Risk Management in Software Development using Artificial Neural Networks

Amrita Gandhi
Computer Dept., VESIT.
Mumbai-74, India.

Ajit Naik
Computer Dept., VESIT
Mumbai-74, India.

Kapil Thakkar
Computer Dept., VESIT
Mumbai-74, India.

Manisha Gahirwal
Computer Dept., VESIT.
Mumbai-74, India

ABSTRACT

IT industry is one of the biggest industries around the world with several software projects being developed which vary in size, cost, complexity, etc. During development, many risks of different types arise such as lack of staff experience, new technologies, budgets, etc. These risks play a huge role in success or failure of a project. Most of the available risk management solutions are too costly and time consuming. There is a need for an efficient risk management technique. To assist the project manager in risk management, we have developed an application which will identify the risks involved during software development and predict the success or failure of the project using Artificial Neural Networks. The prediction will be done using historical data taking the important and common risk factors into account. After risk identification, the probability of success or failure will be determined and suggestions for risk mitigation will be provided for the project. This application will help the project managers in carrying out risk management activities efficiently.

General Terms

Risk Management, Machine Learning

Keywords

Risk Identification, Neural Networks, Backpropagation

1. INTRODUCTION

IT projects are much more risky than we think. A study conducted by Flyvbjerg and Budzier [1] showed that amongst the 1400+ projects surveyed, on an average 27% were over budget, 16% of the projects could easily be categorized as Black Swans. A black swan is a highly improbable event with three principal characteristics: It is unpredictable; it carries a massive impact; and, after analysis it appear less random, and more predictable, than it was. These projects had a cost overrun of over 200% and were late by almost 70%. These were the projects which could cause stunning failures and bring down companies. Presently, there are few software or tools available which assist the developers in determining the impact of these risks in the development lifecycle. Software Development life cycle consists of well-defined steps such as Requirement Analysis and Specification, Design, Coding, Testing and Maintenance. Different type of risks are associated with each of these steps. So, a tool is developed which will help to determine the impact of the risks on the project being developed, so that project failures due to these risks can be minimized. This tool can be used in any phase of the software development process and any number of times, to find out the risk factors which will highly impact the project in the future, if proper actions are not taken to mitigate them.

1.1 Risk Management

Risk management is the identification, assessment, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities [2].

Risk management is the process where first risk identification and calculation of its impact takes place, then a strategy is developed to control those risks. Also we continuously monitor the steps taken to control it. The steps in risk management process are:

1. Risk Identification: Risk Identification is the activity where the all the potential risks which can affect the development of project are determined. For that risk check list can be created.

2 Risk Analysis: It is the understanding of when, where, and why risk might occur, through direct queries to stakeholders about the probability and impact of risk elements.

3 Plan: In this stage, a strategy is made to minimize the impact of the risk. Proper steps are defined to manage risks.

4. Monitoring: The steps taken to avoid or minimize risk are continuously monitored so that risk does not go out of control.

5 Control: The planned actions are carried out if the risk occurs.

1.2 Artificial Neural Networks

A neural network is a parallel distributed processor made up of processing units that is inspired by the way biological nervous systems, such as the brain, process information. An ANN consists of several layers of computing elements called nodes. Each node receives an input signal from other nodes or external inputs and after processing the signals, it outputs a transformed signal to other nodes or final result. The first layer is called the input layer where external information is received. The last layer is called the output layer where the network produces the solution. In between, there are one or more hidden layers which are critical to artificial neural networks to identify the complex patterns in the data. The transfer (activation) function translates the input signals to output signals.

The reasons for using ANN are as follows:

1. It has the ability to learn how to do tasks based on data given for training.

2. It has the ability to generalize i.e. produce reasonable outputs for inputs it has not been taught how to deal with.

1.3 Back propagation Algorithm

Back propagation is a supervised learning method, and is a generalization of the delta rule. It requires a dataset of the desired output for many inputs, making up the training set. It requires that the activation function used by the artificial neurons be differentiable. [3] It finds a way to train multi-layered neural networks such that it can learn any arbitrary mapping of input to output. It is simple to implement.
The algorithm is based on the error correction learning rule. It consists of 2 phases:

Forward phase: Weights are fixed. Input signal propagates through the network, layer by layer, until it reaches the output.

Backward phase: Error signal propagates backwards. Obtained output is compared with the desired one.

1.4 Selection of Risk factors

There are several risk factors which affect the software development process. But after taking a literature survey [4][5][6], we have considered 10 most effective and common risk factors for our project as follows:

1. Changing Scope / Objectives / Requirements.
2. Misunderstanding the Requirement.
3. Failure to Manage End User Expectations.
4. Introduction of New Technology.
5. Lack of Required Knowledge / Skills in Project Personnel.
6. Insufficient / Inappropriate Staffing.
7. Inadequate Documentation.
8. Scheduling and Planning.
9. Level of Complexity.
10. Budget.

2. IMPLEMENTATION

The flow of the project is shown in figure 1. Initially, the risk factors values will be calculated by using a questionnaire and risk exposure values. The calculated risk factors values will be applied to a trained neural network. The neural network will calculate the success rate of the project. Also, using the calculated risk factors values, the factors with high risks will be identified and mitigation steps to reduce the risk will be provided.

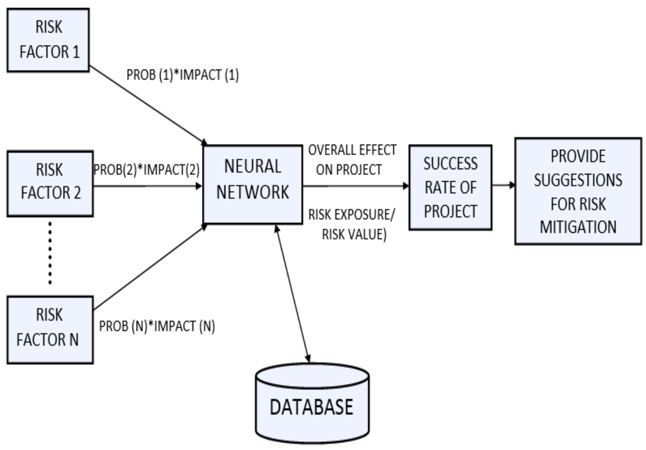


Fig 1: Overview of the architecture

2.1 Creation of questionnaire and calculation of risk factor value

A questionnaire was created which consisted of questions related to the selected 10 risk factors. These questions were selected from the extensive taxonomy-based questionnaire [7]. Each question needs to be answered in YES/NO format as shown in figure 3. Each question is assigned a weight. These weights were determined by carrying out a survey. In the survey, the project managers were asked to rate each question's importance out of 10. The average of the ratings assigned to each question by different project manager was taken to calculate the weight of that question. There are two types of questions: positive and negative questions. Positive questions are those questions which when answered as YES, contribute to the risk. Negative questions are those questions which when answered as NO, contribute to the risk. The value of a risk factor is calculated as follows:

Risk Factor Value= Q1*W1+Q2*W2+...+QN*WN
Where QN= value of question N and WN = weight of WN and so on.
For positive questions, QN=1, if answer=YES. QN=0, if answer=NO.
For negative questions, QN=0, if answer=YES. QN=1, if answer=NO.
The risk factor value is then mapped in the range of 1-10 as:
Risk factor Value= Risk Factor Value*10/W
Where W= W1+W2+W3+....+WN.

2.2 Generation of training data

As back propagation algorithm is used, in order to use the network, the neural network has to be trained. To train the network, we need to have historical data about the value of the risk factors and corresponding result, whether project failed or succeeded. The real data regarding to this is confidential for the company. A survey was conducted in which the impact value and probability of different risk factors were collected from project managers. Combining them with random numbers (in the range 1 to 10), tuples were generated. Each tuple represented the risk factor values for a random project. A threshold value was set to decide the success or failure of the project.

2.3 Creating the neural network

Matlab R2012a was used to create the neural network. The software consists of a neural network toolbox using which we created the neural network. The neural network has 10 inputs and 1 output.

2.3.1 Determining number of hidden layers and number of hidden nodes

In order to select the best performing neural network for the application, we tested the performance of neural networks with different number of hidden layers and different number of nodes. Initially a neural network with 1 hidden layer was created. Then, the number of nodes were varied and performance of each model was noted. Similarly, we tested the performance of neural network with 2 and 3 hidden layers. After comparing the performances, the ANN with 1 hidden layer and 10 nodes gave the best performance for the given input. Some of the results are shown in table 1

Table 1: Determining the best performing architecture

Number of hidden layers	Number of hidden nodes	Performance
1	3	0.1737
1	5	0.1823
1	8	0.2025
1	10	0.2578
1	12	0.1733
1	15	0.1741
1	20	0.2409
1	25	0.1684
2	10,10	0.2109
2	5,5	0.1859
2	15,15	0.1906
2	20,20	0.1952
2	8,10	0.1812
2	12,15	0.2075
2	20,25	0.1407

2.3.2 Training

The neural network was trained using scaled conjugate gradient backpropagation algorithm. The algorithm is a good choice for classification problems. It has lesser memory requirements and is faster than gradient descent algorithms.

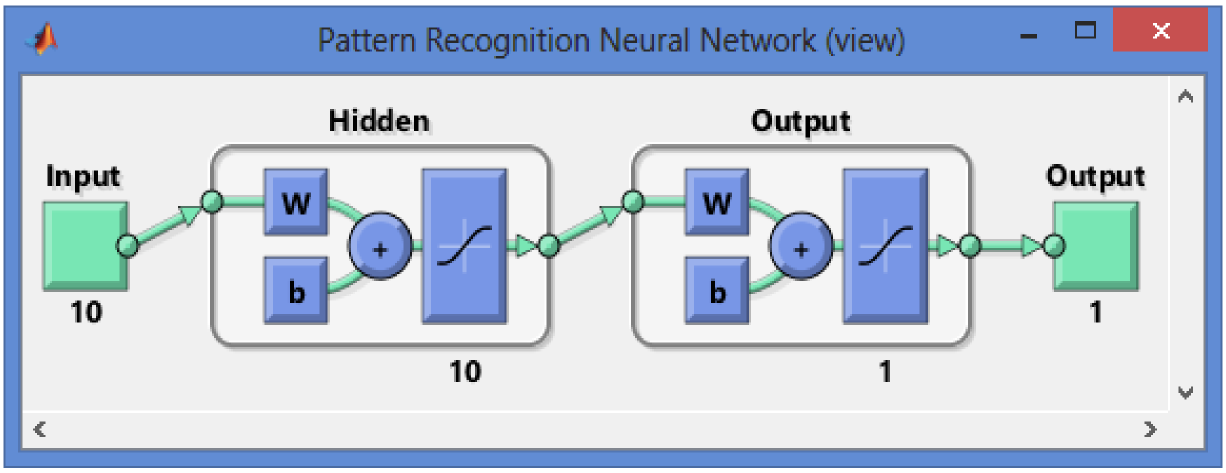
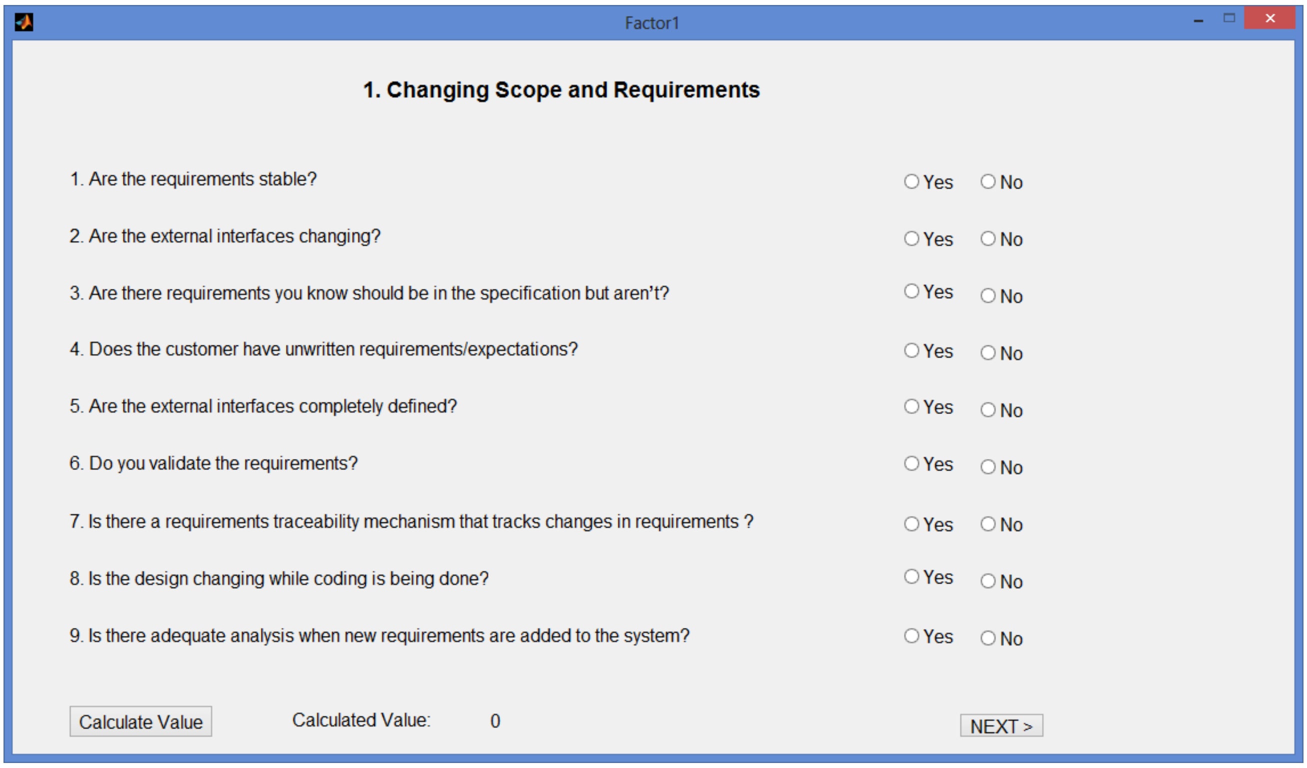


Fig 2: Neural network with 1 hidden layer and 10 hidden nodes

3. RESULTS

The output will contain the bar graph showing the success percentage of the project. Also, it will show past values of the risk factors, when project was run previously. It will help the user to keep track of the risk values in the project. At the end, depending on the risk values, all the risk factors will be highlighted in the category of high, medium and low risk values. By clicking on them, user will be able to see the mitigation steps and also depending on phase the mitigation steps will be provided.

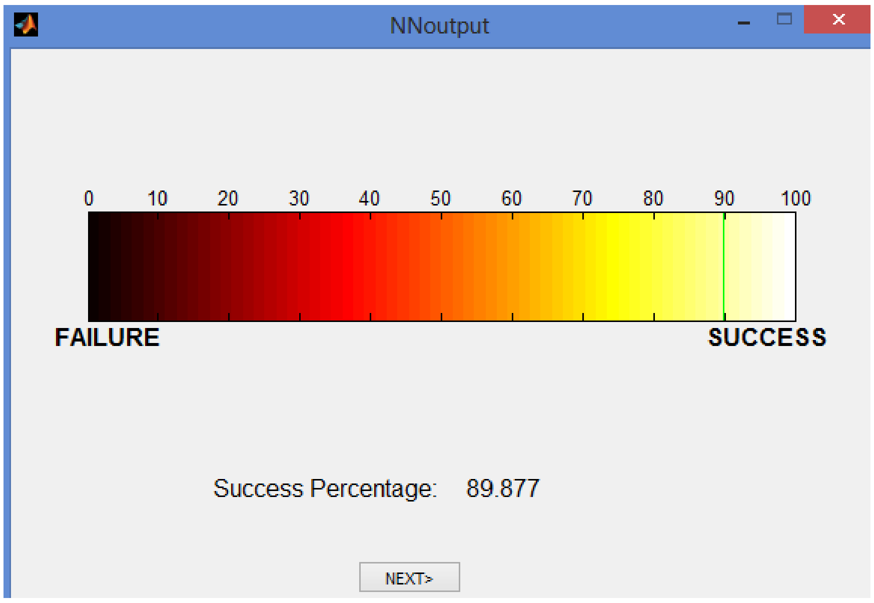

Factor1

1. Changing Scope and Requirements

1. Are the requirements stable?	○ Yes ○ No
2. Are the external interfaces changing?	○ Yes ○ No
3. Are there requirements you know should be in the specification but aren't?	○ Yes ○ No
4. Does the customer have unwritten requirements/expectations?	○ Yes ○ No
5. Are the external interfaces completely defined?	○ Yes ○ No
6. Do you validate the requirements?	○ Yes ○ No
7. Is there a requirements traceability mechanism that tracks changes in requirements ?	○ Yes ○ No
8. Is the design changing while coding is being done?	○ Yes ○ No
9. Is there adequate analysis when new requirements are added to the system?	○ Yes ○ No

Calculate Value   Calculated Value: 0   NEXT >

Fig 3: Questionnaire for risk factor- Changing scope and requirements



Success Percentage: 89.877

Fig 4: Output of neural network

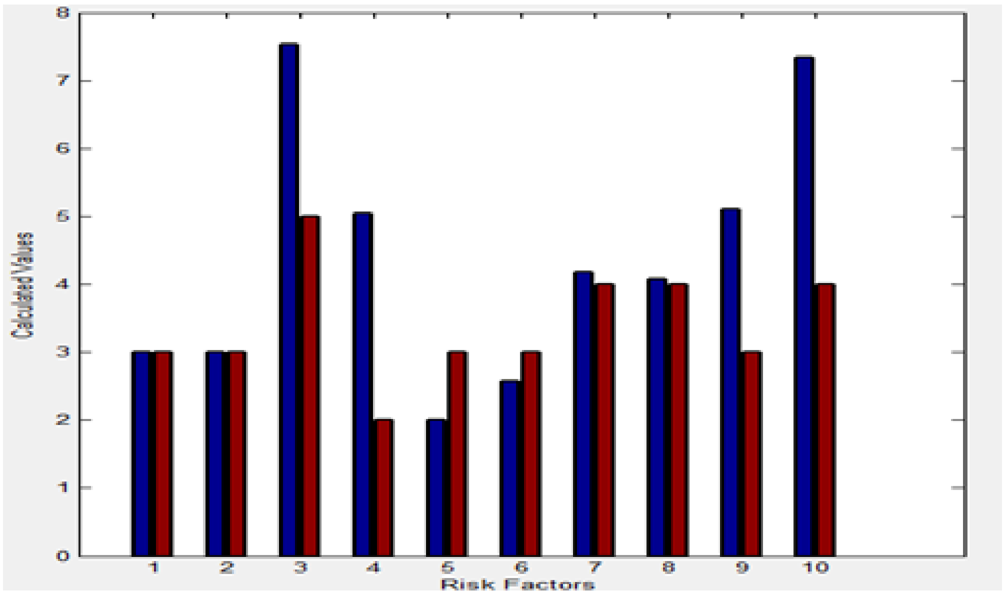


Fig 5: Comparing output with result of previous run

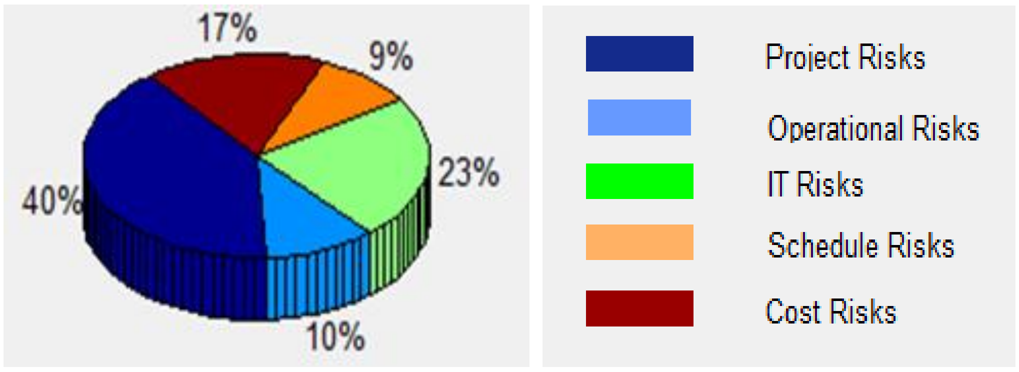


Fig 6: Categorizing the risks into various domains

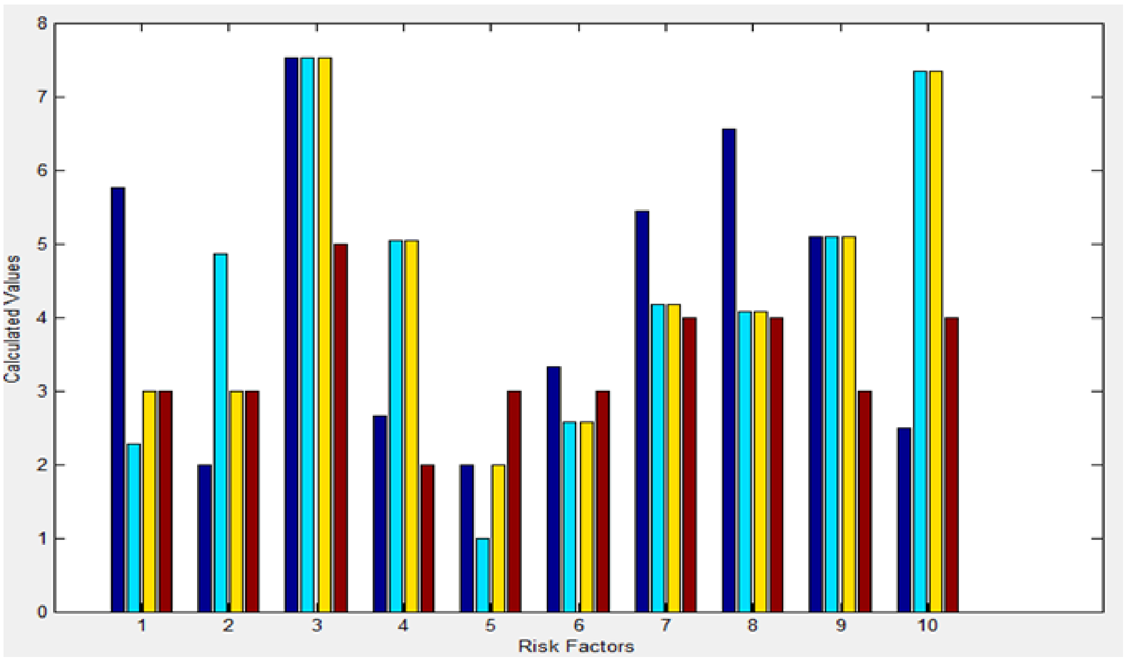


Fig 7: Graph depicting result of all the runs

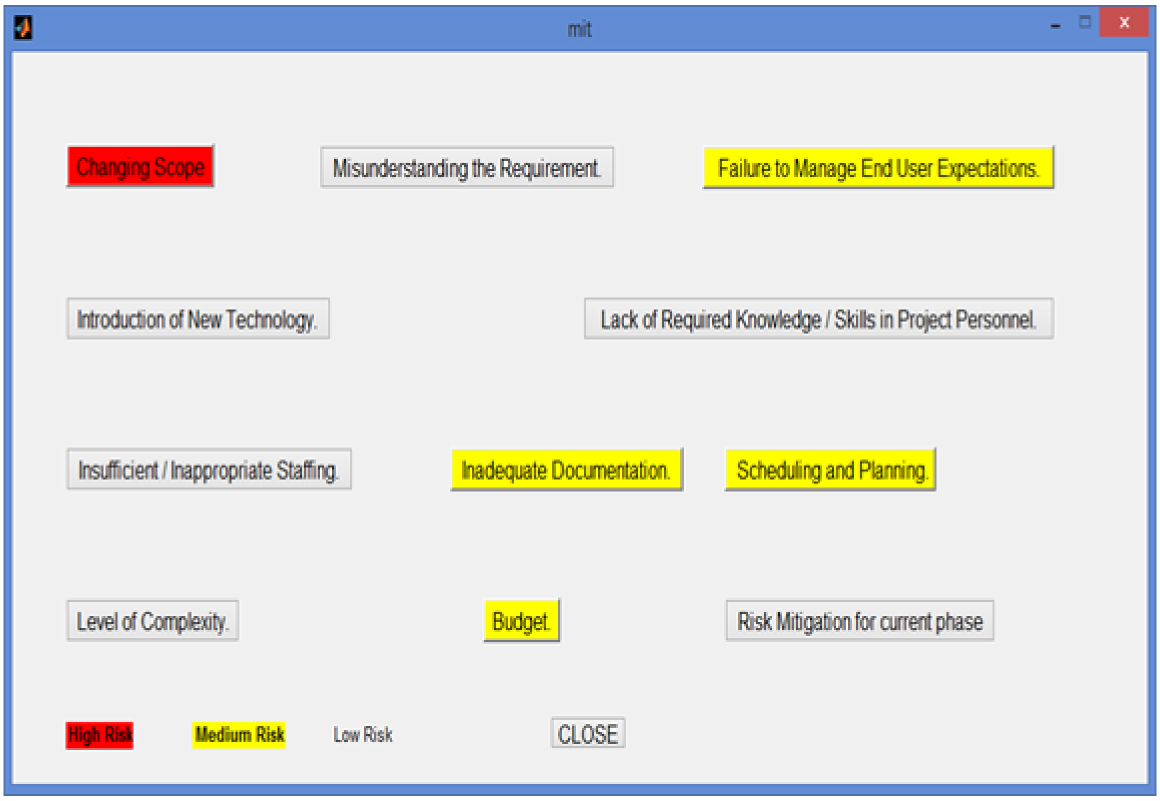
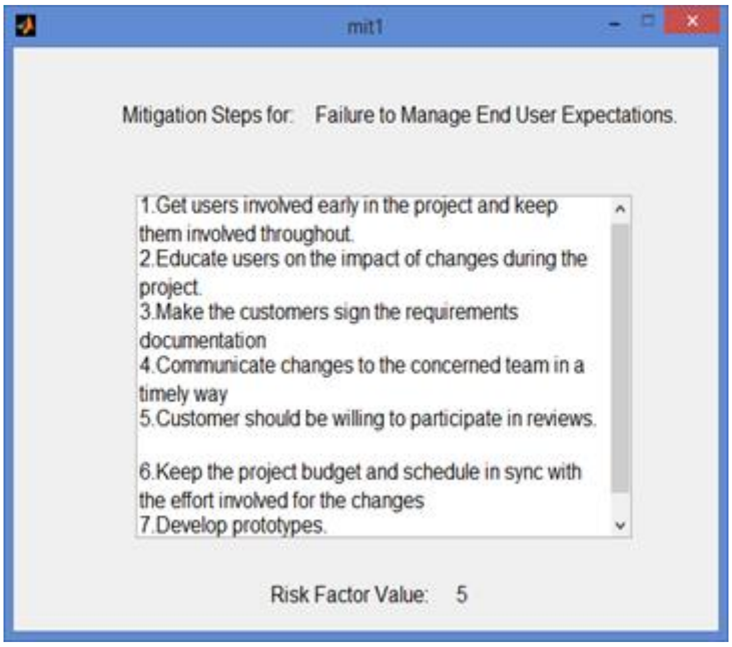

mit
Changing Scope
Misunderstanding the Requirement.
Failure to Manage End User Expectations.
Introduction of New Technology.
Lack of Required Knowledge / Skills in Project Personnel.
Insufficient / Inappropriate Staffing.
Inadequate Documentation.
Scheduling and Planning.
Level of Complexity.
Budget.
Risk Mitigation for current phase
High Risk
Medium Risk
Low Risk
CLOSE

Fig 8: Classification of risks into high, medium and low risk



Mitigation Steps for: Failure to Manage End User Expectations.

1.Get users involved early in the project and keep them involved throughout.
2.Educate users on the impact of changes during the project.
3.Make the customers sign the requirements documentation
4.Communicate changes to the concerned team in a timely way
5.Customer should be willing to participate in reviews.

6.Keep the project budget and schedule in sync with the effort involved for the changes
7.Develop prototypes.

Risk Factor Value: 5

Fig 9: Mitigation steps

4. CONCLUSION AND FUTURE SCOPE

One of the most important phases in the project development is Risk Management. Due to various risk factors, the developer and the project team may not notice some of the risks. Thus, we have successfully developed a tool that predicts the amount of risks involved in a software project, so that they can be detected well in advance and proper steps for mitigation can be taken to increase the success rate of project. So this will be an extremely useful tool for the project managers in the Risk management process.

This is a general model, which can be further extended. As 10 most common factors were considered, in practice, we can consider more factors which have a higher impact on the project. This project contains the generalized questions in software development. Similarly questionnaire can be prepared according to the type of the software developed by the company. Also, facility to add new questions or delete the irrelevant questions from the forms can be provided to the admin of the software. Time factor can also be taken into account, i.e., if risk value of any factor remains high for a long time during development process then more weightage can be assigned to it. Also, depending on the software company in which this software is used, mitigation steps can be set according to different phases of software development process.

5. REFERENCES

[1] Bent Flyvbjerg and Alexander Budzier 2011 Why your IT project may be riskier than you think., Harvard Business Review.

[2] Hubbard, Douglas (2009). The Failure of Risk Management: Why It's Broken and How to Fix It. John Wiley & Sons. p. 46.

[3] Neural Networks and Learning Machines by Simon Haykin (2009)

[4] Barki, H.; Rivard, S.; and Talbot, J. Toward an assessment of software development risk. Journal of Management Infonnation Systems.

[5] Boehm, B. 1991. Software risk management: principles and practices

[6] Moynihan. T. 1997 How experienced project managers assess risk. IEEE Software.

[7] Marvin J. Carr, Suresh L. Konda, Ira Monarch, F. Carol Ulrich, Clay F. Walker (1993). *Taxonomy-Based Risk Identification*