Identifying Risks and Possible Remedies to Mitigate Them during Systematic Software Development Process

Dr. AM Nageswara Yogi  
Professor & HOD, Department of MCA,  
T. John Institute of Technology,  
Bangalore, India  
amnyogi_jalaja@yahoo.co.in

Dr. Mala V Patil  
Professor & HOD, Department of CSE,  
HKBK Institute of Technology,  
Bangalore, India  
malapatil2002@yahoo.co.in

Abstract

Effort estimation and risk analysis are the two important branches of Systematic Software Development Process (SSDP). Wrong estimation is one of the major risk factors leading to the failure of software projects. In addition, a lot many risks have to be faced during the various stages of software development. In literature, we find the failure rate is much more than that of the success rate of software projects. Therefore, explicit and systematic risk management is essential for the successful development of software projects. In the last five decades, many risk management techniques like preparing lists of risk factors, taxonomy of risk drivers, identifying risks through brainstorming sessions and application of Delphi method have been employed. As on today, there is no universally applicable risk management technique since the environment and technologies are different for different projects. In this paper, we use data analysis approach by analyzing the data collected and captured from 30 live software projects whose sizes vary from 1.1 to 200 KLOC. Eighteen risk factors have been discussed based on the analysis of data. Possible remedies to mitigate each of these risk factors are discussed and surmised.

1. Introduction

The word risk originated from the Italian term Risicare means to dare. As per the dictionary, risk is the chance or possibility of loss/damage. Levin and Schneider[15] define risk as “events that if they occur, represent a material threat to an entity’s fortune”. Therefore, risk is an undesirable situation that presents an uncertainty of attaining a program goal, objective or requirement pertaining to technical performance, cost or support or schedule of a project. Risk is also the probability of failing to achieve particular performance, cost, support and schedule objectives and consequence of failing to meet the overall objectives. The risk exposure is the product of probability of loss and size of the software project.

Risk is composed of two components - probability of occurrence and consequence of occurrence. The severity of either of these components is directly based on the nature of the risk itself. Consequence of the risk when it occurs can run from low to high severity. For high severity risks, one must look at the probability of its occurrence. Again, high severity risks might have high or low probability of occurrence depending on the risk itself. Therefore, when determining which risk to be mitigated, we must look for both consequence and occurrence. Assigning probabilities are used to evaluate the likelihood of occurrence of hazards. Combining estimates of loss magnitude with failure probability is the usual way of evaluating risk situations [6, 3].

Software Risk is a measure of the probability of its occurrence and the loss (consequence) due to its exposure. Software risk arises due to uncertainties in management and the ineffectiveness of technical procedures. Charette[8] in his book on risk analysis and management presents the conceptual definition of risk as follows:

First, risk concerns future happenings. Today and yesterday are beyond active concern, as we are already reaping what was previously sowed by our past actions. The question is, can we, therefore, by changing our actions today, create an opportunity for a different and hopefully better situation for ourselves tomorrow. This means, second, that risk involves change, such as in changes of mind, opinion, action, or places.... [Third,] risk involves choice, and the uncertainty that choice itself entails. Thus paradoxically, risk, like death and taxes, is one of the few certainties of life.

The problem with the implementation of software systems is not changed even though the technology...
of software development changed significantly over the years and number of software packages developed/ acquired by the organizations has increased exponentially. Alter and Ginzberg [1] say that the implementation of management systems was fraught with uncertainty during late seventies. During that period it was the exception rather than rule for an organization to develop information systems within the allocated budget and as per schedule [30]. Zmud [30] also brings out that the failure rate of new implementation was very high although improved software practices have benefited the management of software development during 1960-1980. At this point of time Boehm [5] and Charette [8] introduced risk management into the mainstream of software engineering. Identifying risk factors and probabilities associated with risks [6, 1, and 2] has been viewed as steps to reduce risk. At that time quantitative and qualitative models [3] have been developed to measure the magnitude of risks. Several factors have been identified that threaten successful software implementation. Boehm gives ten factors, Alter and Ginzberg eight factors and through literature review Barki et al identified 35 risk factors. Mark Keil et al [18] choose an international approach to identify a broader set of eleven risk factors. Newman [21] cited that lack of management commitment is one of the risk factor for failures in software projects.

Change in the customer requirements, developing technologies, availability of resources, change in environment under which the developed software is required to work, inexperience of members in team and many more intangible parameters affect software estimates leading to failure of the projects. These parameters are known as risk or critical factors which have direct bearing on the cost and schedule of a software project and are known as risk drivers. A methodical analysis of these factors is essential for project success. There are hundreds of such factors that affect project development. In literature the classification of these factors is made depending on their impact on success or failure. Usually, impact is being measured as catastrophic (very high), critical (high), marginal (medium) and negligible (low) effect on the cost and schedule of the project. Experts like Charette [8], Susan [27], Pressman [23] and Waman [29] categorized risks in various ways to analyze the impact of risk factors on the software project.

Charette [8] proposed a general categorization of risks factors into known, predictable and unpredictable risks. For instance, unrealistic delivery dates, lack of documented requirements are grouped as known risks. Staff turnover, poor communications with customer and staff are predictable risks. Identification of unpredictable risks in advance is extremely difficult. The first step a project manager takes towards avoiding known and predictable risks is to identify and control factors categorized in this group. For this purpose, it is required to prepare checklists as done by Boehm [5].

Susan [27] categorized critical or risk factors based on Technical, Organizational and Environment Dimensions. As per Susan Technical risks result from uncertainty in tasks and procedure. Organizational risks occur from poor communication and poor organizational structure. Environmental dimension results from changing environment and problems with external relationships. For example if we take personnel as a risk driver, then personnel lacking necessary technical skill will be categorized as having Technical dimension. Interpersonal relationship hindering the development will be categorized as having Organizational dimension. If the personnel cannot locate or effectively manage external software development then it is categorized in Environmental dimension.

Pressman [23] categorized risk factors into Project risks, Technical risks, and Business risks. Project size, complexity, structure, personnel, resources, customer and requirements are classified as project risks. Design, implementation, interfacing, verification and maintenance problems are categorized as technical risks. Business risks are building a high-end product, loosing management support, loosing budgetary or personnel commitment.

Waman [29] classifies the risk drivers into Technology risks, People risks, Organizational risks, Requirements risks and Estimation risks. Technology risks is the risk of chosen hardware, software and network communication technology becoming obsolete. People risks concerns with capability, skill, maturity etc. Organizational risks – environment at developer as well as the users organizations, Requirements risks are clarity, completeness and confirmation (volatility) of requirements by the end user, Estimation risks are judgment errors in estimation of size, effort and schedule of the project.

In addition development of functions not required, using improper interface, beautifying interface screens, shortfall in externally performed tasks and off-the-shelf components will also lead to failure or cost/schedule overrun of a project.

All these classifications and categorization (Taxonomy) of critical risk factors have been done to analyze their impact on the software development process and to mitigate them as much as possible. Hence a project manager must spend sufficient time in choosing interface, vendors and also discuss with experts or learn from the past experience to identify, categorize and mitigate as many risk factors as possible. Therefore, risk management is essentially the
Identifying and dealing with risks early in development lessen long-term costs and helps prevent software disasters [6]. Marvin J Carr et al [19] have developed a method namely “Taxonomy – Based Risk Identification” based on their previous experience and other published literature. The method facilitates systematic and repeatable identification of risks associated with the development of software-dependent software. For software-intensive systems, effective system integration necessitates that all functions, aspects and components of the system must be accounted for along with an assessment of most risks associated with the system. Marvin Carr [19] observes that the risk management consists of two activities – make informed decisions about risk and take appropriate action to minimize the effects of these risks. Chittister and Haimes[9] said that the process of risk assessment and management is also the sine qua non requirement for ensuring against unwarranted time delay in a project’s completion schedule, cost overrun and failure to meet performance criteria. The analysis of Janne and Kalle[13] shows that awareness of the importance of risk management and systematic practices to manage risks have positive effect on scheduling, requirements management and personnel management. They have derived six software risk components such as scheduling and timing, system functionality, subcontracting, requirement management, resource usage/ performance and personnel management risks. Mark Keil et al [18] using a systematic approach tapped the experience of more than 40 software project managers across the world to identify a universal set of risk factors. The three most important risk factors were judged to be lack of top management commitment to the project, failure to gain user commitment and misunderstanding the requirements.

Critical factors that are likely to have a bearing on the success of the project are to be identified and listed. Several lists of possible risks in software projects have been compiled [5, 2, and 19] and classified [27, 9, 13, 18, 16, 23, and 29]. Risk analysis and risk prioritization rank the identified risk items by assessing the probability and severity of the loss for each risk item. Risk management planning defines how risk reduction will be conducted in a particular project by defining among other things, risk-reduction tasks, and responsibilities, activities and budget. Risk-reduction activities must take into account the process [4], organization [13,11] and technology [12]. Risk resolution produces a situation in which the risk items are eliminated. Risk monitoring involves tracking the progress of a project toward resolving its risk items and performing corrective activities.

Risk management is particularly important in software development projects due to the inherent

2. Software Risk Management

Former US Deputy Assistant Secretary of the Air Force Lloyd Mosemann [32] said: “Software is so vital to military system that, without it, most could not operate at all. Its importance to overall system performance, and the generally accepted notion that software is always inadequate, makes software the highest risk item and must be steadfastly managed... Failure to address risk has been the downfall of many Department of Defence acquisition programs. The system component with the greatest inherent risk has historically been software.” This statement strongly indicates the importance of addressing risks during the software development process.

Risk management of software projects has been the focus of many researchers over the last 50 years due to many failures in the development of the software [31]. Software risk management can be defined as “an organized process for identifying and handling risk factors; including initial identification and handling of risk factors as well as continuous risk management [10]”. Risk Management involves risk assessment and risk control [6]. Risk assessment involves risk identification, risk analysis and risk prioritization, while risk control involves risk management planning, risk resolution and risk monitoring. Therefore risk is often carried out in a number of steps: identification, analysis, planning, monitoring and mitigation [26]. Checklists and brainstorming techniques are used for risk identification. Identified risks are prioritized based on their probability of occurrence and potential impact. Plans are to be made in order to lower the effects of the prioritized risk, lower their probability or to prepare for what to do if they actually occur. In the risk monitoring step, the risks are monitored during the course of the project development and the risks are avoided in mitigation step.

One among the many ways to identify the risks is the collection of data from developed projects and projects under development and its analysis. Based on the data collected from 133 projects in Norway, Italy and Germany, Jingyue Li et al [14] concluded that off-the-shelf components did not contribute negatively to the quality of the software. Therefore a proactive and reactive risk management is necessary to identify and reduce the risks involved. It is required to analyze risk events and drivers and their impact in terms of losses.

discipline tool of determining what risk exists, which risks really affect the software project and what should be done to minimize the risks that do affect the project. Here we would like to quote Peter Drucker [22] who said, “While it is futile to try to eliminate risk, and questionable to try to minimize it, it is essential that the risks taken be the right risks.”
uncertainties that most software projects face. Project management has to anticipate risks, understand their impact on the project, and take steps to avoid them [26]. Although majority of studies have identified several risk items, different risks taxonomies may be required by different project contexts because building one complete and universally applicable risk-taxonomy is quite unrealistic [28]. Although several risk-reduction strategies have been proposed, there is a growing demand for empirical research to provide information about the effectiveness of proposed risk-reduction tools and techniques [25]. Rasmita and Rajshree[24] concludes in their paper that proper and timely risk management control can provide enormous advantage to the organization by cutting down cost and keeping project schedule. Therefore, every project manager is required to perform methodical investigation of risk in order to avoid cost and schedule overruns.

Risk management requires good deal of planning work in order to identify and develop procedures to mitigate or avoid risks. When managing a risk we need to address the cause, the effect of the risk, the possible actions and the benefit of mitigating the risk. Hence Planning, Assessment, Handling and Monitoring are four areas of risk management, which are depicted in Figure 1.

In planning phase, the objective is to determine the events causing risk. Events can be identified by understanding the causes. A cause may be lack of funds, resources limitations. The risk events can be averted through proper risk mitigation planning.

In the assessment phase the objective is to determine the impact of the risk factor on the project. The impact will be shown in cost or schedule of the project. Usually risks are related to each other. One risk may cause another risk, which in turn will cause another. If the original risk can be avoided then the chain is in turn can be avoided. If we are unable to avoid the risk, then the next step is to minimize the effects of the risk. In this way, we strengthen the project and increase the opportunity of successful completion of the project.

To develop a plan to mitigate the effect of a risk is the objective in risk handling phase of risk management. Risk avoidance, risk control, risk assumption and risk transfer are the four ways of avoiding risk. In risk avoidance the objective is to reduce the probability of risk occurring. Minimizing the impact of the risk is the goal of risk control. Acceptance of risk and providing necessary resources are essential when we assume that risk may occur. Risk transfer deals with sharing of risk with others. Examples of risk transfer are warranties, insurance, etc. Staying within cost and schedule of the project and avoiding future risks are the benefits of risk mitigation.

Software development methodologies attempt to reduce risks by gathering information and using structured processes. The implicit assumption is that by following a good methodology and identifying risk factors software risk is reduced enough to avoid failure. However a continuous stream of software failure may be a limit that there are risks that cannot be overcome by traditional approaches.

In this paper we discuss an approach to risk analysis by data analysis process using data collected by distributing a questionnaire to project teams.

3. Risk Analysis by Data Analyses

The process of collecting software project data must be started from the beginning of the software project itself. The data thus collected is required to be modified and refined as the project progresses. A reasonably accurate data helps in validating the software development methodologies and also for decision making by project managers. Based on the objectives of our study, questions were framed which led to the development of a questionnaire to capture the developers’ perception of the project undertaken in terms of risk and complexity involved, size etc. The questionnaire was corrected and improved from time to time based on the lessons learnt from various answers from developers. The final questionnaire was distributed to more than 60 teams working at different organizations for collecting relevant data of the software projects undertaken by them. The authors explained the goals and objectives of data collection to team leaders and team members and requested them to update the information as and when it changes and reasons for the changes. In addition team leaders and members were requested to indicate or list out the problems and risks faced by them. It was gratifying to
note that 30 out of 60 teams returned the questionnaire with the most relevant and useful data as per our requirements[17] which is given in Table 1. In this table for sake of confidentiality, the Project Name (PN) is given code numbers P1 to P30 in column 1. Number of Persons in Team (NPT) is given under column 2. Front End (FE) tool and Backend Database (BD) are given in columns 3 &4. KLOC written to complete the software project is given in column 5. Actual Development (AD) time in Person-Months (PM) taken for the development of each project is given in column 6. Each of the teams have identified and mentioned a number of risk factors, however we have chosen those factors, the team has given maximum emphasis and weight-age. These risk factors are given under the column 7 of Table 1.

Out of 30 projects 21 (70%) projects have chosen Java as front end tool, 4 (10 %) have chosen c#.net, 2 have chosen VB 6 and one each have chosen VB.net, Eclipse, and ASP. Similarly 20 (around 70%) have chosen SQL as backend database, 8(27%) have chosen MSACCESS and 2 projects have chosen Oracle 9i. It appears that Java with its platform independent advantage and SQL database are the most popular among the IT professional. The sizes of the projects vary from 1.1 to 200 KLOC. Number of persons in each team varies from 2 to 15. Actual development time varies from 6 PMs to 157 PMs. Analysis of such a data with vast variations will be interesting.

Some of interesting facts that can be highlighted from table 1 are:

a) The project P1 is of the smallest size with 1.1 KLOC, but three members have taken 11 PMs to complete the project. Compared to the projects P3, P6, P13, P14, P23, and P27 with 3 persons as team members, the project P1 has taken sufficiently more time to complete. The risk factors emphasized by the team are effort estimation and guidance.

b) The project P18 had smallest team with 2 team members and was completed in 6 PMs. The risk factor emphasized by this team is ‘Design’. This is reasonably successful project since 2 members were able to successfully develop the project size of 6 KLOC within 6 PMs.

c) Consider the projects P3, P13 and P14 with 3 members in each of the teams. P3 and P14 have chosen Java as their front tool and p13 has chosen VB.net as front end tool. All three have chosen SQL as their backend database. P13 has taken more time to complete the project. This team has expressed concern about the training requirements as a major hurdle. We infer that the project manager might have given sufficient time for training the team members. Our presumption is that if the team had chosen Java they would have completed the project early.

d) Surprisingly only one team (project p17) has emphasized about lack of management support. We thought that many may express their concern about management support.

e) The project P30 is the largest of all with 200 KLOC developed by a team of 10 members. The risk factors identified by this team are Design, Team skills and Resources. The actual development time is 75 PMs. We conclude that the project is better managed compared to projects P28 and P29 which are also comparatively large projects with sizes 100 and 120 KLOC with teams of 15 and 9 members respectively.

f) Consider the project teams P22, P23 and P24. These teams have taken 23, 14 and 42 PMs to complete the projects of size 7, 10 and11 KLOC with 5, 3 and 6 team members respectively. The team manager of P23 has expressed concern about progress reporting. Even then the team of 3 members have completed the project of size 10KLOC in 14 PMs. This indicates that the team skills are extraordinary. Whereas the project P24 has emphasized that the resources and documentationas the major risk factors. This may be the reason that the 6 members team has taken 42PM to complete the project of size 11 KLOC.

g) The project team P26 with 6 members has emphasized that inspection from user wasthe major concern which may be the reason that the project of size 27 KLOC was completed with a huge effort of 157 PMs. User may be dilly dollying for longer time. The project even though completed might have been a nightmare for the project manager.

h) The project team P27 with 3 members is the best coherent team with each one committed to their tasks. A project of size 50 KLOC has been completed in a record time of 18 PMs. The risk factor they have emphasized is test plans. The skill level of all the three members was so high that nothing was impossible for them.

i) The projects P25 to P30 are reasonable large projects. The main observation here is that more number of team members does not mean that the project will be completed early. For example the teams P27 & P30 consisting of 3 and 15 members have completed the projects faster than 6 and 16 member project teams P26 and P29. Here we mention Brooks Law [7] which states that “Putting more people on a late project makes it latter.” Coordination and communication with more people may lead to occasional idle period for some team members. Similarly more the time allotted for the project completion, the team (staff) will work less
hard. Parkinson law states that “Work expands to fill the time available.”

It is our presumption that most of the smaller projects may be prototypes to understand user requirements or sub-modules for bigger projects. Now we discuss and surmise various risk factors and possible remedies in the following paragraphs

3.1. Management commitment

The commitment of management is very important. If this is not there, then there is no point in taking up the project. The project team will not be able to address any of the issues and it is impossible to do any work. Mark Keil et al [18] chose an international approach to identify broader set of risk factors and explored the possible cultural differences relating to the identification and ranking specific risk factors through Delphi method. The panelist chose the term “management commitment” instead of “management support”. This shows that the experts chosen for the Delphi method indicated the strong-active role the top management must play from the initialization to the implementation of the project.

3.2. Understanding user requirements

It is required to check whether requirements stated by the customer are well understood by the development team or not. It is very important for the developers to understand what the customer wants. First step in this direction is analyzing the requirements as stated by the customer for consistency. To overcome the risk of not understanding requirements, it is required to arrange group discussions with users and development team as many times as required. Recording minutes of meetings between the developers and customers will also benefit in saving time and helps in avoiding discussing the same details many times. It is certain that users during the discussions usually will come up with additional or modifications to the requirements. At some point of time both users and developers must freeze the requirements. Developing team should not be allowed to think that there was lack of coordination between the users and them. Prototyping will help in overcoming this risk. A prototype is worth one million words. Written requirement specifications

**Table: 1 Data collected about software projects**

<table>
<thead>
<tr>
<th>PN</th>
<th>NP</th>
<th>FE</th>
<th>BD</th>
<th>KL</th>
<th>QC</th>
<th>AD (PM)</th>
<th>Risk factors emphasized</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3</td>
<td>Java</td>
<td>SQL</td>
<td>1.1</td>
<td>11</td>
<td></td>
<td>Effort estimation, guidance</td>
</tr>
<tr>
<td>P2</td>
<td>4</td>
<td>Java</td>
<td>SQL</td>
<td>1.2</td>
<td>12</td>
<td></td>
<td>Data</td>
</tr>
<tr>
<td>P3</td>
<td>3</td>
<td>Java</td>
<td>SQL</td>
<td>1.2</td>
<td>11</td>
<td></td>
<td>Guidance</td>
</tr>
<tr>
<td>P4</td>
<td>4</td>
<td>c#.net</td>
<td>SQL</td>
<td>1.5</td>
<td>12</td>
<td></td>
<td>Motivation</td>
</tr>
<tr>
<td>P5</td>
<td>4</td>
<td>Java</td>
<td>SQL</td>
<td>1.5</td>
<td>16</td>
<td></td>
<td>Design</td>
</tr>
<tr>
<td>P6</td>
<td>3</td>
<td>Java</td>
<td>MSA</td>
<td>1.7</td>
<td>12</td>
<td></td>
<td>Changes in req.</td>
</tr>
<tr>
<td>P7</td>
<td>4</td>
<td>Java</td>
<td>SQL</td>
<td>2</td>
<td>12</td>
<td></td>
<td>Implicit req.</td>
</tr>
<tr>
<td>P8</td>
<td>4</td>
<td>Java</td>
<td>MSA</td>
<td>2</td>
<td>14</td>
<td></td>
<td>Understanding req.</td>
</tr>
<tr>
<td>P9</td>
<td>4</td>
<td>Java</td>
<td>MSA</td>
<td>2.3</td>
<td>12</td>
<td></td>
<td>Data</td>
</tr>
<tr>
<td>P10</td>
<td>4</td>
<td>Java</td>
<td>SQL</td>
<td>2.5</td>
<td>16</td>
<td></td>
<td>Tech. issues</td>
</tr>
<tr>
<td>P11</td>
<td>4</td>
<td>Java</td>
<td>MSA</td>
<td>2.7</td>
<td>14</td>
<td></td>
<td>Effort est.</td>
</tr>
<tr>
<td>P12</td>
<td>4</td>
<td>Java</td>
<td>MSA</td>
<td>3</td>
<td>16</td>
<td></td>
<td>Team expertise</td>
</tr>
<tr>
<td>P13</td>
<td>3</td>
<td>.net</td>
<td>SQL</td>
<td>3.7</td>
<td>18</td>
<td></td>
<td>Training req.</td>
</tr>
<tr>
<td>P14</td>
<td>3</td>
<td>Java</td>
<td>SQL</td>
<td>4.5</td>
<td>14</td>
<td></td>
<td>Team expertise</td>
</tr>
<tr>
<td>P15</td>
<td>4</td>
<td>Java</td>
<td>SQL</td>
<td>5</td>
<td>14</td>
<td></td>
<td>Training</td>
</tr>
<tr>
<td>P16</td>
<td>4</td>
<td>c#.net</td>
<td>MSA</td>
<td>5</td>
<td>14</td>
<td></td>
<td>Responsibilities</td>
</tr>
<tr>
<td>P17</td>
<td>4</td>
<td>Eclipse</td>
<td>SQL</td>
<td>5</td>
<td>20</td>
<td></td>
<td>Mgt. commitment</td>
</tr>
<tr>
<td>P18</td>
<td>2</td>
<td>Vb6</td>
<td>SQL</td>
<td>6</td>
<td>6</td>
<td></td>
<td>Design</td>
</tr>
<tr>
<td>P19</td>
<td>4</td>
<td>Java</td>
<td>SQL</td>
<td>6</td>
<td>16</td>
<td></td>
<td>Communication, documentation</td>
</tr>
<tr>
<td>P20</td>
<td>4</td>
<td>Java</td>
<td>MSA</td>
<td>6.5</td>
<td>12</td>
<td></td>
<td>Tech. issues</td>
</tr>
<tr>
<td>P21</td>
<td>4</td>
<td>Java</td>
<td>MSA</td>
<td>7</td>
<td>14</td>
<td></td>
<td>Tech. issues</td>
</tr>
<tr>
<td>P22</td>
<td>5</td>
<td>Vb6</td>
<td>SQL</td>
<td>7</td>
<td>23</td>
<td></td>
<td>Test plans</td>
</tr>
<tr>
<td>P23</td>
<td>3</td>
<td>Java</td>
<td>SQL</td>
<td>10</td>
<td>14</td>
<td></td>
<td>Progressreporting</td>
</tr>
<tr>
<td>P24</td>
<td>6</td>
<td>ASP</td>
<td>SQL</td>
<td>11</td>
<td>42</td>
<td></td>
<td>Resources, doc. umentation</td>
</tr>
<tr>
<td>P25</td>
<td>5</td>
<td>Java</td>
<td>SQL</td>
<td>20</td>
<td>25</td>
<td></td>
<td>Guidance</td>
</tr>
<tr>
<td>P26</td>
<td>6</td>
<td>Java</td>
<td>SQL</td>
<td>27</td>
<td>157</td>
<td></td>
<td>Inspection from users</td>
</tr>
<tr>
<td>P27</td>
<td>3</td>
<td>c#.net</td>
<td>SQL</td>
<td>50</td>
<td>18</td>
<td></td>
<td>Testing plans</td>
</tr>
<tr>
<td>P28</td>
<td>15</td>
<td>Java</td>
<td>OR9I</td>
<td>100</td>
<td>45</td>
<td></td>
<td>Design, team skills, resources</td>
</tr>
<tr>
<td>P29</td>
<td>9</td>
<td>Java</td>
<td>OR9I</td>
<td>120</td>
<td>153</td>
<td></td>
<td>Design, team skills, resources</td>
</tr>
<tr>
<td>P30</td>
<td>10</td>
<td>c#.net</td>
<td>SQL</td>
<td>200</td>
<td>75</td>
<td></td>
<td>Design, team skills, resources</td>
</tr>
</tbody>
</table>

trying to describe the look and feel of a user interface will be nowhere near an effective user interface prototype. Demonstration of prototype to the users will help in precisely evolving and formulating the customer requirements. Our experience shows that user’s involvement during the development of GUI forms will focus the user’s attention in evolving and freezing the requirements.

3.3. Managing changes in requirements

It is very important that the developers are kept informed about the changes in user requirements. It is an essential skill for a project manager to manage appropriately the changes in requirements. Many changes in requirements during the development will confuse the developers and results in a risky project.
In literature the problem is known as “requirements uncertainty”. The users will come up with new changes in requirements as and when the developer demonstrates the software was the experience of first author during the development of many defence related software projects. Another problem that the project manager may face will be that some stubborn team members will not easily accept even the small changes in requirements during the development phase and put forth their arguments against these minute changes, whereas user will be insisting to accommodate. The project manager has to make a balancing act to convince his team members and at same time make the user understand difficulties involved in accommodating the changes during the development phase. The project manager can suggest and convince users to take up new version of the software. The risk management techniques to be adopted are prototyping, requirement scrubbing, incremental development, etc.

3.4. Implicit requirements of Users

Users do not always have sufficient domain/technical knowledge regarding their needs. The developer should try to elucidate sufficient requirements which may be required to develop quality software but not stated by the user. In addition all the developers and managers should organize and analyze the project plan and understand the objectives of project which may bring consistency in thinking level within team members and help in understanding user’s perspective.

3.5. Documentation

At each phase of the project, the required documents have to be generated at least in the draft form. Documents are the ones which show the progress of the project. The software is an invisible product the documents will give a feel about the software being developed. The risk involved here is both inadequate and excessive documentation hinder review process. Documents must be controlled appropriately by giving version numbers with date of release to avoid using old versions.

3.6. Technical issues

Many times developers in their enthusiasm will be over-optimistic about estimating technical issues. Usually over-optimism regarding the estimation tends to cause underestimation. For example, in the beginning of the project developers may not consider the need to prepare activities dealing with test plans or risks. This lands the manager in trouble at some point of time during the development. Therefore project manager must able to grasp the ground level realities and act accordingly to overcome the risk related to this type. Based on the answers to the questionnaire, we found that some teams at the beginning faced network and LAN problems. They were also in dilemma in choosing proper software platform like Windows or Linux. Once chosen it was not available in the organization and procurement was a lengthy process. Another difficulty was incompatibility of operating system to run the software. Therefore, it is the responsibility of the Management to provide the required resources both for development and training. Some members also mentioned that Client and communication problems were tackled by discussing with seniors, browsing internet and consulting system administrator etc. Such team members who take initiative will be an asset to the organization because they have made an attempt to tackle the problems themselves. Technical risks also include identifying required algorithms, knowing the availability of asoftware if any about the selected problem, understanding the differences between the existing software if any and software to be developed.

3.7. Effort estimation using analogy

Project managers usually depend on the experience obtained from previous projects for estimation of effort for the future projects. If the current project is similar in nature then risk involved in effort estimation will be minimal provided all the best practices and experience from the previous projects are considered. At the same time it is very dangerous to depend on previous projects too much with inadequate analysis. It should be remembered that unlike other engineering projects, technology and environment for software engineering projects will be changing very fast. Boehm suggests that effort must be estimated using at least three to four methods and analyze the differences. If we are able to substantiate the reasons for difference in values obtained from different methods then the project manager will be on the safer side.

3.8. Expertise

It is required to check whether skill level needed for the development has been specified and whether sufficient number of developers with required skill level can be placed in the team. Some team members will be shrewd enough to learn and practice the required skill level in short notice. Based on our experience it is observed that important tasks were neglected in the early stage of development due to the limitations of human resources having required skill levels. To avoid such a risk outsourcing can be planned in the beginning or considered at an appropriate time. Then the problems with outsourcing have to be carefully looked into.
3.9. Training

An observation from data analysis extracted for the questionnaire was that many team members were not aware of executing server software and found difficulty in installation of software etc. Many team members initially would have only theoretical knowledge. They have to be trained appropriately. It is responsibility of the project manager to arrange training in the required fields to team members. The team members can also browse internet and consult their seniors and experienced members to learn and obtain the skill levels required. This type of risk can be mitigated by arranging discussions/training on technology adoptions, analysis of user requirements, preparation of documents and communication skills etc.

3.10. Delegateresponsibilities

It is required that project has to be systematically divided into technical activities by using work breakdown structure. Responsibility for each technical activity has to be specified. Delegation of responsibilities and authorities will help in early detection of any serious risks concealed in the project. Once the responsibility is fixed then it is easy for the project manager to allot the activities to specific members having the required skill. Planning training to the members also becomes clear.

3.11. Design complexities

It requires patience, capabilities and skills to make a design based on the user’s requirements. If the design is complex the teams will feel difficulties in understanding the logic. It is always better to correlate requirements and the design before proceeding for coding. To avoid the risk of wrong design the team members must sit down together, discuss and analyze the complexity involved and document the steps. The discussions with seniors will help to some extent in mitigating this kind of risk. We found that the presence of user representative during the discussions became motivating factor for team members.

3.12. Team skills

The team members developers should have not only sufficient knowledge for the project but also command specific domain knowledge regarding the customers’ applications. Data analysis indicated that more than 50% of the team members were lagging in skills required for the project. Team members found difficulty in coding. They found difficulty in understanding the requirements. They were not sure about the concepts. To overcome such a situation the teams dedicated sufficient time for learning and practicing utilization of technology for the project. Discussing with experts will also help to some extent to mitigate this risk. The team members indicated that some of the problems were taken care of with the help of seniors. However, these activities will affect the schedule of the project. The risk management techniques to be employed are staffing with top talent, team-building, training, skill-mix etc. However, top talent persons may leave the project at any time if others offer better perks and facilities. Personnel motivations and self-interest of team members may also contribute to the project failures.

3.13. Testing

During the analysis of data and discussion with various project teams, we understood that test phases were not generated for most of the projects initially. This was due to the fact that requirements were not understood and there was no proper guidance. In some cases not generating test cases was due to unclear responsibilities also. Project manager should see that testing started with design and development phase itself. The problem can also be tackled by planning adequate time initially for preparing test cases.

3.14. Communications

In the answers to the questionnaire, a few members indicated that team leaders were so busy and communicating with them became big problem for the teams. Lack of communication will result in delay in completing the project. The project manager may be overloaded with many responsibilities. To overcome such problems it is required to develop second line leadership. In addition communication is hindered because of lack of clearly defined and accurate terms. Sometimes ambiguous or conflicting terms may be used to refer to conceptually different aspects. Conducting weekly meetings and encouraging developing and promoting second line leadership will mitigate such risks to some extent.

3.15. Resources

It is responsibility of the management to provide required resources. It is an organizational risk. Management has to take care, otherwise the teams become directionless. For example non-availability of computers and library facilities is a resource risk. Taking too much time and constraints in procurement of required resources will hinder timely completion of the project. We found that there are restrictions towards utilization of internet at the work place. A member of the management indicated that restrictions were due to misuse of these facilities by a few team members.
3.16. Guidance
It is essential that the team members must be guided properly. Lack of guidance will again result in not meeting deadlines and users’ requirements. The risk management techniques to be adopted are appointing experts from outside to review the status of the project, taking suggestions and guidance from them. In addition prototyping may also help as a guide.

3.17. Inspection and auditing
Inspection and auditing the software project development will be helpful in filling the gaps in communication between developer and users. Some of tools like groupware, walk-in through have to be utilized to save the time for development. Auditing by outsiders would be very useful in detecting the deviations from the project plan and requirements.

3.18. Progress reporting
This risk will be outcome of lack of communication, guidance and inspection. It is very important that the developers must give feedback to project manager with progress reports at regular intervals of time. Some team members do not report either due to shyness or afraid of answering the queries from the project manager. Fixing milestones for progress reporting will help in avoiding such problems. It is also required to peer review the progress at regular intervals.

4. Conclusions
We have identified and listed eighteen risk factors and discussed in detail along with possible remedies. Based on our investigations from the answers to questionnaire, experience in leading the software projects and from the literature survey, taxonomy of risk factors can be Technical, Resource, Supportability, Cost and Schedule risks by grouping the factors discussed above.

It is well known that excessive, immature, unrealistic or unstable requirements drastically affect the success of project which is a part of technical risk. The technical risk is also the risk associated with the evolution of a new design to achieve the same or greater level of performance than the previous design. The difficulties faced during specification preparation, system analysis, preliminary design, critical design, testing and troubleshooting can be grouped as technical risks. It is important to collect data from previous projects and analyze in detail to gain the ability to detect and diagnose failures in software project development activities. The responsibility lies with the project manager in reducing the technical risk.

Resource risk is the risk associated with obtaining and using applicable resources in order to successfully complete the program. Many times the resources required will be outside the authority of the project leader. The main concern is to utilize proper resources whether people or machines which are likely to affect cost and schedule. The responsibility of management in reducing resource risks is very high.

Once the developed system is in service, issues of maintainability and usability arise. Supportability issues are comprised of both technical and resource issues. A fielded software is not supported, then it is worthless because the user will not be able utilize effectively all the features provided in the software. Developer is required to provide training about the usage of the software and maintain for some specified period. The ability to field, train and maintain the developed system is very important and brings goodwill among the user community and future gains. Here again the responsibility lies with both the project manager and management. Project manager must look into the technical activities in supporting the software at the user’s site in addition to training user meticulously and also providing well written user manuals. The management must clear any administrative hurdles for smooth transfer and supporting the developed software at the user’s place.

Personnel with excellent knowledge on software design and development may not be available at the time of need. Therefore, options have to be worked out for such situations which involve cost. In addition diversion of funds allotted to other important assignments cannot be ruled out. Such a situation will adversely affect the budget and the performance of the project. These risk factors can be categorized under cost risks. Many times cost risks cannot be avoided and hence flexibility must be built in planning for this type of contingency. Technical, resource and supportability risks will have direct bearing on cost. Again the responsibility in addressing cost risk lies with both project manager and the management.

Usually a PERT (Project Evaluation Review Technique Path Method) chart is drawn to indicate parallel and sequential tasks. The chart will also indicate milestones to be accomplished. When risk elements are encountered the effect will be delay in schedule and cost. Schedule risk will be driven by technical, resource and supportability risks. Technological changes, non-availability of human resources etc. drive the schedule risk. Again the responsibility in minimizing schedule risk must be shared by both project manager and the management.

5. Acknowledgements
Authors wish to express their gratitude to Dr. Thomas P John, Chairman, T. John Group of
Institutions, Dr. S Padmanabha, Principal, T. John Institute of Technology, Bangalore and Dr. T.C. Manjunath, Principal, HKBK College of Engineering, Bangalore for their invaluable support for our investigations carried out for the preparation of this research paper.

6. References

www.ijert.org