CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

It has almost been a decade since the software industry has detonated. It has witnessed a remarkable growth and a tremendous escalation not only in the core activities but also in the IT enabled services. Despite the uninterrupted expansion, the software industry still has the highest number of project delays and failures. According to the Standish report [9], 44% of the software projects are challenged (late, over budget and/or with less than the required features and functions) and 24% have failed (cancelled prior to completion or delivered and never used). Thus, making a total of 68% (both challenged and failed) which is quite exponential. Boehm [10] found that 15-35% of all the software projects were cancelled outright while the remaining projects suffered either from schedule slippage, cost overruns or failure to meet the project goals.

Software development projects are collections of larger programmes with many interactions and functional dependencies. It involves a creation of a product that has never been created before although the development processes are similar among other projects. As a result, software development projects have a dismal track-record of cost and schedule overruns and quality and usability problems [12]. Further, it becomes very difficult to predict the success of project because the scope of the project keeps changing depending upon the market; hence the resources have to be re-allocated leading to schedule slippage and cost overruns. Many software projects involve multiple entities such as companies, divisions, etc., that may have varying interests. There is often a feeling of disconnection between software developers and their management, each believing that the others are out of touch with reality resulting in misunderstanding and lack of trust [12]. According to Doherty and King [27] and Warkentin et al. [30] organizational risks stemming from organizational culture, structure and business processes impacts the technical software development issues, creating a wide range of potential trouble points. A study revealed that 65% of the project failures were due to management issues [67].
Literature Review

Apparently, this implies that software project development is extremely risky. Therefore, managing the involved risks is of primary importance in software project development, especially in the large-scale software projects [68]. If the risks are not controlled at the early stages of the project, it will result in an exponential increase in the cost of the project as shown in Figure 2.1 [69] [70].

![Figure 2.1: Relationship between Project Risk and Cost/Return](image)

The extant literature has produced a number of conceptual frameworks to explain different types of software development risk, risk management strategies, the preferred organizational climate and measures of software project performance [24] [35] [71]. This chapter aims to enlist and critically review the studies that have been conducted in the area of software risk, organizational climate and software projects.

### 2.2 BASIC CONCEPTS

#### 2.2.1 Project

**Project**, by definition, is a temporary activity with a starting date, specific goals and conditions, defined responsibilities, a budget, a planning, a fixed end date and multiple parties involved [72]. According to Turner and Muller [73], a project is an endeavour in which human, material and financial resources are organised in a novel way, to undertake a unique scope of work, of given specifications, within the constraints of cost and time, so as to achieve beneficial change defined by quantitative and qualitative objectives.
A software project is a project which encompasses a unique scope of work with given specifications which needs to be completed in a given time at a given cost [74].

The main stakeholder of a software project is the customer, the party that is going to use the delivered system to its business purposes and will benefit from the value added by applying the system to its business. The second most important stakeholder of a software project is the developer, the party that is building the system to be used by the customer.

The success or the failure of the software project is broadly assessed in three dimensions as shown in Figure 2.2 [74] [75] [76]:

a) Cost – refers to any or part of the project’s materials, supplies or external contracts.
b) Time – refers to any part of the project schedule, including the duration of individual tasks, milestones and deadlines.
c) Product performance – refers to the specifications, quality, scope or standards that part or the entire project is planned to achieve.

![Figure 2.2: Project’s Triple Constraint [77]](image)

However, the project cannot be called successful or failed on the basis of these three parameters if viewed from the eyes of all the stakeholders involved in it. The same outcome of a project may
mean different things to different people [78]. Management’s view of what constitutes a successful project may be different from that of a project manager, while developer and users may have different take on the success [79] [80]. The difference in viewpoint is due to different perspectives, motivation and responsibilities associated with the roles [78] [81] [82] [83]. Linberg [84] found that software developers considered a project successful albeit it got cancelled before arriving at a conclusive outcome because it resulted in substantial learning that could be applied to future projects. Baker et al. [85] has defined project success as follows: the project is considered an overall success if the project meets the technical performance specification and/or mission to be performed, and if there is a high level of satisfaction concerning the project outcome among the key people in the parent organization, key people in the project team and key users or clientele of the project effort. One of the studies identified two criteria namely project management criteria and product success criteria, for defining project success. It noted that project management success covers meeting time, cost and quality objectives, while product success deals with the ability of the project’s final product to meet the project’s owner’s strategic organizational objectives, satisfaction of users’ needs and satisfaction of stakeholders’ needs where they relate to the product [86]. The same was reiterated in other studies [79] [82].

According to Prifling [87] success of software project in literature has been defined from two perspectives - efficiency and the effectiveness perspective. He says that one school of researchers defines success from the efficiency perspective, pinpointing the measures of efficient management of a project, such as adherence to time, budget, and quality requirements. While the other school, by contrast, places more emphasis on the effectiveness of projects, i.e. fruitful overall project outcomes for the organization as a whole, such as future profits or improved business process performance [87] [88]. The Standish report also defines success of the software project as meeting time, cost and product performance [9] [87]. Hence, the literature has no comprehensive view on what constitutes the success of an IT project [89].

However, on the basis of the above definitions, the success of the software projects can be viewed on two different perspectives – one from the customer’s point of view and another from the developer’s point of view.

The customer’s success criteria of the software project are:

a) Keeping expenses within the budget,
b) Meeting the schedule,
c) Acquiring the product of acceptable functionality & quality.

The developer’s success criteria of the software project are:

a) Reaching customer satisfaction confirmed with timely payments,
b) Matching the time to market demands,
c) Building the product of high maintainability and reusability.
d) Substantial learning that can be applied to the future projects.

In principle, the success of a software project can be claimed only if both the perspectives have been successfully achieved. This sometimes becomes quite difficult to achieve due to various factors influencing the software development. These can be the internal or external risks that the software projects get exposed to during the software development lifecycle or various organizations climate factors that may either help or hamper the otherwise smooth development and execution of the software projects. Before proceeding to the risks and organization climate factors that impact the success of the software projects, it’s imperative to understand the stages of Software Development Life Cycle.

2.2.2 Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a framework that is used to understand and develop information systems and software successfully. It is a process used by almost all developers and software development companies as the standard in the software process development. SDLC has many models and each model has its own strengths, weaknesses, advantages and disadvantages [90] [91].

The typical activities involved in the software development life cycle include:

Requirements Gathering

Gauging the accurate set of requirements is the first step in software development process. System requirement may vary depending on the software product that is going to get developed.
Therefore, a careful analysis has to be made about the system requirement needed for the development of the product.

**Requirement Analysis**

This step is marked by undertaking a feasibility study on software requirement gathered in the first step. In this phase, development team has to communicate with the customers and make analysis of their requirements and system. An exhaustive document is prepared in this phase which has details like project plan or schedule of the project, the cost estimated for developing and executing the system, target dates for each phase of delivery of system developed and so on. This phase is the pedestal of software development process since further steps taken in Software Development Life Cycle would be based on the analysis made in this phase.

**Systems Analysis and Design**

This is an important phase in software development. Here analysis is made on the design of the system that is going to be developed. In other words, database design, functional specification design, low level design documents, high level design documents and so on takes place. Care must be taken to prepare these design documents because the next phase, namely the development phase, is based on these design documents. If a well structured and analysed design document is prepared, it would reduce the time taken in the subsequent steps namely development and testing phases of the Software Development Life Cycle.

**Code Generation**

This is the phase where actual development of the software takes place. That is based on the design documents prepared in the earlier phase. Code is written in the programming technology chosen. Here, the code is converted into executables in this phase after code generation.

**Testing**

In order to ensure quality of the software it is extremely crucial to ensure that the software so delivered is defect free. This can be ascertained by testing the developed code. Various tools and techniques are available for testing at different levels such as regression testing, performance testing, stress testing etc. Based on the need, the testing methods are chosen and reports are prepared about bugs. After this process the system again goes to development phase for
correction of errors and again tested. This process continues until the system is found to be error free.

**Deployment**

This is one of the last phases of software development cycle. This phase is marked by documentation of internal design of software for maintenance and enhancement.

**Support, Maintenance and Enhancement**

The last phase of the software development life cycle is to provide support and maintenance of the delivered software. This is an incessant process. With ever changing environment, new problems are discovered and new requirement are identified, new dimensions need to be added to the existing software. All this is covered under support and maintenance phase of software development cycle.

The entire software development cycle is constantly exposed to both internal and external risks. The risks are present in all the stages of SDLC and it is the obligation of the project manager to either remove or reduce its impact on the project. The various kinds of risks that software project is exposed to is discussed below.

2.2.3 **Risk**

Risk is any potential situation or event that could negatively affect a project’s ability. A risk is an exposure to loss or injury or a factor, thing, element, or course that involves uncertain danger [92] [93]. Risks specific to the software have been widely studied in the literature. An examination of the available literature reveals that software project risk has been conceptualized in several different ways. For instance, Charette [94] defines it as “the possibility of loss with some choice involved”, later he further worked on the definition of risk and stated that risk is “an event with a likelihood of occurrence and some potentially negative consequence” [95]. While, Barki et al. [96] [97] have defined risk as “the degree of exposure to negative events and their probable consequences” and “a combination of the probability of an undesirable event with the magnitude of each and every foreseeable consequence”. Higuera and Haimes [98] and Madachy [99] have defined risk as “a measure of the probability and severity of adverse effects. Furthermore, Lytytinen et al. [47] and Ropponen and Lytytinen [26] have stated that software risk is “a state or
property of a development task or environment, which, if ignored, will increase the likelihood of project failure”. Keil et al. [15] have defined software risk as “a contingency that constitutes a serious threat to the successful completion of a software development project”.

Recently, Benaroch [100] has defined software risk as “failure to respond to threats”. He has further defined it as “the downward or upward variation in expected outcomes” [101]. While Purao et al. [102] states that risk is “a particular aspect of the development task, process, or environment, which, if ignored, will increase the likelihood of project failure”. Gefen et al. [103] laid down “unforeseen contingencies related to changes and additions to the software specifications during the development period” as the definition of software risk while, Wallace et al. [33] defined it as “a set of factors or conditions that can pose serious threat to the successful completion of a software project”. Masri et al. [34] stated that software risk is “the probability that risk sources would lead to risk events that in turn increase the negative variance from expected outcomes with predetermined magnitudes as well as the degree of which risk management mechanisms influence risk sources and the variance of expected outcomes”.

Thus, it can be clearly seen that valuable attempts have been made to specify definition of the software project risk. However, the most common definition of risk in software projects is in terms of exposure to specific factors that present a threat in achieving the expected outcomes of a project [36]. On this basis, risk in software projects is usually defined as the probability – weighted impact of an event on a project [19] [36] [94] [95]. Mathematically, it can be written as, 

$$R = P \times I$$

where $R$ is the risk exposure attributable to a particular risk factor, $P$ is the probability that the undesirable event will be realized and $I$ is the impact or magnitude of the loss if the event does occur. Risk exposure is usually measured in dollars or time in commercial projects.

The general view as used today in software projects is to reduce the likelihood of an unfavourable project outcome. Therefore, all probable risk factors should be identified at the onset of the project. The risk exposure for each factor is then estimated (using the above formula) and the exposures are prioritized to identify the risks that embody the greatest threat to the project. Attention is then focused on the high risk factors to decrease the likelihood of their occurrence and/or the magnitude of impact if they are realized, through control measures such as mitigation
strategies and/or contingency plans. A progressive status of identified risk factors is maintained and periodically updated [36].

However this concept of risk has certain flaws:

First, even before software engineering adopted the definition, management research found that this approach does not match actual managerial behaviour. It was found that, in practice, the likelihood of outcomes and their impacts tend to enter into managers’ calculations of risk independently, rather than as their products. They believe that risks can be reduced or dissolved by using their managerial skills to control the dangers. That is, “managers look for alternatives that can be managed to meet targets, rather than assess or accept risks” [105].

A second limitation of this definition is that it is very difficult in practice to estimate the probability of impact of many risk factors, especially in software projects. Probabilities can only be significantly determined for activities that are repeated many times, under controlled circumstances. Since software projects are often about enabling change through new applications using new technologies in dynamic environments, the extent to which previous patterns are applicable to the future is essentially uncertain in these projects [36].

A third limitation of this definition is that it securely pairs the risk event with the risk consequence, ignoring the interceding influence of organization-specific susceptibilities and capabilities to mitigate and respond [36] [106]. The capability of the organization to respond to a threat may increase or decrease an organization’s exposure to a risk event. These variables are not explicitly accounted for in the traditional definition of risk. They are usually left to be implicitly considered during risk identification and evaluation processes.

A fourth limitation is that the definition encompasses only known or foreseeable threats. It provides limited options for managing realized threats and it does not recognize unforeseeable threats. This is a consequence of defining risk in terms of probability of impact. To assess the probability of an impact one needs the ability to foresee an eventuality [36]. Despite all the flaws in the definition of risk, this is still the most widely acceptable definition by the software practitioners.
2.3 STUDIES RELATING TO RISKS AFFECTING THE SOFTWARE PROJECTS

Considerable amount of research has been conducted in the areas of risk identification, analysis and management of software projects. Investigators in this area have tried to identify the various risk factors that affect the success of the software projects and have also proposed various risk management models for better supervision of these threats [19] [22] [24] [26] [35] [36] [42] [107] [108]. Inspite of all this, the success rate of software projects is not very positive. Risk researchers corroborate the high failure of software projects, estimating that one-third of software projects fail or are abandoned. The unacceptably high rate of system failure across the board regardless of the size or complexity of the project, has commanded investigations of why these problems occur and what can be done to prevent them [109] [110] [111]. Furthermore, recent evidence suggests that reports from project managers of on-going projects may be optimistically biased, distorted or even non-reported, thereby further exasperating managers efforts to detect potential project pitfalls [112] [113]. This section deals with various work done on identification of risks and their mitigation.

The extant literature has produced a number of conceptual frameworks to explain different types of software development risk, risk management strategies and measures of software project performance [24] [36]. The majority of risk management studies deals with normative techniques of managing risk [47] [96]. Some empirical studies have tried to understand how one can effectively manage software risk. Using case study data they have discussed which risk management principles were not followed and tried to identify reasons for not following the same. Overall, these studies provide illuminating insights into risk management deliberations, but are weak in explaining the true impact of risk management in generalising from observations. A few studies have gone further to establish systematic models of risk management [47] [48]. They all conclude that risk management efforts reduce the exposure to software risk and can thereby increase software quality and improve software development. Some studies focus solely on project delays or deals indirectly with software risks [114] [115] Overall, the understanding of how software risk management can improve software development has remained fragmented and largely anecdotal.
In terms of previous efforts to identify risk factor, **Boehm’s work [19]** has probably had more influence on practitioner’s community than any other [15]. Boehm [19] identified top ten list of software risks based on his experience in defence industry. According to him 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, straining computer science capabilities are the top risks that a software company faces. To manage these risks Boehm proposed theory “W: Make everyone a winner”. This theory worked on two subsidiary principles: “plan the flight and fly the plan and identify and manage your risks”. Accordingly, various mitigation strategies to counter the above risks were proposed such as staffing with top talent, requirement scrubbing, prototyping, reference checking, detailed multisource cost and schedule, compatibility analysis etc. Although, the study details out the risks and their mitigation but fails to convey the tools used to derive these risks other than the fact that Boehm [19] used his own experience and a survey of the experienced project managers to identify these risks. According to Ropponen and Lyytinen [26], the study lacks a theoretical foundation. It has multiple items which refer to the same phenomenon.

**Sherer [116]** used a three dimensional framework to explain the software risks. She identified technical, organization and environment as three dimensions on which the software risks were explained. To combat these risks various risk mitigation technique were also proposed. In the study, technical risks were identified as the most important risks affecting the software projects. Use of HIPO charts, data flow diagrams, prototyping, simulation modelling, benchmarking, fault tolerant methods were some of the risk management tools proposed in the study. Her study fails to link these risks to the Software Development Life Cycle. Moreover, the study fails to provide constructive tools to mitigate the risks relating to software vendor relationships and outsourcing.

**Field [117]** in his article states that, “projects fail too often because the project scope was not fully appreciated and/or user needs not fully understood”. He developed a comprehensive list of pitfalls that must be avoided to execute a successful software project. The list includes misunderstanding user requirements, project scope ill defined, poorly managed changes; change in the chosen technology, business needs change, deadlines unrealistic, resistant users, lost sponsorship, lack of experienced personnel, best practice and lessons ignored by managers. This
list was re-used by Reel [118] to adduce various risk management strategies such as building the right team, giving team what they need, involving customers or user in the development, setting up procedures and expectations for high level of quality before the development, tracking the progress religiously and institutionalizing a process for learning from past mistakes.

Keil et al. [15] employed a variation on the traditional Delphi survey approach to elicit opinions from a panel of experts through iterative, controlled feedback. Groups of experienced project managers from USA, Hong Kong and Finland were formed and list of top ten risks along with their mitigation were identified. The list includes the following key risk factors and their mitigation: a) Lack of top management commitment to the project, lack of adequate user involvement and failure to gain user commitment; can be mitigated by relationship management, trust building and political skills. b) Misunderstanding the requirements and lack of frozen requirements can be managed by educating the user/customer on the impact of scope changes in terms of project cost and schedule; tools such as multi-criteria decision making and function point analysis can also be used to mitigate these risks. c) Failure to manage end user expectations, lack of required knowledge/skills in project personnel, introduction of new technology and insufficient/inappropriate staffing can be mitigated by internal reviews coupled with external reviews, use of work-breakdown structure, development of contingency plans to manage personnel shortfall and use of new technology. d) Changing scope/objectives and conflict between user departments can only be managed by contingency and disaster planning. Inspite of such detailed analysis this study fails to link it with the software development cycle.

Oz et al. [107] collected quantitative and qualitative data about reasons why software projects fail. They conducted a factor analysis of the survey and identified lack of corporate leadership, poorly communicated goals/deliverables, inadequate skills and means, poor project management and deviation from timetable/budget as the key risk factors.

Ropponen and Lyytinen [26] using survey instrument on eighty project managers delineated six components of software development risk. These were: scheduling and timing risks, systems functionality risks, subcontracting risks, requirements management risks, resource usage and performance risks and personnel management risks. They proposed various risk mitigation strategies to extenuate the identified risk, mainly concentrating on the following: standardization
of the process, making risk management an integral part of the project, hiring of experienced project managers and controlling or decomposing the size of the project.

Schmidt et al. [119] developed a comprehensive list of fifty three factors affecting the software project, using inputs from multicultural set of forty one practicing project managers. Three panels were formed on the basis of their cultural background and Delphi technique of decision making was used to identify the critical risk factors affecting the software projects. The risks identified were lack of top management commitment, failure to gain user commitment, misunderstanding of requirements, lack of adequate user involvement, lack of required knowledge/skills in project personnel, lack of frozen requirements, changing scope/objectives, introduction of new technology, failure to manage end user expectations, insufficient/inappropriate staffing, conflict between user departments.

Jiang et al. [120] made use of software risk measurement instrument pertaining to various characteristics of a software development projects developed by Barki et al. [97] and pointed out that project size, application complexity, technology acquisition, insufficient resources, lack of team expertise, lack of user support, lack of user experience, lack of clear role definition, intensity of conflicts are the top nine risk factors that a software project can be exposed to.

According to the study conducted on the Indian software industry, the Indian software industry is not only facing the problem of attrition from the home front but is also facing problem from the client side as well. There were a number of cultural and political issues that US managers perceive as irritants or barriers. One such issue is the apparent unwillingness of Indian software professionals to point out potential problems up-front, and in general, an unwillingness to say no for fear of offending the clients. Another related weakness is the lack of familiarity of many Indian firms and professionals with the work culture and work norms in the west, and especially in the United States. Other difficulties include resistance within the US to foreign programmers, poor telecommunication infrastructure, and the delays in obtaining the required visas for Indian programmers [66].

Addison and Vallabh [121] conducted a three phase Delphi survey of the South African software industry and identified unclear or misunderstood scope/objectives, misunderstanding the requirements, failure to gain user involvement, developing the wrong software functions,
unrealistic schedule and budgets, continuous requirement change, inadequate knowledge/skills, lack of effective project management methodology, gold plating as the key risk factors. They also studied the effectiveness of various controls or strategies to reduce the occurrence of risk factors. They found that experienced project managers use controls such as assigning of responsibilities to team members and stabilizing requirements and specifications more than inexperienced project managers. The study lacked in the area of data collection as the sample size taken was too small to generalize the risks and their mitigation strategy.

DeMarco and Lister [122] identified schedule flaw, requirements inflation, employee turnover, specification breakdown and poor productivity as the core risk factors of software development projects. While Lu and Ge [123] analysed the risk of software development projects in two aspects - one for owners (project’s feasibility and knowledge management problem) and another for contractors (scope change risk and management risk) in China using AHP technique.

In-depth interview with IT professionals from leading firms in Australia were undertaken by Baccarini et al. [37], to ascertain the IT risks and their mitigation. The top ten risks identified in the study were: personnel shortfalls, unreasonable project schedule and budget, unrealistic expectations, incomplete requirements, diminished window of opportunity due to late delivery of software, continuous changes to requirements by client, poor production systems performance, poor leadership, inadequate user documentation, lack of agreed user acceptance testing and sign-off criteria. Rich and valuable array of mitigation techniques were identified and categorised as avoidance, reduction, transfer or acceptance strategies. However, the study failed to provide any framework for software risk management.

In a study conducted by Smith [124] in Africa, lack of top management commitment to the project, unclear/misunderstood objectives, schedule flaw, lack of client responsibility; ownership and buy-in of the project and its delivered systems, no planning or inadequate planning, project not based on sound business case, lack of available skilled personnel, not managing change properly, lack of adequate user involvement, poor risk management came out to be the major risk factors affecting the software project.

Costa et al. [24] in their study presented a technique for assessing the risk level of a software project based on its systemic and specific risks. They introduced an approach to estimate the
probability distribution of earnings and losses incurred by an organization according to its software project portfolio. This technique was supported by an empirical study to assess the relative importance of risk factors for software development projects. The following risk factors were keyed out: client risk, control risk, analysis risk, team risk, testing risk, policies/organization structure risk and design risk. However, no attempt was made in the study to address the issue of managing and mitigating these risks.

**Dey et al. [42]** through a case study on TCPO in Barbados identified the key risk factors as: the unavailability of key personnel, employee turnover and incorrect/incomplete requirement. The study also developed an integrated framework for managing risk in software development with the involvement of the stakeholders in TCPO. To control risks in the project they proposed constant project monitoring, dynamic scope management plan, involvement of client in the development process and effective communication between developer and client as some of the strategies for mitigating the risks.

**Verner et al. [126]** conducted an exploratory statistical analysis to identify determinants of project success and used logistic regression to predict project success. According to the study, success is more likely to happen if the project manager is involved in schedule negotiations, adequate requirements information is available when the estimates are made, initial effort estimates are good, take staff leave into account, and staff are not added late to meet an aggressive schedule. The study is quite comprehensive as far as prediction of project success is concerned but it does not cover all the factors that affect the success of the project. Defining success only from the perspective of estimation, staffing and scheduling is incomplete in itself.

**Zhou et al. [41]** analysed ten case studies in the UK, USA and New Zealand to identify critical risk factors (scope creep, unwillingness of customer to accept final systems, poor project management, etc.) at the pre-implementation and implementation stage of the software project. The study is limited to identification of risks at pre-implementation and implementation stage and ignores the risk occurring in post-implementation stage of software projects. The study fails to provide any insight into the risk management and mitigation strategies.

**Bannerman [36]** conducted a study in government agencies in Australia to investigate the practices of a state government in dealing with software projects. Analysis of the study uncovered
ten categories of risk factors namely: project governance, project setup, partner engagement, business proprietorship, project management, change management, management of projects, recognition of red flags, management of risk and benefit realization. The study also identified the various risk management techniques used in these projects. Overall, the study aimed at reviewing and reassessing the status of risk management research in literature and practices in a sample of Australian public sector agencies. The study is helpful as it provides a complete understanding of risk and risk management in public sector. Whether the same can be applied in private sector software companies is not clear from the study.

Iacovou et al. [124] identified lack of top management commitment, original set of requirements miscommunicated, language barrier in project communications, inadequate user involvement, lack of offshore project management know-how by client, failure to manage end user expectations, poor charge control, lack of business know-how by offshore team, lack of required technical know-how by offshore team and failure to consider all costs as the core risk factors. The study aimed at providing an insight into risks affecting offshore – outsourced development projects but fails on delivering the risk management techniques to combat these risks.

Anudhe et al. [22] using case-based methodology and structured and semi-structured interviews with senior management of various Indian software companies keyed out various risk factors affecting the software projects. Schedule and budget management (developing a collaborative work culture with clients), client expectations (educating the client to involve deep level of association with the customer), requirements capture (elaborate data collection and proactive analysis), staffing (maintaining buffer resources, involving client in resource recruitment), changes in client’s corporate structure (transparency and adequate communication) are few of the risk factors and its mitigation mentioned in their study. However, the study fails to define any model for risk management.

Thus, it can be clearly seen that numerous researches have been devoted to the identification and management of the software risk. Various methodologies have been used for keying out the list of major risk factors. As a result, there have been various lists of risk factors with some similarities and some differences [40]. Therefore, a comprehensive and exhaustive list encompassing all the risk factors affecting the software projects is prepared in the following section.
2.3.1 A Risk Classification Framework

Many different risk factors impacting the success of the software projects have already been identified by various researchers. Some of these risk factors are quite detailed and affect only specific projects in specific conditions. Some factors are, however, reported as very commonly encountered and having strong impact on the project’s chance of success. Some of the risk factors are controllable while others cannot be controlled by the project manager [15]. While some risks are more important as they have a direct impact on the project’s outcome than compared to the other. Thus, on the basis of this proposition the risks identified by the researchers have been classified on the basis of importance and control as shown in figure 2.3 and have been explained in the following section.

**Figure 2.3:** Risk Classification Framework
2.3.1.1 Stakeholder Management Risk

As the name suggests, this quadrant captures the notion that successful projects are very often those that have the commitment of both senior management and the end-users i.e. those who will actually use the system. Without a clear charter, or mandate, the project is simply not viable [15]. Projects in which either top management or user commitment is lacking represent a high-risk proposition. But initial commitment is not enough. Once a project has started, project managers must periodically gauge the level of commitment from both top management and the user community to avoid being caught in a situation where support for the project suddenly evaporates. These risks fall under high risk and low control category as they cannot be controlled by the project manager, but they can be moulded. Project managers must take reasonable steps to ensure that they have the support and commitment needed to deliver a successful project. The list of risks that fall under this category are:

1. *Lack of top management*

   Project is disrupted from achieving its objectives owing to management playing politics within and between departments or external agents. Furthermore, users may not support the project if they perceive that there is a lack of top-level management sponsorship. Keil et al. [15] found that lack of senior management commitment is the most critical risk impacting the software projects. Studies have time and again empathized that the top management support is needed throughout the implementation and top management needs to publicly and explicitly identify the project as top priority [127] [128] [129] [130].

2. *Corporate culture not supportive*

   Corporate culture may be project adverse owing to other hidden agendas, factions within the company, organizational culture under continuous change or threat of change, and other internal priorities. All this will result in weak management support for the project and consequential failure of meeting objectives [37] [127] [131] [132].

3. *Inadequate user involvement*

   This risk has been repeatedly mentioned by various researchers. This risk features in the top ten causes of software failure list of many researches. If the client is not involved in the project it will
result in wrong interpretation of scope and objectives of the project and thus may lead to loss of money, time and even life [15] [41] [121] [124].

4. **Lack of client responsibility and ownership**

This risk was also fundamental on the list identified by Keil et al. [15]. If the users are not involved, there is a risk that developers may assume detailed functionality and business requirements, leading to project objectives not being achieved. It also means laying the blame for ‘lack of client responsibility’ on the project manager rather than on the users [133].

5. **Friction between clients and contractors**

Personal animosity or enmity can occur between clients and software contractors as a result of misunderstandings, unanticipated changes in the scope of the contract, missed or delayed delivery, or some other item of dispute that polarizes clients and contractors into opposing camps [134].

### 2.3.1.2 Requirement and Schedule Risk

Many projects face uncertainty and turmoil around the product’s requirements. While some of this uncertainty is tolerable in the early stages, the threat to success increases if such issues are not resolved as the project progresses. If requirements-related risk factors are not controlled, it might result in either building the wrong product, or building the right product badly. Either situation results in unpleasant surprises and unhappy customers. The above statements clearly show how crucial this quadrant is. The risks in this quadrant can be largely controlled by the project manager, but do require skilful interfacing with the user or customer. The risks that fall under this quadrant are:

1. **Miscommunication of requirements**

Sometimes it is seen that the client himself is not sure what he desires from the project. It has also been found through literature that conflict among the users further accelerate this problem of miscommunication of requirements. Infact, miscommunication of requirements is one of the biggest risk factors as miscommunication can complicate the transmission of the original set of requirements and subsequent information exchanges and change requests [125].
2. **Unclear scope/objectives**

Boehm [19] point out that the different stakeholders in a software development project have different objectives, which often conflict with the objectives of another stakeholder. For instance, users require a robust, user-friendly system with many functions that can support their tasks while on the other hand, development team members hope to encounter interesting technical challenges. These differing expectations create fundamental conflicts when simultaneously approached, resulting in unclear or misunderstood scope/objectives of the project. Furthermore, ill defined or ambiguous requirement specifications are equally dangerous and are likely to originate problems of usefulness and deviations from both timelines and budget [19] [41] [135].

3. **Changing requirements**

Stakeholders (includes users) continuously make changes to software functionality throughout the project life-cycle. As the users’ needs change, so do the requirements of the project. One of the researchers suggested that by freezing a part of the functionality and delivery date, completion of the system is enabled. While, on the other hand it has also been argued that requirements should not be frozen as in today's fast moving business environment, a frozen design does not accommodate changes in the business practices. With a frozen design, the developer has little flexibility in changing the specifications. Continuous and uncontrolled changes in requirements, however will inevitably lead to a delay in the project schedule [10] [15] [133] [134] [136] [137].

4. **Improper change management**

Without proper software change management, enterprises lack a full understanding of how software running in production automates their business processes. This includes management of changes to software in development, changes to software in production, and changes to associated artefacts like requirements, models, and test cases. This also includes management of both individual changes and the coordination of dependent changes. Researchers have mentioned that improper change management as one of the leading causes of software failure [124] [138] [139] [140] [141].

5. **Unrealistic schedule and budget**

The project is unable to realize its objectives owing to unrealistic restrictions placed on the projects budget, schedule, quality or level of performance. A project failing to meet its committed
deliverables or being significantly over budget can result in termination of the project. A number of researchers have stated that the ‘scheduling and timing’ risk is a major complicating factor as it is difficult to estimate schedules with acceptable accuracy and consistency. Very often, organizations embark on a large project having underestimated its size and complexity. This risk leads to the difficulties in scheduling the project correctly. Ropponen and Lyytinen [26] believe that performance with scheduling and timing risks improves with project experience. A fixed schedule may lead to schedule pressures and people under pressure do not necessarily work better, resulting in the inability to produce satisfactory results or deliver any software at all [19] [136] [142] [143] [144].

6. Misunderstanding of requirements

It may be time consuming and difficult to collect and record all of the required details from all prospective users, resulting in the project team not knowing enough about what is required to complete the project successfully. This may lead to the possibility of developing a system that cannot be used, mainly because, a proper systems’ analysis to develop a complete and accurate set of requirements has not been performed. A number of researchers have identified misunderstanding of requirements as one of the top ten risks affecting the software projects [15] [117] [119] [121] [133] [145].

7. Unrealistic expectations

According to Keil et al. [15], problems with user expectations can occur whenever user expectations are not realistic. This happens due to inadequate planning and failure to gain sign off from the client. Sometimes the project team is also exposed to unrealistic expectations from the top management or from the project manager. This de-motivates the team thus leading to serious issues in software projects.

8. Gold plating

Gold plating means that the team is focused on analyzing and generating excessive levels of detail while losing sight of the project’s objectives. Often developers and analysts think of additional capabilities or changes, known as gold plating, which they think would make the
system better and more attractive in their view. These deviations may result in unsatisfied users and unnecessary costs [19] [146] [147].

9. Inaccurate estimation of schedule or cost

An inaccurate initial estimation can lead to inadequate allocations of budget, or time, or both, to the project, which can then cause it to fail. The estimate drives every aspect of the project, constrains the actions that can be taken in the development or upgrade of a product, and limits available options. It is believed that most impromptu estimates of project scope based on the engineering or management experience are incorrect and are often based on simple assumptions and over-optimism; or worse are made to accord with what others want to hear. Needless to say, such estimates often lead to disaster. If the estimate is unrealistically low, the project will be understaffed from its outset and, worse still, the resulting excessive overtime or staff burnout will cause attrition and compound the problems facing the project. In turn, overestimating a project can have the same effects as any other inaccurate estimate [93] [148].

2.3.1.3 Project Management Risk

The risks under this are related to the actual execution of the project. The project manager has reasonable control over these risks, and hence these risks are regarded as moderate. Under normal circumstances, these risks do not pose a serious threat to the project; hence, they fall under low risk category. But project managers cannot afford to become complacent in their handling of these risks. Failure to manage the risks in this quadrant can result in poor quality software that is delivered late and/or over budget. The following risks belong to this quadrant.

1. Inadequate plans and procedures

Inadequate planning can cause the entire project to go haywire. Planning helps in identifying the loopholes in the project and also enables the project manager to provide tools and techniques to overcome these loopholes. As a matter of fact meticulous planning at the time of pre-project stage helps in identifying risks at early stages of implementation. This risk has also been mentioned as one of the top ten risks affecting the software projects. [124].
2. Lack of project management methodology

Poor project management is universally accepted as a major cause of risk and failure in software projects. A full and complete project plan may not necessarily be presented but a comprehensive and proper project management strategy with clear definition of roles and responsibilities must be initiated as soon as possible. According to researchers requirements are frequently not explicitly fixed in any reasonable document. Project management from the customer side is most often done by just forwarding e-mails (and forwarding them again, and again) until they reach suppliers. It is then difficult to figure out the chain of responsibility, management, and resourcing for each separate change request. Therefore, it is very important to define project member roles and establish proper communication liaisons [41] [149] [150].

3. New technology being introduced

This risk occurs by using new or ‘leading edge’ technology that has not been used successfully at other companies. This risk may be increased further if there is a shift in technology during the project. Studies have stated that stability and compatibility of hardware and software platforms is the major cause of problem. Unproven or unfamiliar technologies may cause disappointment and lead to under-performance or conflicts [41] [133].

4. Lack of single point accountability

It is typical of large software projects to have many team leaders but no single point of responsibility for deliverables, resulting in the project failing to meet its objectives. If there are multiple contact points with the client as well, it might create a situation wherein the client keeps on informing different things to different contact points thus delaying the project [151].

5. Lack of technical knowledge

Project personnel may not have adequate knowledge of the technology, or the business, or may just not have the experience to handle the project. McLeod and Smith [152] indicate that ‘people’ risk arises from inadequate skills (both technical and managerial) as well as level of experience. The lack of experience with technology also increases the likelihood of occurrence of this risk.
Infact, this is again one of the major causes of delay and failure of the software projects. [15] [17].

6. **Inappropriate staffing**

Inappropriate project staffing leads to failure of software projects. One of the main reasons behind this is the lack of proper project planning and estimation. Excess staff will result in increase in cost to the project where as under-staffing will result in excessive overtime or staff burnout thus causing attrition and poor performance [153].

7. **High level of attrition**

Employee turnover during the project has tremendous negative impact, as it gets extremely difficult to get competent experienced technical persons within a short period and furthermore, it takes time for them to adjust in a new environment. These have negative impact on productivity of software development projects [41].

8. **Lack of commitment from project team**

Lack of commitment from project team results in poor performance and delivery of poor quality product. Studies have confirmed that lack of commitment from the team is due to lack of trust, lack of balance or diversity, poor socialization, and inadequate communication [45].

9. **Lack of mechanism of validation and verification**

System development and testing is probably the most critical phase of any software development project. Adequate programming and testing methods and techniques need to be adopted. The use of unstable and sometimes incompatible software and hardware platforms may pose significant risk to the project. Tools needed for testing and verifying the application or product are indispensable for a successful implementation and deployment of the software. If correct tools are not available on time, it may delay the deployment, thus affecting the overall project schedule [41].
10. Inadequate tools for reliability

Reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Reliability is an important attribute of software quality, together with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software reliability is hard to achieve. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software. Realistic constraints of time and budget severely limit the effort put into software reliability improvement [154].

2.3.1.4 Environment risk

Risks in this quadrant can be traced to the project environment that exists both within and outside the organization. Risks falling under this quadrant are those over which the project manager has little or no control. They have a low likelihood of occurrence and are, therefore, not viewed as being crucial. However, if they do occur, they can be serious threat to the project. These risks are most difficult to predict. The risks under this quadrant are:

1. Inadequate third party performance

The contractor selected is not fit for the purpose of the project. The contractor is unable to provide a solution that meets time, cost, quality and performance objectives. This may gravely affect the overall performance of the project as any delay from the third party may derail the entire project schedule resulting in increase cost and time [142].

2. Competition alters schedule

Globalization has given rise to new challenges and one such challenge is to keep up with the competition. Competitors may build software solutions more quickly, with greater functionality at cheaper cost, aggressively deploy the final product within the same market space [36] [134].

3. Change in scope due to change in business model
With ever changing environment it is very important to keep up with the project schedule. Studies have found that sometimes due to changes in competitive environment the software becomes outdated and is no longer needed by the user. According to a number of researchers business return on investment in IT can be eroded owing to changing consumer market conditions or advancements in software engineering. Sometimes, the entire project may be scraped or terminated due to change in the business model. The top management may withdraw support to the project due to restructuring of the business or even client’s organization may suddenly become disinterested. These have been confirmed by various studies as well [37] [134] [136].

4. Natural disasters

Natural disaster has a very rare occurrence yet it is one of the most crucial risk factor. One it is uncontrollable and two it can cause a complete damage to the project resulting in loss of money, time and even life.

After identifying and classifying the risks on the basis of control and importance, it is important to understand how these risks can be abrogated and success of the software project be ameliorated. One of the approaches towards increasing the rate of success of the software projects can be presence of strong, robust, open and free organizational climate in the software development companies. “What is an organizational climate?”, “How does it impact the software project?”, “Is there any relation of success of software project with the organizational climate?” The next section aims to answer these and present an overview of the organizational climate and its related work in the area of software projects.

2.4 ORGANIZATION CLIMATE

The construct of climate was first introduced in the 1960s, primarily based on the “social climate” and “social atmosphere” variables proposed by Lewin et al. [155] [156] and followed by empirical research conducted in organizational settings [157] [158] [159] [154]. Since Likert’s [158] early empirical work, organizational climate has been viewed as a fundamental building block for describing and analyzing organizational phenomena [160]. Cumulative research demonstrates that employee climate perceptions have important effects on both individual and organizational outcomes, such as work attitudes and satisfaction, job performance, service
quality, customer satisfaction, workplace accident rates, TQM outcomes, and organizational financial performance, among other things [161] [162] [163] [164] [165] [166] [167] [168] [169] [170].

2.4.1 Definition
Organizational climate has been studied quite elaborately and various researchers have defined climate in numerous ways. According to Kopelman et al. [171], organizational members are active perceivers and interpreters of their work environments, and employees tend to form their perceptions by observing how the daily operations of the organization are conducted and what goals the organization appears to be pursuing. Schneider and White et al. [172] noted that the organization transmits this information to employees through its policies, practices, and procedures (e.g., human resources policies, marketing practices, operations management procedures), which collectively send messages about what is important—what behaviours the organization rewards, supports, and expects. Based on these behaviours and activities, employees develop a summary sense of “what is important around here,” which represents climate. Climate represents the patterns or themes that employees perceive in what they experience; it is one way to conceptualize the totality of the experiences organizational members have of their workplace [172]. It can be viewed as subjective, temporal, and potentially subject to managerial manipulations [173].

Climate has also been defined as “the shared perceptions of employees concerning the practices, procedures, and kinds of behaviours that get rewarded and supported in a particular setting” [167]. According to Litwin et al. [159], organization climate can be defined as “the perceived attributes of an organization and its sub-systems as reflected in the way an organization deals with its members, groups and issues”. Isaksen et al. [174] defines climate as “the recurring patterns of behaviour, attitudes and feelings that characterize life in the organization. At the individual level of analysis, the concept is called psychological climate. At this level, the concept of climate refers to the individual perceptions of the patterns of behaviour. When aggregated, the concept is called organizational climate”.

Ekvall [175] and Isaksen and Ekvall [176] have defined organizational climate in terms of the interplay of institutional policies, goals, strategies, tasks, workload, resources, technology and
staff. They suggested that creative outcome is most likely to happen if the organizational climate does the following:

1) Challenges individuals with tasks, goals and institutional operations. Work must be meaningful and “the development and survival of the organization is important” to employees.

2) Employees must have opportunities and initiative. This may be apparent in communication within and outside the organization and in the methods available to obtain information.

3) There must be support for new ideas. They should be encouraged and rewarded.

4) Employees must be trusted and employees should feel that trust. Risk is minimal because employees know they are trusted and in turn trust the organization.

5) Risk taking is supported. Risk is viewed as a part of the creative process.

They described a measure of organization climate that covers 10 areas: support for ideas, challenges, time for ideas, freedom, trust and openness, dynamism, risk taking, playfulness and humour, debates and conflicts and impediments. Yet another most widely used framework for identifying organization climate has been given by Litwin et al. [159]. This framework emphasises on motivational linkages and seems to be quite relevant for studying organizational climate [177]. The framework considers six dimensions of organizational climate that include: structure, responsibility, reward, risk, warmth and support. Thus, climate is a perceptual mediator through which the effects of the work environment on employee behaviour pass [171] [178].

It is necessary to distinguish climate from culture, because the term culture is often used when climate is the more appropriate term [179]. Climate is about experiential descriptions or perceptions of what happens; it can most accurately be understood as a manifestation of culture [179]. In contrast, culture is a deeper phenomenon based on symbolic meanings that reflect core values and fundamental ideologies and assumptions [180] [181].

2.4.2 Organization Climate and Software Projects

Ein-Dor and Segev [182] perhaps conducted the first study of climate in Information Technology. He examined the relationship between climate toward Management Information System and the quality of developer-user relationships, the degree of system use, and system integration in the
Literature Review

organization. Boynton et al. [183] developed a conceptualization of the overall “IT management climate”, which is related to managerial IT knowledge and IT management-process effectiveness. More recently, Watts and Henderson [184] explored the concept of innovative IT climate and discussed how it is related to IT innovation. Jia and Reich [71] developed a new construct, “IT service climate” and tried to establish relation between IT service climate and service quality. They also developed a theory–based framework to help project managers identify the causes of service failure. Hyvari [185] evaluated the critical success/failure factors and their relationship with organizational background variables such as project type, project size etc. using chi-square analysis.

Huang and Trauth [186] conducted a cross-cultural study on how culture influences the globally distributed software development between U.S and China. They conducted in- depth interview with Chinese IT professionals and found improving language skills, fostering organizational culture of valuing diversity and generation of innovative ideas along with adaptation of such organizational culture will help in improving the software quality and overall software development. An empirical study examining the work climate within the software development teams was conducted by Acuna et al. [187]. Team selection inventory test and team climate inventory test were used to gauge the developer’s climate preference before and after the project and developer’s perception of the climate respectively. According to the study, participative safety and team vision preferences and perceptions fit correlated with improved software quality. Keil et al. [35] conducted a controlled experiment on 134 graduates and found information asymmetry and organizational climate to have significant indirect effects on the willingness of the employees to report negative project status information.

Researchers have also studied the impact of organizational climate factor on software developer’s job satisfaction, job performance and project success [48] [49] [50]. Woodruff [48] studied organizational climate and its impact on job satisfaction and its impact on job satisfaction and job performance. His study included 202 software developers from 12 different organizations in United States. The studies identified management’s ability to make decisions, selection of personnel along with inter-group cooperation and receptiveness by management significantly affected the job satisfaction. Mclean et al. [49] studied the impact of compensation on job satisfaction of the software developers. A study by Rasch et al. [50] examined the relationship
among self esteem, goal difficulty, goal specificity, role ambiguity, effort, achievement needs, locus of control and ability on individual software developer performance. They found that individual ability and achievement needs have a direct effect on software developers’ performance.

There is also work on individuals’ psychological climates, such as creativity climate for IT professionals, climate for business process change, ethical climate of IT professionals, technical updating climate, climate for reporting bad news in software projects, climate for knowledge-sharing [188] [189] [190] [191] [192] [193] [194]. Besides this, a number of researchers have also have mentioned that organizational climate has an impact on technical software development issues; thus, creating wide range of potential trouble points in the project [27] [28] [29] [30] [87].

2.5 PERSPECTIVES OF SOFTWARE PROFESSIONALS

The concept that different stakeholders can perceive software projects in different ways is also well established in the literature [46]. Keil et al. [141] have demonstrated that users and project managers differ in terms of their project risk perceptions. While Warkentin et al. [30] [195] and Stephen et al. [196] have exhaustively studied the perception of risk among various demographic characteristics of software professionals and have suggested that professionals with more experience in project leadership were more likely to view projects, and their associated risks, more holistically and assign and resolve risk as if they were organizational in nature. Researchers have also studied the perceived importance of risk factors among the experienced and inexperienced project managers and concluded that experienced project managers perceived different risks to be important compared with inexperienced project managers [124]. Liu et al. [197] have compared the senior executive and project manager perceptions of IT project risk in Chinese software industry. They concluded that project managers tend to focus on lower-level risks with particular emphasis on risks associated with requirements and user involvement, whereas top management focus on higher-level risks such as those risks involving politics, organization structure, process, and culture. Besides this, studies have also been done on the perception of success of the software project among various stakeholders by [74] [78] [80] [84].
These studies have concluded that there is a marked difference in the perception of the success of the software projects among various stakeholders of the project.

2.6 CONCLUDING REMARKS

Thus, it can be clearly seen that tremendous amount of research has been conducted on software risk management. Ample literature is available which explains how software projects can be successfully delivered and various studies have been conducted to study the organization climate and its impact on software quality, innovativeness, team satisfaction etc., yet the success rate of software projects remains a mere 32% [9]. This clearly shows that there exists a definite gap between the researchers and practitioners approach. Some of the gaps identified in the contemporary research are as under:

- Most of the studies done on identification and management of software risk have been conducted in the developed countries but there are very few studies on the identification of risk dimensions affecting the software projects in India.

- In the literature, affect of organizational climate on success is well documented but its impact on software risks is largely anecdotal. Further, the dynamics of organizational climate on software risks and success is missing in the Indian context.

- It is also seen that there are few studies done on the perception of software risks among various groups of software professionals. There is a paucity of research on the study of perception of software risks and organizational climate factors among various groups of software professionals based on demographic and project characteristics.

- There is a paucity of research on gauging the collaborative impact of organizational climate and risk factors on the success and the three success constructs namely budget, schedule and quality.

In view of the above literature, this research is directed towards understanding the dynamics of organizational climate factors on software risks and success of the software projects in India. It also aims to gauge the perception of risk and climate factors among the various groups of software professionals based on the demographics and project characteristics.