CHAPTER 2

REVIEW OF LITERATURE

Section 2.1 presents a detailed literature review covering different facets of software development project risk. Section 2.2 describes previous research on risk management in software development projects. Section 2.3 presents some of the instruments for identification of risk items and risk management items. Section 2.4 reviews previous studies linking risk, risk management and project outcome. Gaps in literature are discussed and motivation for the present study is explained in Section 2.5. The summary and conclusion is provided in section 2.6.

Quality and success of a research is often a reflection of the time and effort invested in developing research ideas and concepts. The immediate goal of a literature survey is to determine whether the idea is worth pursuing or not. The first step of the procedure entails specifying the domain of the constructs. (Pinder, Wilkinson and Demack, 2003). This includes outlining what is included and excluded from the concept under study (Churchill, 1979). Hence this study of software project risk and risk management began with an examination of the literature.

In order to obtain a better understanding of software project risk and risk management constructs, an extensive literature review was performed. It was conducted mainly to identify those features of software development projects which researchers and practitioners have pointed out as factors that increase the riskiness of a development effort and the strategies they adopt to counter these factors. There have been a number of research studies on the issue of “risk in software development” and attempts have been made to classify them into various categories based on their similarities (Sumner, 2000). An extensive amount of literature was surveyed in order to ensure that no important factor was overlooked. In order to identify as many factors as possible, two general resources served as the basis.

First source of literature was articles within IS research which addressed the problems associated with software development projects. Majority of IS articles dealt with the types of problems that occurred in specific phases of the software
development process. These articles either used empirical data to draw conclusions as to the effects of a particular risk factor or they proposed models that hypothesized how a few of the risk factors might impact a development effort. These articles taken individually do not provide a clear picture of the spectrum of the constructs. However, they provide a clear picture of the topics which have been studied by researchers. Second source of literature was articles written by practitioners detailing their experiences with software development projects (e.g., Boehm, 1983; Burchett, 1982; Casher, 1984; Keider, 1984; Kindel, 1992). Majority of articles in this group described the author’s experiences with a particular development project, or consisted of a summary of their generalized observations from previous studies. These articles tell us about problems that appear to be encountered frequently in software development projects and how these problems can be mitigated.

2.1 SOFTWARE DEVELOPMENT PROJECT RISK

Cambridge Learner's Dictionary defines "risk" as the possibility of something bad happening. Researchers and practitioners in various domains have conducted studies on this topic. Though there are differences in perceptions and approaches to the same, an examination of literature reveals a great deal of similarity in conclusions. Typically, risk is described as some kind of an event that may or may not occur, coupled with a consequence that follows if the event occurs (Dedolph, 2003).

A simple definition of project risk states that it is a problem that has not yet occurred but which could cause loss to one’s project if it did (Wiegers, 1998). The concept of risk is associated with a number of human endeavors ranging from space exploration and company acquisition to information systems development (Barki et. al., 1993).

The classical decision theory states that risk is perceived as reflecting variations in the distribution of likely outcomes and their subjective values. Hence a risky alternative is one where the variance is large and risk forms an important factor in evaluating alternative options. Decisions are said to be taken under risk when there is the possibility of more than one outcome resulting from the selection of an option. Furthermore, it is assumed that the probability of occurrence of each
is known to the decision maker in advance. The variation in outcomes is said to be a consequence of factors which are beyond his control (Radford, 1978).

Empirical studies on how managers deal with risks show that the managers are not necessarily rational in reacting to risks. They look at a risky choice as one that contains a threat of a very poor performance (March and Shapira, 1987). Also, risk is not a probability concept; it deals with the magnitude of the bad outcome. Accordingly, managers act in a loss-aversive manner rather than a rational manner as predicted by the traditional theory. They seek to avoid risks rather than just accept them. They make fast decisions to avoid risks, negotiate uncertainty absorbing contracts, or just delay decisions if possible (MacCrimmon et al., 1984).

A software project risk points to an aspect of a development task, process or environment, which if ignored tends to increase the likelihood of software project failure (Lyytinen et al., 1993). Software project risks are a major dilemma to information systems project managers (Jiang et. al., 2000). The reasons for variations in success can be attributed to risk factors which are technical, economic and behavioural in nature (Barki et. al., 1993; Lyytinen, 1988). Such incidents pose danger to the development of a successful project leading to inadequate software operations, software re-work, implementation difficulty, delay or uncertainty (Boehm, 1991). McFarlan (1981) viewed project risks as failure to obtain all of the anticipated benefits because of implementation difficulties, much-higher-than-expected implementation time, and thus resulting in the development systems whose technical performance is considerably below estimates.

To summarize, risk has two components

1. The chance / probability that an undesirable event will occur.

2. The negative consequences / magnitude of loss because of the occurrence of this event.

Boehm (1989) defined Risk Exposure (RE) combining these two components as:

\[ RE = \text{probability of an unsatisfactory outcome} \times \text{Magnitude of loss arising from this outcome.} \]

Sherer (1994) viewed that software project risk could be estimated from the possibility of failure multiplied by the magnitude of its loss. Similarly, Rainer,
Snyder and Carr (1991) described risk as a function of the vulnerability of an asset to a threat multiplied by the probability of the threat becoming a reality.

A precise calculation of the probabilities of negative outcomes and their magnitude is necessary in order to calculate risk exposure. However, there are numerous complexities in software development that make a proper estimation of outcome probabilities hard. Hence, assessing risk via a quantitative evaluation of probabilities could be very difficult and unreliable (Kaplan and Garrick, 1981).

In lieu of estimating the probabilities of a negative outcome, an alternative method has been devised. Kangari and Boyer (1989) adopted a method of risk assessment based on the use of natural language. Accordingly people were asked to express in a natural language, the relative weight and severity of loss arising due to the identified risk factors.

Barki et. al. (1993) put forth a modified definition of software project development risk by referring to the uncertainty surrounding a project.

As risk is a potential problem, an effective step to risk management would be through proper identification of risk factors (Fairley, 1994). The process of risk analysis can be broken down into three; risk identification, risk estimation and risk evaluation. This information enables managers to take steps to avoid potential problems before they become crisis situations. The initial step in the research process is the identification of potential software risks.

The extensive literature review resulted in the identification of over 100 risk factors. The next step was to try to group similar factors together in order to get a clearer picture of the general types of software project risk factors. This resulted in the creation of 12 general types of software project risk categories.

1. Team related factors
2. Effectiveness of Project Communication
3. Project Manager Characteristics
4. Organizational Climate and Support
5. External Factors
6. Role of the user
7. Formalization of project charter
8. Project estimation and planning
9. Tools and technology
10. Requirement stability and accuracy
11. Effectiveness of Project Monitoring
12. Cross cultural and gender issues

This is a list of potential risk categories associated with software development projects which tend to increase a project’s likelihood of experiencing problems. Each of them is discussed elaborately in the following sections.

2.1.1 Team Related Factors

Team related risk items are very common in software development projects. These issues include frequent shuffling of team members, a highly diversified team, employee attrition, lack of skills among members, conflicts in the team, level of staff motivation and improper definition of responsibilities.

Employee turnover is a much studied phenomenon (Shaw, 1998). Team member turnover results in huge costs for the company, in terms of both direct and indirect costs. Direct costs are incurred on account of replacement and recruitment. Indirect costs arise out of pressure on remaining staff, costs of learning, product or service quality, organizational memory and the loss of social capital (Dess & Shaw, 2001). Considering the high costs associated with replacing IT staff and the value of their experience, companies need to devise mechanisms designed to keep IT staff longer (Mak and Sockel, 1999; Moore, 2000; Campion, 1991). Studies in an engineering services company revealed that employee turnover was noticed more among professionals who faced stress due to changing technologies and company requirements (Rao, 2006).

Studies have proved that higher levels of diversity within a team results in high levels of conflict, which is counter productive. Many projects also suffer from overstaffing where project teams are staffed with large numbers of unnecessary people who waste resources and add coordination problems (Corso, 1993). However studies conducted by Thomas (2000), Katzenbach and Smith (1993) have suggested that diversity provides an environment that encourages every individual to contribute her/his own ideas. This in turn leads to higher motivation and consequently higher quality team output.
“Team member skills” refers to the level of experience, knowledge, and skills that software development team members have. The literature has identified a range of skills that team members may not possess, but that can impact the development project. They include lack of experience and lack of software development knowledge which are critical for project success (Carmel and Sawyer, 1998; Curtis, 1988).

Team member experience is a crucial factor for the success of a project. It has been observed that those with a higher level of design skills and experience produce better design (Beaver et. al., 2006). The standard of work may turn out to be below the expected levels owing to lack of ability and experience of staff (Cooper, 1994). This lack of know – how can extend to hardware, operating systems and other software (Fuerst and Cheney, 1982). Charette (1996) observes that the presence of inexperienced personnel can lead to troubled software development projects.

The level of application experience possessed by the development team has also been identified as a critical risk factor in the IS literature. There are instances where software professionals do not possess the right skills or background to understand the business requirements or apply the right tools to model and produce the corresponding systems (Morgan, 2005). Jones (1994) identifies lack of application knowledge among project team members as a major cause for project failure. Casher (1984) argues that when project team members do not understand the application they are developing, the likelihood of software failures increases.

The quality of people in software teams is one of the most important factors in improving productivity and quality in software projects (Blackburn et al. 1996). Lack of adequate and frequent training adversely affects capabilities (Dominiak, 2006; Cappelli, 2000).

Conflicts among team members hamper the development of a software project. The reasons for conflicts may be task based, process based and relationship based (West, 1994). The most frequent source of conflict is usually disagreement over goals and work processes, which is often a result of communication gaps between team members. Such conflicts cause severe damage to team performance (Heckhausen, 1989). Collaborative problem solving
frequently leads to disagreements among individuals or groups which can lead to conflicts within an organization. Robey et al. (1989) came across instances of open confrontations among software development team members in a troubled project.

The motivation level of the project team has a direct impact on the project. There are many studies linking motivation to productivity increase. A number of factors including reward structures and performance measures affect the motivational levels (Jerome and Kleiner, 1995). Gilb (1988) placed importance on staff motivation in determining whether project goals tend to be achieved satisfactorily. A team where members have a less satisfactory salary package, where promotions are conspicuous by their absence, staff motivation tends to be low. Setting deadlines and consequences for not meeting the same, is another way of motivating employees. Mc Comb and Smith (1991) also propose that the level of motivation of staff members determines the pace of their work. Reward and recognition are considered by organizations and managers as an important element in motivating individual employees (Cacioppe, 1999). High employee turnover, which is considered to be detrimental to organizational performance, is often the result of lack of reward for good performance and lack of opportunities for career advancement (Newcomb, 1999).

Another factor that has an impact on software development risk lies in the manner in which responsibility is assigned to each team member. Several sources have mentioned that if the responsibility of each team member is not clear, the project may have problems. Projects are likely to fail if each member is unclear about her/his responsibility (Evans, Piazza and Dolkas, 1983). One of the biggest problems in software project management has been to establish a proper accountability structure (Boehm, 1979). In other words, team members are often disorganized and managed with a very vague delineation of responsibilities. Thayer, Pyster and Wood (1980) found that an uncertainty surrounding the responsibility for various project functions led to a negative impact on the success of the project.

2.1.2 Effectiveness of Project Communication

Communication is a vital issue in software development projects. The three major stakeholders – the customer who financially sponsors the project, the contractor who performs the system development and the eventual user of the
system – have to communicate regularly during the project (Deutche, 1991). Keider (1984) in his survey of MIS professionals found that ineffective communication was the single major factor which caused most number of software failures. Communication may breakdown because of different reasons including disagreement over goals and objectives, preconceived notions and opinions often leading to prejudice, semantic differences and even misunderstandings in non-verbal communication (Staehle, 1999).

Curtis (1988); Corbato and Clingen (1979) have indicated that communication problems are more likely to occur if there is a geographic separation among members. They view that projects involving multiple cities or countries are more likely to have problems. There also exist a number of management problems between the off – shore and on – site teams.

Lack of English communication skills is another major factor which affects the performance of a project team. Research conducted by India's National Index of Communication Skills shows that only 10 percent of IT applicants have adequate language skills (www.itbusinessedge.com). Good communication skills, especially spoken English skills, are mandatory and important in today's global business environment.

Organizations need to make provisions for effective intra organizational communications. There are a number of interfaces which enable the same (Malone, 1985).

Documentation is an integral part of a software system. It contains the information that is necessary to effectively and successfully develop, use and maintain a system. However in practice, the creation of appropriate documentation is largely neglected (Bayer and Muthig, 2006). Upon completion of the project, members are spread all over the company without adequate documentation of the essential components. This leads to loss of knowledge and experiences. Weiser and Morrison (1998) are of the view that project information is rarely retained in the appropriate manner by which it can be used for future reference. Project documentation should support communication during the project and address the information needs of various people— project members, project management, project supervision (Disterer, 2002). From the user's perspective, the
completeness of documentation with respect to the user’s tasks is an important indicator of the quality of software developed (Bayer and Muthig, 2006).

Teleconferencing and video conferencing systems and services are the main set of technologies developed to support group work (Sabri and Prasada, 1985). Unreliable telecommunication facilities such as slow internet connections or telephone lines, slow computer networks etc may seriously hamper the progress of work. Software companies in India face a number of telecommunication problems including shortages of telecommunication links, time lag in accessing such links, poor transmission and high cost of installation and use. Such limitation on telecommunication facilities has discouraged many foreign clients from considering offshore software development in India as a feasible option (Chakraborty and Dutta, 2001).

2.1.3 Project Manager Characteristics

The project manager plays an important role in the success of a software project (Mc Donough, 2000; Brown and Eisenhardt, 1995). The project manager should have a clear understanding of the issues, have an adequate grasp of relevant technologies and be capable in the political sphere (DeMarco, 1982).

An important component of providing a work atmosphere in which employees can perform well is the communication style of the project leader (Thacker and Yost, 1997). An effective manager understands the effects of mutual interactions between different parts of the project and steers them in the direction of continuous learning and adaptation (Augustine, Payne, Sencindiver and Cock, 2005).

Discharging the leadership tasks effectively in the challenging business environment of today requires experienced hands. It is not merely a question of following a process (Goleman, 2000). In the absence of adequate experience, the leader is unable to guide his team-mates properly. Brown and Eisenhardt (1995) and McDonough (2000) highlight how the leader’s people management skills affect project development. A project manager should benchmark himself against other leaders to measure his performance from time to time (Townsend, 2006).

Literature also mentions other risk factors such as personal bias in the selection of team members and multiple projects in hand to be completed, which adversely affect software development (Wallace, 1999).
2.1.4 Organizational Climate and Support

Organizational politics has been discussed with utmost importance in the literature over the last two decades. It is considered to be a primary component in contemporary business practices (Aronow, 2004). Organizational politics refers to behaviors that occur on an informal basis within an organization and involve intentional acts of influence which are designed to protect or enhance individuals’ careers when conflicting sources of action are possible (Drory, 1993; Porter, Allen and Angle, 1983). Several sources have indicated that political conflicts and power plays can increase software project risk and negatively affect its outcome. Jones (1994) on the basis of his experience with cancelled projects estimates that corporate politics has been associated with more than one-third of them.

There are various risk factors which are related to organizational climate. The lack of top management support has been cited as a possible risk factor in software development projects (Boehm, 1989). Ewusi – Mensah and Przasnyski (1991) have said that it is an accepted principle or maxim in IS literature and practice that senior management support is essential for the success of software project. It has also been suggested that top management is necessary for successful project development (Jarvenpaa and Ives, 1991; O’Toole and O’Toole, 1966; Walton, 1989). Sauer (1993) suggests that support should come from senior management, user managers and from the organization itself.

Organizational politics may showcase themselves in a number of ways. Fox (1982) indicates that development projects will encounter a number of problems if organization pressurizes team members to keep development costs low. Further, Burchett (1982) notes that project managers faced problems of arbitrary and unrealistic deadlines. Schedule and budget should be based on a clear assessment of achievability. Organizational issues such as team building, group dynamics, project organization, and inter organizational relations are equally important (Nash and Redwine, 1999).

An unstable and unprofessional corporate environment could add to software project risk. It manifests itself through competitive pressures which radically change user requirements, making an entire project obsolete. Research has recognized and confirmed the existence of a work culture of long working hours coupled with work intensification and heightened levels of occupational
stress. Work pressure and stress can lead to deterioration in the efficiency of staff. This can have an impact on the atmosphere in the workplace and the quality of the work produced.

Organizations with little or no experience implementing large projects face a number of risks such as not being able to access a similar industry benchmark, committing resources to an unsubstantiated plan and timeline etc. Hence they need to conduct a detailed scoping exercise which would define business and functional requirements (Kesner, 2006).

Risk is said to increase if tools and resources such as facilities, personnel, funding, finance, technology etc are not available for the project (Command, 1988). Humphrey et. al. (1991) suggests that organizations need to make development tools and expertise available to software project teams. Tait and Vessy (1988) found that if the time and funds available to complete a project are insufficient, system designers would resort to short cuts rather than undertaking the normal development procedures. This raises the risk of project failure. The limited availability of financial and managerial resources is one contributing factor to problems in software development (Anderson and Narsimhan, 1979). Keider (1984) notes that changes take place when the scope of the project is redefined, the availability of resources changes, or the cost of equipment and/or personnel changes.

2.1.5 External Factors

External factors refer to the interdependence on third parties such as vendors, which may affect the ability of the internal development team to complete a software development project on time without problems.

Boehm (1989) indicates that shortfalls on the part of sub – contractors may be a major source of risk. These include supplied components which are a poor match to a new application, mandated support tools and environments that are incompatible or poor in performance or functionality and commercial components that turn out to be immature and poorly supported.

Dependence on external agencies affects the progress of software projects (Rothwell and Dodgson, 1991). Risk associated with a software development project increases with the number of hardware and software suppliers. Products from different vendors are often incompatible and a system composed of
equipment from assorted vendors makes the diagnosis of problems extremely complex (McFarlan, 1981; Ruthberg and Fisher, 1986; Kemerer and Sosa, 1988). Schmidt et. al. (1996) concluded that lack of control over consultants, vendors and subcontractors leads to problems regarding the schedule and quality of the outcome. This problem is further aggravated when there is no legal recourse due to poor contract specification.

External risk also includes risk factors arising out of interactions with regulatory and government agencies. For example, Indian workers are facing more visa restrictions, especially in the USA which largely affects the work assigned to them (www.bizasia.com). This causes the risk of work not being completed on time.

2.1.6 Role of the User

Boehm and Ross (1989) state that many software problems surface because users are unaware of the possible results that can be achieved with information systems and this in turn leads to a set of unrealistic expectations. Research on this topic from a risk perspective found user-related risks are a key threat to successful software development projects. These risks include unwilling IS users, lack of user support, lack of user experience, user shirking responsibility, user resistance to change and sheer number of users. Failure to assess the user-related risks and to adapt management methods accordingly can be a major source of system failure (Jiang et. al, 2000).

Information systems become obsolete as the mapping between user requirements and software deteriorates. Monitoring reliability of business software can provide an indication of how well user requirements are currently mapped to the corresponding information system (Heales, 1995). The process of obtaining user requirements varies according to the situation prevailing in the organization (Newman and Sabherwal, 1991). Keider (1984) says that if no effort is made to know what client requirements are, it may adversely affect the final outcome of project. Robey and Farrow (1982) indicate that an assessment of user needs and resulting client support aids in better understanding of the system.

Other sources have also suggested that project failure may be the result of inflated or unreasonable user expectations. Ginzberg (1981) identifies five areas of expectation which are likely to be important in determining user acceptance: the
expectations pertaining to the goals and objectives of the systems, the importance of the problem being addressed, the way the system will be used, the impact the system will have on the organization, and the criteria that should be used to evaluate the system.

Lack of software experience on the part of the client is another major risk factor that affects its outcome (Barki et. al. 1993; Mc Farlan, 1981). Without adequate knowledge of the application, user requirements tend to be misstated leading to irrelevant results.

The general attitude of the client is a factor which influences project risk. Client attitude refers to unwilling users, resistance towards a new system and their feelings of responsibility (Alter, 1979; Alter and Ginzberg; 1978; Lyytinen et. al. 1993; Tait and Vessy, 1988; Andersen and Narsimhan, 1979; Charette, 1996). The users must feel a need for a system without which it cannot be successfully implemented (Guthrie 1974).

Communication obstacles between users and system developers are caused by a lack of understanding, as users and system developers have different jargons and terminologies (Junhe, 2004).

2.1.7 Formalization of Project Charter

The formalization of project charter forms an integral part of the software development process. Brooks (1995) says that defining the goals of the project is a difficult task. If goals are ambiguous and ill-defined, development problems may be encountered (Lyytinen and Hirchheem, 1987; Schmidt and Kozar, 1978). A clear cut goal is to be initiated failing which programmers would end up making their own assumptions about the project. Gaining frequent feedback from the customer helps promote synthesis of different points of view, but if the project goals are not well defined from the outset, individuals will tend to focus on their own goals (Jiang et. al., 2000).

Incidence of project failure is high in situations where the project is not adequately defined (Keider, 1984). Kindle (1992) suggests that, in order to prevent failures, a company should have a clear understanding of the project objectives and there should be a clear consensus on how the end results should be. Doll (1985) indicates that successful firms are more likely to have reached an agreement between top managers and project manager on a set of criteria for the
development priorities. If multiple organizations are working on a project then differences in organizational goals may cause problems (Corbato and Clingen, 1979).

2.1.8 Project Estimation and Planning

A common problem encountered in software development projects is the incidence of poor estimation techniques for determining a project's budget and schedule.

There are several reasons why estimation poses a problem in software development projects. Non-existence of standards for estimating the time period of a project is the most common reason. Estimation also becomes a problem if it is not done by the project leader, but by anyone who is available to do so (Keider, 1974). It may also be difficult to estimate schedules if personnel availability for the project is unknown. An estimator needs to know who will be working on a project in order to estimate its duration of completion (Keider, 1974). DeRoze and Nyman (1978) indicate that inadequate estimates may occur because there are no simple rules for predicting software costs accurately and estimates generally do not allow for anticipated problems and changes.

According to Farquhar (1970), in an absence of accurate estimates managers do not know what resources to commit to a development effort. This often results in cost and schedule overruns. Poor estimates can lead to excessive schedule pressure or unrealistic schedules that can increase project risk (Boehm, 1987; Jones, 1994; Jones, 1995; Lyytinen et al., 1993). Tait and Vessey (1988) indicate that the likelihood of project failure increases in situations where severe financial or time constraints are present. Whittaker (1999) found that incorrect estimation leads to unrealistic project deadlines especially when the new development tools are difficult to understand.

Poor planning is another problem which has been identified in many software development projects. McCarthy (1979) notes that even if goals are clearly specified development projects often suffer from inadequate planning. Metzger (1981) based on his survey on unsuccessful projects, states that approximately half of them failed because of poor planning. Doll (1985) suggests that firms with more successful IS development efforts are almost three times as likely to have a written overall plan for systems development. Plans should be
made for defect prevention (Jones, 1995), task assignments (Keider, 1984), establishment of milestones (Keider, 1984) and for backup and disaster recovery (Kindle, 1992).

2.1.9 Tools and Technology

Inappropriate methodology and tools used in the project is cited as a reason for the project not delivering what it is designed to deliver (Cash, McFarlan and McKenny, 1988). The approach that fits one kind of project very well may not be well suited to others.

The appropriateness of selected tools and technology refers to the risk of choosing hardware, software, language, methods, tools, etc. that are a bad fit with the task or team members. Thayer et al. (1980) have indicated that decision rules for selecting the correct software design techniques and strategies to be used in designing and testing software are generally not available. There are several reasons why selected technology and tools may be inappropriate for a project. Technology may be inappropriate and increase project risk because of inadequate physical capabilities. For example, if the selected equipment is unable to handle the transaction volume then the likelihood of project failure increases (Casher, 1984). The system used may turn out to be incompatible with the software selected for the project (McFarlan, 1981).

The mandatory use of inappropriate or unsuitable technology causes problems in a number of software projects (Command, 1988). Such instances, which are the result of political pressure, pose a risk to the project being developed (Jones, 1994). A certain technology is likely to be selected due to organizational politics and company rules (Boehm, 1989a; Command, 1988).

The selection of appropriate development tools is an area of concern in literature. Scacchi (1984) says that most companies do not have adequate software development tools and production methodologies. Anderson and Narsimhan (1979) observed that inappropriateness of development methodology was the risk factor having the greatest impact on project success. The respondents to Schmidt et al’s (1996) Delphi survey observe that the lack of an effective development methodology tends to cause quality problems resulting from poor estimation and relatively less flexibility for change.
Several sources have suggested that organizations should change their tools and techniques from time to time. If the technology is unreliable, inefficient, or exhibits problems, project outcome may be doubtful (Boehm, 1981; Command, 1988; Fairley, 1994; Lyytinen et. al. 1993). Technology may also be inappropriate if it is not easily modifiable (Casher, 1984).

2.1.10 Requirement Stability and Accuracy

Fox (1982) suggests that even if management can specify what users want, requirements are not likely to remain stable. Jones (1994) suggests that 80% of all projects face the risk of creeping user requirements. Creeping user requirements refer to changes in requirements which occur after the formal requirements phase.

Some of these changes are essential but most projects underestimate the ripple effects that a requirements change has on design, coding, testing, personnel, assignments, communication, budget schedule and performance (Boehm, 1991). Requirement changes can result in excessive schedule pressure. Numerous authors have indicated that frequent fluctuations in requirements can increase project risk and uncertainty.

Boehm (1989) identifies gold plating as another type of change in requirements. Gold plating occurs when complex requirements are added on to the project. The job gets bigger and disproportionately more expensive as more and more features are added to the project. The final outcome is not of much use to the user.

Changing requirements result in unsuccessful projects. According to Bruce and Pederson (1982) a change in requirements makes it difficult to estimate software potential cost. Project schedules which overlook requirement changes may result in budget overruns (Schmidt et.al., 1996). Ruthberg and Fisher (1986) observe that the chances of project failure loom large in a situation requiring a large number of requirement changes.

2.1.11 Effectiveness of Project Monitoring

Software development problems are also attributed to the fact that adequate time was not spent on various phases of software development. Thayer (1980) notes that techniques and aids for the same are not generally available. In order to control a large project, there must be a list of milestones which are clear, concrete
and measurable (Brooks, 1995). A project needs to have an unbiased feedback in order to accurately monitor its progress (McComb and Smith, 1991).

Boehm and Ross (1989) stress upon the fact that once a development plan has been made, an effort should be taken to see that work progresses accordingly. The need for reviews at frequent intervals has been emphasized by practitioners in software development (Keider, 1974; Humphrey et al, 1991; Kindle, 1992). However, managers tend to overlook this fact (Boehm, 1979).

Henderson and Lee (1992) have suggested that problems may occur if a project manager lacks an adequate standard against which the quality of work done is to be compared (Scacchi, 1984). Keider (1984) has noted that between the completion of the feasibility study and formulation of the work plan, there could be a change in estimates which necessarily has to be reflected in the estimated time and budget, or else the project may fail. Projects should be able to accommodate changes in requirements in the project, in order to succeed (Boehm, 1983; Ruthberg and Fisher, 1986)

2.1.12 Cross Cultural and Gender Issues

Culture may be defined as a shared set of beliefs, attitudes, norms, values, and behaviour organized around a central theme and found among speakers of one language, in one time period, and in one geographic region (Triandis, 1995).

This particular risk factor discusses the significance of developing an awareness of cross-cultural factors in software development. It is rare for organizations to bother with the details of how personnel at lower levels in the hierarchy will run meetings, make decisions, solve problems, manage staff and communicate proposals. Yet, people from different cultures carry out all these procedures in diverse ways. The trouble is that each culture assumes their way is the correct one. Unexplained deviations from these norms are perceived as deviant and even devious (Pooley, 2005).

Language is a major issue, and perceptual differences in understanding language lead to communication barriers. Though English is the universally accepted language, there are instances where the same word denotes a different meaning to people. For example, ‘sanctioned a project’ means ‘allowed a project’ to the British and ‘prohibited a project’ to the Americans. Likewise, Asians prefer to work in groups and strongly value collectivism. However Westerners are more
individualistic by nature. Further there are differences in risk taking capacity, techniques of conflict resolution, general business etiquette, attitude to relationship etc. These together affect efficient functioning of the project team which comprises of members from various nationalities.

Joan Mann et.al proposes a research model linking cultural traits to risk factors in IS projects. Based on the Hofstede’s (1991) dimensions of values framework, they argue, taking Thailand as an example, about how the intensity of IS project risk factors vary based on the difference in cultural dimensions.

Failures in IS projects are also caused by ethical issues. Cultural differences may give rise to divergent perceptions of ethical issues and contradictory approaches to ethical reasoning (Oram and Headon, 2002).

Women now enjoy equal opportunities to higher education with men and hence the gender ratio at workplaces is more or less equalized. Given the growing presence of women in the workforce, gender specific issues need to be given more attention. Women employees have to face pressures of unearthly work timings (hindu.com). Compared to men, women’s time commitments to paid employment are more influenced by the need to reserve time for dependent care and other family responsibilities (Fagan, 2001). There is overwhelming evidence that the burden of dependent care falls on women (Buffardi et al., 1999). Women respondents claimed to have suffered physical ailments as a result of working long hours. Other negative aspects about this job include traveling at night and traveling abroad and the social stigma attached to it.

2.2 RISK MANAGEMENT PRACTICES

This section provides an overview of the risk management practices suggested to address the risk factors identified in software development projects. Risk management is concerned with a phased and systematic approach to analyse and control the risks occurring in a specific context (Charette, 1996). The predominant purpose of risk management is to take the appropriate course of action to strike an optimal balance between likely benefits of such techniques and the exposure to risks (Powell et. al. 1996). Software project risk management is risk management applied to the development and/or deployment of software-intensive systems. Considerable improvements can be made in the software development process through the systematic applications of risk management

Boehm (1991) defined risk management as an emerging discipline whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful operation or major sources of rework. A formal risk management programme is a structured way of evaluating risks to the software development process. A typical risk management framework involves implementing and monitoring measures to reduce risk. Project risk management encompasses both hard skills such as estimating and scheduling tasks, and soft skills, which include motivating and managing team members (Kirsch 1996).

Risk management strategies use observations from the past; they learn from analogical situations, and they use deductive reasoning to detect risky incidents. Over time, observations are generalized by crafting specific theories of cause-effect chains into generic risk items. In addition, risk management approaches feature a repertoire of risk resolution techniques. These are derived from local causal theories on how risky incidents affect software development and how interventions affect development trajectories. The techniques help formulate schematic plans for interventions that decrease the likely impact of risky incidents, or avoid it altogether.

A thorough review of literature on risk management strategies for software projects, helped to identify a range of risk resolutions techniques which are discussed under nine categories, namely:

1. Leadership Strategies
2. HR Policies
3. Training
4. Project Coordination
5. User Coordination
6. Requirement Management
7. Estimation Techniques

24
8. Appropriate Methodology

9. Project Control

Each of them is discussed elaborately in the following sections.

2.2.1 **Leadership strategies**

Making informed decisions by consciously assessing what can go wrong as well as the likelihood and severity of the impact is at the heart of risk management. The need to manage risk increases with system complexity (Higuera et. al. 1994; Haimes, 1991).

Communication lines, the structure of authority, and lines of accountability are significant in organizing the risk-based management process. It is important to effectively communicate risks to everyone involved and to reward reporting of omissions and errors (Keil, 1995). The management structure is also important to create a sense and discipline of accountability (Rochlin, 1993). Obtaining the top management commitment is also a crucial factor (Alter and Ginzberg, 1978). This can be achieved through increasing payoffs associated with successfully completing the project, creating opportunities for senior managers to publicly display their support for the project, and aligning the project with other goals that are viewed as central to the organization (Ropponen and Lyytinnen, 1997). Management should pay attention to the organizational behavioral aspects of the project and build a supporting base for the project within organization so that the project would not be abandoned even if the main advocate was to leave (Ewusi-Mensah et.al, 1991).

McFarlan (1981) suggested selection of a technical expert as the project manager to lead the team. The technical background of the project manager plays a very important role in risk minimization (Ropponen and Lyytinnen, 1997). The project manager/leader is to be backed by an assistant leader, who works closely with the project manager and is able to take over in his absence (Jurison, 1999). Finding skilful, open-minded software managers is instrumental in improving the risk management capability (Boehm, 1989). In order to reduce the need for self-justification, project managers should be rotated in and out of projects so that the people who initiated the project are replaced with people who will naturally have greater objectivity (Keil, 1995). Keil 1995 argued that the failure of the project could be avoided if the project managers adopt a broader view of project management spanning both the rational approach and the psychological approach.
2.2.2 HR policies

There is extensive empirical literature investigating the relationship between HR policies and organizational performance. The relationship between HR and business results is built on a simple premise that better deployment and use of HR should correlate with higher business performance. Those organizations where the HR department is supportive and helpful, encounter fewer incidents of software failure (Evans, 1986; Huselid, 1995; Ulrich, 1997).

Job matching is an essential feature of risk management. Matching an individual to a particular job is based on the idea that personnel differ to a considerable extent in abilities relevant for the successful performance of a task (Zeidner, Johnson and Scholarios, 1997). Tools which are used to match computer professionals to jobs include assessment centres and simulations (Nash, Redwine, 1999). Personnel systems designed around ability profiling have been shown to enhance organizational productivity and translate into economic benefits for employers (Hunter and Schmidt, 1982). When individuals are considered for a position they should be consulted beforehand, lest it results in a problem (Anastasi and Urbina, 1997; Cronbach and Gleser, 1965). It will be easier to trace and correct errors if accountability is established for a particular task (Zmud, 1980).

Attendance systems are designed to tackle employee absenteeism and track the number of hours employees work. Flexibility in working time includes a variety of arrangements for employees such as part-time work; job sharing, flexitime, fixed-term contracts, subcontracting and career/employment break schemes (Papalexandris, Kramar, 1997). These arrangements have been introduced for a variety of reasons which include economic factors, improvement in productivity and competitiveness, timely completion of work etc. (Brewster, 1994). Flexible hours of work and employment schedules affect family and employee satisfaction (VandenHeuval, 1993). Dissatisfaction with the job, stress and poor health have been found to be outcomes of heavy work schedules (Karasek et al., 1981; Karasek, 1979).

Bonuses and commissions do have a motivational effect on employees. Targets appear all the more achievable if there are a series of rewarded steps in the form of promotions and incentives (Garbett and Morton, 2006).
2.2.3 Training

The growing demand for software development requires increasingly productive people. For effective project management, team members need global experience and adequate training (Collins and Kirsch, 1999).

To prevent large variations in employee performance, effective methods of training should be adopted. Training is gaining importance owing to the magnitude of the problem of producing a sufficient number of well-trained software engineers (Martin, 1981). Training of employees is regarded as one of the most important functions of efficient resources management (Prytherch, 1986). Adequate and frequent training increases individual capabilities (Dominiak, 2006). Effective training in technical aspects, project management, communication and other relevant areas can help newcomers become more effective (Deephouse et. al, 2005).

Training on risk management is essential for project managers. Symptoms such as “escalating commitment” (Keil, 1995), the “no-problem syndrome”, “risk aversion” etc can be improved by increasing their awareness of risk management methods (Boehm, 1989; Keil, 1995)

Firms are just beginning to emphasize a ‘software development team culture.’ Sawyer and Guinan’s (1998) study of a major software manufacturer revealed that exposure to group dynamics, conflict management, and listening skills seminars helps to develop this culture among software developers.

2.2.4 Project coordination

The project requires a series of coordination measures. Internal integration focuses on coordinating the project team members while external integration focuses on the external agencies (MaFarlan, 1981).

Projects involving new technology should rely more on internal integration tools that are designed to enhance the team’s technical competence and operation as an integrated unit (Jurison, 1999). Software project management requires different types of coordination at different stages and a major portion of the organizational design problem is choosing the particular type of coordination that matches the given uncertainty. Given the temporary nature of software projects (project teams are usually dissolved once their objectives are achieved) minor
slippages in the control process can have a greater adverse impact than the same slippage in a more permanent organization (Nidumolu, 1995). Zmud (1980) suggests an impersonal mode of project coordination for low risk/uncertainty projects and group mode of coordination for high risk projects. This is endorsed by Alter and Ginzberg (1978) also. Nidumolu (1995) recommends vertical mode of coordination in high risk projects. Barki et. al’s study (2001) shows higher levels of formal planning in high risk projects when cost control is taken as a measure of performance. But when system quality is studied as the performance objective, it is seen to have positive correlation with user planning (Barki et. al., 2001).

Nidumolu (1995) recommends adoption of a risk-based perspective rather than a structural perspective for coordination of software projects. He argues for exerting higher level of leadership and authority (vertical control) to ensure that the objectives are met. The risk-based perspective of software project coordination also takes into account the temporary nature of software projects and teams.

Coordinating with suppliers and sub-contractors is also an important risk management task (Boehm and Ross, 1989). In order to have the whole-hearted cooperation of sub contractors and suppliers, companies need to maintain excellent relationships and operations with them. The open minded attitude in the interactions with suppliers will enable companies to understand their needs better and handle conflicts better (Wong, 2002).

2.2.5 User coordination

Proper communication systems should be designed to integrate users into the development environment. Some of the strategies recommended include selection of a user as the project manager, creation of a user steering committee, frequent and in-depth meetings of this committee, a user-managed change control process, training of the user, frequency and detail of distribution of the minutes of the project meeting to key users, selection of users as team members, formal user specification approval process, progress reports prepared for corporate steering committee, giving users the responsibility for the installation of the system and letting users manage decisions on key action dates (McFarlan,1981; Boehm, 1991). Another measure is to insist on the mandatory use of the system developed and rely on diffusion and exposure (Alter and Ginzberg, 1978).
The risk that arises out of a lack of proper user commitment and support can be minimized by establishing the right contacts and creating a “home base” in the user organization (Alter et. al, 1978). User involvement during the definition stage of the project will also help to make users more realistic in their expectations of the outcome (Ginzberg, 1981).

The key to minimizing user risk includes user involvement, user participation and a clear statement of user requirements (Jiang et.al., 2000; Lin et al., 2004). Teaming with customers helps to reduce the overall risk to both sides. Formation of a risk team comprising of the developer, customer, systems architecture group, and contract management create a forum for open discussion of risks that crossed organizational lines (Dedolph, 2003). Projects with relatively little structure can benefit from external integration tools that create effective links between the project team and the client’s organization (Jurison, 1999). To integrate the technical and end-user perspectives, Kokol et al. (1991) put forward three propositions. First, during the requirement specifications phase, analysts must understand the user requirements and the environment in which the software will operate. Second, during the implementation phase, all parts of the specifications should be implemented correctly. Third, the implemented system must be validated, i.e. software developers must ensure that the implemented system represents a correct mapping of the specifications.

Another useful practice is to have the client’s representatives participate in technical reviews to assure a common understanding of client needs and avoid future surprises (Jurison, 1999).

Project managers should be able to create and maintain long-term relationships with users and promote user commitment to the project (Addison and Vallabh, 2002). Boehm and Ross (1989) argue that the primary job of the software project manager is to structure the project to meet the ‘win’ conditions of various stakeholders.

User manuals need to be prepared with a lot of care. Wright (1988) observes that documentation involves the integration of three basic decisions - decisions about the content of the manual, decisions about the presentation of the information in the manual, and decisions about how the effectiveness of the manual should be evaluated. Carroll et. al. (1986) has developed a manual format.
called the "Minimal Manual" which significantly addresses the first two decisions. Good documentation can minimize the effects of a flawed interface (Gong, 1990).

2.2.6 Requirement Management

Requirements management has been identified as one of the most critical aspects in controlling technology related risks. Organizational analysis, user surveys, information hiding, task analysis, user characterization, requirement scrubbing (Boehm, 1991) have been discussed in this regard. Experimentation, synthesis from characteristics of the utilizing system, paying early attention to poorly defined parts and system functionality, allowing the project to be driven by the user community and not by the developers (Schmid et al, 1996; McFarlan, 1981) etc are measures suggested to bring requirement changes under control. Poor execution of requirement elicitation will almost guarantee that the final project is a complete failure (Hickey and Davis, 2004). Requirement issues need to be resolved as the project progresses (Keil et. al., 1998). Unnecessary changes or requirements should not be entertained.

The risk of unrealistic estimation of budget and cost can be minimised through software reuse and requirement scrubbing (Boehm, 1991). The system should be kept as simple as possible (Alter and Ginzberg, 1978) and should be designed to cost (Boehm, 1991). The project success rate increases with standardized processes (Deephouse et. al., 2005) and design methodologies (McFarlan, 1981). Multi context analysis of the features required will also reduce such risk (Charette, 1996).

2.2.7 Estimation techniques

Empirical studies have demonstrated a positive relationship between effective estimation and satisfactory project outcomes (Deephouse et al., 2005; Guinan, Cooprider, and Faraj, 1998). Reliable estimates are critical for effective project planning and monitoring (Boehm, 1983; Abdel-Hamid, 1993; Lederer et al., 1990; Kemerer and Sosa, 1998). For a project to be termed as effective, the essential pre requisites are visibility of objectives, plans, status and other indicators.

Software Cost Estimation is an empirical process that is applied to estimate the effort and budget required for the software product which is going to be developed. The process starts with the planning phase activities and is refined
throughout the development. This is very important for managing and scheduling the software project (Arifoglu, 1993).

Improving planning effectiveness will, in turn, help to meet budgets and schedules (Deephouse et. al., 2005). Given the uncertainties involved in software project estimations, there may be flaws in the initial estimation. Rather than driving project managers to pursue unrealistic goals, the project manager should be asked to set explicit goals and be judged on how effective his decisions are in achieving those goals (Abdel-Hamid, 1993).

2.2.8 Appropriate Methodology

Project managers need to make a series of decisions at the initial stage and during the course of a software project to ensure a high quality software product (Ferrin et al., 2002). Suitable design approaches like prototyping, evolutionary approach, modular approach, simulation, modelling etc reduce the risk of lack of familiarity of the designers (Zmud, 1980; Boehm, 1991; Alter and Ginzberg, 1978).

Simulation has been used to address a variety of issues in software development projects including strategic management, project planning and control and process improvement. Simulation support also reduces the development time. Using simulation models of the designs, we can increase the accuracy of early estimates of performance (McBeath and Keezer, 1993).

Practitioners view prototyping as an ideal approach for communication with users, as it provides more flexible designs, and is better for the early detection of problems (Verner and Cerpa, 1997).

Software re-use, the use of software artifacts across multiple projects, is an important strategy for improving software development efficiency and increasing the quality of software systems. When re-use is attempted, developers usually have access to the implementation code which can be modified to match a new project’s requirements (Ravichandran, 2003; Rothenberger and Dooley, 1999). The major savings are realized in the detailed design, coding and testing phases. For these phases, software can be used or adapted rather than uniquely developed (Margono and Rhoads, 1992).

Benchmarking has been defined as “the search for and the implementation of best practices” (Jones, 1995). In the area of software development it is
perceived as an assessment method, which is concerned with the collection of quantitative data on topics such as size, effort, defects, schedules and costs (Jones, 1995; Beitz and Wieczorek, 2000). It also helps managers to identify the quantum of improvement required to be the best (Beitz and Wieczorek, 2000).

2.2.9 Project Control

Project control includes comparing actual progress to planned progress and taking corrective action when performance deviates significantly from the plan (Boehm and Papaccio, 1988). It involves collecting information about costs, schedules, and technical output such as code, designs, documentation, test plans, training materials, and procedures (Weinberg, 1985). Information can be gathered via meetings, interviews, walk-through, and formal technical reviews.

Quality assurance (QA) makes sure that the product meets user requirements and that it provides the desired functionality and quality. While the whole project team should be committed to building quality into the product, it is a general practice to have a separate individual or a group whose primary responsibility is quality assurance (Jurison, 1999). Abdel-Hamid (1993) asserted that there should be a mechanism for comparing activities performed with a standard of what should be carried out, a procedure for changing behavior if there is a need and a feedback method or mechanism.

Review meetings play a major role in project control. Their purpose is to assess progress and identify areas of deviations from the plan so that corrective action can be taken. Project review meetings provide visibility to plans and create opportunities for obtaining and enforcing commitments from the participants (Jurison, 1999). Project review meetings are most effective when they are scheduled at regular time intervals and follow an established agenda. These meetings are used to resolve interpersonal conflicts as well as technical issues. Team members discover that, as they interact, they spend less time dealing with interpersonal issues at these meetings and more time on solving important technical problems (Pattit and Wilemon, 2005).

Any significant deviations or variances from the plan require prompt attention from the project manager so that timely corrective action can be taken. Keil et al. (1998) stated that to avoid the problem of scope creep, project managers should inform users of the impact of scope changes about details of project cost.
and schedule. Project managers should be able to distinguish between desirable and absolutely necessary functionality. The project manager must be able to identify the source of the problem. If there is a major deviation from the plan, the project manager must decide whether re-planning future activities is warranted (Jurison, 1999).

Technical performance control, the process of assuring that all technical requirements are met, is normally exercised through a variety of design reviews. These reviews are usually held at major milestones (e.g. completion of requirements definition phase, design phase, or coding) but it can be held at other times during the project also. The progress towards important technical goals should be tracked through appropriate metrics during the project. The metrics provide project managers visibility of what has been achieved, and their trends offer predictions of what can be expected in the future (Jurison, 1999). Software engineers use different types of software development technical review for the purpose of detecting defects in software products (Sauer et. al., 2000).

2.3 CHECKLISTS ON SOFTWARE PROJECT RISK AND RISK MANAGEMENT

One of the most common methods for identifying the presence of risk factors and risk management strategies in a particular project has been the use of checklists. These checklists present a list of all potential risks and risk management factors that might be applicable in a software development project.

McFarlan (1982) was one of the initial researchers in identifying the risk factors in software development. McFarlan Risk Framework was a general framework applicable to any IT projects. He highlighted the failure of project managers to consider the aggregate risk of the portfolio of projects. Also, they did not recognise the fact that those different projects require different managerial approaches. He categorised projects based on Project size (Size in cost, time, staffing level, or number of affected parties), Experience with technology (Familiarity of the project team and the IS organization with the target technologies) and Project Structure (how well structured is the project). He suggested appropriate risk mitigation strategies for these risk groups.

One of the pioneering studies in this regard is the top 10 risk list of Boehm (1991). His list has been compiled by probing several large software projects and their common risks and is thus empirically grounded. The top ten items are:
Personnel Shortfalls, Unrealistic schedules and budgets, Developing the wrong functions and properties, Developing the wrong user interface, Gold-plating, Continuing stream of requirements changes, Shortfalls in externally furnished components, Shortfalls in externally performed tasks, Real-time performance shortfalls and Straining computer-science capabilities. The list is also well-known and has been widely applied in practice to orchestrate risk management plans (Ropponen and Lyytinnen, 1997). But in spite of its popularity and simplicity, Boehm's list forms an inductively derived collection of risk items and thus lacks a theoretical foundation (Lyytinen et al, 1998; Barki et. al. 1993).

Barki et al (1993) tried to overcome some of these limitations to produce a more comprehensive list of risk factors. Based on a review of the IS risk literature, he prepared a list of 35 risk factors. From the data collected from the project leaders and user representatives from 120 ongoing projects in 75 organizations, the list was reduced to 32 which, after factor analysis, became 23. Factor analysis revealed 5 general categories or dimensions of risk. Those dimensions are: organizational environment, technological newness, expertise, application size, and application complexity.

Jiang and Klein (2002) supplemented this study through a survey among project managers asking them to rank these risk categories in order of importance. They also identified the necessary risk management focus areas (user focus, institutional focus, commitment focus and simple focus) to counter these risk factors.

One of the most quoted international studies on software project risk factors was conducted by Schmidt et al. in 1996. In an attempt to compensate for some of the previous shortcomings in checklists of risk factors, Schmidt et al. (1996) conducted a survey of project managers and developed an extensive list of risk factors in software development. Their research was accomplished through three simultaneous Delphi surveys in three different settings: Hong Kong, Finland and the United States. In each country, a panel of project managers was formed and a “ranking-type” Delphi survey was used to solicit risk items from the panel. These risk factors were then consolidated into a comprehensive list of software project risk items. The initial list produced by the expert panels was grouped into 53 unique risk items and then further reduced to a list of 11 risk factors common to all countries. These factors are Lack of top management commitment to the project,
Failure to gain user commitment, Misunderstanding the requirements, Lack of adequate user involvement, Failure to manage end user expectations, Changing scope/objections, Lack of required knowledge/skills in the project personnel, Lack of frozen requirements, Introduction of new technology, Insufficient/inappropriate staffing and Conflict between user departments. The third and final phase of the study involved the actual ranking of these risk factors. It was seen that the ranking of the factors varied significantly across the countries. These factors were categorized into four segments and risk management strategies were suggested for all the factors.

Keil et al (1998) improved upon their international Delphi study exploring the issue of IT project risk from the user perspective and compared it with risk perceptions of project managers. By understanding the differences in how users and project managers perceive the risks, insights could be gained that may help to ensure the successful delivery of systems. The Delphi study revealed that these two stakeholder groups have different perceptions of risk factors. Through a comparison with Schmidt et al. (1996) study on project manager risk perceptions, zones of concordance and discordance were identified.

Anja Mursu (2000) repeated Schmidt et al’s Delphi study design in Nigeria to identify the major software development risks. The study produced a rank-order list of software risk factors which are significant different with the rankings of the earlier study.

Addison and Vallabh (2002) also found the top ten risk factors to be different in their survey. They also found significant variation in risk perception when respondents were grouped based on the number of years of experience.

The Software Engineering Institute (SEI) has contributed considerably over time to the study of project risk management and has produced a significant amount of literature with a comprehensive inventory of variables related to the assessment of software development project risk. SEI’s Taxonomy-Based Risk Identification Instrument, which contains 194 questions, is probably one of the largest checklist of software development risk factors. These 194 questions are classified under various categories falling under three classes namely Product Engineering, Development Environment and Program Constraints.
Moynihan (1997) through his survey on project managers who had managed custom-built, software-intensive application and development projects that originate from external clients produced a huge collection (113) of risk related constructs. He made a comparison of these risk factors with the Barki variables and SEI themes. Though there was considerable overlap, he found that many of the real world issues are not captured in the Barki list and the SEI list.

The work by Ropponen and Lyytinen (1997) contributes to the empirical studies about the commonality and type of software development risks. Using a survey instrument, they empirically delineated six components of software development risk namely scheduling and timing risks, system functionality risks, subcontracting risks, requirement management risks, resource usage and performance risks, and personnel management risks.

Alter (1979) identified eight risk factors: nonexistent or unwilling users; multiple users or implementers; turnover among all parties; inability to specify purpose or usage; inability to cushion impact on others; loss or lack of support; lack of experience; and technical or cost effectiveness problems after studying the implementation of 56 Decision Support Systems.

Zmud (1980) found that technological complexity, the degree of novelty or structure of the application, technological change, and project size influence the outcome of large software development efforts.

McBeath and Keezer (2003) suggested that several sources of uncertainty should be taken into account in the management of software development projects. These are: complexity, lack of structure, or instability of project objectives; newness of the technology; users; IS management; upper management; and project size.

Linda Wallace (1999) developed a valid and reliable measure of software project risk to study risk from a common perspective and to compare findings across studies in a more meaningful manner. Following established practices of instrument development, her study developed and empirically tested a model of software project risk. The final instrument had 44 items measuring 23 risk factors. Their factor analysis revealed 6 general categories or dimensions of risk. Those dimensions are: User, Development Team, Organizational Environment, Project Complexity, Project Management and Requirements. Scales were designed to measure each of the six first-order dimensions of risk.
Boehm's model (1991) suggests a comprehensive set of steps and guidelines to manage software development risks. For each of the top 10 risk items, Boehm developed a set of risk-management techniques that “have been most successful to date in avoiding and resolving the source of risk”. The idea is that after detecting the most important risk items risk-managers can compile the associated set of risk management measures and plans. These risk management practices include Award-fee contracts, Benchmarking, Contracts, Design to cost, Early user's manuals, Incremental development, Information hiding, Instrumentation, Mission analysis, Morale building, Pre-award audits, Presheduling, Prototyping, Requirements scrubbing, Software Re-use, Team building, Tuning, User Surveys, Compatibility analysis, Competitive design, Cost-benefit analysis, Cross-training, Detailed multisource cost and schedule estimation, High change threshold Incremental development, Inspections, Job-matching, Modeling, OPS-concept formulation, Organizational analysis, Reference checking, Scenarios, Simulation, Staffing with top talent, Task Analysis, Technical analysis and User characterization.

Davis' model (1982) is concerned with selecting procedures that lead to complete and correct information requirements. His argument is that one of the reasons for the high risk in systems development is the inappropriate methodologies used for obtaining and documenting user requirements. Therefore he suggested “packaging” different methods into alternative strategies that are then carefully explained. These strategies are linked to a risk management model which suggests the most successful requirements determination strategy for a given situation. These strategies include Asking from users, Deriving from existing systems and Synthesis from characteristics of the utilizing system and Discovering from experimentation.

Alter and Ginzberg's (1978) focused on problems associated with the organizational acceptance and implementation of the information system. They argued that implementing any system would involve uncertainty from the managerial point of view. Therefore, these uncertainties should be detected and appropriate measures should be taken to minimize their impact. They recognized and classified several risk resolution strategies into inhibiting or compensating strategies. Inhibiting strategies are ex ante—this means to avoid a particular problem, while compensating strategies are ex post—this means to make up for a
previous error or problem. These strategies are: Avoid change, Hide complexity, Insist on mandatory use, Keep the system simple, Obtain management support, Obtain user commitment, Obtain user participation, Permit voluntary use, Provide training programs, Rely on diffusion and experience, Sell the system, Tailor system to people's capabilities, Use evolutionary approach, Use modular approach and Use prototypes.

McFarlan (1982) classified risk resolution techniques into four types, namely External integration, Internal integration, Formal planning and Formal control mechanisms. The strategies are Selection of user as project manager, Creation of user steering committee, Frequency and depth of meetings of this committee, User-managed change control process, Frequency and detail of distribution of project team minutes to key users, Selection of users as team members, Formal user specification approval process, Progress reports prepared for corporate steering committee, Users to be responsible for education and installation of the system, Make users manage decisions on key action dates, Select a manager to lead the team, Conduct frequent team meetings, Implement regular preparation and distribution of minutes on key design decisions, Conduct regular technical status reviews, Manage low turnover of team members, Include high percentage of team members with significant previous work relationships, Encourage participation of team members in goal setting and deadline establishment, Get outside technical assistance, Milestone phases selection, Systems specification standards, Feasibility study specifications, Project approval process, Project post audit procedures, Periodic formal status reports versus plan, Change control disciplines, Conduct regular milestone presentation meetings and Communicate deviations from plan.

2.4 REVIEW OF STUDIES LINKING PROJECT RISK, RISK MANAGEMENT AND PROJECT OUTCOME

The studies referred above consider software risks along several dimensions and have provided some empirically founded insights of typical software risks and risk management strategies to mitigate them. Overall, these studies provide insights into risk management deliberations, but are weak in explaining the true impact of risk and risk management practices on the project outcome. A few studies have gone further to establish how risk management
efforts reduce the exposure to software risk and can thereby increase software quality and improve software development (e.g., Fairley, 1994; Nidumolu, 1995; Wallace et. al., 2004).

A number of system performance criteria have been developed and empirically tested. They include IS usage, user satisfaction, quality of decision making, cost/benefit analysis, team effectiveness, and project effectiveness. The triple criteria of project success – meeting cost, schedule and performance targets - have been widely used by researchers to analyze project success. Saarinen (1990) proposed a system success measure with four dimensions: system development process, system use, system quality, and organizational impacts. The identification of these distinct dimensions of system performance illustrates that a project can be both successful and unsuccessful at the same time depending on the metric selected. One of the most popular approaches is to categorize these measures under process performance measures and product performance measures (Barki et. al 2001; Nidumolu 1995; Deephouse, 2005; Wallace, 2000; Al-Hindi, 1996; Ravichandran, 1996). Product outcome refers to measures of the “successfulness” of the system that was developed. It looks at how the software developed scores on important parameters of software quality: reliability, maintainability, easiness to use, response time, meeting the requirements, user satisfaction etc. Process outcome measures refer to the “successfulness” of the development process of the project. The focus is on completing the project within budget, within schedule and the on the overall quality of the development process. Both aspects are important as the software delivered by the project may be of high quality but the project itself may have exceeded the time and cost projections. On the other hand, well managed projects which come in below cost and time budgets may deliver poor products.

Due to the difficulty in quantifying costs and benefits, measures based on perceptions have become particularly prominent in IS literature.

Linda Wallace (1999) validated the second order factor model of risk through the establishment of co-alignment, a structural model of the relationship between risk and project outcome – both product and process outcome. The result of this research has established a tentative link between project risk and project outcome and shows that the level of risk associated with a project can have an impact on the ability of the project to be finished on time and within budget.
Jiang et. al. (2000) has independently done a study similar to the one described above and arrived at similar conclusions. He also found that software project risk can better be expressed as a second order factor model and that there is negative link between risk and project success.

Based on her previous research, Wallace et al (2004) developed a model linking risk and project performance. This was guided by project management literature and socio technical theory. Six components of risk were extracted through principal component analysis and these six dimensions were further grouped under three dimensions namely social subsystem risk, technical subsystem risk and project management risk. The relationship of these second order dimensions of risk with product and process outcome of the project was studied through structural equation modeling.

Drawing from contingency research in Organizational theory and IS literature, Barki et al (2001) developed an integrative contingency model of software project risk management. The central hypothesis in the model is that the outcome of the software project is influenced by the fit between the project risk and how the project is managed. The outcome measures used are cost over run and quality of the final software delivered. The risk management practices studied are formal planning, internal integration and user participation. High risk score and the low risk score projects are separately analyzed. In each of these groups, sub groups scoring high and low on performance factors are separated. Thus the following four categories emerged: low risk high quality, low risk low quality, low risk high cost performance, high risk low cost performance. The ideal profiles for each risk category are calculated. The fit is measured as the deviation from the ideal profile. The deviations are seen to be negatively correlated with the performance supporting the contingency model.

Nidumolu's (1995) study was a pioneering effort in linking software project risk to project performance. His study linked coordination mechanism and risk drivers to project performance. Two types of coordination mechanisms were studied. Vertical coordination is the interaction through formal systems and procedures, and horizontal coordination is how they coordinate through mutual adjustments and interaction. A new research model was developed along the structural contingency perspective in Organizational theory and risk based perspective in software engineering. This model introduced residual performance
risk, i.e., the difficulty in estimating the performance related outcomes in the later stages of the project, as an intervening variable clarifying the relationship between risk, coordination mechanisms and performance.

Na et al (2006) study is an extension of the Nidumolu (1995) study in Korea. They utilized measure development and analysis similar to the process described in the Nidumolu study. Three distinct models were proposed in this study. Model 1 assessed the impact of two popular risk management strategies and residual risk on objective performance. Model 2 assessed the impact of functional and systems development risk on objective performance. Finally, Model 3 assessed the impact of functional and systems development risk on subjective performance. The study provides insights into managerial strategies to reduce the possibility of software project overruns. Their findings reveal that both functional and system development risks are important predictors of software project performance.

Jiang and Klein’s study (2000) relates to various software development risks on project effectiveness. The 12 most common software development risks proposed by Barki et al. (1993) were examined. Project effectiveness, which is a specific dimension of the system performance, was studied through the following measures: meeting project budget and schedules, amount of rework, quality of work, and operation efficiency. The results of the study indicated that different project risks would impact different aspects of system development to differing extents. The following two risk factors were seen to have a more significant impact on effectiveness: lack of general expertise of the team and lack of clear role definitions for team members.

Claes Wohlin’s (2002) research work links project success to project characteristics. Project characteristics are variables such as complexity, competence, requirements stability, personnel turnover, geographical distribution, methods and tools, time pressure, information flow, top management priority of project, and project management. The project success variable studied is the timely completion of the project. He ranks and classifies projects into three categories based on the success parameter. Then the projects are ranked and classified again based on the scores on the project characteristic variables. These two rank lists are compared and correlated. This is the basis for computing an index to predict project success/failure based on project variables.
Deephouse et al (2005), through an exploratory study, developed a conceptual model linking effectiveness of software processes to the project outcome. Seven software process variables namely planning, process training, stable environment, user contact, design reviews, prototyping and cross functional teams were considered. Product quality and time overrun were taken as indicators of project performance. The study also tested the planning effectiveness as a mediating variable. Project characteristics such as the staffing level, percentage of work outsourced, type of application developed are tested as control variables.

Ropponen and Lyytinen (2000) researched on how risk management practices and environmental contingencies help in addressing the risk components in software projects. The study assumed that software development risk could be split into several distinct dimensions and that various risk management methods and practices could influence different components of the software risk. In the same vein, they assumed that there existed a connection between environmental contingencies and the capability to handle software risk. The study provided encouraging evidence of how the use of risk management methods could address some of these risks. At the same time, the study recommended that software organizations must tailor their risk management efforts to their development environment.

Jiang et al (2000) studied the relationship between the major risk factors and the responses to these risk factors (mitigation strategies). Response strategies were classified under four implementation focus areas namely user focus, institutional focus, commitment focus and simple focus. To examine the specific implementation focus adopted by project managers in different risk situations, a series of regression analyses were conducted with risk factors as dependent variables and each implantation focus area as an independent variable. He found that different risk situations warranted different approaches.

A similar analysis was performed by Addision and Vallabh (2002) to determine whether there were significant relationships between some of the demographic data (such as number of years of experience of the project manager) and the risks and risk controls. These tests were performed using regression analysis, to compare the controls to each of the risk factors to determine if they were effective in mitigating the occurrence of each risk factor. Some of them were seen as significant though none of the pairings resulted in high values of R².
Keil (1995) looked at another aspect of software project failure called project escalation. It refers to the continued support given to projects which are in deep trouble and are all set to fail. Why further resources were wasted on these projects was the research problem. His model linked a set of project factors, psychological factors, individual factors and organizational factors to project escalation behaviour.

Kirsch (1996) proposed to build an integrated contingency model of software project management linking project management practices to the characteristics of the project and attributes of the individuals involved. He hypothesized that the project characteristics directly affected project management practices while individual attributes may have both direct and moderating effects on such practices. These project management practices are, in turn, believed to affect project performance.

Walsh and Schneider (2002) had looked at software development risk from the agency theory point of view. He studied the causal relationship between team decisions and project success. According to him, team decisions are influenced by the agency effects (alignment between the interest of the member and that of the organization) as well as development ability of the member.

2.5 OBSERVATIONS FROM LITERATURE REVIEW AND MOTIVATION FOR THE CURRENT RESEARCH WORK

The researchers have adopted different approaches to developing checklists on risk and risk management. One approach has been to develop these checklists based primarily on their personal experience with software development projects. Boehm's top 10 is largely based on his experiences at TRW. The same was true with many studies in the 1980s and early 90s (e.g., Casher, 1984; McFarlan, 1982). Many of these checklists were criticized as they were not very systematic and coherent and lacked any theoretical basis (Ropponen and Lyytinen 1997; Schmidt et al. 1996).

Another approach adopted was to develop the checklist based on extensive literature review. The major work in this regard includes the literature from Software Engineering Institute. But many subsequent researchers have questioned the lack of empirical validation of these instruments developed. No steps were
taken to contact practicing software project managers for input into the relative importance and accuracy of the identified risk factors. (Wallace, 1999)

The third approach was to elicit the list of risk factors from practicing managers. Many researchers have conducted surveys among the members of software projects (Moynihan, 1997, Jiang et. al. 2002). Many of these studies have been criticized for their limited focus on IS literature. They made no attempt to reconcile their findings with the IS literature in this area. This has limited their usefulness as a comprehensive practical tool for gauging project risk.

Linda Wallace (1999) study stands out as an attempt to develop a comprehensive measure of software development risk based on literature which is later validated with software professionals working in software projects. But she has focused only on in-house software development with USA companies most of which were non IT companies. Hence many of the non USA risk factors as well as risk factors specific to software development companies may not have been captured in her list.

Also, most of the previous research takes an isolated view of software project risk and risk management strategies. Very few studies have taken an integrated and comprehensive view of risk and risk mitigation strategies linked to project outcome. Arguments are largely based on anecdotal evidence or armchair theorizing. Empirical evidence on the relationship between risk and project outcome is rare and often fails to take into account various risk factors that may hinder success.

Linkages among software development risks, risk management strategies and various dimensions of system success are generally overlooked in IS literature. Yet, this is an important step for advancing our knowledge on project risks because it is very likely that different project risks may affect the various dimensions of system success differently. A particular control procedure or method may reduce only certain aspects of software development risk and not others. Linkages between risks and various dimensions of system success can help project managers to select the needed implementation strategies to achieve their desired project outcomes.

Most of the studies in these domains have been done in developed countries but have come out with generalized conclusions regarding the risk
factors in software development projects. This has been acknowledged as a major limitation of the research on software development risk factors. Many researchers have argued with empirical evidence against this generalization across countries.

The most quoted international study on software risk factors by Schmidt et al. in 1996 in Hong Kong, Finland and the United States showed that there is no consensus on the top risk factors across countries. Mursu (2000) replicated the Delphi study of Schmidt in Nigeria. He found significant differences in the risk factors and their importance in Nigeria compared to what the original study showed. Similarly Na et al. (2006) found that models developed with data collected in USA do not apply to organizations in Korea where the IT capability is known to be lower than in the USA. Specifically, their study suggests that, unlike the Nidumolu (1995) study conducted in the USA, residual risk is not a significant predictor of subjective performance measures such as software project process and product performance.

Many researchers acknowledge that cultural differences can impact work related values and play a significant role in the success or failure of projects (Hofstede, 1991). Joan Mann and James P. Johnson proposed a research model for risk associated with information systems projects in Thailand. His model is based on the premise that the Thai culture is likely to impact the propensity for risk to occur. A risk factor significant in one culture may not be significant in another.

Most of the previous studies focus on software professionals working in non IT companies. This has influenced many of the risk factors identified in the studies (e.g. factors such as lack of support from top management, lack of expertise available etc). Not many studies have been done focusing only on software development in software companies.

Also, most of the studies focus either on in-house projects or completely outsourced projects where the end user is well defined. The outsourced projects covered in these studies are projects outsourced to IT companies in the same country. There is a substantial increase in software development outsourcing to firms overseas, especially in India. Multi-level outsourcing is also very popular. This will lead to project management issues such as lack of visibility about the final user, contractual problems, information asymmetry etc.
All these point to the need for more studies to be able to generalize across varying socio-economic contexts and also to develop insights into the risk-risk management-project outcome models in different contexts.

India presents many unique characteristics which will have impact on the risk and risk management. The working environment in software companies in India is different from that of USA. Some of the notable differences include

- Flexible working hours to absorb the risk impact without causing cost and time overruns.
- Low cost per employee in India resulting in high profit margins. This helps Indian companies to absorb the financial impact of the occurrence of a risk event.
- Infrastructure support in Indian companies is inferior to their USA/UK counterparts.
- The regulatory issues cause considerable risk—visa rules, work permits, minimum wage salaries, laws on working environments etc can pose specific problems
- The employee turnover is very high in India. Absence of proper in-house systems and documentation practices put companies at an immense disadvantage when employees leave without any notice.
- Legal and political risk is involved given the fact that the major chunk of the software development in India is for foreign clients.

Hence the validity of the findings of the international researchers needs to be tested in different environments such as India in order to assess their universal applicability.

2.6 SUMMARY AND CONCLUSION

The chapter has reviewed previous research on software development project risk, risk management, measurement scales and models linking risk and risk management to project outcome. The research by Wallace (1999) is specially noted for its rigour and comprehensiveness.

The major limitations of the study are also noted. The literature still lacks a comprehensive and validated study linking risk, risk management and project outcome. Also no major studies on these constructs are reported from India. The motivation for the present research is derived from these limitations.