Compatibility Measurement of Components in Component Based Software Engineering Using Boltzmann Learning Technique

A Dissertation Submitted

By

Jasneet Kaur

11301523

To

Department of Science & Technology

In partial fulfillment of the Requirement for the

Award of the Degree of

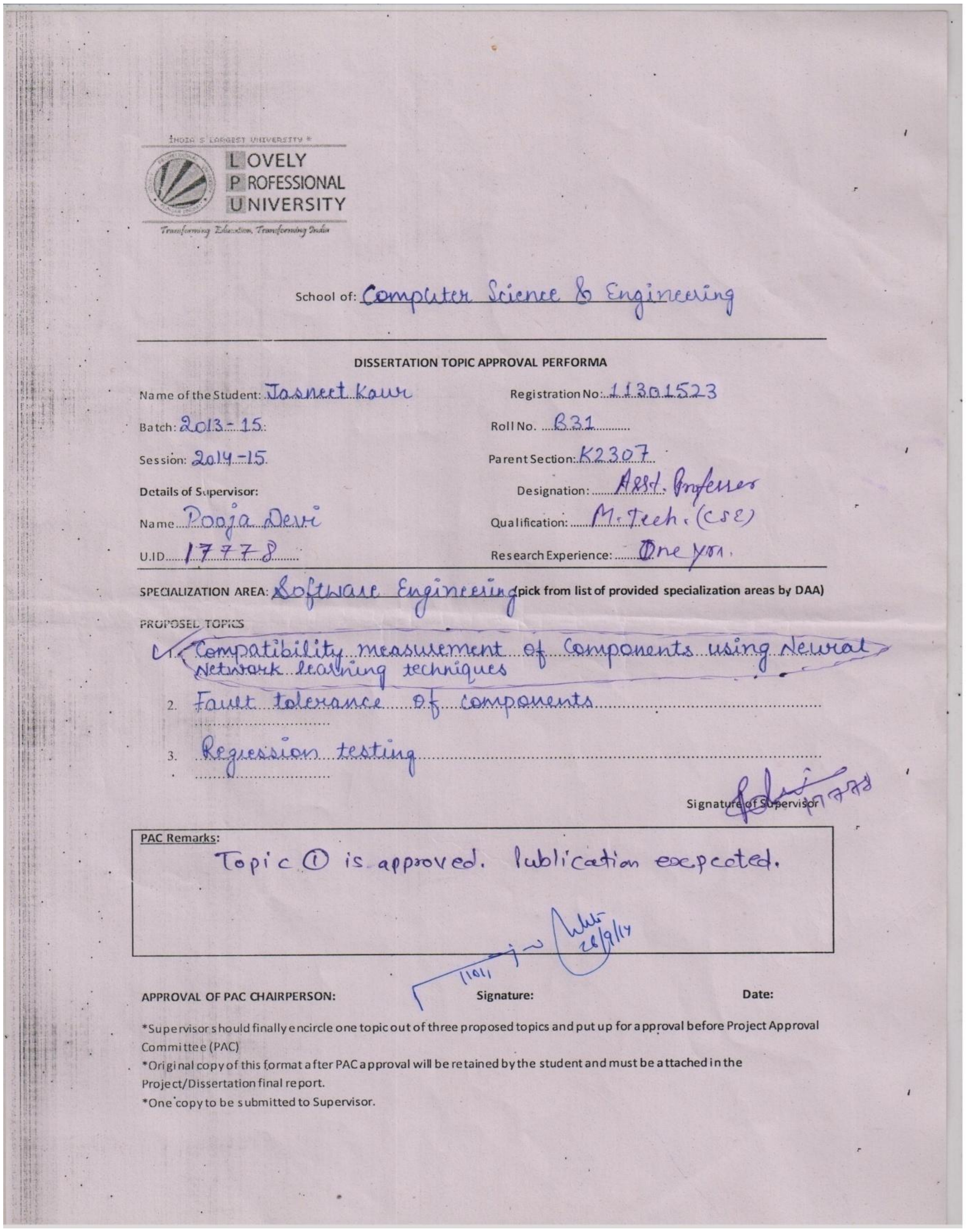
Master of Technology in Computer Science

Under the guidance of

Ms Pooja Devi

(May 2015)

APPROVED PAGE



INDIA'S LARGEST UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education, Transforming India

School of: Computer Science & Engineering

DISSERTATION TOPIC APPROVAL PERFORMA

Name of the Student: Jasneet Kaur	Registration No: 11301523
Batch: 2013-15	Roll No. B31
Session: 2014-15	Parent Section: K2307
Details of Supervisor:	Designation: Asst. Professor
Name: Pooja Devi	Qualification: M.Tech.(CSE)
U.ID: 17778	Research Experience: One yrs.

SPECIALIZATION AREA: Software Engineering (pick from list of provided specialization areas by DAA)

PROPOSEL TOPICS

1. ✓ Compatibility measurement of components using Neural Network learning techniques
2. Fault tolerance of components
3. Regression testing

Signature of Supervisor [illegible] 17778

PAC Remarks:

Topic 1 is approved. Publication excpected.

[illegible] 26/9/14

APPROVAL OF PAC CHAIRPERSON: Signature: Date:

*Supervisor should finally encircle one topic out of three proposed topics and put up for approval before Project Approval Committee (PAC)

*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.

*One copy to be submitted to Supervisor.

ABSTRACT

In modern days developers are giving less interest in starting any development from the scratch. In these days developers just deploy the already built components for their project or system. One of such approach is Component Based software Engineering (CBSE) in which various developers use already built components according to their requirements and reuses the components as per need. In this research, the work has done on checking compatibility of such components with other components of same system so as to reduce reliability issues between components. The effort of entire work done proposed an approach in which using dependency values of components; one can compute the compatibility percentage of those components using Boltzmann learning techniques of neural networks over the values. This approach will give the best suited component from all the alternatives by analyzing compatibility values.

ACKNOWLEDGEMENT

I would like to express my special thanks to Department of CSE of Lovely Professional University for giving me this opportunity of writing this thesis and providing such nice peoples who was there always to help me in my dissertation. Secondly, big thanks goes to my mentor “Ms. Pooja Devi” who gave me this topic “Compatibility measurement in Component Based Software Engineering” for my dissertation work, I am heartily thankful to Pooja mam for being my mentor and helped me in doing a lot of Research and I came to know about so many new things. Very special thanks to all the authors whose paper I referred for this dissertation. Their effort made me to think about new ideas and due to what I am able to implement them in my research. Source: Internet gave me so many short definitions that’s included here.

At last, I would also like to thank my parents and friends who helped me a lot in my research within the limited time frame. This is just 40% completion of my work, I hope my mentor, parents and friends will help me till my thesis is under working

DECLARATION

I am **Jasneet Kaur** hereby declare that the dissertation proposal entitled "**Measuring the Compatibility of Components in Component Based Software Engineering Using Boltzmann Learning Technique"** submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: 5-May-2015

Investigator

Regn. No. 11301523

CERTIFICATE

This is to certify that **Jasneet Kaur** has completed M.tech dissertation proposal titled **Compatibility Measurement of Components in Computer Based Software Engineering Using Boltzmann Learning Technique** under my guidance and supervision. To the best of my knowledge, the present work is the result of her original investigation and study. No part of the dissertation proposal has ever been submitted for any other degree or diploma.

The dissertation proposal is fit for the submission and the partial fulfillment of the conditions for the award of M.tech Computer Science & Engg.

Date: 5-May-2015

Signature of Advisor

Name: Pooja Devi

UID: 17778

TABLE OF CONTENTS

List of figures

Chapter 1
Introduction

1.1 Introduction:

A software is a collection of instructions. Software Engineering is a branch of computer science and system engineering which deals with the development of complex and large software based systems and applications. Basically a software engineering is concerned with building the software system and it is an applications of skills, principles and art to construct and design the various programs. Software engineering covers technical aspects of developing the software through designing, implementation and modification of the software. System software and application software are two categories of a software. To manage the hardware components we use system software because these hardware components should seems like a functional unit to users and software. A system software contain: OS, file manager, memory manager, resource manager, memory manager and many other utilities. On the other hand we have “Application Software”. To accomplish the specific task we use application software. An application software is different from system software, it may or may not have a single program. In this dissertation work the work’s concern is with software engineering. There are many software engineering definitions are given by authors and research, some of them are mentioned in a section.

1.2 Software Engineering Definitions:

There are various definitions of software engineering are given by researchers and authors. Some of the definitions are mentioned below:

- **Sonzmerville:** According to author Sonzmerville: Software engineering is concerned with building the “Software Systems”. The software system consist of various technical and non-technical features. For the development of the software product various engineering principles are used but software systems are normally tackled by the single individual.

- **Dennis:** By the Dennis point of view, software engineering is the application of skills, principles and art to construct and design the various programs.

- **Parnas:** Author Parnas said: Software Engineering (SE) is based upon two conditions.
  First, in the construction of software more than one person is involved.
  Second, more than one version of the program should be produced.

- **Fairly:** Fairly gave the definition about software engineering is: It is a managerial and technological discipline which deals with maintenance as well as production of software product. The software product must be modified on time and within its cost estimation.

- **Eoehm:** According to Eoehm:Software Engineering (SE) is the practical application of scientific knowledge. Software engineering is helpful in constructing and designing the computer programs. Software engineering also represents the complete documentation of the software.

- **Blaschek & Pomberger:** According to Blaschek and Pomberger: Software engineering is related to the use of high quality software and it is the real time application of economical production as per scientific knowledge.

1.3 Software Engineering Objectives:

The objectives in software engineering are given to design a system which will have following characteristics:

- **On Time:** The product should be delivered at the established date.
- **Reliable:** Software should be reliable enough to use and shouldn’t get crashed while using.
- **Complete:** Product must be done with good documentation and is fulfilling customer requirements.

1.4 Software Development Stages:

Following are the software development stages:

- Requirements Specification and Analysis
- Conceptual or Architectural Design
- Detailed Design
- Implementation
- Unit and Integration Testing
- System Testing
- System Delivery and Deployment
- Maintenance

1.5 Software Engineering Principle:

In the software development some principles are used:

- First principle of software engineering is the “Quality of Software”. Quality of developed software must be high so that software should be easily learn and easily in use.
- To implement an accurate solution of the software, first determine the problem issues which are related to the software. After determination of problem issue write down all the requirements.[19]
- The software product delivery to the customer must be done in easy way.
- During the development of the software various changes in the software can be occurred. So the software must be designed in such a way that software should be able to tolerate various changes.
- Software developers who are involved in the process of software development should be very highly skilled. So that they can make a software with good quality.[26]
- Apply the various testing techniques to make sure that the software is bug and error free. If there is any error and bug, remove it and make the software bug and error proof.
- Do write the software design documentation. This written documentation will be helpful if you want to modify the software or want to make any other change.

- There are various process models are available which are used to develop the software. If you are using the appropriate process model, it can give you best solution of a problem, so use the appropriate process model while developing the software.

1.6 Software Engineering Disciplines:

In software engineering, basically we have 8 effective disciplines:

- **Prototyping**:
  Prototype model is basically a dummy model which is for small types of systems. Prototype is used to design the good interface. The prototype has limited functionality. Prototyping usually understands the requirements of users and demonstrates physical designs.

- **Abstraction:**
  Abstraction is used to find out the key aspect of a problem which is might be occurring in the software without getting any detailed information.

- **Design Methods and Analysis:**
  For design methods and analysis purpose team work is required. When team building process is there, usually people communicate with each other by common notations.

- **Software Process:**
  Process of the software contains all activities that are necessary at the time of software development. Activities can be quality of the software, speed, efficiency of the software and many others.

- **Software Architecture:**
  Software architecture defines various components of the system with relationships and the environment which acts as a guide in future.

- **Reuse:**
  Here, the application or the prototype are available for reuse by the other similar systems too.

- **Tools: -**
  Computer Aided Software Engineering (CASE) tools are used to improve the quality of software development. These tools are used to visualize the software development by all the stakeholders of project at each and every stage of development

- **Measurement:**
  Measurement is useful and helpful in decision making process. The process in terms of improvement, resources acts as the support in the development of good software.

1.7 Crises of Software:

During the process of development of the software, many problems can occur. These problems are known as crises of the software. Following are some characteristics:

- Unsatisfactory of software system.
- Extremely late.
- Software is not matching with user requirement.
- Viability to complete.
- Delivery is expensive.

1.8 Some Myths Related To Software:

There are many myths that are related to the software. In the software development many myths are used. These myths are discussed below.

- **Easy To Change:** The one of the myth in software development is software is easy to change. As the editing in software code may be possible up to some extent. But the software is not easy to change. In the software development process, the source code is easy to change, but it is not easy to make changes in the software without introducing the errors. If any change is required in the system, it should be re verified.

- **The replaceable device has the less reliability than the computer system:** The other myth in computer system is that the computers that are used in the software development process are more reliable than the devices that are needs to replace in the software system. That is with the use of this no software errors is occurring or the probability of the error occurrence should be less. But it is totally the wrong myth.

- **Software Testing Remove All the Errors:** The other myth in software development is that if the system is tested, it should remove all the errors that are presented in the system. It is totally the wrong myth. As the testing can only shows that how many errors are presented in the system. The testing cannot show that how many errors are absent at that time [3].

- **Software Safety Can Be Increased With Reusability:** This myth is totally wrong, because reuse of the code may give the false or no sense to the security of the system. The reuse code is also need to analyze, whenever the code need to reuse, it should be analyze properly.

- **Software Can Work Properly On Its First Time:** One of the myth in the software engineering is that software can be work properly only its starting days. As the time pass out the software application may not work properly at that environment.

- **Software with More Features Is Better Software:** This myth is totally unbelievable. With the extra or more features the software may become complex. So it may be very difficult to manage the software. So it is totally the wrong myth that with the more features software is better.

1.9 Software Engineering Approaches:

We can divide software engineering approaches into three major parts which are:

- **Structured Approach:**

Structured approach includes the fundamental steps i.e. Requirements, design, coding, testing and maintenance of software development. In this approach we use following elements:

- **Data Flow Diagrams:** These are used for showing flow of data. In this it gives flow of data through a system. A data flow diagram includes various processes, flow of data, actors, and data repositories.
- **Data Dictionary**: To contains the details of data flow diagrams. In this all the information about the data flows and data repositories in the system is given.
- **State Transition Diagrams**: These diagrams visualize the time dependent behavior of system by showing various states it undertakes.
- **Entity Relationship Diagrams**: These diagrams highlight the relationships between the various kinds of data repositories (stores).

- **Object Oriented Approach:**

The Object oriented approach is used for carrying out the real time problems to the considerations and using those considerations software development can be done. It is a sensible strategy in which we deal with the softwares which consist of various objects to construct and manage itself. [8] This approach uses concept of classes and objects which also provide facility to reuse the code and design. It is further classified into four parts:

- **Unified Modeling Language (UML)**: UML is the standard language which is used to write the software dummy models or blueprints. It is used to visualize, construct, specify, and document the various characters of the project. UML classifies the Structural things into seven parts.

- **Class**: Class is basically a bunch of objects. These objects share the same properties, relationships, functions and the syntax.

- **Interface**: It is the set of various functions that are performed during implementation. It specifies the service of a class. Interface always defines the set of operations specifications, it never tell us about the set of operation implementations.
- **Collaborations**: collaborations stand for combinations of two or more than two things. Collaborations have the structural as well as the behavior dimensions.
- **Use Case**: it is defined as the observed set of actions that are performed by actors of system.
- **Active Class:** An active class works same as class. But it has one major difference that in an active class the objects represent only those elements whose behavior are similar with each other.
- **Components**: these are physical and replaceable part of the systems.
- **Note**: the nodes are the physical entities. Nodes are presents only at the run time.

- **Component-Based Approach:**

Component Based Approach is based upon Reusability principle. In this we use already built components for our project and deploy them as they are. The components are selected as per the requirements. This approach is reuse based approach in which we deal with definition, Implementation and composition of prebuilt independent components [15]. Even if we don't have skill to develop any module or component we can take the available components built for that purpose and use it as per our need.

1.10 Software Components

As it is too complex to deal with objects and classes in case of large projects which includes various modules and components. Dealing with components using codes includes a number of classes and objects which leads to very complicated structure and bulky development code for each component because we have to code each module or component separately. So to overcome this software engineering introduced component based software engineering in which we deploy already built independent components rather than coding for them from start. Components can be hardware or software which has certain functionality and some role to play in the system. Components act as plug n play for the system we are developing using CBSE.

A software component is a part of system which is developed by any third party and can be deployed independently wherever required along with interaction of other components of system. A software system includes its:

- **Specifications** which includes the description about the component
- **One or more implementations** to make it platform independent internally so that it can fit in every situation.
- **Component Model** which tells us about the set of services that are supported by the component as well as set of rules associated with component[2].

1.11 Software Component Models and Technologies

As already explained above, a software component model specifies conditions for composition and communication between various software components.

A software component technology is a collection of software products that supports the use of software component model.

A component model is a definition of standards for

- component implementation,
- documentation and
- deployment.

Examples of component models are: EJB model (Enterprise Java Beans), COM+ model (.NET model), CORBA Component Model. The component model specifies how interfaces should be defined and the elements that should be included in an interface definition.[20] one of the most widely used component models is Microsoft’s Component Object Model (COM) [21].

1.12 Neural Network:

Neural Network or Artificial Neural Network (ANN) is a field of artificial intelligence which is defined as a computing system created by a number of interconnected nodes in layers which process the external inputs and gives dynamic response according to situation as it can change its patterns accordingly.[30]

ANNs follows some learning rule which provides results according to the input patterns. One of such rule is:

Boltzmann Learning Rule: This is learning rule according to which we can calculate probabilities by taking neuron values as parameters.[33] This is a type of machine learning in which we have following cases:

Clamped condition when we have all connections among the units

Free running condition when we have no connections among units and they operate freely and independently

Other learning rules that can be used for compatibility analysis are:

- Knowledge based learning rule
- Hebbian learning rule
- Memory based learning rule
- Competitive learning rule etc.

Chapter 2

Review of Literature

During the preparation of this dissertation work, work and hypothesis collected from many books, journals, magazines, whitepapers and articles. Some of them are very useful for my current and further research. They have worked on this greatly. Some abstracts from those resources with references are given below.

A research paper by “Inti Gonzalez-Herrera and Johann Bourcier” 2014, entitled **“Scapegoat: An Adaptive Monitoring Framework for Component-Based Systems”** [9]. The paper is published under “IEEE Digital Library”, 2014. This paper proposed a method named “Scapegoat” which used an open source dynamic platform “kevoree” by using models@run.time for making systems dynamic adaptable. The defined approach in the paper checks each component's resource usage such as CPU, Memory and Time etc. Then this approach use model@run.time to predict the faulty components and helps in fine grained monitoring.

A research paper by “Gordon Blair, Nelly Bencomo and Robert B. France” 2009, entitled “**Models@Run.Time”**[7]. This paper is published under “IEEE computers”, 2009. The paper is defining models@run.time which tells about various aspects of the system in the case of higher abstractions. A model@run.time represents system by considering the nature, architecture and objectives of the system from a problem space view. models@run.time focuses on software architecture and functional representations of architecture and main target is to upgrade the level of abstraction to that of requirements. It also deals with the runtime variability of the systems for adaptations of new components or technology.

A research paper by “Marco Autili and Paolo Di Benedetto” 2012, entitled **“A Hybrid Approach For Resource-Based Comparison Of Adaptable Java Applications”**[14]. This paper is published under “Science of Computer Programming”, 2012. In this paper authors are showing that heterogeneity of resources creates problems because java based mobile applications have to run on huge amount of different devices. This research paper proposed a model called “Chameleon Model” to decide java programs to be adapted by analyzing their resource consumption. It helps to choose among multiple alternatives which will suite better with the available resource provisions.

A research paper by “Ardhendu Mandal Lecturer, Department of Computer Science and Application, University of North Bengal (2009)” entitled “**BRIDGE: A Model for Modern Software Development Process to Cater the Present Software Crisis”**[3]. Large and complex projects are often late to market, have quality issues, and not always delivered on promised functionality called software crisis i.e problems associated with software development process. To handle such software crisis a new software development process model named BRIDGE is introduces in this paper. The bridge model works in 13 phases and each phase includes verification and specification

A research paper by “Luciano Baresi, Elisabetta Di Nitto and Carlo Ghezzi” October 2013, entitled **“Toward Open-World Software: Issues & Challenges”** [13]. This paper is published under “IEEE Digital Library”, 2013. This paper showed that software should answer to changes by organizing itself accordingly and adapting the behaviors. This paper explained many concepts like:

1. SOA (Service Oriented Architecture)
2. Web Services
3. Publish/Subscribe Middleware
4. Grid Computing
5. Autonomic Computing

An another research paper by “Walter Binder, Jarle Hulaas , Philippe Moret and Alex Villazón” 2010, entitled **“Platform-Independent Profiling In a Virtual Execution Environment”** [26]. This Paper states that Java virtual machines have problems in analyzing algorithms performance, tools, complexity, bottleneck, CPU time and consumption etc. This paper gave an approach based on program transformation techniques to build a profiling data structure with independent profiling metrics to make it portable and generate reusable profiles.

Aresearch paper by **“**Steen Becker, Heiko Koziolek and Ralf Reussner**”** January 2010, entitled **“The Palladio Component Model for Model-Driven Performance**

Prediction" [24].This paper has provided a model named as Palladio Component Model (PCM) to specify structure of Component Based Software Development by presenting a model for it as well as model based simulation tool. This can make performance predictions. This model measures the response time of various alternatives and at the end we can compare and choose according to our requirements. It provides a simulation model which compares various alternatives with each other as well as various running systems too.

A research paper by "Francois Fouquet, Olivier Barais, Noel Plouzeau and Jean-Marc Jezequel" 2012, entitled **"A Dynamic Component Model for Cyber Physical Systems"** [6]. The paper is published under "CBSE, V. Grassi, R. Mirandola, N. Medvidovic, and M. Larsson", 2012.In this paper authors have presented a system which works in volatile environments where the components of the system themselves adapt the conditions in their environment. It evaluates embedded domains and provides reconfigurable component-based model using model@runtime which provides efficient and safe reasoning for benchmarking for assessing the usability and overhead. Physical devices are controlled by low power microcontrollers to bind them with software components than to deploy extra components.

Research paper by "Murat Gunestas" January 2005, entitled **"A Study on Component Based Software Engineering"** [15].This research paper is explaining about the three main approaches that are used by software engineering. These three main approaches are: Structured approach (Traditional), Object oriented approach and Component Based approach. This component based approach has changed the view of the implementation and maintaining software systems. As estimated by the Gartner Group from year 2003, 70% of new applications are implemented as a composition of already built components and latest built components are integrated to develop complex business systems. This increases the time to market, software lifecycle costs and quality. As compare to structure and object oriented approaches, Component-based approach of software engineering provides more advantages in terms of reuse, flexibility and maintenance. In this paper component based approach is examined as this approach provides interoperability as well as maintenance cost is reduced because all components are independent.

A research paper by “Ruslan Salakhutdinov and Geoffrey Hinton” 2009, entitled **“Deep Boltzmann Machines”** [23].This paper showed an approach that helps the Boltzmann machines which have multiple variables or hidden layers. Data dependent assumptions are estimated using an resemblance that varies and concentrate on a single mode, and data independent expectations are estimated using Markov chains.

Research paper by “KyunHyun Cho, Tapani Raiko and Alexander Ilin” 2012, of entitled “**Deep learning and Boltzmann machines”** [11]. In this paper the author suggested an approach to understand the large and complex problems which uses deep learning i.e many internal hidden layers. For constructing deep neural networks the new approach the author has proposed parallel tempering instead of gibbs sampling concept which will analyze the dynamic properties and gives out the output. This approach has implemented for Boltzmann machines because it can handle multiple hidden layers better than other techniques.

A research paper by “Murali Krishnan Gunasekaran” entitled **“Component-Based Software Engineering – New Paradigm of Software Development,Ivica Crnkovic”** [19] This paper emphasized that the component based software construction is use to gain the large momentum. It focuses on the software engineering research and computing. For developing component based applications many standards are needed. In these kind of systems the single language based approach is not required. In this paper author discuss the mixed language programming approach, this approach can be used to build the component based software systems. This method is used to solve the design problems that occur while the development of the software. The MLP is very useful approach, because it allows to use the existing components to develops the new software systems. Then these systems are combined to develop a larger system. The concept of inheritance is used in this paper. Here the genetic algorithm and component based approach are used. these approaches are used to build the new components system. These systems are use to analyze and develop a component based solution for a ship design problem.

Research paper by “Ms. Sonali. B. Maind , Ms. Priyanka Wankar”, of title **“Research Paper on Basic of Artificial Neural Network”** [18]. In this paper the authors tells about

Artificial Neural Networks basics. It says that an ANN information processing system which includes many highly interconnected processing elements called neurons which is inspired by human biological brain system. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. This paper gives overview of Artificial Neural Network, working & training of ANN. It also explain the application and advantages of ANN.

Research paper by ''Michelangelo Diligenti, Marco Maggini, and Leonardo Rigutini", of title **"Learning Similarities for Text Documents using Neural Networks"** [16]. In this paper the author explained that neural networks follow either a supervised or a unsupervised learning. The author proposed an approach which lies in between these two learning schemes. A set of relations is specified by the supervisors for various input patterns an neural network is trained to work according to the relationships specified by supervisor for dimensionally reducing space of text documents.

Chapter 3

Present Work

3.1 Problem Formulation

Now days, most of the companies use the component based system to develop the new software. If a company wants to develop new software, it can use the existing modules. But the main problems occur in the size of that software. Sometimes the size of the software is small, but the module size is bigger than the software size. In this case the software became of large size. Hence the functionality of the software is less. The existing software systems face the problem of compatibility. The compatibility is the big issue in the CBSE. Because when a software is developed, its components should match. Their compatibility effect, when the environment of the components that use in the software is totally different. If the software developers selects the components manually and they are unable to check the functionalities of these components. It also leads to a problem called compatibility problem. In the compatibility various components are checked. The Components are the building blocks of software systems. It comprises the elements of reuse in software architecture. Component is a primary functional unit. Components allow the users to represent a high level software model. Components must be generic enough to work in a variety of contexts. The compatibility of two components is determined by the behavior at their interfaces. If two components wants to communicate with each other, the services of these components must be compatible.

3.2 Objectives of study

The component based software engineering is used to develop the new software using existing components. Software system's performance can be increased by measuring compatibility of the components. Hence, reliability of components increases with the use of component based software engineering. CBSE is playing an important role in achieving the objective of this dissertation. Following are the objectives of this dissertation:

- To study and analyze various compatibility testing techniques for component based software
- To propose techniques for compatibility testing of component-based software.
- To Implement knowledge based learning to test compatibility of component based software modules
- To propose Boltzmann learning to test compatibility of component based software modules
- To Implement proposed technique, analyze and compare results with existing technique in terms compatibility

3.3 Research Methodology

Methodology is the theory based analysis. Methodology came from two words, "methods and logic". The combination of methods and logic is known as methodology.

The component based software modules are the plug and play based modules means independent modules and these modules are highly reusable components. The main problem exists in the component based software's is of compatibility. In this work, a technique will be developed which is based on to test compatibility of the component based software's using Boltzmann leaning algorithm. In this work, we will take various component based software modules and test their compatibility for re-using to make a new or to update existing software. The dependencies have been calculated between various component based modules to analyze the compatibility. To analyze the compatibility neural network technique has been applied. The dependency between the modules will be analyzed on the basis of number of values transferred from one module to other module. To analyze the compatibility dependency graphs has been drawn. To draw dependency graph, input training dataset values has been considered. The training dataset values are initial assumption that how many values are exchanged between module. To calculate dependency at every iteration initial value will be incremented by 1 and check the error if the error will reduced further increment is done . In the project one stage will come at which error become stable or keep on increasing with every increment, now that value will be considered as the final dependency value of modules

After the preprocessing phase, we select the modules from the various modules to check compatibility. To test the compatibility of the selected components Boltzmann learning algorithm is applied on the proposed dependency values. The modules which are most compatible are selected to make new software.

3.3.1 Flow Chart:
As, problem area for research has been defined already, Now how the work will be carried out, various methods, technologies and platforms I have to follow for the completion of my goal is defined through flowchart below:

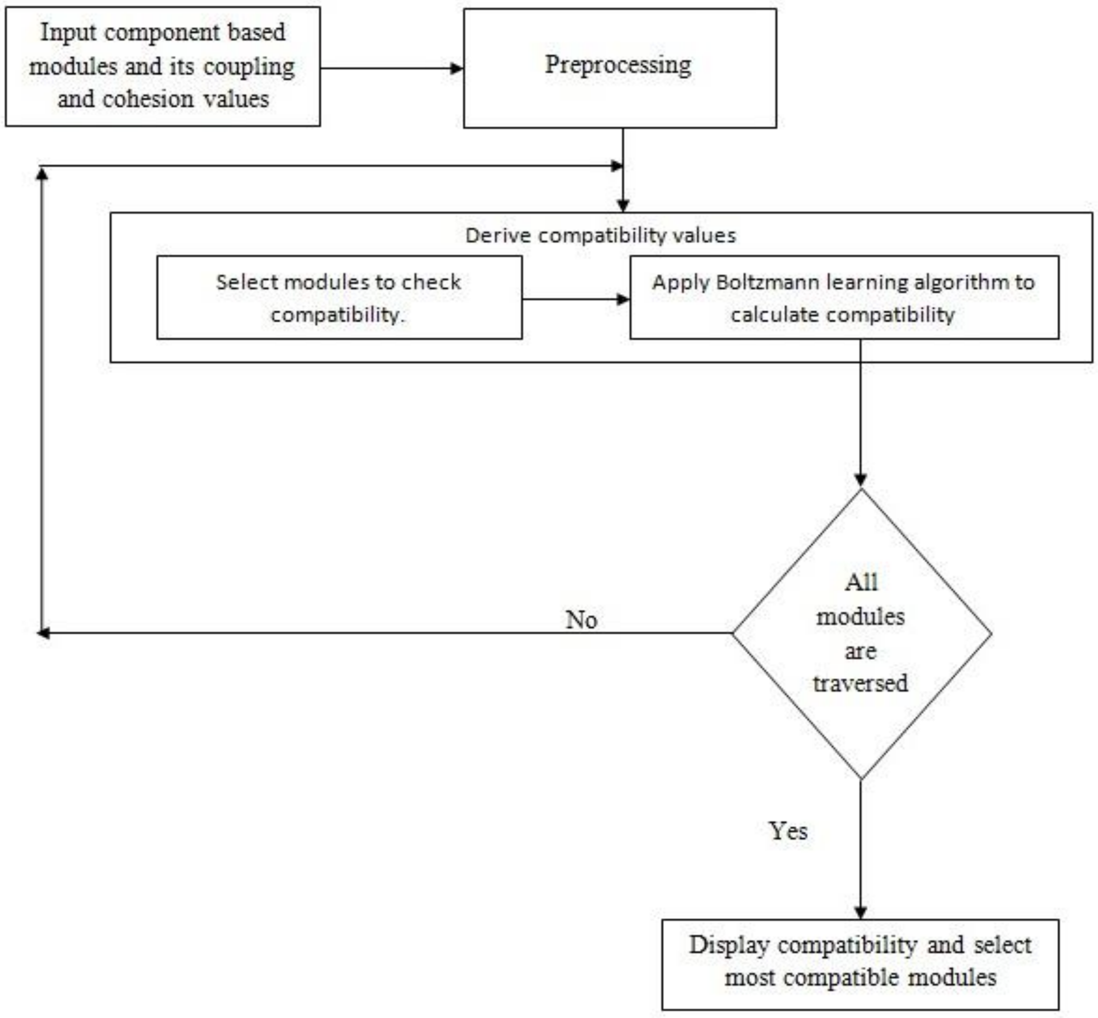

Input component based modules and its coupling and cohesion values
Preprocessing
Derive compatibility values
Select modules to check compatibility.
Apply Boltzmann learning algorithm to calculate compatibility
All modules are traversed
No
Yes
Display compatibility and select most compatible modules

Fig 3.3.1 Proposed Methodology

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 Dependencies:

A neuron is an information-processing unit which is fundamental to the operation of the neural. The three basic elements of the neural models are:

- Synaptic Weights
- Linear Combiner
- Activation function

SW is a set of synapse which is characterized by the strength or weight of its own.It is also known as connecting links.

An adder for summing the input signals, weighted by the respective synapse of the neuron. This operation is called **linear combiner (LC)**.

An **AF** for limiting the amplitude of output of the neuron. Sometimes it is also called squashing functions [33]

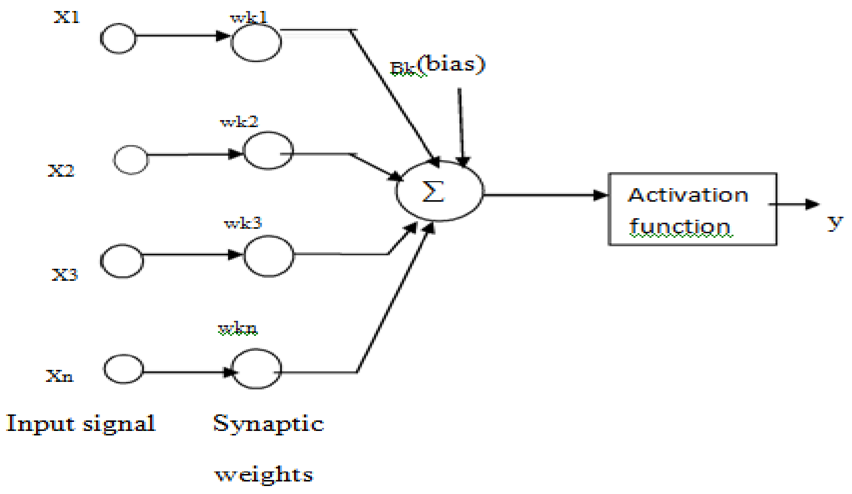


Fig 4.1: Neural Networks [31]

It is a learning in which synaptic weights are corrected according to the error of the neuron output Here the output generated is compared with target output and desired response.

Error=Desired Response – Actual Output

4.2 Boltzmann Learning

A restricted Boltzmann machine is a two-layer undirected graphical model where the first layer visible units are held to values given by the teacher and second layer are unlearning component (where the output units are free to vary) The visible layer is fully connected to the hidden layer via pair-wise potentials, while both the visible and hidden layers are restricted to have no within-layer connections.BM is a stochastic recurrent neural network consisting of binary neurons arranged in two layers. Each neuron v_i in the visible layer is connected to all the hidden neurons, and each neuron hj in the hidden layer is connected to all the visible neurons. It can be denoted by v a binary column vector containing the states v_i of the visible neurons and similarly by h a vector of hidden states hj .

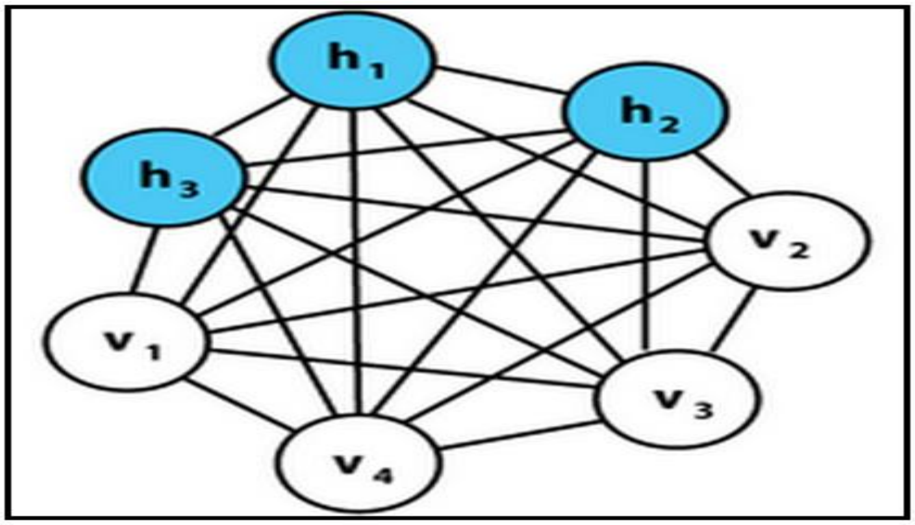


Fig.4.2 Boltzmann Learning [23]

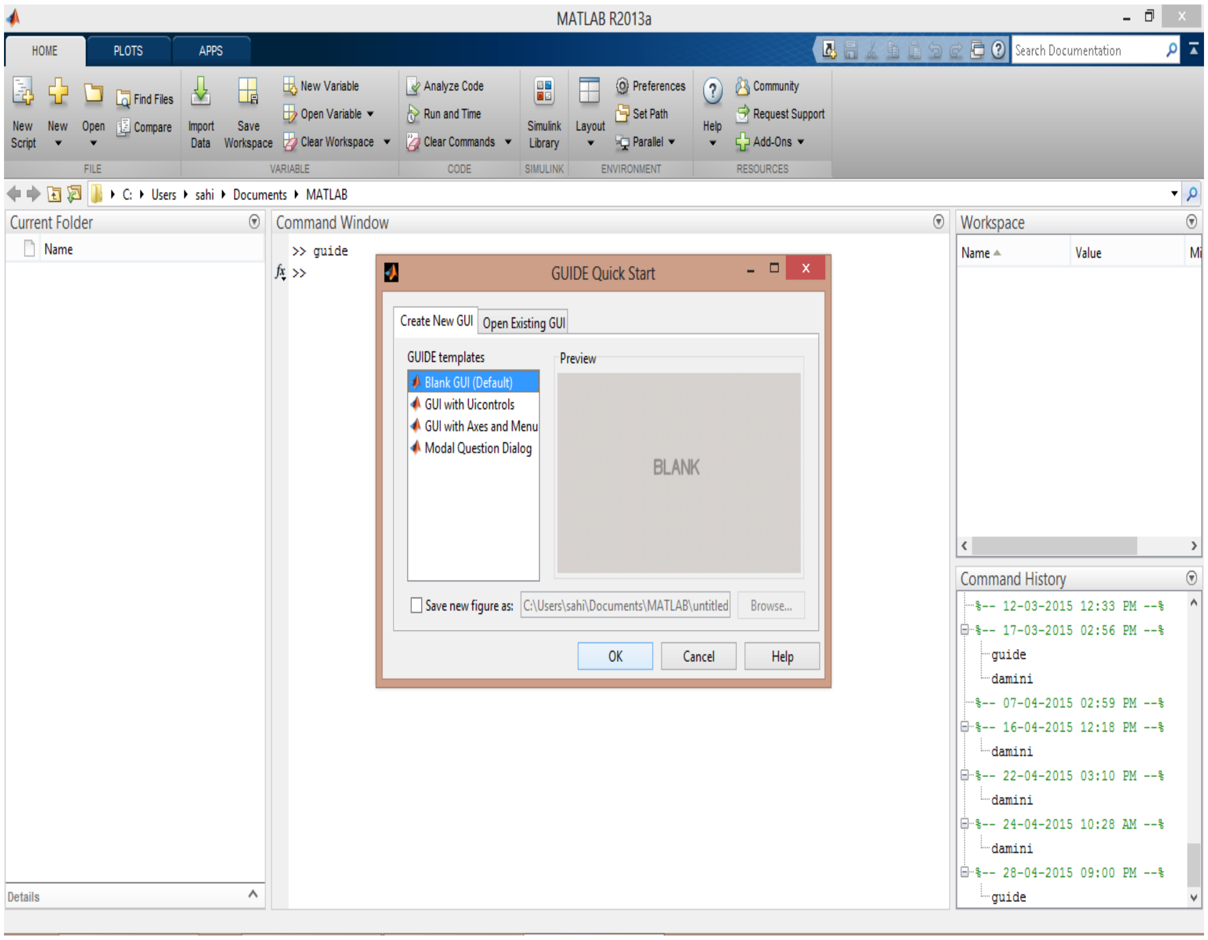
Algorithm

INPUT: Training rules to derive compatibility and component based modules

OUTPUT: Percentage of compatibility

```
Start ()

{

    1. Assign training rule to each component which are available
       While (traversed all components)

    2. a=Calculate dependency of each component ()
       a=a++;

       if (a<a(i-1));
       {
       A=calculate dependency of each component ();
        a=a++;
       else
       {
       Assign final value of dependency;
       }

    3. A=b;

    4. B=final compatibility;
       End

    5. End
```

4.3 Introduction to MATLAB



```
>> guide
>>
```

```
%-- 12-03-2015 12:33 PM --%
%-- 17-03-2015 02:56 PM --%
  guide
  damini
%-- 07-04-2015 02:59 PM --%
%-- 16-04-2015 12:18 PM --%
  damini
%-- 22-04-2015 03:10 PM --%
  damini
%-- 24-04-2015 10:28 AM --%
  damini
%-- 28-04-2015 09:00 PM --%
  guide
```

Fig 4.3.1: Matlab

This figure shows the guide tool box that has been used to implement the interfaces as well as handling various tools

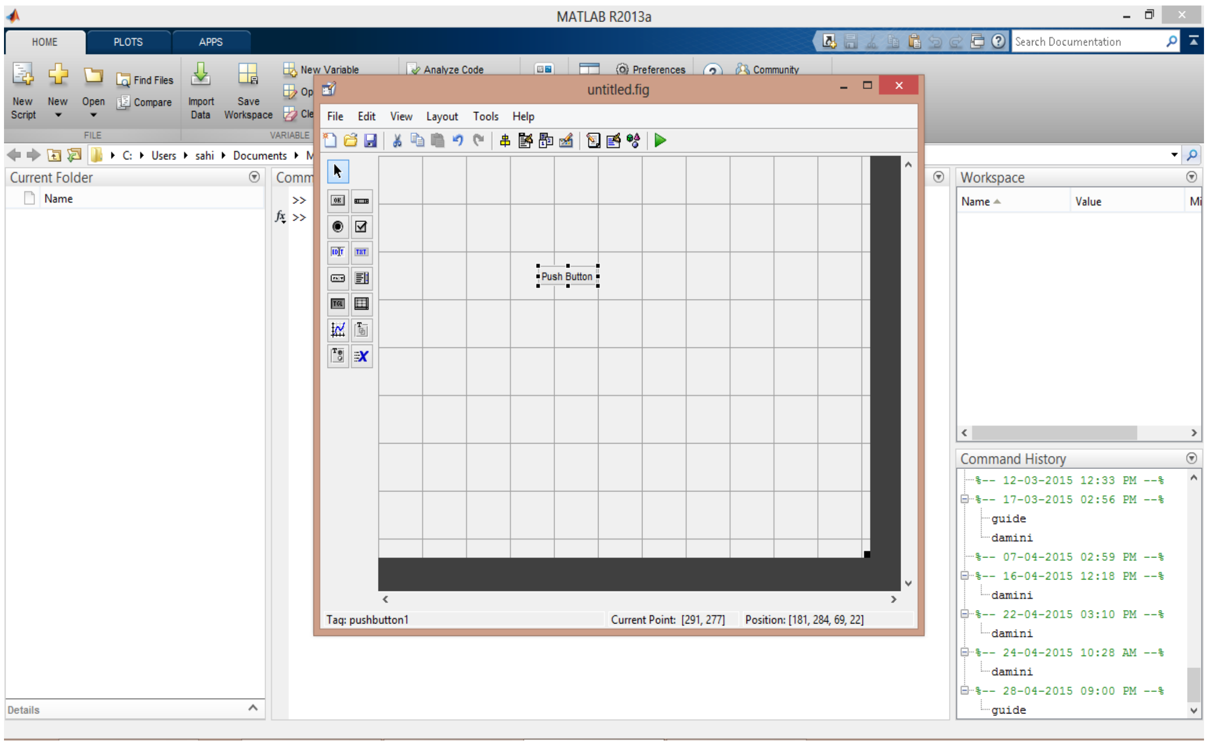


Fig 4.3.2: Matlab Tool

This idea is developed in MATLAB which is mostly used in all areas of research and industry these days. MATLAB gives an advantage of implementing ideas which includes mathematical equations [27] as the proposed approach in this paper has mathematical equations to calculate compatibility so most preferred tool is MATLAB. It is a programming language for coding mathematical programs. It has toolboxes which are used as per requirement.

4.4 Working Of Developed Tool:

The new tool that is develop in here provides following things

- This tool is used to check the compatibility of the system.
- The tool check the compatability of each module with the other.
- If the faulty objects are present in the module it drops that module and choose the new one.
- The new tool always check for the uncompatible nodes.
- These uncompatible nodes affect the reliability of the software.

- When the tool complete with the checking of nodes, it further checks the compatability of the software.
- It selects the two or three objects, if the percentage of the faulty nodes present in the system is high then the tool drop that combination and choose the next.
- This helps to maintain the compatability of the system.
- This tool also increase the quality of the software system.

4.5 Snap Shots

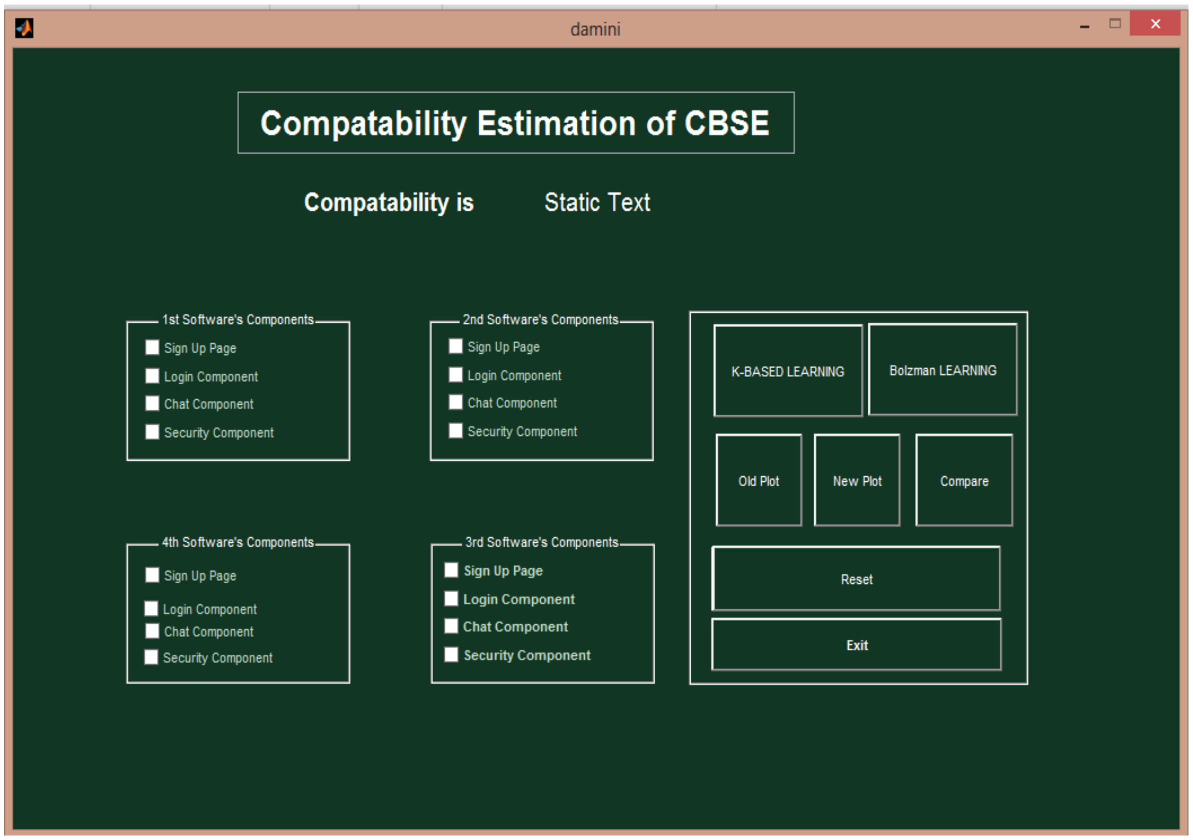


Fig 4.5.1: Initial state.

In this figure, software is in idle state. Here four component boxes are there. These component boxes contain the various components.

Dependencies check

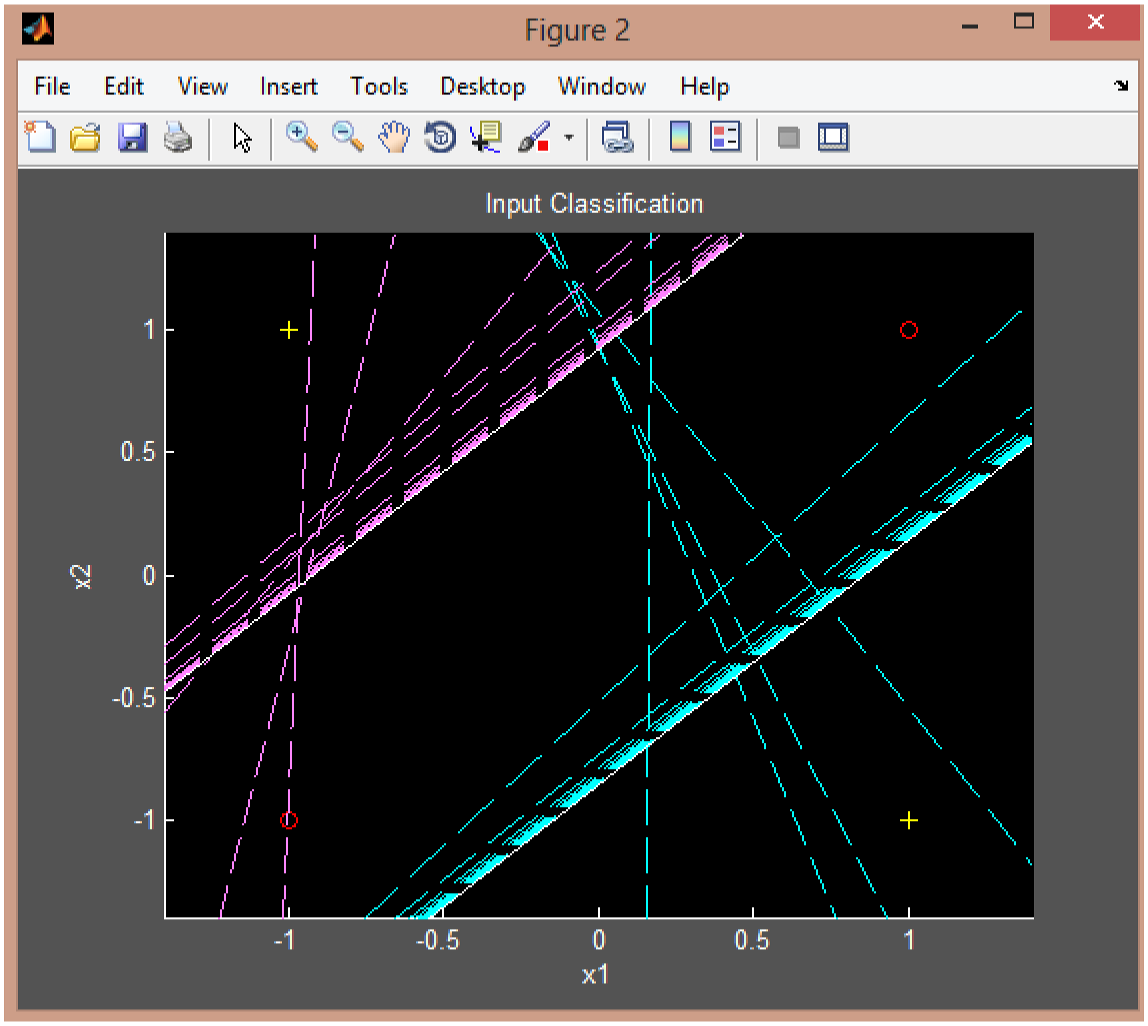
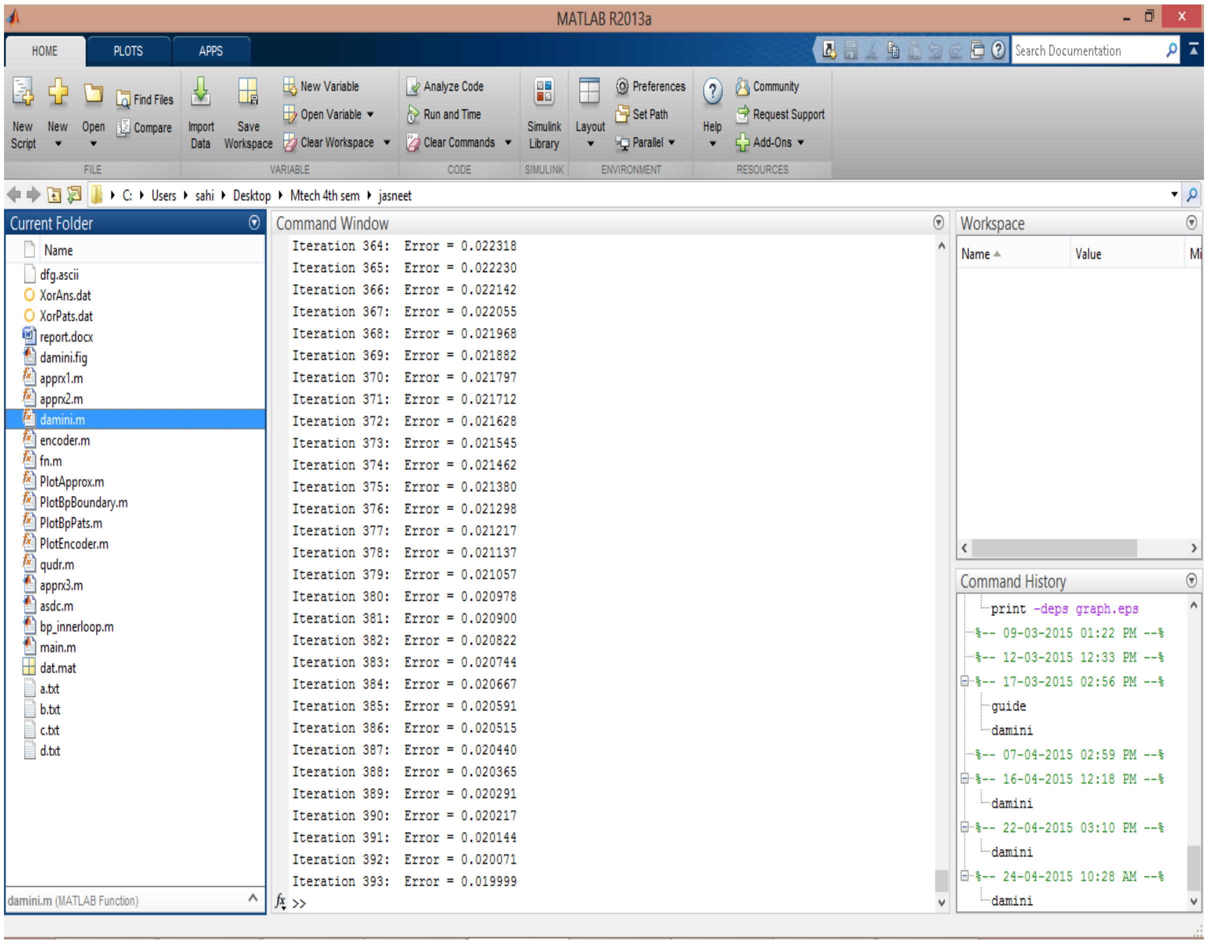


Fig 4.5.2: Dependencies graph

In above figure, various dependencies between components are shown in form of graphs between the four components

Iterations

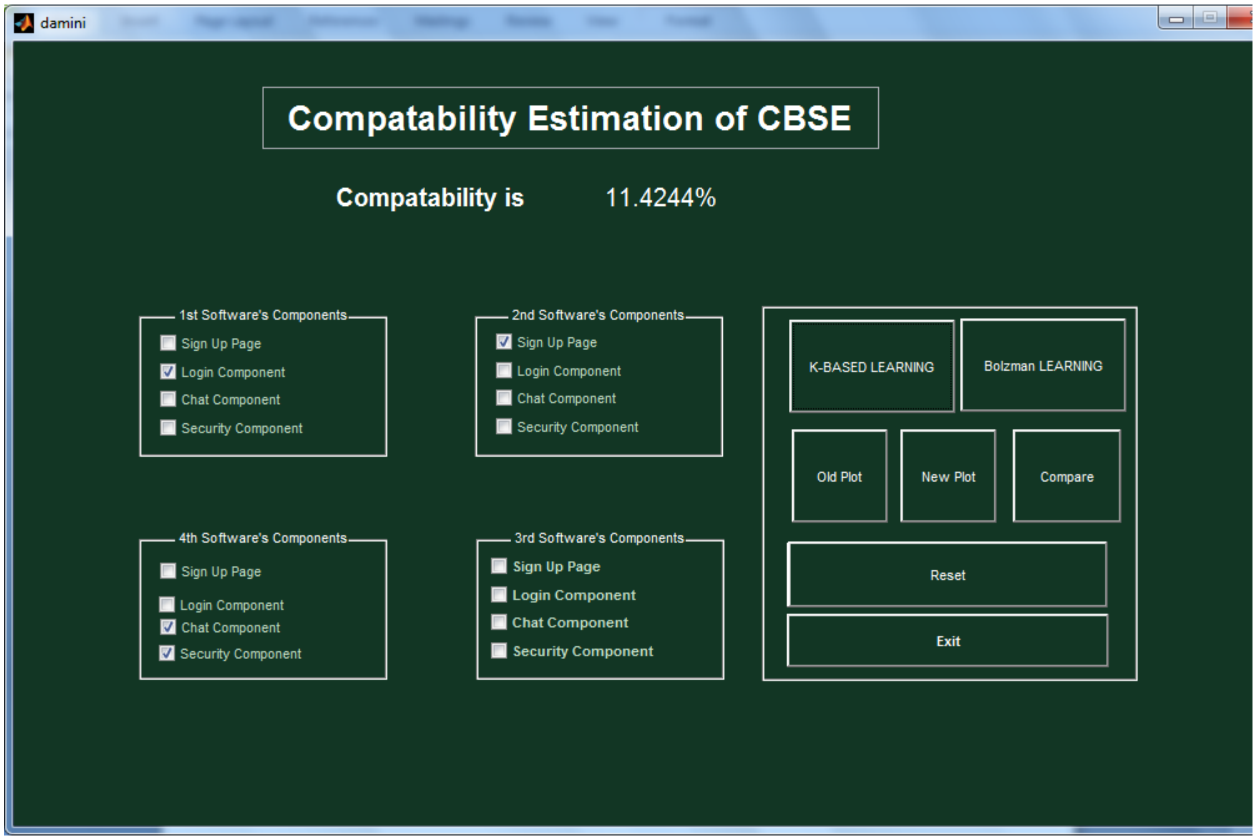


```
Iteration 364:  Error = 0.022318
Iteration 365:  Error = 0.022230
Iteration 366:  Error = 0.022142
Iteration 367:  Error = 0.022055
Iteration 368:  Error = 0.021968
Iteration 369:  Error = 0.021882
Iteration 370:  Error = 0.021797
Iteration 371:  Error = 0.021712
Iteration 372:  Error = 0.021628
Iteration 373:  Error = 0.021545
Iteration 374:  Error = 0.021462
Iteration 375:  Error = 0.021380
Iteration 376:  Error = 0.021298
Iteration 377:  Error = 0.021217
Iteration 378:  Error = 0.021137
Iteration 379:  Error = 0.021057
Iteration 380:  Error = 0.020978
Iteration 381:  Error = 0.020900
Iteration 382:  Error = 0.020822
Iteration 383:  Error = 0.020744
Iteration 384:  Error = 0.020667
Iteration 385:  Error = 0.020591
Iteration 386:  Error = 0.020515
Iteration 387:  Error = 0.020440
Iteration 388:  Error = 0.020365
Iteration 389:  Error = 0.020291
Iteration 390:  Error = 0.020217
Iteration 391:  Error = 0.020144
Iteration 392:  Error = 0.020071
Iteration 393:  Error = 0.019999
>>
```

```
print -deps graph.eps
%-- 09-03-2015 01:22 PM --%
%-- 12-03-2015 12:33 PM --%
%-- 17-03-2015 02:56 PM --%
guide
damini
%-- 07-04-2015 02:59 PM --%
%-- 16-04-2015 12:18 PM --%
damini
%-- 22-04-2015 03:10 PM --%
damini
%-- 24-04-2015 10:28 AM --%
damini
```

Fig 4.5.3: Dependency Iterations

Dependencies are calculated by iterations again and again until the optimal value is obtained


damini

Compatability Estimation of CBSE

Compatability is 11.4244%

1st Software's Components
- [ ] Sign Up Page
- [✓] Login Component
- [ ] Chat Component
- [ ] Security Component

2nd Software's Components
- [✓] Sign Up Page
- [ ] Login Component
- [ ] Chat Component
- [ ] Security Component

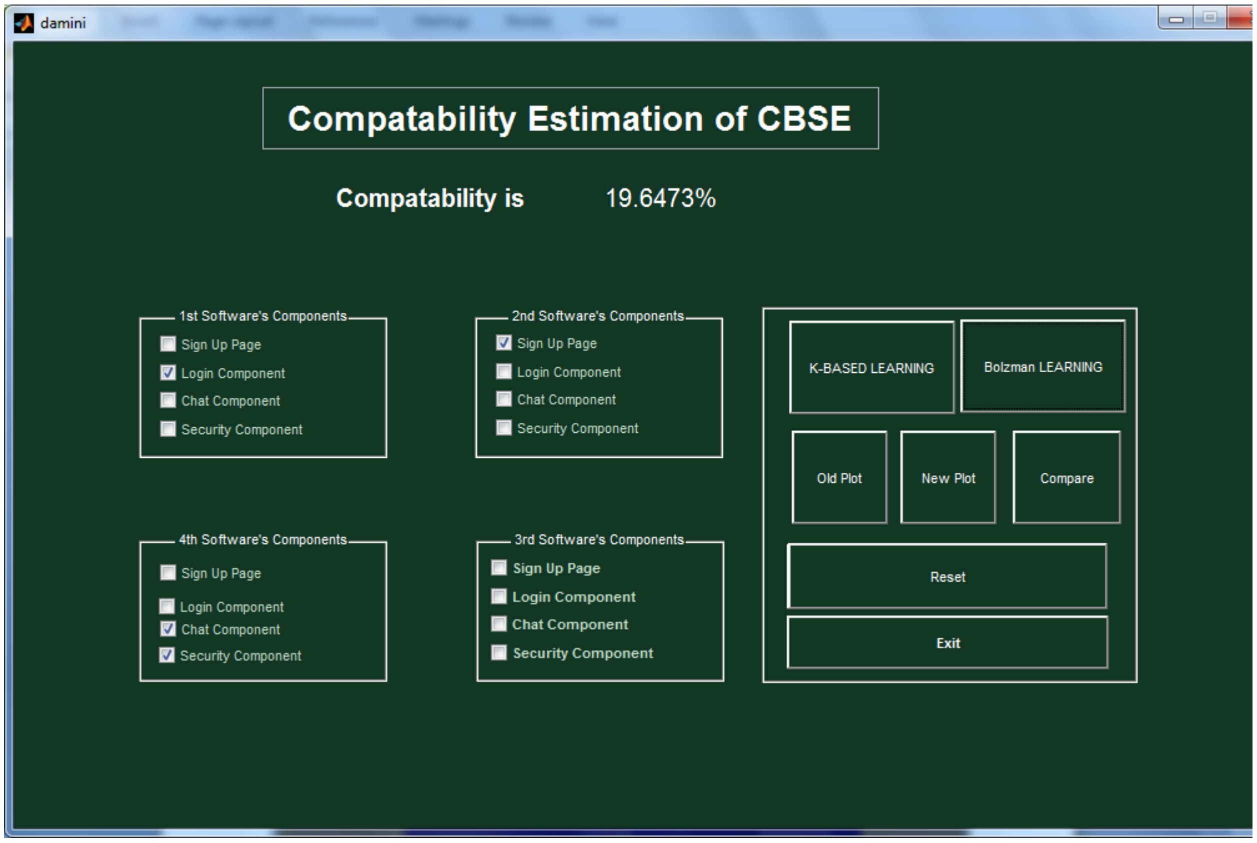
4th Software's Components
- [ ] Sign Up Page
- [ ] Login Component
- [✓] Chat Component
- [✓] Security Component

3rd Software's Components
- [ ] Sign Up Page
- [ ] Login Component
- [ ] Chat Component
- [ ] Security Component

Fig 4.5.4 Compatibility Check

In this figure, the compatibility between the various components is shown. The computability can be shown by the bar graph by clicking on the right hand side of the figure.

Compatibility Check



Compatability Estimation of CBSE

Compatability is 19.6473%

1st Software's Components
- [ ] Sign Up Page
- [✓] Login Component
- [ ] Chat Component
- [ ] Security Component

2nd Software's Components
- [✓] Sign Up Page
- [ ] Login Component
- [ ] Chat Component
- [ ] Security Component

4th Software's Components
- [ ] Sign Up Page
- [ ] Login Component
- [✓] Chat Component
- [✓] Security Component

3rd Software's Components
- [ ] Sign Up Page
- [ ] Login Component
- [ ] Chat Component
- [ ] Security Component

Fig 4.5.5 Using KBL

In this figure, the compatibility between the various components is shown. The computability between various components are shown using Knowledge Based learning.

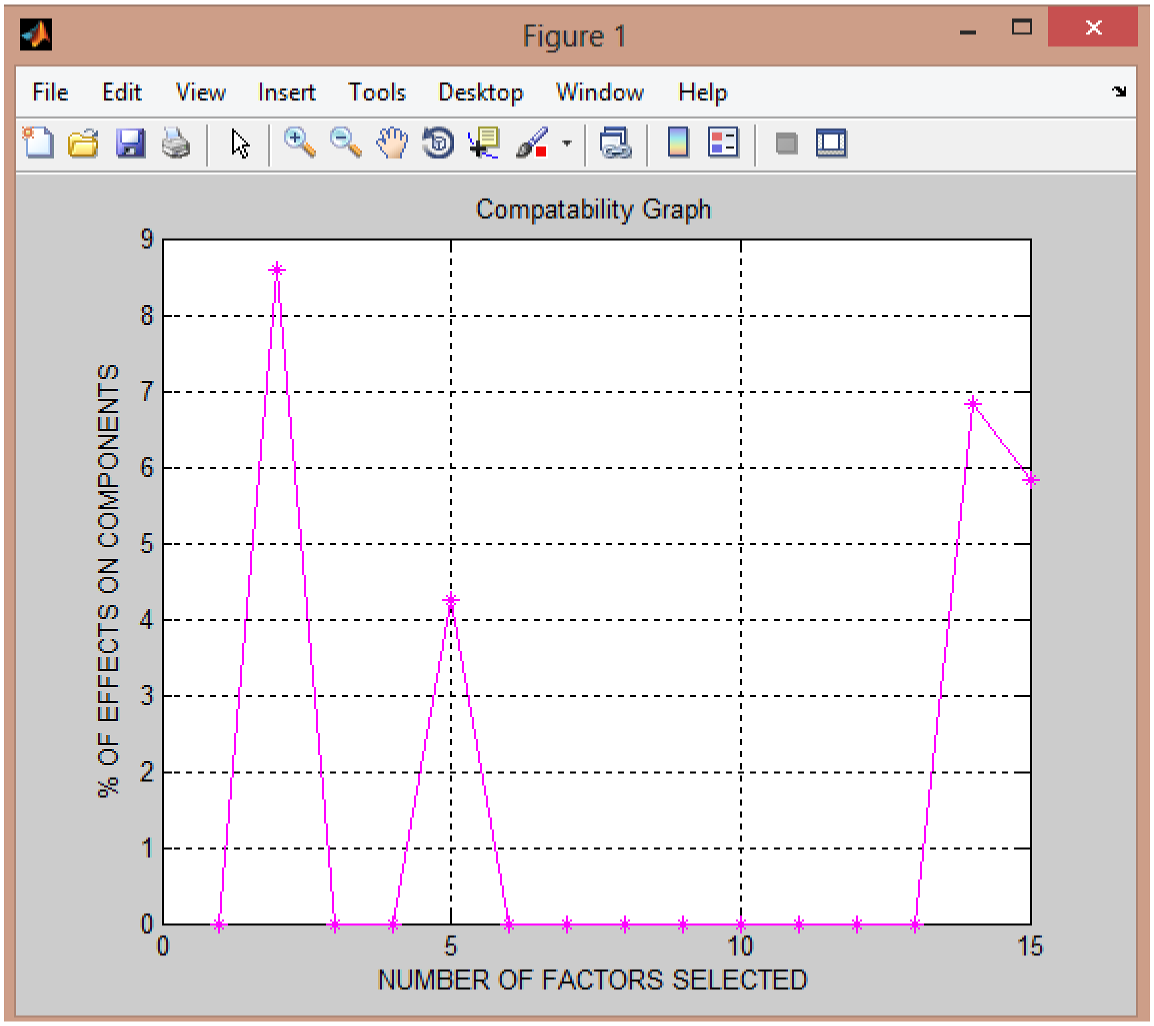
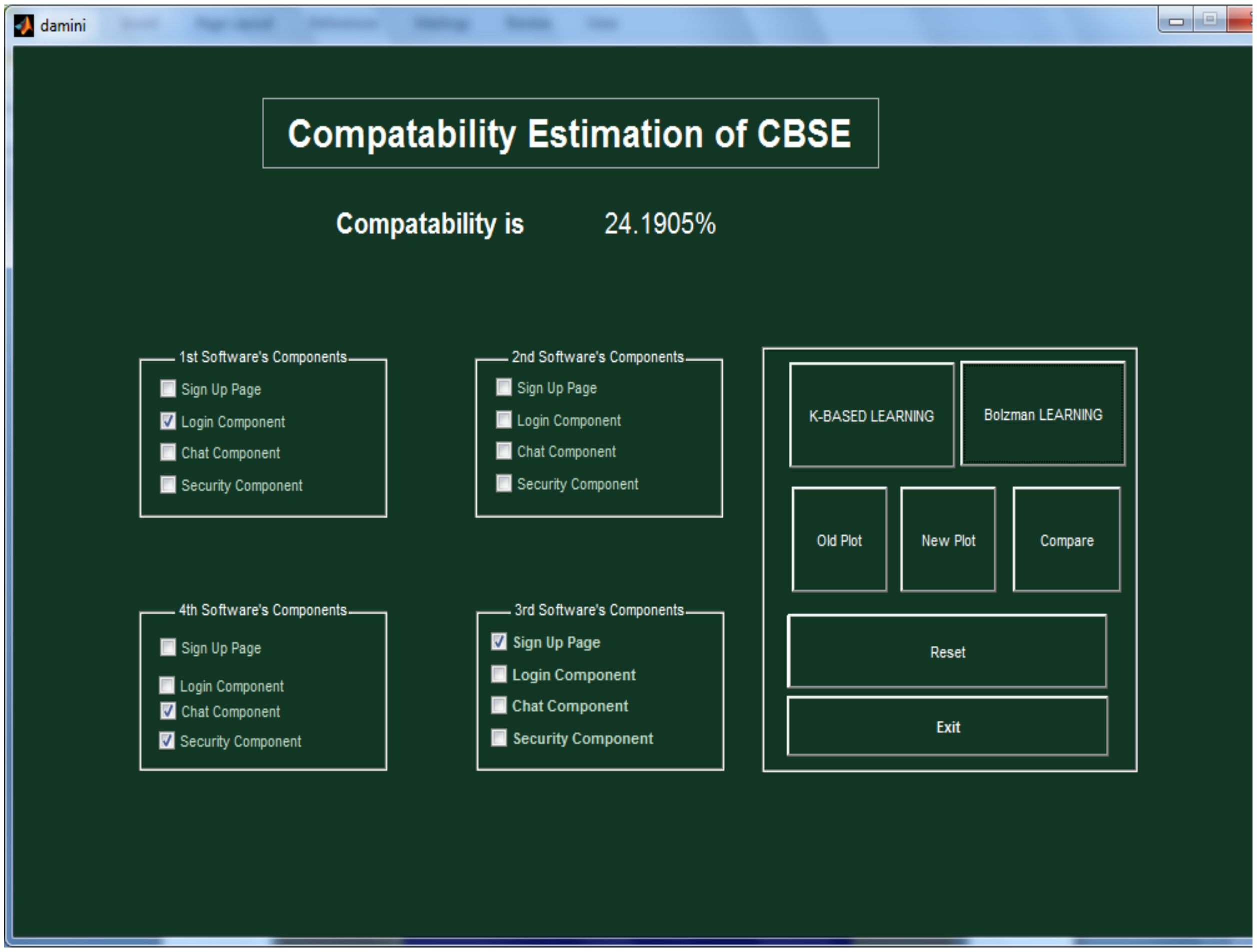


Fig 4.5.6 Graphical representation of KBL

This figure shows the graph of the compatibility values of various components and according to value of components overall compatibility is shown using KBL

Compatibility Checking


damini

Compatability Estimation of CBSE

Compatability is 24.1905%

1st Software's Components
- [ ] Sign Up Page
- [✓] Login Component
- [ ] Chat Component
- [ ] Security Component

2nd Software's Components
- [ ] Sign Up Page
- [ ] Login Component
- [ ] Chat Component
- [ ] Security Component

4th Software's Components
- [ ] Sign Up Page
- [ ] Login Component
- [✓] Chat Component
- [✓] Security Component

3rd Software's Components
- [✓] Sign Up Page
- [ ] Login Component
- [ ] Chat Component
- [ ] Security Component

Fig 4.5.7 Using Boltzmann Learning

In this figure, the compatibility between the various components is shown. The computability of the components are shown using Boltzmann learning

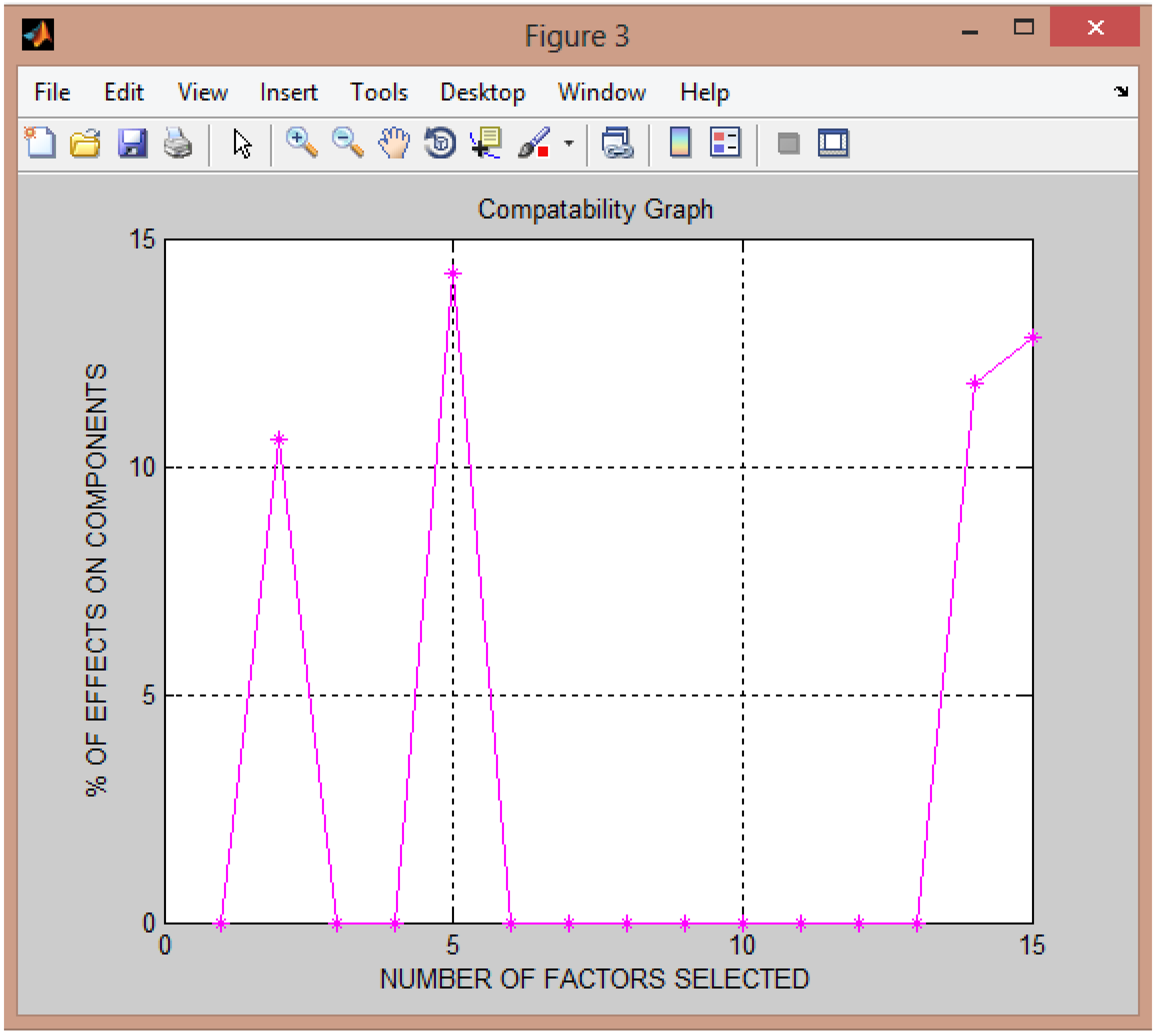


Fig 4.5.8: Graphical Representation of Boltzmann learning

This figure shows the graph of the compatibility values of various components and according to value of components overall compatibility is shown using Boltzmann Learning

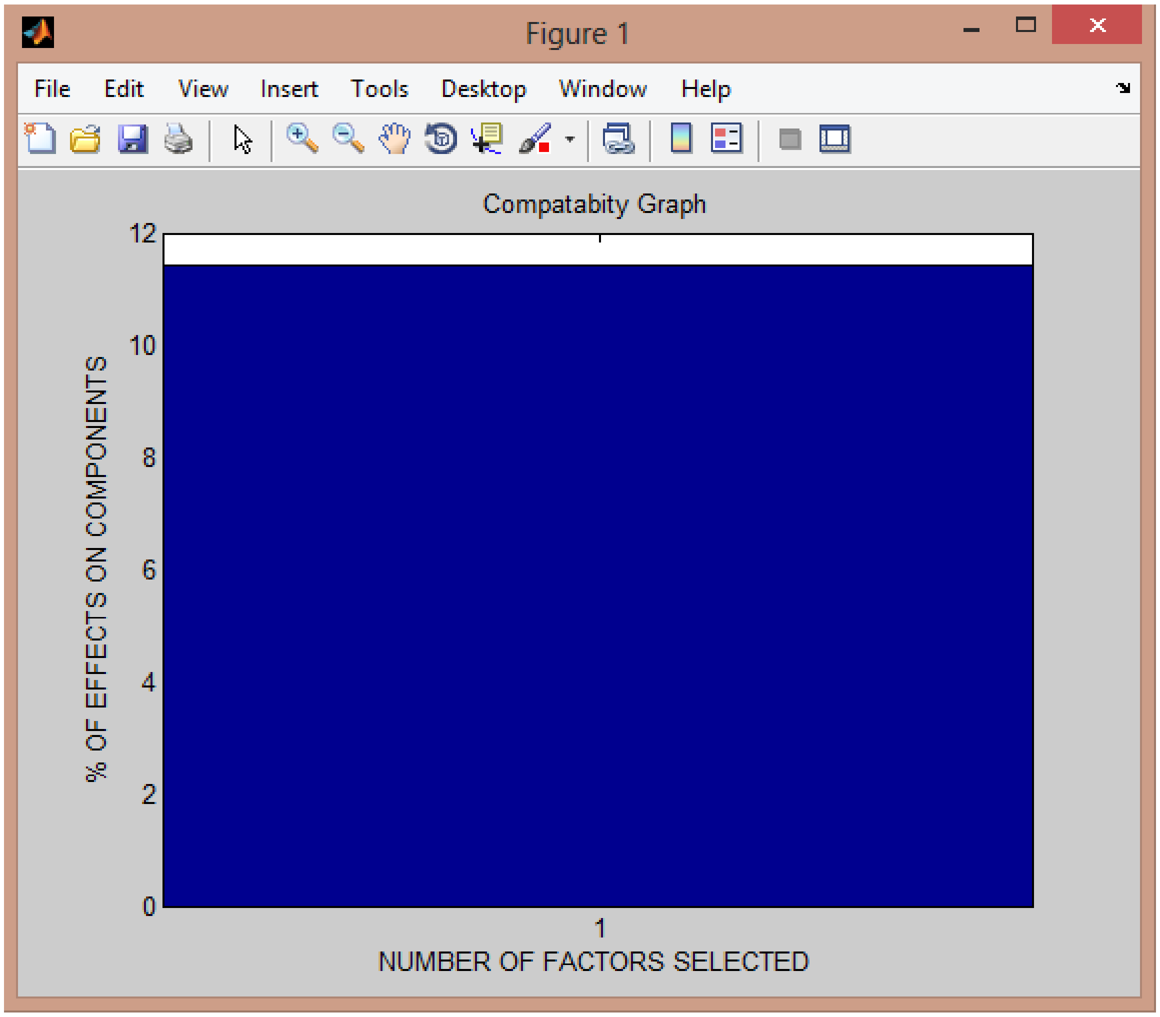


Fig 4.5.9: Old Plot

This figure shows the bar graph of knowledge based learning rule after calculating the compatibility of components using the same

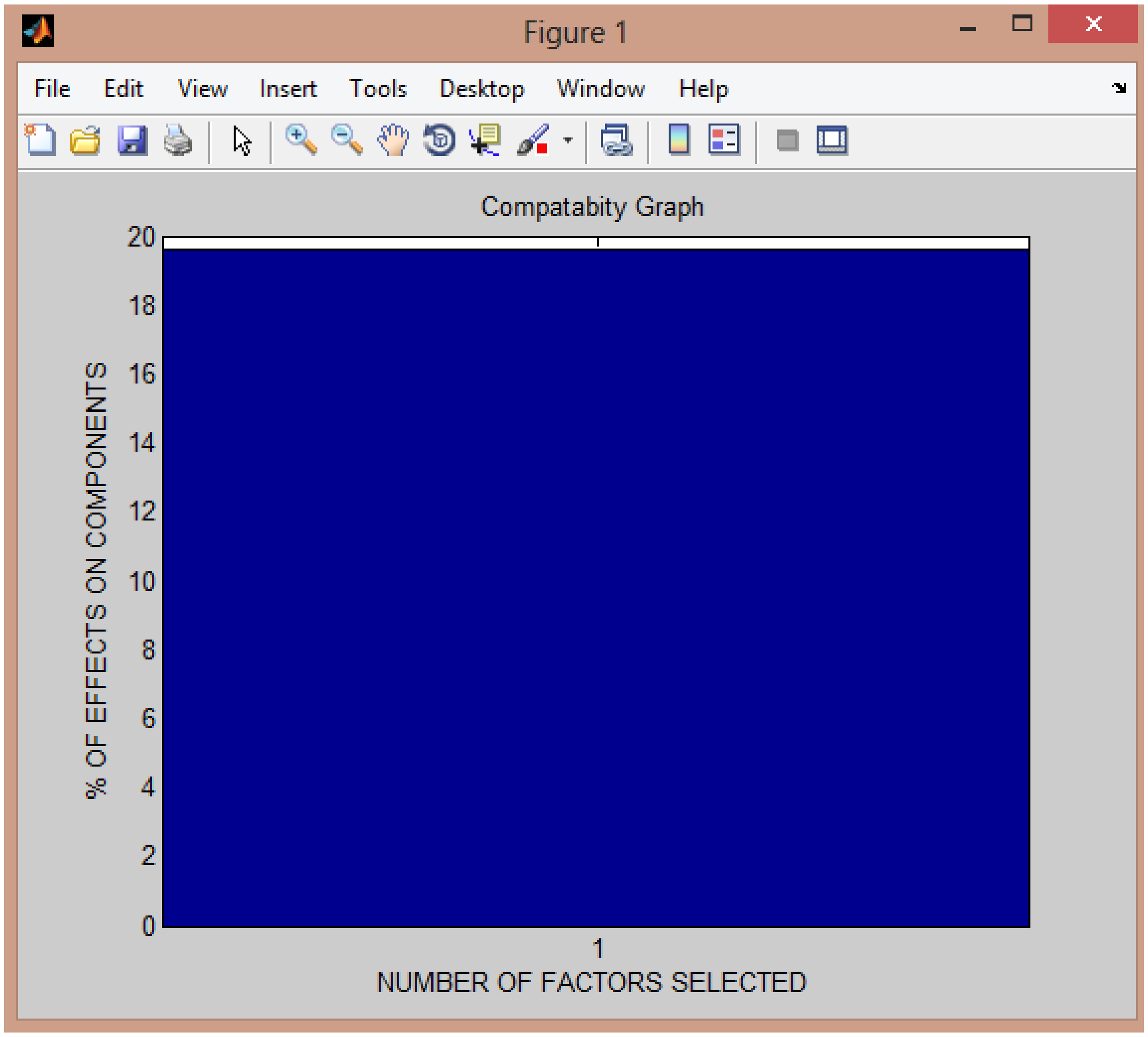


Fig 4.5.10: New Plot

This figure shows the bar graph of Boltzmann learning rule after calculating the compatibility of components using the same

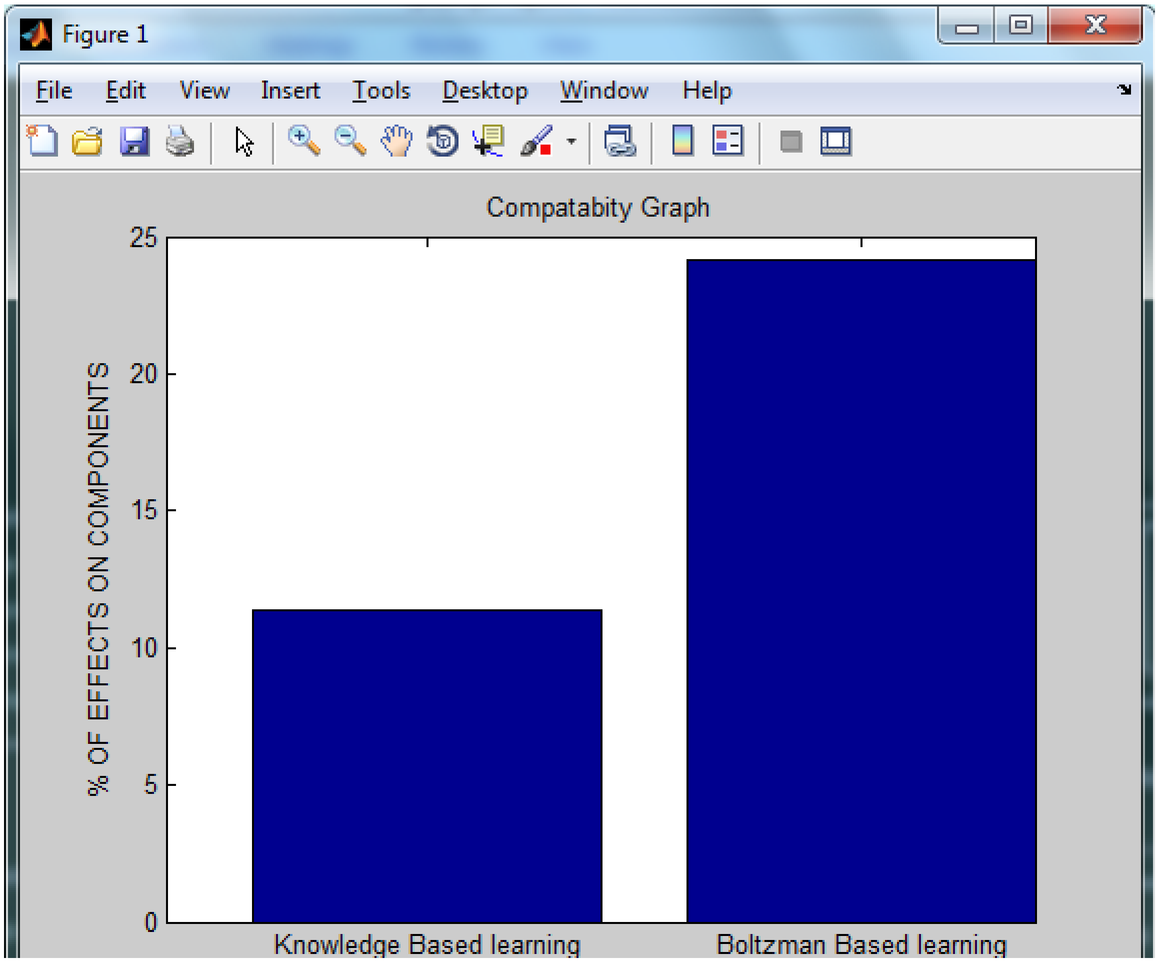


Fig 4.5.11: Comparison Graph

As illustrated in figure 4.5.11, the techniques of knowledge based learning and boltzmann learning technique is applied and in figure compatibility of the two techniques are shown graphically.

Chapter 5

Conclusion and Future Work

The Components are the building blocks of software systems. Component is a primary functional unit and the compatibility of components is a big issue in the CBSE. Because when a software is developed, its components should match in terms of compatibility and resources even when the environment of the components differs. If the software developers selects the components manually and they are unable to check the functionalities of those components. It also leads to a problem called compatibility problem. If two components want to communicate with each other, the services of these components must be compatible.
The component based software modules are independent modules which are plug and play by nature. The component based software modules are integrated when required and the basis of compatibility. To analyze the compatibility of the component based software modules technique of neural network had been proposed. In the previous times, knowledge based learning had been applied to check compatibility of the software modules. In this work, technique of Boltzmann learning is applied to analyze compatibility of the software modules. The simulation results shows that Boltzmann learning perform well in terms of compatibility testing as compared to knowledge learning.

FUTURE WORK

In the future work, Boltzmann learning technique will be applied on the aspect ratio programming to test compatibility of aspect ratio programming software modules.

Chapter 6

List of References

I. BOOKS :

[1] Dr.-Ing. Michael Eichberg Components and Component-based Software Development Introduction

[2] Large-Scale, Component-Based DevelopmentAlan W. Brown Publisher: Prentice Hall PTR First Edition May 30, 2000

II. RESEARCH PAPERS

[3] A. Mandal "BRIDGE: A Model for Modern Software Development Process to Cater the Present Software Crisis"2009

[4] A.S O'Fallon "Component Based Software Engineering: Qualification of Components during Design" 2004.

[5] Dr. RW. Jensen "An Economic Analysis of Software Reuse" Software Technology Support Center December 2004

[6] F. Fouquet, B. Morin,V.Grassi et al. "A dynamic component model for cyber physical systems", CBSE, 2012.

[7] G. Blair, N. Bencomo and R B. France,"Models@Run.Time", 2009, IEEE Digital Library.

[8] G.Gossler , J.Sifakis "Composition for Component-Based Modelling"2005

[9] IG.Herrera, J.Bourcier, E.Daubert, et al.“Scapegoat: an Adaptive monitoring framework for Component-based systems”, 2014, IEEE Digital Library.

[10] K.Kaur, J.Bedi, and H. Singh “Towards a Suitable and Systematic Approach for Component Based Software Development”,World Academy of Science, Engineering and Technology 2007

[11] KH.Cho, T.Raiko, and A. Ilin **“**Deep learning and Boltzmann machines”,2012

[12] Kuljit Kaur Chahal,Harpeep Singh,”A metrics Approch to Evalutate Design of software Components”

[13] L. Baresi, E.Di Nitto, and C. Ghezzi“Toward open-world software: Issue and challenges”, October 2013.

[14] M. Autili, P.Di Benedetto, and P.Inverardi, “A hybrid approach for resource-based comparison of adaptable java applications”, Science of Computer Programming, 2012.

[15] M.Gunestas, “A Study on Component Based Software Engineering”, January 2005.

[16] Michelangelo Diligenti, Marco Maggini, and Leonardo Rigutini, of title “Learning Similarities for Text Documents using Neural Networks”, 2013

[17] Microsoft Corporation, The Component Object Model Specification, v0.99, 1996.

[18] Ms. Sonali. B. Maind , Ms. Priyanka Wankar, of title “Research Paper on Basic of Artificial Neural Network” , January 2014

[19] Murali Krishnan Gunasekaran "Component-Based Software Engineering – New Paradigm of Software Development,Ivica Crnkovic", [2012]

[20] Nikolay K. Diakov, Marten van Sinderen, Dick Quartel "Monitoring Extensions for Component-Based Distributed Software"

[21] Pascal Poizat, and Gwen Sala¨un "Model-Based Adaptation of Behavioural Mismatching Components,Carlos Canal"

[22] Q.Wang, J.Shen, et al"A Component-Based Approach To Online Software Evolution", 2004

[23] R.Salakhutdinov and G.Hinton,"Deep Boltzmann Machines", 2009

[24] S.Becker, H.Koziolek, et al."The Palladio Component Model for Model-Driven Performance Prediction", 2010.

[25] Sanjay Misra, Ibrahim Akman and Murat Koyuncu, " An inheritance complexity metric for object-oriented code, A cognitive approach", [2011]

[26] W. Binder, J. Hulaas, et al.,"Platform-independent profiling in a virtual execution environment" January 2010.

III. LINKS:

[27] http://en.wikipedia.org/wiki/Artificial_neural_network

[28] http://en.wikipedia.org/wiki/Component-based_software_engineering

[29] http://en.wikipedia.org/wiki/Software_engineering

[30] http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/Ch19.pdf

[31] http://www.codeproject.com/Articles/175777/Financial-predictor-via-neural-network

[32] http://www.cs.ccsu.edu/~stan/classes/CS530/Slides/SE-19.pdf

[33] http://www.tutorialspoint.com/matlab/matlab_variables.htm

Chapter 9
Appendix

List of Abbreviation:

AF	Activation function
ANN	Artificial Neural Network
BM	Boltzmann
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Of The Shelf
EJB	Enterprise Java Beans
KBL	Knowledge Based Learning
LC	Linear Combiner
SOA	Service Oriented Architecture
SW	Synaptic Weights