nature of data required for this research is a bit different from that which is used in forensic speaker identification in criminal cases.

Fitzgerald, M.E.; Ross, A.M. proposed that, Data collection was a tedious task for the researcher. She faced problems in convincing people to participate in her data elicitation process. There were also a few people who grew skeptical about their voices being recorded, on the other hand there were some who were excited on this very idea of their voice being recorded and used in a research work.

A common question that the researcher faced was "what will you do with my voice sample?" It was not an easy task for the researcher to satisfy the participants with an answer because the participants failed to understand what a spectrogram is. Most of them had probably no idea about formants, resonance, fundamental frequency, etc. The researcher tried to explain to them in a simplistic way, but they were keen to know the technicalities involved in a forensic speaker identification process. As a result, many people simply refused to participate in the research.

The researcher met more and more people and tried to convince them. Finally, she managed to collect data from 10 participants. The voice samples were to be recorded under controlled conditions. Hence, all the voice samples were recorded in a closed room with fans and mobile phones switched off.

According to Perepletchikov, M.; Ryan, C., Analysis, particularly in case of survey or experimental data, involves estimating the values of unknown parameters of the population and testing of hypotheses for drawing inferences. Analysis may, therefore, be categorised as descriptive analysis and inferential analysis. Descriptive analysis is largely the study of distributions of one variable. For example, in some linguistic studies, researchers have been interested in the variants of a morpheme or phoneme across age groups or their distribution between male and female speakers or so on. When research aims at finding only a specific variable or only one type of a variable among a group of speakers, among people of different age groups, social class, or any other group, the analysis is called unidimensional. Analysis is termed bivariate or multivariate when it is done in respect of two or more than two variables respectively.

Sometimes, research requires that variables are compared or correlated to figure out the amount of correlation between two or more variables. Such analysis is called correlation analysis. In some other cases, researchers look into the effect of a variable on other dependent variables. It is thus a study of functional relationships existing between two or more variables. This type of

analysis is termed Causal analysis. This analysis can be termed as regression analysis. Causal analysis is considered relatively more important in experimental researches, whereas in most social and business researches our interest lies in understanding and controlling relationships between variables then with determining causes per se and as such we consider correlation analysis as relatively more important.

Poshyvanyk, D.; Marcus, A., claims that, Analysis of speech samples for speaker identification is generally done at the word level. However, the data that was collected for the research consisted of many sentences in Hindi. So, there was a need for some software that could help slice sentences into disparate audio files. The software, namely Praat and Goldwave, have the option of clipping audio samples into small files. Therefore, any of these two would be used at the first stage of assessment. The words selected for comparison will be extracted from the sentences and will be saved as .WAV files. Next, Praat will be used for generating spectrograms of the extracted words and with the help of the software, voice, resonance and manner features like pitch, frequency, amplitude, duration, intensity, formant values and others will be measured. A comparison of these features, as the researcher believes, will help in identifying the relationship between the voice, resonance and manner features and this may have implications for speaker identification. In the next chapter, we have discussed the process of segmentation of sentences, words, sounds and their analysis using Praat in detail.

Geppert, B.; Mockus, Robler, researched that like every forensic speaker identification process, our first step was auditory analysis. As mentioned earlier, our primary focus is on acoustic analysis, therefore, we have made a very brief account of auditory analysis which is based on the parameters used by CFSL (Central Forensic Science Lab). These parameters do not include the socio-linguistic cues present in the voice of an individual, such as tone, accent etc. This is because though these features give us some knowledge about the regional or social background of the speaker, but these are articulatory features which can be easily modulated. These features also change with time. Also, in some cases, one can use a falsetto or disguised speech . and measure these outcomes. This result about the above case study shows a significant decrease in customer reported defects and in effort needed to make changes.

Chapter-3

Objective and Problem formulation

3.1 Problem Definition

Changeability is the main process used in software development to use the existing software system, resources and code to generate the new software system. Changeability actually reduces the user efforts, development cost and time. It actually affects the complete development process for all stages. But according to the changes performed in the software system, some new risks, faults can occur. In some situations, it is possible that the changeability add the overheads to the software system. Because of this there is the requirement of some measures that can estimate the effectiveness of changeability. It means to perform the cost and reliability estimation of changeability process. This estimation is performed for all kind of software system. In this work, a component based software system is defined to analyze the degree of changeability. This changeability estimation is performed under different metrics.

3.2 Objectives

The objectives associated with presented work are given here under

- The main objective of work is to perform the changeability estimation by performing the metrics based estimation for existing and changed software system.
- The aim of this work is to analyze the individual component as well as complete software system.
- The aim of this work is to identify the degree of changeability and the cost estimation over the software system.
- The aim of this work is to implement the metric based changeability analysis in matlab environment.

Chapter-4

PRESENT WORK

Today most of the software or applications are derived from the existing software systems. These existing software systems are defined in the form of components or the library. But when they are used in a new system, they require some changes or the configuration. This improvement to the existing system is called changeability. But, as an existing system is modified to form a new software system, it is required to identify the complexity of changeability. It means to identify the feasibility or the cost estimation to identify the changeability is effective for the software system. In this present work, metrics based estimation is defined to identify the degree of changeability. The metrics includes the individual component analysis, component interconnectivity analysis and the system level analysis. The analysis is here performed at three stages. In first stage, the existing system analysis is performed. In second stage, the changed software system is analyzed under same metrics. At the final stage, the difference analysis is performed between the existing and changed/modified software system. The work is here defined to identify the reliability and accuracy of this software system.

4.1 Research Design

The presented work is about to design a changeability analysis model under software metrics based estimation. The initial work is here to identify the component description of the existing software system. This description is here given in terms of component specification, component interaction specification and the communication within as well as outside the component specification. Once the input component system is defined, the next work is to perform the metrics based estimation for the software system. This estimation is performed under individual component specification, component interaction analysis and system analysis.

The presented system model is shown here under :

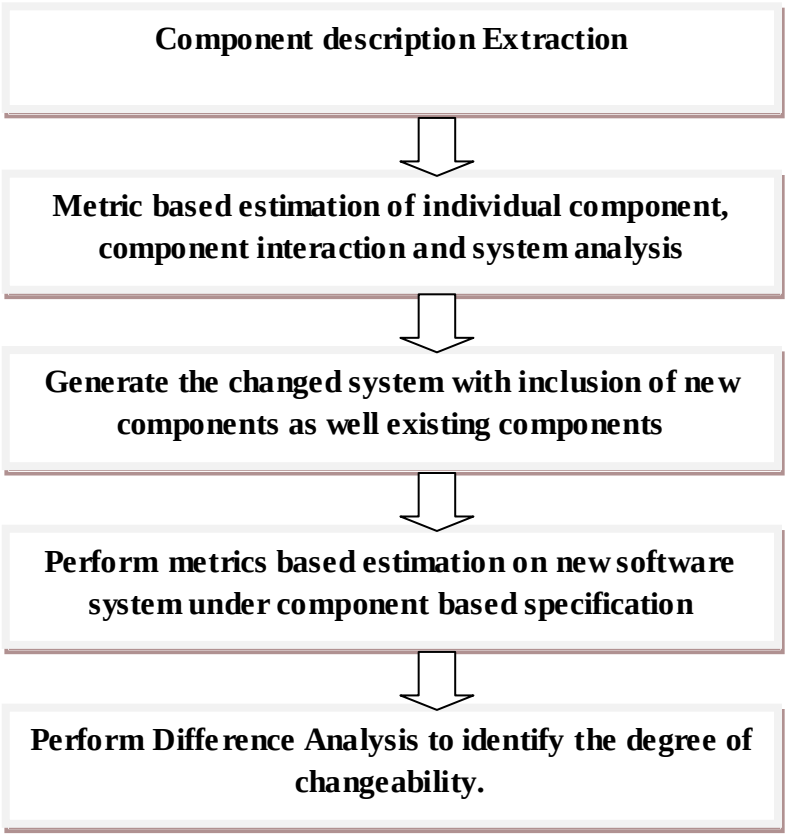

Figure 4.1 :- Flow of Work

Once the individual system metrics are evaluated the next work is to integrate the existing system with new system with the inclusion of existing system components as well as interaction with new components. Based on this new software system is defined. In next stage of work, the metrics based analysis on new software system is performed. Here the analysis is again performed in terms of individual component analysis, component interaction analysis and system analysis. At the final stage, the existing and new software system metrics difference analysis is performed to estimate the degree of changeability.

4.2 Methodology

4.2.1 Phase One

Consider we have a component based software system which consist of four components as shown in table 1 i.e A,B,C,D.

Table 1:- components complexity raw data

Component	M 1j	V 1j	A	A	B	B	C	C	D	D	Iimax	Oimax	Ec	Ii	Oi
I	M 1	V 1	Mvi j1	Mvi j2	Mvi j1	Mvi j2	Mvi j1	Mvi j2	Mvi j1	Mvi j2					
A	3	4	0	0	2	1	2	2	4	1	5	6	5	2	3
B	2	5	2	4	0	0	0	0	3	2	3	2	3	5	4
C	3	2	0	0	0	0	0	0	0	0	2	3	2	6	6
D	2	4	3	4	0	0	4	3	0	0	2	4	5	2	4

- 'M' is the no. of methods in each components.
- 'V' is the no. of instance variables in each components.
- 'Mvij1' is the no. of methods invoked by component Ci form component Cj.
- 'Mvij2' is the no. of variables invoked by component Cj form component Cj.
- 'Ii' Available no. of incoming interactions.
- 'Oi'Available no. of outgoing interactions.
- 'Iimax' is the maximum number of inputs in each component.
- 'Oimax' is the maximum number of outputs in each component.
- 'EC' is the set of pairs (v,m) for each variable V in the component that is used by method M.

The above raw data is used in calculation of the component specification, component interaction specification and the communication within as well as outside the component specification. In this we use coupling matrix, cohesion matrix and interface matrix.

4.2.1.1 Coupling Matrix

Coupling among components is defined as the total number of the other components that are to be coupled with this component.

In component based software system, we can defined coupling as : the two components or blocks coupled with each other if and only if out of these two components one component depend upon another component. Generally , two types of coupling is used :-

1) Afferent coupling

   We can define it as the actual no. of another components or blocks which depend upon the sub components or child components in the same component or it is an indication of the responsibility of block or component.

2) Effernt coupling

   We can define it as the actual no. of another components which are dependent upon the child components or sub component in the same component depend upon & it is an indication of the indepandance of the components

Now in order to generate coupling matrix, we start by taking any component of the CBSS. Let a software system that consists a group of components or group of blocks C:{c1,c2,…,cm}.Let Mj and Vj is the set of method and variables of Cj. MVji is a set of method & variable in component Ci called by component Cj.

MVj is a group which consists of methods & variables in another components.ie $1 \leq i \leq m$ and $i \neq j$:

$$MV_j = \frac{\left|\bigcup_{1 \leq i \leq m \cap i \neq j} MV_{j,i}\right|}{|M_j|+|V_j|} \qquad (1)$$

According to above equation, coupling metrix software system, as shown in the table given below: **Table 2:-** coupling metrix for comp.

Compont	Mj	V1j	A	A	B	B	C	C	D	D	MVj
I	M1	V1	Mvij1	Mvij2	Mvij1	Mvij2	Mvij1	Mvij2	Mvij1	Mvij2	
A	3	4	0	0	2	1	2	2	4	1	1.7143
B	2	5	2	4	0	0	0	0	3	2	1.5714
C	3	2	0	0	0	0	0	0	0	0	0
D	2	4	3	4	0	0	4	3	0	0	2.3333

Now, The coupling metrics(SCOP) for the whole system can be calculated by adding all values of MVj and division is performed by the total no. of components in the system by using the below given equation:

$$\mathbf{SCOP}=\frac{\sum_{j=1}^{m} MV_j}{M} \qquad (2)$$

Here, MVj = the individual coupling metrix i.e component j. and

M is the total no. of comp. in the component based software system.

So, according to equation no 2 , coupling metric for the system is:

$$\mathbf{SCOP = 1.4048}$$

4.2.1.2 Cohesion Matrix

Cohesion can be defined as the similarity of functions or methods in the component.It is a measure through which we can determine the different actions done by the comp. are linked with each other.

Consider a comp. J ie. There is a class which consist of a group of methods Mj(c)≡ $\{Mj1, Mj2, \ldots\ldots, Mjm\}$ and also a set of variables Vj(c) ≡ $\{Vj1, Vj2, \ldots\ldots Vjn\}$. Ej(c) is a set of pair of (Vj,Mj) to different variables V in the V(c) which can be used by function M in Mj(C).

Now cohesion metrix can also be termed as Cohesion of methods (COM) .

The equation for cohesion metrix is defined as :

$$\boldsymbol{COM_j(C)} = \frac{|\boldsymbol{E_j(C)}|}{|\boldsymbol{V_j(C)}| * |\boldsymbol{M_j(C)}|} \qquad (3)$$

According to the equation no. 3, we calculate cohesion metrix to every comp. in component based software system , as in the given table:

Table 3:- cohesion metrix for comp.

	M1j	V1j	EC	COMj
A	3	4	5	0.4167
B	2	5	3	0.3000
C	3	2	2	0.3333
D	2	4	5	0.6250

Now, The cohesion metrics(SCOH) for the whole system can be calculated by adding all values of COMj and division is performed by the number of components in the system by using the below given equation:

$$\mathbf{SCOH} = \frac{\sum_{j=1}^{m} COM_J}{m} \qquad \mathbf{(4\ )}$$

Here, COMj refers to the cohesion metric of component j.So, according to equation no 4, cohesion metric for the system is:

$$\mathbf{SCOH = 0.4188}$$

4.2. 1.3 Interface Matrix

Component Interface Metrix (CIM) can be defined in terms of Afferent and Efferent coupling. Component Interface Metrix can be defined in the form of:

- No. of inlet interactions available (II).
- No. of outlet interactions available (OI).
- The ratio of inlet to outlet, ie. II/OI.

There is other interaction metrix which is linked with a comp. is the Actual Interaction Metrix (AIM). AIM maps density of interface in a comp. It is the division of Actual no. of interaction to the potential ones:

$$\mathbf{AIM_j} = \frac{\mathbf{II_j + OI_j}}{\mathbf{II_{j,max} + OI_{j,max}}} \qquad (5)$$

Here, IIj,max is the max. number of inputs in comp. j & OIj, max is max. no. of outputs in component j. In this work we use the AIM instead of CIM.

Now According to equation no 5 , we get the interface metric for each component in component based software system , as shown in the table given below:

Table 4:- Interface metrix for comp.

Componet	M 1j	V 1j	A	A	B	B	C	C	D	D	Iimax	Oimax	I I	O I	AI Mj
I	M 1	V 1	Mvi j1	Mvi j2	Mvi j1	Mvi j2	Mvi j1	Mvi j2	Mvi j1	Mvi j2					
A	3	4	0	0	2	1	2	2	4	1	5	6	2	3	0.4545
B	2	5	2	4	0	0	0	0	3	2	3	2	5	4	1.8000
C	3	2	0	0	0	0	0	0	0	0	2	3	6	6	2.4000
D	2	4	3	4	0	0	4	3	0	0	2	4	2	4	1.0000

Now, Based on the actual intraction measurement for a comp, the system actual interface metrix (SAIM) is calculated. It is the combination of all interface metric of component and dividing it by number of components. It can be calculated by the equation :

$$\mathbf{SAIM} = \frac{\sum_{j=1}^{m} AIM_j}{M} \qquad \mathbf{(6)}$$

So, by the equation the whole system interface metric is:

$$\mathbf{SAIM = 1.4136}$$

4.2. 1.4 Sole System Complexity Matrix

To calculate the sole system complexity metrix (SSCM), we now integrating the above three component metrix (coupling metric, cohesion metric, interface metric) by assigining different weights to each metrix.

$$SSCM_j = \alpha * SCOP + \beta * SCOH + \gamma * SAIM \qquad (7)$$

Where α ,β and γ is the weights for the three metrics (coupling, cohesion and interface metrics) for component j respectively. But there is a condition that $\alpha + \beta + \gamma = 1$.

If we provide values to the coupling, cohesion and interface metric like 0.5, 0.2, 0.3 then by using equation no.7, we will get the sole component complexity metrics as :

SSCM = 1.2102

4.2. 1.5 Sole Component Complexity Matrix

To calculate the sole component complexity metrix (SCCM), we now integrating the above three component metrix (coupling metric, cohesion metric, interface metric) by assigining different weights to each metrix.

$$\boldsymbol{SCCM_j = \alpha * MV_j + \beta * COM_j + \gamma * AIM_j} \qquad \textbf{(8)}$$

Where α ,β and γ is the weights for the three metrics (coupling, cohesion and interface metrics) for component j respectively. But there is a condition that $\alpha + \beta + \gamma = 1$.

If we provide values to the sole component coupling, cohesion and interface metric like 0.5, 0.2, 0.3 then by using equation no.8, we will get the sole component complexity metrics as:

Table 5:- complexity metrix for comp.

Component	MVj	COMj	AIMj	SCCMj
A	1.7143	0.4167	0.4545	1.0768
B	1.5714	0.3000	1.8000	1.3857
C	0	0.3333	2.4000	0.7867
D	2.3333	0.6250	1.0000	1.5917

From the table 5, it is clearly seen that the component D has heighest complexity metrix so it needs much investment i.e time/cost whereas the component C has lowest complexity metrix so it needs less investment.

4.2.2 Phase Two

Here, we consider a new component based system in which we use the two components (comp A and comp B) from the previous table and rest are new components.The detailed description of the components taken in this phase are given below ;-

Table 6:- components complexity raw data for new system

Comp	M1j	V1j	A	A	B	B	C	C	D	D	Iim Ax	Oi max	EC	II	OI
I	M1	V1	MV Ij1	MV Ij2	MV Ij1	MV Ij2	MV Ij1	MV Ij2	MV Ij1	MV Ij2					
A	3	4	0	0	2	1	2	2	4	1	5	6	5	2	3
B	2	5	2	4	0	0	0	0	3	2	3	2	3	5	4
C	1	5	1	0	0	0	0	0	0	0	2	3	2	6	6
D	2	4	1	2	2	2	2	2	2	2	2	4	5	2	4
E	3	3	1	2	2	2	2	2	2	2	2	4	5	8	7
F	1	2	1	2	2	2	2	2	2	2	2	3	6	7	5

- ‘M’ is the no. of methods in each components.
- ‘V’ is the no. of instance variables in each components.
- ‘Mvij1’ is the no. of methods invoked by component Ci form component Cj.
- ‘Mvij2’ is the no. of variables invoked by component Cj form component Cj.
- ‘Ii’ Available no. of incoming interactions.
- ‘Oi’Available no. of outgoing interactions.
- ‘Iimax’ is the maximum number of inputs in each component.
- ‘Oimax’ is the maximum number of outputs in each component.
- ‘EC’ is the set of pairs(v,m) for each variable V in the component that is used by method M.

The above raw data is used in calculation of the component specification, component interaction specification and the communication within as well as outside the component specification for new system. In this we use coupling matrix, cohesion matrix and interface matrix similar to the phase one.

4.2.2.1 Coupling Matrix

Coupling metrix for this component based software system can be calculated similar to the phase one by using equation number (1).

$$\mathbf{MV_{j1}} = \frac{\left|\cup_{\mathbf{1 \le i \le m \cap i \ne j}} \mathbf{MV_{j,i}}\right|}{\left|\mathbf{M_j}\right| + \left|\mathbf{V_j}\right|}$$

According to this, coupling metrix to every comp. in this system is as in the table given below:-

Table 7:- coupling metrix for new comp.

Comp	M1j	V1j	A	A	B	B	C	C	D	D	MVj1
I	M1	V1	MV Ij1	MV Ij2	MV Ij1	MV Ij2	MV Ij1	MV Ij2	MV Ij1	MV Ij2	
A	3	4	0	0	2	1	2	2	4	1	0
B	2	5	2	4	0	0	0	0	3	2	2.3333
C	1	5	1	0	0	0	0	0	0	0	0
D	2	4	1	2	2	2	2	2	2	2	0
E	3	3	1	2	2	2	2	2	2	2	0
F	1	2	1	2	2	2	2	2	2	2	0

Now, Similar to the phase one, the coupling metrics(SCOP1) for the whole system can be calculated by adding all values of MVj & division is performed with the total no. of comp. in the system by using equation no (2) :-

$$\mathbf{SCOP1} = \frac{\sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{MV_j 1}}{\mathbf{M}}$$

So, according to the above equation , coupling metric for the system is:-

$$\mathbf{SCOP1 = 0.9365}$$

4.2. 2.2 Cohesion Matrix

Cohesion metrix for this component based software system can be calculated similar to the phase one by using equation number (3) :-

$$\boldsymbol{COM_j 1(C) = \frac{|E_j(C)|}{|V_j(C)| * |M_j(C)|}}$$

According to this, we provide cohesion metrix with every comp. in this system as in the table given below:-

Table 8:- cohesion metrix for new comp.

Comp	M1j	V1j	EC	COMj1
I	M1	V1		
A	3	4	5	0
B	2	5	3	0
C	1	5	2	0
D	2	4	5	0.6250
E	3	3	5	0
F	1	2	6	0

Now, Similar to the phase one, the cohesion metrics(SCOH1) for the whole system can be calculated by adding total values of COMj & division is performed with the no. of comp. in the system by using the equation no (4) :-

$$\mathbf{SCOH} = \frac{\sum_{j=1}^{m} \boldsymbol{COM_J}}{\boldsymbol{m}}$$

So, according to this equation, cohesion metric for the system is :-

$$\mathbf{SCOH1 = 0.3833}$$

4.2. 2.3 Interface Matrix

Interface metrix for this component based software system can be calculated similar to the phase one by using equation number (5) :-

$$\mathbf{AIM_j 1} = \frac{\mathbf{II_j + OI_j}}{\mathbf{II_{j,max} + OI_{j,max}}}$$

According to this, we get the interface metrix for each component in this system as shown in the table given below :-

Table 9:- Interface metrix for new comp.

Comp	M1j	V1j	A	A	B	B	C	C	D	D	Iimax	Oi Max	II	OI	AIMj1
I	M1	V1	MV Ij1	MV Ij2	MV Ij1	MV Ij2	MV Ij1	MV Ij2	MV Ij1	MV Ij2					
A	3	4	0	0	2	1	2	2	4	1	5	6	2	3	0
B	2	5	2	4	0	0	0	0	3	2	3	2	5	4	0
C	1	5	1	0	0	0	0	0	0	0	2	3	6	6	0
D	2	4	1	2	2	2	2	2	2	2	2	4	2	4	1
E	3	3	1	2	2	2	2	2	2	2	2	4	8	7	0
F	1	2	1	2	2	2	2	2	2	2	2	3	7	5	0

Now, Based on the actual intraction measurement for a component ,the system actual interface metrix (SAIM1) is calculated. It is the integration of all the interface metric of component and dividing it by number of components. It can be calculated by the equation no (6) :

$$\mathbf{SAIM1} = \frac{\sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{AIM_j}}{\mathbf{M}}$$

So, by this equation the whole system interface metric is :-

$$\mathbf{SAIM1 = 1.1091}$$

4.2. 2.4 Sole System Complexity Matrix

To calculate the sole system complexity metrix (SSCM1), we now integrating the above three component metrix (coupling metric, cohesion metric, interface metric) by assigining different weights to each metrix similar to phase one by using equation no (7):-

$$SSCM_j = \alpha * SCOP + \beta * SCOH + \gamma * SAIM$$

Where α ,β *and* γ is the weights for the three metrics (coupling, cohesion and interface metrics) for component j respectively. But there is a condition that $\alpha + \beta + \gamma = 1$.

If we provide values to the sole system coupling, cohesion and interface metric like 0.5, 0.2, 0.3 then by using equation no.7, we will get the sole component complexity metrics as:

SSCM1 = 0.8776

4.2. 2.5 Sole Component Complexity Matrix

To calculate the sole component complexity metrix (SCCM1), we now integrating the above three component metrix (coupling metric, cohesion metric, interface metric) by assigining different weights to each metrix similar to phase one by using equation no (8).

$$SCCM_j 1 = \alpha * MV_j + \beta * COM_j + \gamma * AIM_j$$

Where α ,β *and* γ is the weights for the three metrics (coupling, cohesion and interface metrics) for component j respectively. But there is a condition that $\alpha + \beta + \gamma = 1$.

If we provide values to the sole component coupling, cohesion and interface metric like 0.5, 0.2, 0.3 then by using above equation, we will get the sole component complexity metrics as:

Table 10:- complexity metrix for new comp.

Comp	MVj1	COMj1	AIMj1	SCCMj1
A	0	0	0	0
B	2.3333	0	0	0
C	0	0	0	0
D	0	0.6250	1	0.6250
E	0	0	0	0
F	0	0	0	0

From the table 10, it is clearly seen that the component D has heighest complexity metrix so it 1needs much investment i.e time/cost whereas the other components have zero complexity metrix so they needs less or negligible investment.

4.2.3 Phase Third

In this phase, we will calculate the updation cost for the changed component based system and on the basis of these updation cost, we will calculate the changeability of the system.In this case we use only two components.

4.2.3.1 Coupling cost metrics

Coupling cost for the system(SCMj) can be calculated by subtracting the coupling metrix ((MVj)calculated in phase one) from the coupling metrix ((MVj1)calculated in phase two).

$$SCM_j = MV_j1 - MV_j \quad (9)$$

Here, MVj1 is the coupling metrix calculated in phase two.

MVj is the coupling metrix calculated in phase one.

Now, According to the equation no. (9) , we get the coupling cost metrix for the system as shown in the table given below :-

Table 11:- coupling cost metrics for component

Component	SCMj
A	-1.7143
B	0.7619

From the table, it is clearly seen that the couling cost for the component A is negative while component B is slightly positive. So, There is no need of changability for the component A while only component B needs small **changability**.

4.2.3.2 Cohesion cost metrics

Cohesion cost for the system(SCOMj) can be calculated by subtracting the cohesion metrix ((COMj)calculated in phase one) from the cohesion metrix ((COMj1)calculated in phase two).

$$SCOM_j = COM_j 1 - COM_j \quad (10)$$

Here, COMj1 is the cohesion metrix calculated in phase two.

COMj is the cohesion metrix calculated in phase one.

Now, According to the equation no. (10) , we get the coupling cost metrix for the system as shown in the table given below :-

Table 12:- cohesion cost metrics for component

Component	SCOMj
A	-0.4167
B	-0.3000

From the table, it is clearly seen that the cohesion cost for both the component A and component B is negative . So, There is no need of **changability** at all.

4.2.3.3 Interface cost metrics

Interface cost for the system(SICMj) can be calculated by subtracting the interface metrix ((AIMj)calculated in phase one) from the interface metrix ((AIMj1)calculated in phase two).

$$SICM_j = AIM_j 1 - AIM_j \quad (11)$$

Here, AIMj1 is the interface metrix calculated in phase two.

AIMj is the interface metrix calculated in phase one.

Now, According to the equation no. (11) , we get the interface cost metrix for the system as shown in the table given below :-

Table 13:- interface cost metrics for component

Component	SICMj
A	-0.4545
B	-1.8000

From the table, it is clearly seen that the interface cost for both the component A and component B is negative . So, There is no need of **changability** at all.

4.2.3.3 Sole component cost metrics

Sole component cost for the system(SCMj) can be calculated by subtracting the sole component complexity metrix ((SCCMj)calculated in phase one) from the sole component complexity metrix ((SCCMj1)calculated in phase two).

$$SCM_j = SCCM_j 1 - SCCM_j \qquad (12)$$

Here, SCCMj1 is the sole component complexity metrix calculated in phase two.

SCCMj is the sole component complexity metrix calculated in phase one.

Now, According to the equation no. (12) , we get the interface cost metrix for the system as shown in the table no. geven below :-

Table 13:- sole component cost metrics for component

Component	SCMj
A	-1.0768
B	-1.3857

From the table, it is clearly seen that the interface cost for both the component A and component B is negative . So, There is no need of **changability** at all.

Chapter-5

RESULTS

The results for the whole work done in this research is shown individually for defferent types of complexities. It is shown here in the form of graphs.

5.1 Complexities for existing system

5.1.1 Element Communication Analyasis

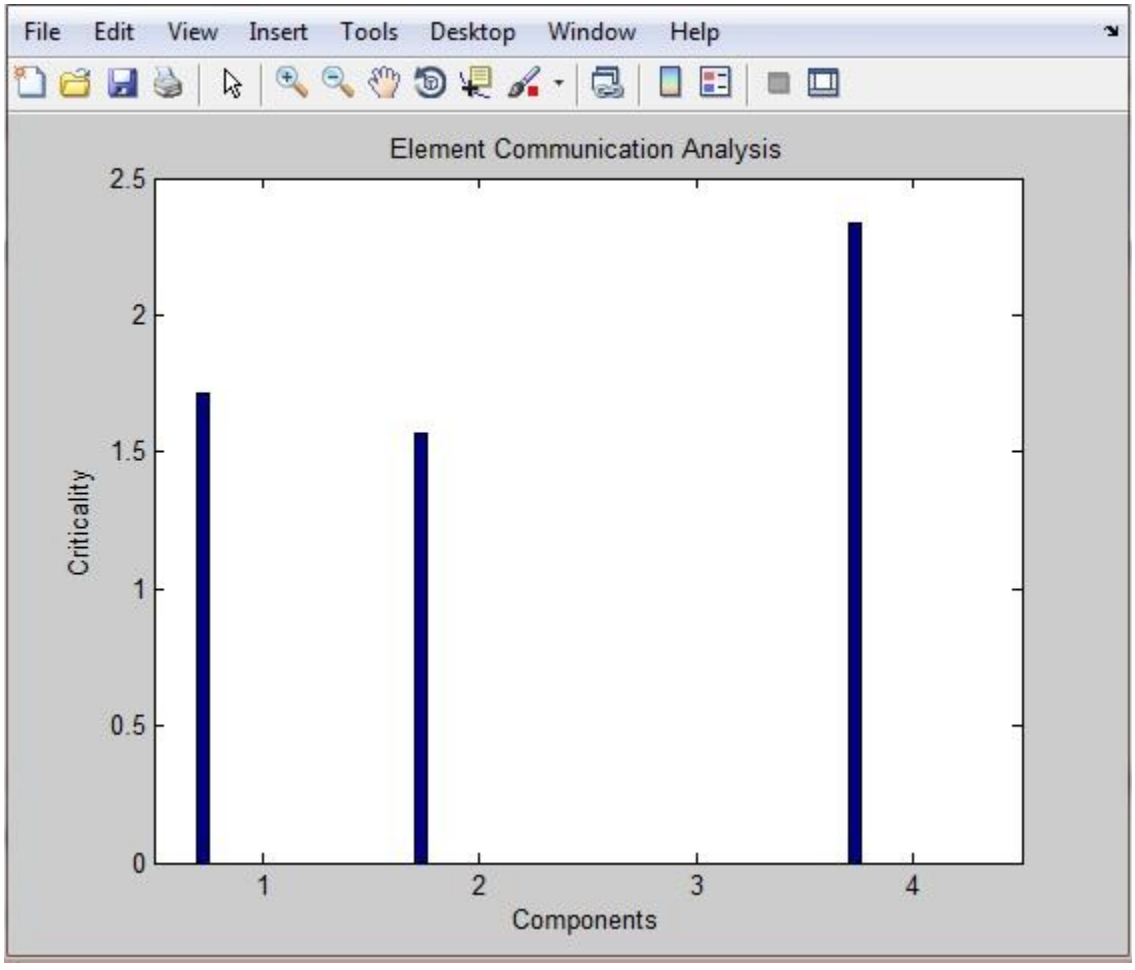

Fig. 5.1.1

5.1.2 Internal Element Analyasis

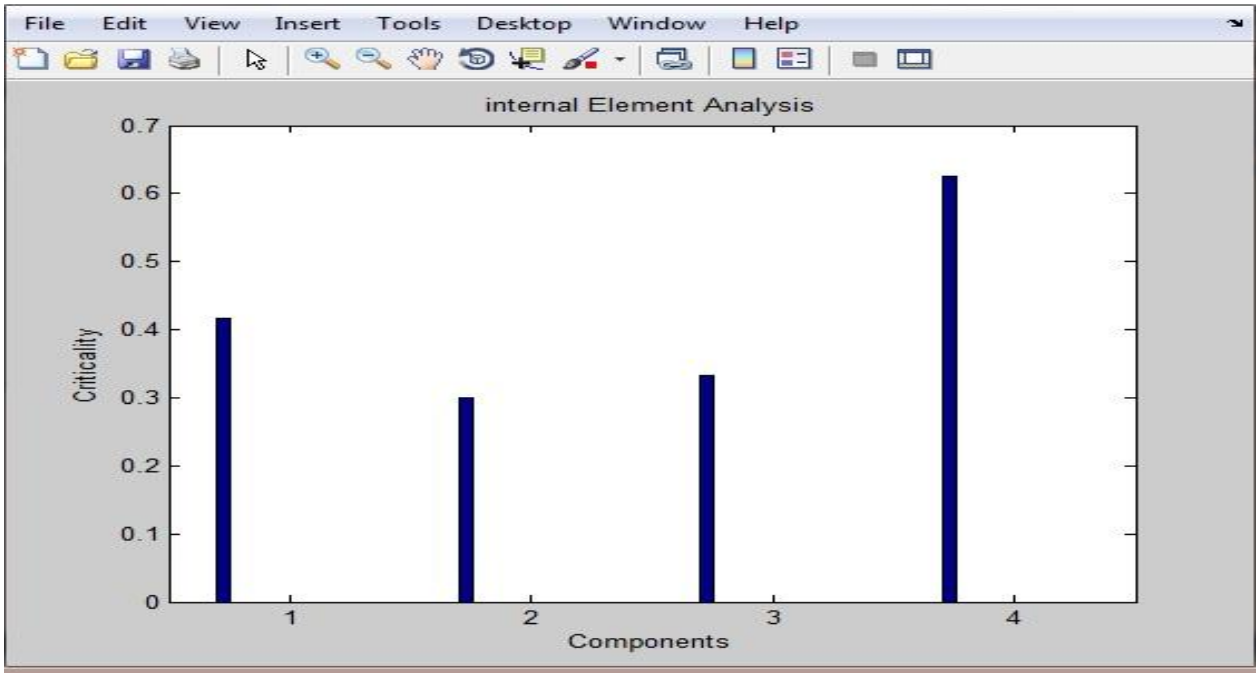

Fig. 5.1.2

5.1.3 Interfacing Analyasis

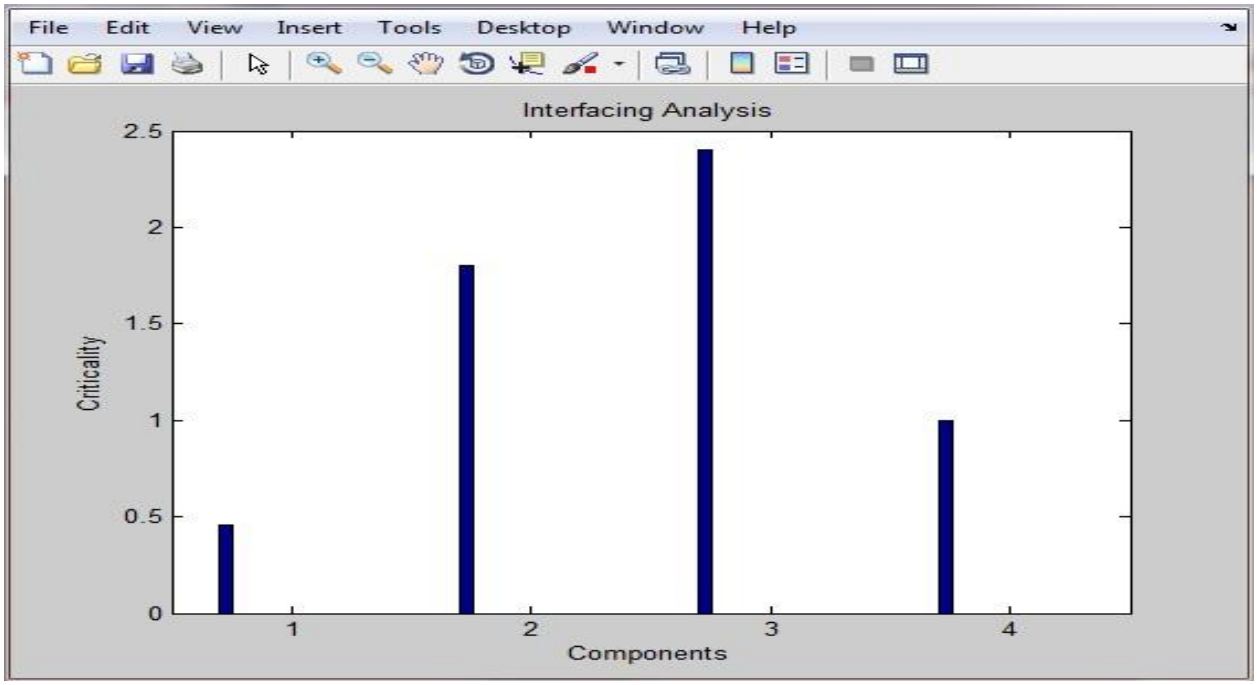

Fig. 5.1.3

5.1.4 Whole System Analyasis

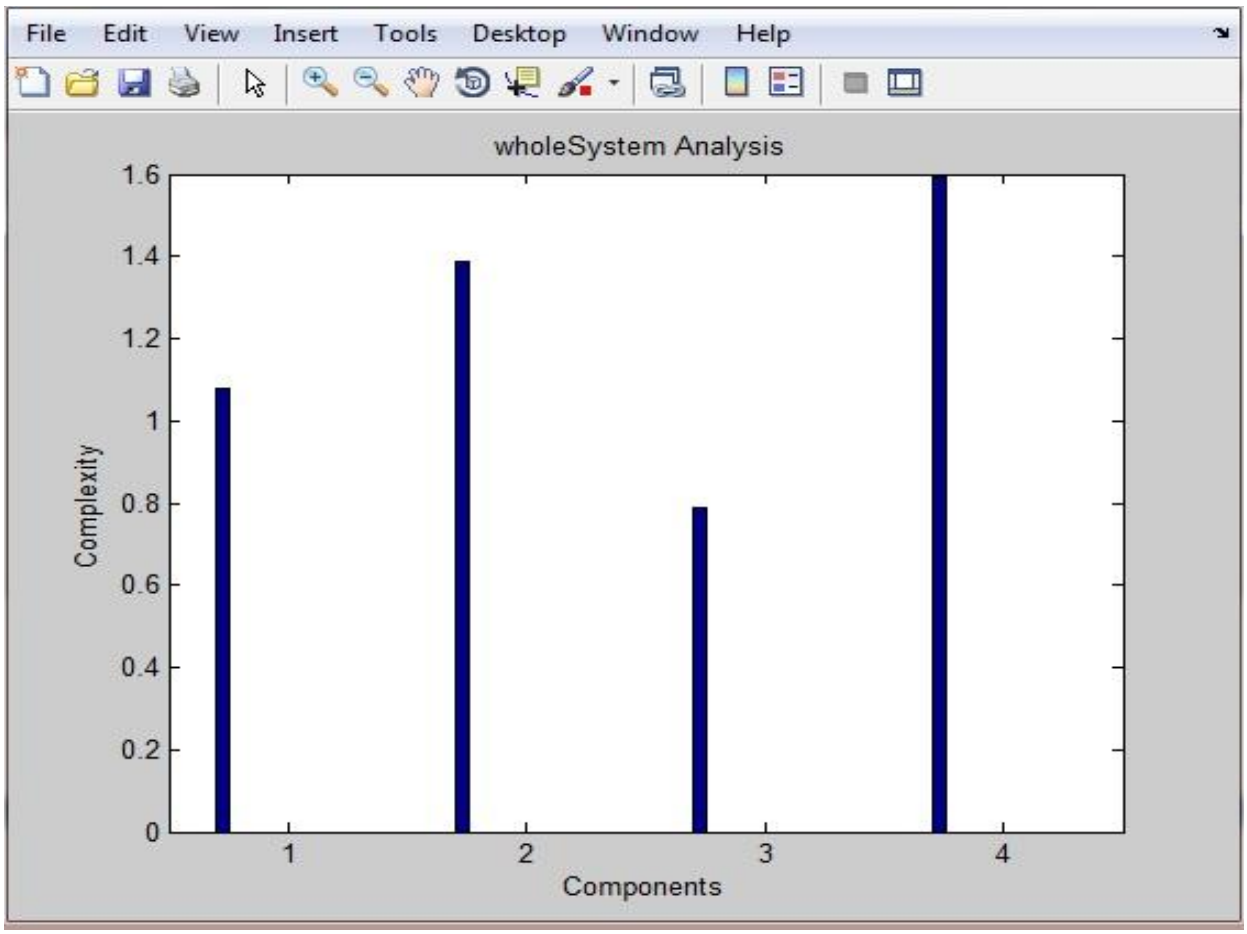

Fig. 5.1.4

5.2 Complexities for changed system

5.2.1 Element Communication Analyasis

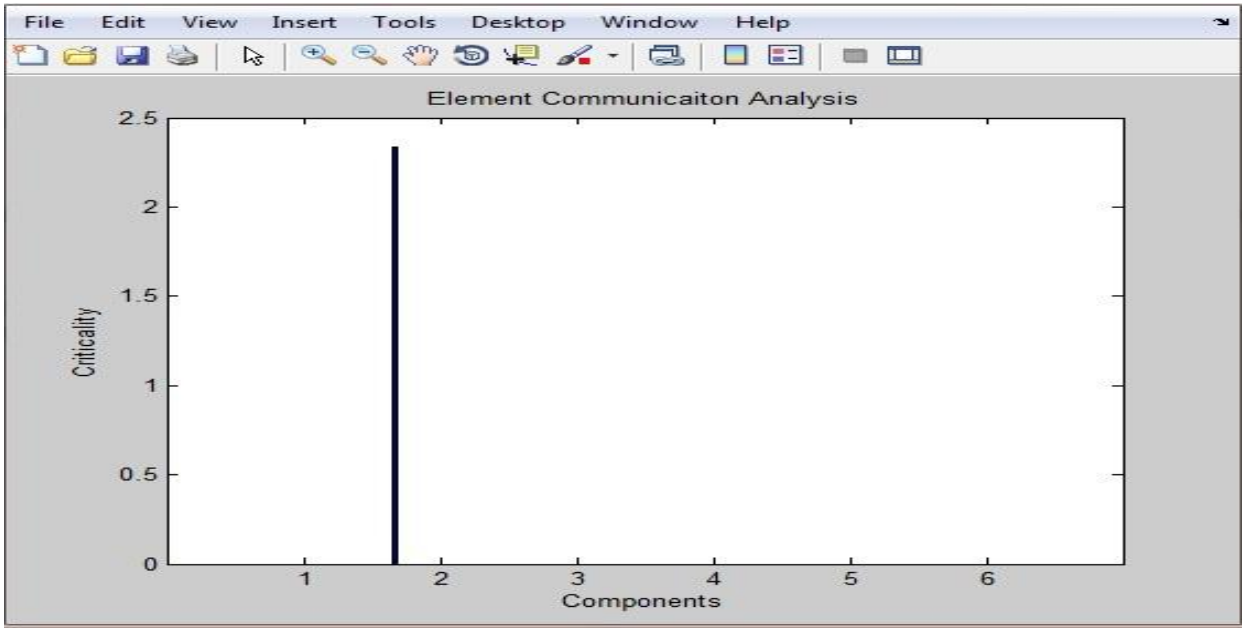

Fig. 5.2.1

5.2.2 Internal Element Analyasis

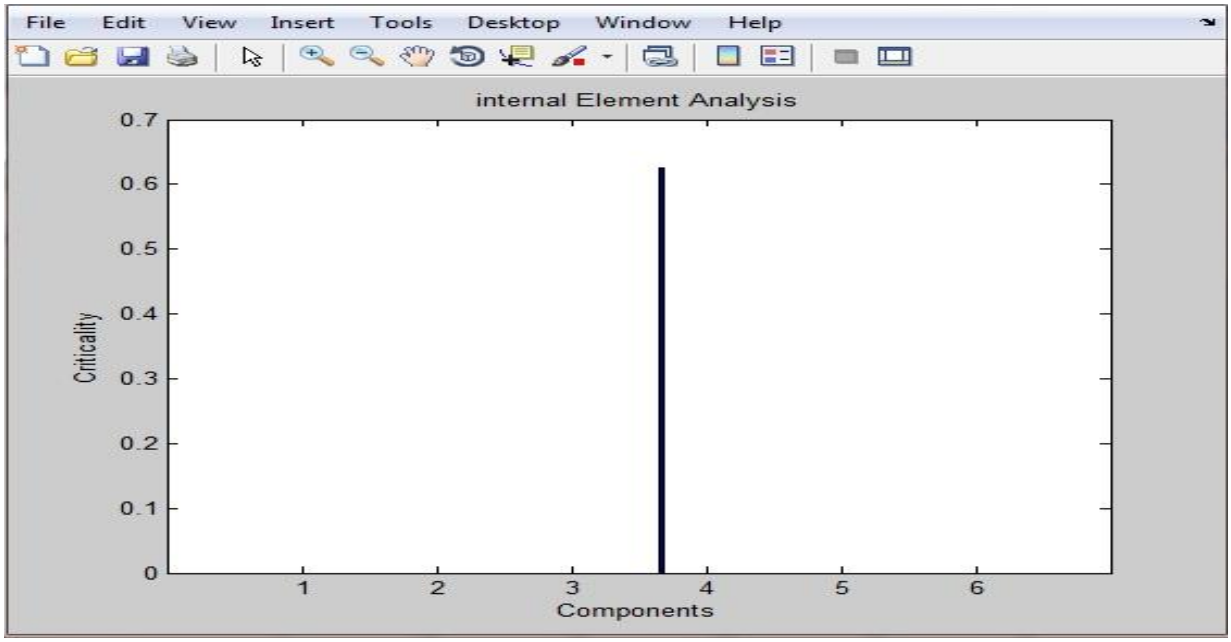

Fig. 5.2.2

5.2.3 Interfacing Analyasis

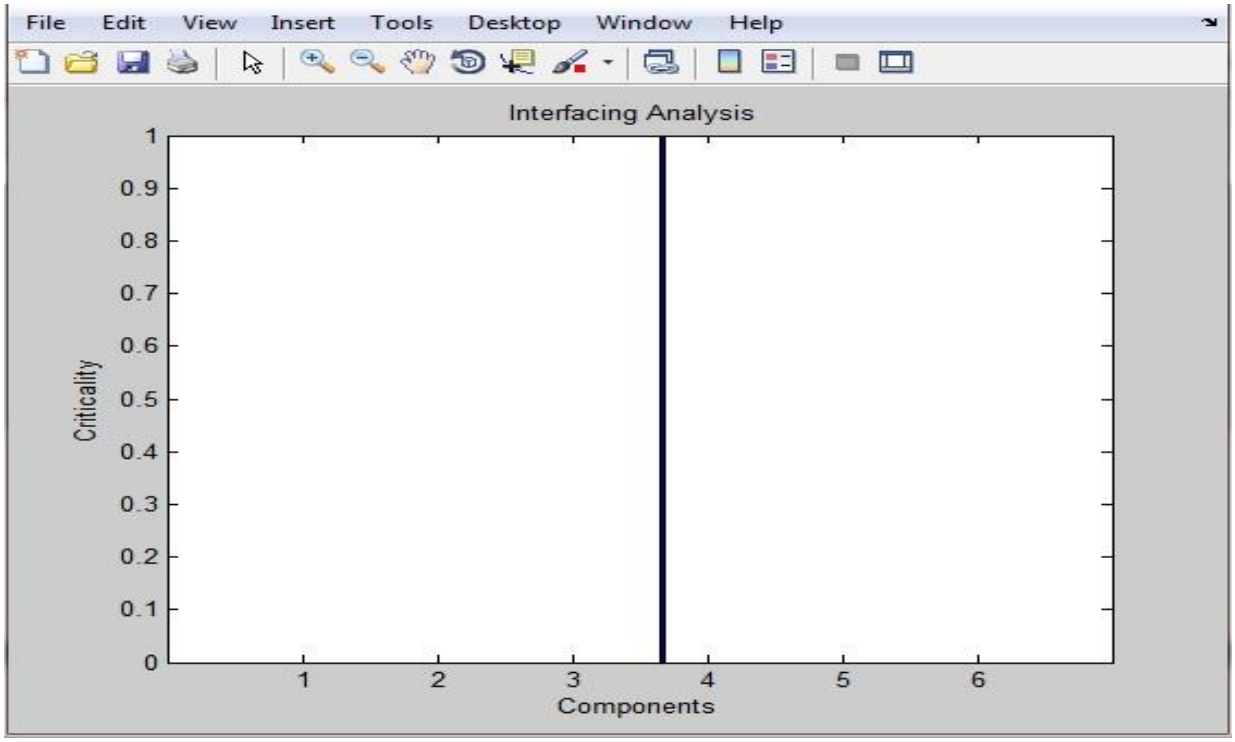

Fig. 5.2.3

5.2.3 Whole System Analyasis

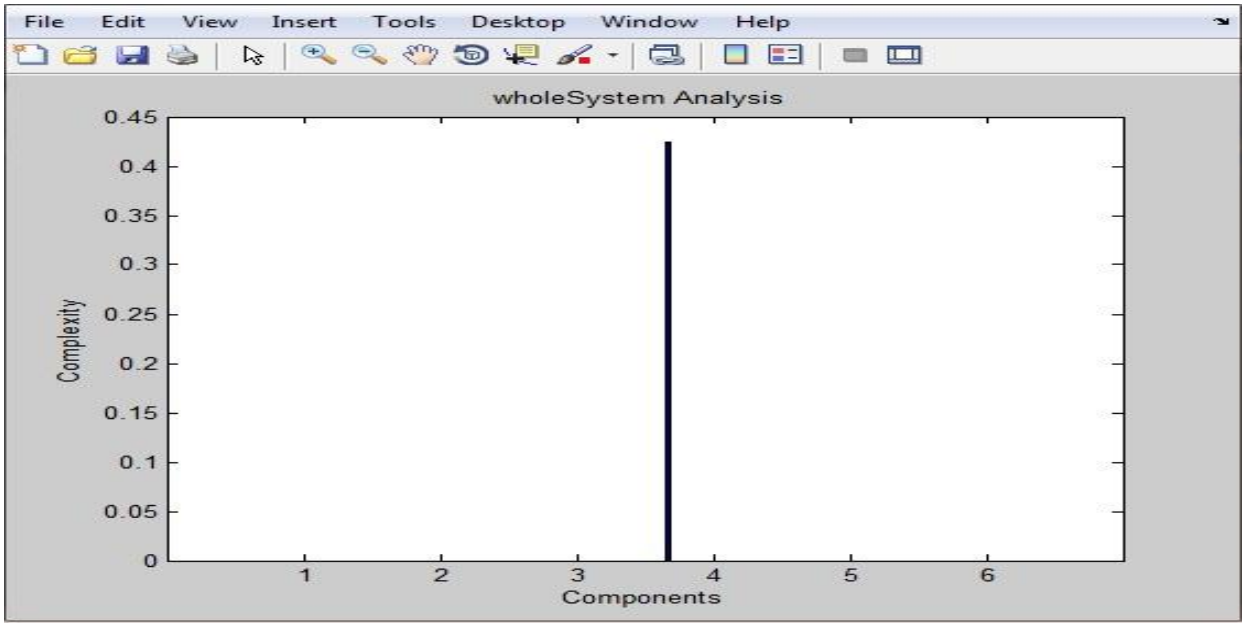

Fig. 5.2.4

5.3 Changeability Indicator

5.3.1 Module Interfacing Analyasis

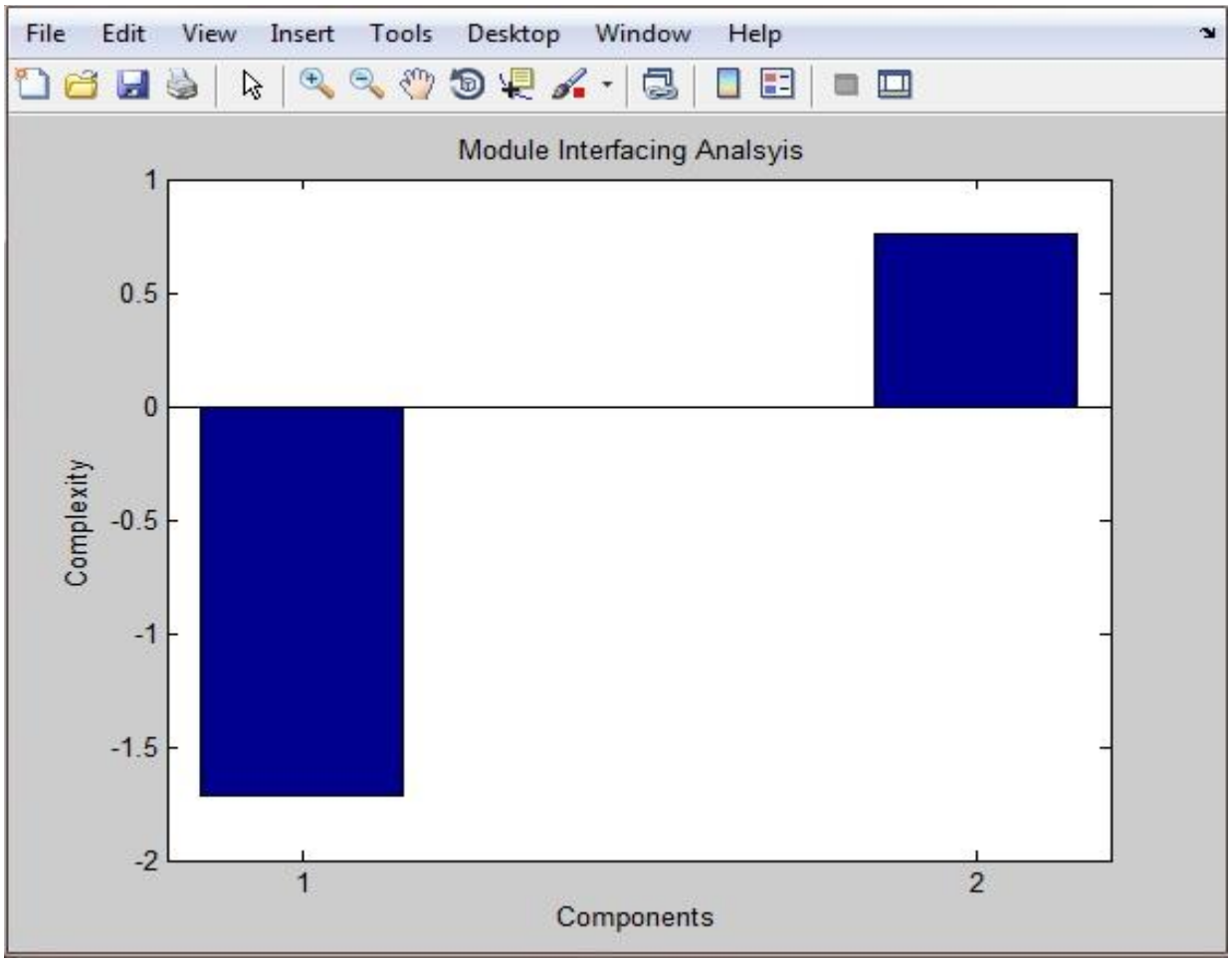

Fig.5.3.1

From the figure 5.1, it is clearly seen that module interfacing analyais is negative for component A. So, there is no need for changeability or negligible **changeabilty** required for component A But there is positive module interfacing for component B. So, there needs some **changability**.

5.3.2 Individual Component Analyasis

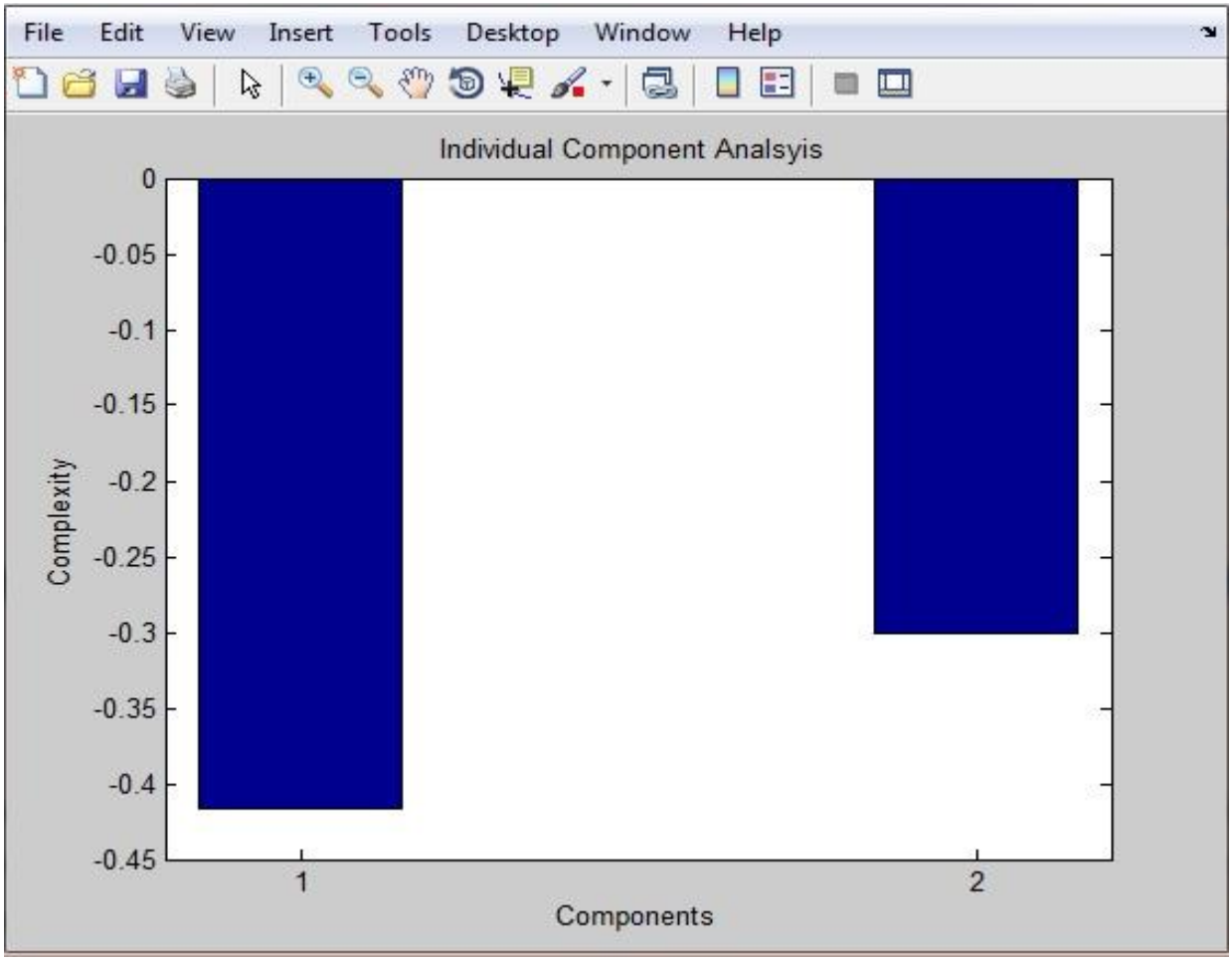

Fig. 5.3.2

From the figure 5.2, it is clearly seen that Individual component analyasis is negative for both the components. So, there is no need for changeability or negligible **changeabilty** required for at all.

5.3.3 Component Interfacing Analyasis

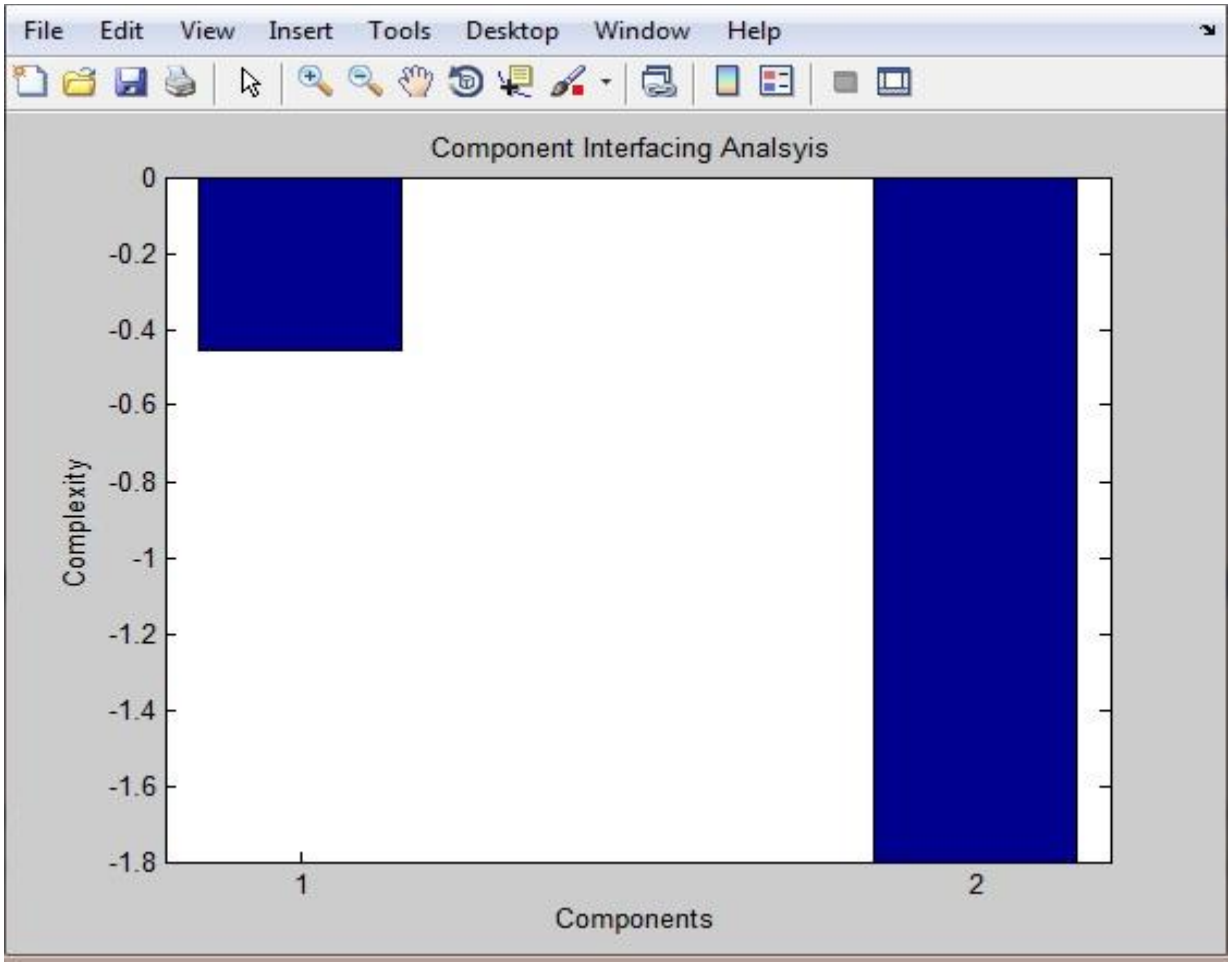

Fig. 5.3.3

From the figure 5.3, it is clearly seen that Individual component analyasis is negative for both the components. So, there is no need for changeability or negligible **changeabilty** required for at all.

5.3.4 System Criticality Analyasis

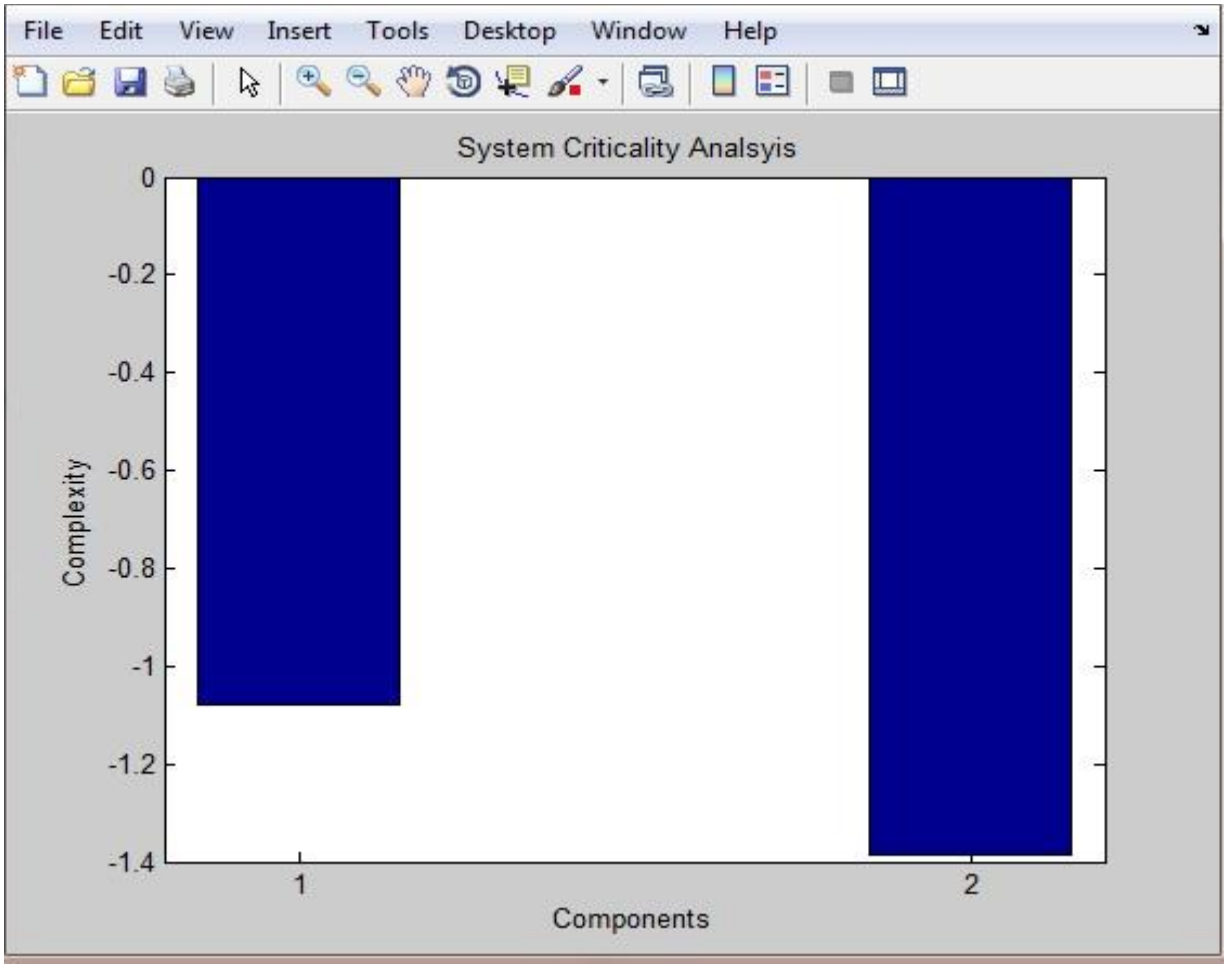

Fig. 5.3.4

From the figure 5.4, it is clearly seen that Individual component analyasis is negative for both the components. So, there is no need for changeability or negligible **changeabilty** required for at all.

5.4 Significance of work

The significance of the presented work are3 listed as under

- In this present work, the metric based analysis is performed so that the quantitative results are obtained.
- The word is able to identify the cost and reliability estimation of individual component as well as comleted system.
- The work is able to identify the relation analysis over the software system.

5.5 Motivation

In order to understand the importance of the present work we first need to understand the significance of software metrics. We are aware that no software can be presented as a system till it is not verified. There are lots of metrics which are available but no one provide such a suit which is capable to collect some metrics to perform an automated software measurement. The presented work is about to analyze a software system based on object oriented matrices. These matrices include the individual component analysis, component interaction analysis and system based analysis.

5.6 Future Scope

In future, the research work may be considered for also calculating the risk associated with proneness to change using metrics which can calculate severty,occurrence or the frequency of the possible changes and timely detection for corrective measure which would lead to regain of stability and maturity of the software development. We can also implement model FMECA (failure mode efforts and criticality analysis), for tracking the relationship between change proneness of the software to the total estimated risk.

5.7 Limitations

This research work is only limited to java programs. Also as all the actual values which we have introduce in labeling are taken from human analysis which may lead to any human error and human bias.

Chapter-6

REFRENCES

[1] Ingram, C.; Riddle, S., "Using early stage project data to predict change-proneness," *Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on* , vol., no., pp.42,48, 3-3 June 2012

[2] Sebastian G. Elbaum John C. Munson,; "Code Churn: A Measure for Estimating the Impact of Code Change".

[3] The Challenges Of Software Change Management In Today's Siloed IT Organizations, A Commissioned Study Conducted By Forrester Consulting On Behalf Of Serena Software, November 2006

[4] N. Fenton, S. L. Pfleeger and R. Glass, "Science and Substance, A Challenge to Software Engineers", IEEE Software, July 1994, pp.86-95.

[5] D. German, A. Hindle, and N. Jordan. Visualizing the evolution of software using softChange. In *Proc. of the 16th Internation Conference on Software Engineering and Knowledge Engineering (SEKE 2004)*, pages 336–341, 2004.

[6] Nary Subramanian, Lawrence Chung, "Metrics for Software Adaptability" 2Department of Computer Science University of Texas, Dallas Richardson, TX, USA,

[7] Gentzane Aldekoa1, Salvador Trujillo2, Goiuria Sagarduil, Oscar Díaz2"Experience measuring maintainability in software product lines"; XV Jornadas de Ingeniería del Software y B.ases de Datos JISBD 2006

[8] Chidamber, S. R. and Kemerer, C. K. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, Vol. 20 (June 1994), pp.476-493.

[9] Ajrnal Chaumun, M.; Kabaili, H.; Keller, R.K.; Lustman, F., "A change impact model for changeability assessment in object-oriented software systems," *Software Maintenance and Reengineering, 1999. Proceedings of the Third European Conference on*, vol., no., pp.130,138, 1999

[10] J. C. Munson, S. G. Elbaum, and R. M. Karcich, "Software Risk Assessment Through Software Measurement and Modeling", will appear *Proceedings of the 1998 IEEE Aerospace Applications Conference*, IEEE Computer Society Press.

[11] en.wikipedia.org/wiki/Cluster_analysis

[12] Jukka Kainulainen "Clustering Algorithms: Basics and Visualization"HELSINKI UNIVERSITY OF TECHNOLOGY Laboratory of Computer and Information Science T-61.195 Special Assignment 1

[13] Jon Kleinberg;" A Theorem for Clustering", Department of Computer Science Cornell University

[15] MA Chaumun, RK Keller, F Lustman - Advances in software engineering, 2002 - dl.acm.org

[16] Bieman, J.M.; Andrews, A.A.; Yang, H.J., "Understanding change-proneness in OO software through visualization," Program Comprehension, 2003. 11th IEEE International Workshop on, vol., no., pp.44,53, 10-11 May 2003

[17] Kabaili, H.; Keller, R.K.; Lustman, F., "Cohesion as changeability indicator in object-oriented systems," Software Maintenance and Reengineering, 2001. Fifth European Conference on , vol., no., pp.39,46, 2001

[18] Adam M. Ross, Donna H. Rhodes, Daniel E. Hastings, Volume 11, Issue 3, pages 246–262, Autumn (Fall) 2008

[19] 2007Changeability in operations: A critical strategic resource for European manufacturing " Universities Engineering Conference, 2007. UPEC 2007. 42nd International , vol., no., pp.930,937, 4-6 Sept.

[20] Monden, A.; Nakae, D.; Kamiya, T.; Sato, S.; Matsumoto, K.-i., "Software quality analysis by code clones in industrial legacy software," *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on* , vol., no., pp.87,94, 2002

[21] Geppert, B.; Mockus, A.; Robler, F., "Refactoring for changeability: a way to go?," *Software Metrics, 2005. 11th IEEE International Symposium* , vol., no., pp.10 pp.,13, 1-1 Sept. 2005

[22] Poshyvanyk, D.; Marcus, A., "The Conceptual Coupling Metrics for Object-Oriented Systems," *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on* , vol., no., pp.469,478, 24-27 Sept. 2006

[23] http://www.performancezoom.com/per

[24] Bissonnette, Victor L. "Statistical Tables. " Victor Bissonnette'sHome Page. 23 Mar. 2004.Dept. of Psychology, Berry College. 23Oct. 2007 Jamieson, Susan. "Likert Scales:How to (Ab)Use Them." Medical Education 38 (2004): 1217-1218.

[25] http://satc.gsfc.nasa.gov/metrics/codemetrics/oo/thresholds/index.html

[26] http://www.ingrid.org/jajakarta/turbine/en/turbine/maven/reference/metrics.html

[27] http://www.objectmentor.com

[28] http://www.objectmentor.com/publications/dip.pdf

[29] Rahmat Widia Sembiring & Jasni Mohamad Zain;" The Design of Pre-Processing Multidimensional Data Based on Component Analysis", Faculty of Computer System and Software Engineering, Universiti Malaysia Pahang Lebuhraya Tun Razak, 26300, Kuantan, Pahang Darul Makmur, Malaysia