Risk Identification and Preemptive Scheduling In Software Development Life Cycle

GJCST Computing Classification
D.2.6, D.2.2 & K.6.1

Basit Shahzad, Abdullah S. Al-Mudimigh, Zahid Ullah
College of Computer & Information Science
King Saud University, Riyadh, Saudi Arabia
Basit.shahzad@gmail.com , mudimigh@ksu.edu.sa, zahid@ksu.edu.sa

***Abstract*- Software development has emerged as a disciplined discipline and the use of process models to develop the software has increased over time. Although the software industry is blessed with quite a few tool driven approaches, and the usage of technology is increasing yet the amount of risks faced by the software development life cycle have also increased to an extent. This paper focuses on the avoidance and mitigation strategies for the already identified and prioritized risk factors.**

Keywords- Preemptive risk identification, software risk handling, risk mitigation

I INTRODUCTION

Software risk management has been a very hot area of research since last three decades. Recently, the research community looks seriously interested to identify not only the risk factors but also the causes of the appearance of the risk factors in software development life cycle and how these risks can either be handled or avoided. A recent survey of 600 firms indicated that almost 35% of them had at least one 'runaway' software project [1]. In another study, conducted on almost 13,000 projects, it was investigated that almost 25% of the projects were either delayed or faced a failure. It has been observed that most problems in the software industry are faced just because of the poor software risk handling mechanisms or due to the absence of any such mechanism at all. In this regard it is important to note that currently strong emphasis is being given on this domain to identify more and more risk factors. Pressman [3] has made an effort to identify the software risks, and has provided the ten broader risk factors. Bohme, in his work has also provided a list of top ten risk categories[4]. Basit Shahzad, [5] has also worked in this domain to identify a relatively more detailed list of software risk factors and also identifying the relative impact of each risk factors. In a recent paper on risk management, the risk factors have been prioritized according to their frequency of occurrence and the impact that they possess [6], and thus a list of eighteen risk factors with respect to their total impact has been prepared. The list is presented in the table 1 and table 2. Table 1 presents the list of all 18 risk factors, while table 2 presents the ordered list of software risk factors w.r.t. the overall impact of each risk factor.

Risk #	Impact	Probability	Overall Impact	Risk #	Impact	Probability	Overall Impact
2	519.5	50%	259.75	1	55.2	50%	27.6
15	191.4	50%	95.7	6	208.5	10%	20.1
11	185	40%	74	16	195.6	10%	19.6
13	348.5	20%	69.7	10	61.3	30%	18.39
18	208.5	30%	62.55	9	145.6	10%	14.6
17	83.5	50%	41.75	8	56	10%	5.6
5	136	30%	40.8	12	15	20%	3
4	114.8	30%	34.44	7	6.6	30%	1.98
14	292.5	10%	29.3	3	3.6	30%	1.08

Table 1: The risk factors w.r.t their identifier

The risk factor identified in this list is expected to cover a border range of the risks that may come into the software development process. Still the author feel himself restricted, not to claim that this list covers all possible risk factors. It is strongly believed that the risk identification, particularly, is an ongoing process, and apparently there is no full stop as the risk factors keep on increasing with the arrival of new technologies, people, environment, management and the circumstances. So a claim about the identification of all risk factors available in the entire software process, may not be realistic. Table 2, presents the ordered list of available risk factors, by calculating the overall impact and frequency of each available risk factor [6].

In table 2, the term "Impact" means the impact of that specific risk factor, e.g. the risk factor number 2 has the impact of 519.5 and risk factor number 7 has the impact of 6.6. The term "Probability" means the possible occurrence of the risk factor. The term "Overall Impact" describes the impact of a risk factor with respect to the probability of each factor. The risk factors have been ordered with respect to the overall impact they possess, in ascending order, showing the maximum overall impact of 259.75 for risk factor number 2 and minimum overall impact of 1.08 for risk factor number 3.

Risk #	Impact	Probability	Overall Impact	Risk #	Impact	Probability	Overall Impact
2	519.5	50%	259.75	1	55.2	50%	27.6
15	191.4	50%	95.7	6	208.5	10%	20.1
11	185	40%	74	16	195.6	10%	19.6
13	348.5	20%	69.7	10	61.3	30%	18.39
18	208.5	30%	62.55	9	145.6	10%	14.6
17	83.5	50%	41.75	8	56	10%	5.6
5	136	30%	40.8	12	15	20%	3
4	114.8	30%	34.44	7	6.6	30%	1.98
14	292.5	10%	29.3	3	3.6	30%	1.08

Table 2: List of prioritized risk factors

II STATE OF THE ART

Software risk identification and mitigation has been a prime area of research since last two decades, and this area of research has received a highly overwhelming response and contribution from the researcher both: in industry and academia, world-wide. In order to identify the recent trend and practices in the domain of software risk identification a comprehensive literature survey was conducted that has helped in the more effective management of risk factors.

Danny Lieberman [31] has worked to reduce operational risks by improving the software quality. Dannay focuses on the classification and quantitative evaluation of removing the software risks by effective software management, thus contributing to the classified risk mitigation. In a study that was conducted in 2005 [32], a sample of 167 customer's data breaches were analyzed to view the distribution of risks and threats and it were identified that 3% of the total risks are caused by accidental disclosure bye-mails, 7.8% of risks are oriented due to the human weaknesses, 40.1% risks are caused by unprotected computer/backup media and 49.1% of risks are caused due to the malicious exploitation of software risks. Thus, suggesting way mitigates the risk factors more appropriately.

The SEI reports that 90% of all software risks are due to already known defects [33], while all of the SANA top 20 internet security problems are result of poor coding, testing and sloppy software engineering.[34]

Jhon Stiuby (2009) and his team have worked on the management of risks in distributed software projects, which proposes a framework for handling the software projects that are not developed at geographically same location, and have advised a framework to e followed in this regard [35].

B.J. Alge, C. Wiethoff, and H.J (2003) . Kelin have emphasized on the effective handling of risks and problems in the software development lifecycle and in team structure by the usage of knowledge building process and effective communication[36]. E. Bradner, G. Mark, and T.D. Hertal

(2005) have worked to identify the correct team sizes for the different project sizes and have focused the problems that are experienced by over, low and poor staffing [37].

R. N. Burn (2001) and his team have discussed the risks that are oriented due to the in-appropriate application selection methodology, specially in the database projects [38]. R. N. Charatte (1989), has proposed the analysis and management of the risk factors in software development process [39].

The surveyed literature has been identified greatly in the favors of categorical identification of the risk factors as the existence of risk factors can be extremely harmful, if not attended at the proper time by giving due consideration.

III HANDLING AND AVOIDANCE MECHANISM

Table 2, summarizes the Impact, Probability and Overall Impact of each risk factor. The aim to establish the prioritized list is to help the interested community to better handle the software risks, thus, the risk factor with the highest overall impact is proposed to be addressed first and with the highest attention, perhaps even leaving all other activities at hold. While the risk factors like factors number 7 or 3 require least attention, and can be given importance only when the ample staff is free to invest time on the management of these risks factors. After having established the prioritized list of risk factors based upon the overall impact it is necessary that the risks are either to be handled or avoided, it is necessary that a strategy is proposed for each risk factor. Sub-sections 2.1-2.14, discuss the handling and avoidance strategies against each risk factors, presented in Table 1 and Table 2.

A. Requirements Are Not Properly Stated

i. Multiple requirement acquisition approaches must be used; this includes the questionnaires, interviews and direct communication. The team deployed on the requirement acquisition should be capable enough to extract the accurate/valuable information from the information lot coming from different sources. The capabilities of the analysts in terms of requirements acquisition can be determined by their performance in the previous projects. An analyst having a very good track record of determining the requirements may be more trustable for deployment in the requirement acquisition process.
ii. Facilitated Application Specification Techniques (FAST) [7] should be used to ensure the elaborated understanding of the requirements at both ends, i.e. the customer and developer. This informal way of requirement collection helps the development team to understand the requirements in the actual context. [12]
iii. The customer must allow the development team to have a flexible schedule if the requirements are expected to change dynamically. Only minor changes, which don't have the impact on the architecture of the software, can be changed dynamically. The major changes, requiring the change in architecture, cant be completed in the same time and cost. Therefore, if the customer requires or expects the dynamic changes in the requirement definition, it must expect a relatively higher cost and time to complete the project. [13]
iv. The development team must be familiar with the Enhanced Information Deployment [7] technique, to take care of the default requirements that are not explicitly mentioned by the customer.

B. Low Estimation And Time And Cost

i. The development team while bidding for the project must have a clear idea of the requirements that are explicitly stated and also of those that are expected by default. It is appropriate that the management acquires multiple estimates from different sources, and suggest a flexible schedule in terms of time and cost. Only the acquisition of estimates from multiple sources is not sufficient but a mechanism should be in place to identify the best possible estimates out of available. It is recommended that this process be governed by the team of experienced analysts, developers and managers, in order to make this exercise more effective and result oriented [14].
ii. It has been observed that if the funding and time are not flexible, the incremental model [4] of development may be a solution. As it grows in increments, if the funding or time collapse, at least there is something presentable to the customer, rather than having nothing at all. Although the product may be incomplete yet the time and cost incurred can be presented to customer to grab the future funding for development purposes.
iii. The development team must try to find the maximum amount of reusable code, the availability of reusable code will have three dimensional positive effects. First it will decrease the time required for the software development by making available the code that was to be developed if the reusable code were not available. Secondly, it will decrease the cost of development as less development is required in the presence of reusable code, the higher the usage of re-usable code the lower the cost of software development comes. Thirdly, the re-usable code is already tested component and hence does not require re-testing, therefore, saving time of testing the component.
iv. The team of experienced developers and management may decide, in consultation with the customer, that if there are any scrubable requirements that may not harm the overall working of the software. Such requirements may be eliminated to save time and cost[15].
v. Clean room engineering may not be implemented in the projects that have tight time and cost schedule.

C. More Stress Of Users Than Expected

The developer must always expect and consider that the customer is not capable of describing all the requirements. The developer, if possible, must design and implement the system in a way that it can tolerate with the extra burden as well.

The developers must also do the extensive stress testing to ensure that the software is capable of handling the load and stress of the users. The development teams can stress test the software at component level, environment level, architecture level and end-to-end level. In *component level* we assume that although unit testing has its existence yet it has a disadvantage that in the domain of web services, it can't work to check the concurrency and deadlock of the simultaneous requests, adequately. Therefore it is necessary that each component residing on the web server is tested through the stress testing, in order to check that no deadlock occurs during the simultaneous access, and the consistent position of data is maintained and also no deadlock occurs while the records are being accessed and updated. In *environment level* and after the completion of the requirement engineering phase, the development team decides the hardware and software infrastructure that they plan to provide for the development life cycle. The infrastructure may include a database application, a front end application, a hardware platform and a load balancer. This infrastructure helps in determining the scalability, reliability and cost of application. Hence, all available

infrastructural option are to be reviewed categorically in order to identify and estimate the performance and the cost of performance. *Architectural level* stress testing is also called benchmarking. The basic purpose of the stress test on the application's architecture is to measure the cohesiveness of the component residing at the different levels. A well responsive application would ensure that all components at all tires are well associated and working properly. During the development process, the sample components may be taken from each tire of the cohesive modules to detect any flaws during analysis and design of the application. *End-to-End* stress test has a flavor of real test that may be prolonged to several hours and in some cases even to some days. These End-to-End test (if accurately designed) test the application as whole and at length [8].

D. Less Reuse Than Expected

i. While estimating for the projects cost and resource requirement, the developers must know that what amount of software is available for re-use, this should be an rational decision as, if the reusable code is not available the effort to develop such code will be duplicated. As not only code is to be developed, but also the component is to be tested before integration with other components. The person investigating for the availability of re-usable code must have adequate knowledge of existing libraries of components and must also know about the active libraries being updated. The active knowledge of web is also essential in this regard.
ii. If the component is to be developed, it is necessary that a clean room engineering approach is applied is the development so that the time required for testing the component is minimized if not completely eliminated [16].
iii. The best developer, among the available lot, should be deployed to develop the components so that the expected time on development and testing is minimized.

E. Delivery Deadline Tightened Or Manager Change Circumstances

i. The managers somehow try changing the circumstances because of the deadline pressure or because of the orientation of new requirements. The absolute definition of requirements at the beginning ensures that circumstances remain constant and deadlines are not tightened.
ii. The development team and management of the development firm must have the foreseeing capability, and should try adhering to the dynamic circumstances without disturbing the firm itself. For this purpose the firm must try and maintain the experienced staff who can use their intuition at the required time and contribute for the betterment of the firm.
iii. The FAST approach may be used to speed up the requirement acquisition, thus decreasing the negative impact of tightened deadlines. Although FAST session has the build-in capability to speed up the requirement acquisition process yet it is necessary that the FAST session is conducted with the sincerity, spirit and motivation. A FAST session that can't deliver positively causes the wastage of extra time that is very hard to manage in the coming time if the project is already behind the already agreed schedule.

F. Funding Will Be Lost

i. Inorder to ensure that funding issues remain in order, the development team must first ensure that the software is developed within time, developing within time will not only help to improve the revenues and profits but would also ensure that the funding remains available throughout the software development lifecycle. This is the win-win situation in which neither the development firm seeks extra time nor the customer is to pay anything extra for any requirement change.
ii. Its important that friendly relationship is maintained with the funding agency. A state of trust should be estabiliehed between both parties and they should be able to communicate with each other which utmost ease and without involving any other third party channel. The informal meetings of both parties at social events may be of great help in improving the warmness of the relation.
iii. Along with the cordial relationship with the funding agency, it is also important that the funding agency is kept updated regarding the progress of the software development process, and also any problem that is faced during the process. Being informed about the problems and achievements, the funding agency will be in a better place to help the development firm with the continuation of the funding.

G. Technology Does Not Meet Expectations

i. The decision about the choice of technology should be taken only after a very through consideration of the available tools and technologies and only by the experienced practitioners. The customer in some cases may allow the change in technology, but this change must not have any negative effect on the quality of the software, it is also important that any change in the already agreed tools and technologies is done only after the mutual consultation of the development team and the customers. It is the moral responsibility of the development firm to advise the most suitable solution to the customer if he does not have the adequate knowledge of the possible tools and technologies that are available to choose from the available lot [17].

ii. If the change of tool, is agreed between the customer and the development team the development team must try to choose the best available tool in consultation with the customer. The development team should choose the tool in which they have very good expertise so that the expertise in tool may be translated into the company's revenues and profits.
iii. The tool chosen should not only be acceptable to the customer but the customer should have necessary training on the tool. It is also important for the customer to argue with the development firm about the future acceptability of the product being developed by using that specific tool. The choice of tool must not only meet the current needs of the customer but should also be able to meet the future expectations of the customer.

H. Lack Of Training On Tool Or Staff Inexperience

i. The rapid advancement in the current tools and technologies force the developer to remain up-to-date. The development firm can keep its employees updated by offering them training on the emerging tools and technologies. Along with the training on the emerging tool, it is also important that the employees be also provided the advanced knowledge of the current tools in which the firm is doing the development currently. It is also important that someone in the organization have the vision and wisdom to use his intuition about the arrival of future technologies, so that the training can be arranged and provided to the employees in advance and market benefits can be obtained by having this advanced availability of the usage of technology [18].
ii. The firm may hire the new graduates from the leading universities, having some knowledge of the current tools. The firm can train them and provide them small assignments to do, in order to complete their training and making them a useful member of the firm, but all this requires a long planning and a visionary leadership at the firm, who can have the knowledge and wisdom about the emerging trends in technologies. In order to hire the graduates from the reputed universities, the firms may plan to schedule the seminar in the universities for the final term students and may opt to arrange on the spot job interviews to identify potential candidates for the possible hiring to meet the future needs of the development firm. This approach has been observed to be extremely helpful in not only fulfilling the industry-academia gap but in also producing the quality products for the industry by using the knowledge imparted by the academia [18].
iii. It is important that the teams are made for each project. Developing the team structure will help in not only promoting the efficiency of the work but will also help in providing experience to new members. This will also help the new members to learn about the smooth flow and effective handling of the tedious work. Such exercise will help them to learn the art of working in a team in also producing the outcome by doing smart work.

I. Staff Turnover

i. Staff, and particularly the experienced staff is an asset to any firm, and firms generally do their best to retain such individuals. But this is very obvious that learned individuals still want to change the jobs, although this trend may not be eliminated yet it can be reduced. The employer should keep the honest estimations of the salaries available in the market for experienced people. By giving less salary, the employer should not assume that the employee will work sincerely and with the best of his effort, rather the employee may keep on wasting his and firm's time by searching for other employment opportunities during the office hours [19].
ii. Proven experience show that employee enjoys working with an employer who has more care for the families of the employee. The employer may offer the services like, free family medical; children school fee, car allowance, house rent, etc in order to keep the employee attracted.
iii. The employer should provide other social gathering and meeting opportunities to the employees, inorder to help establish a family culture at the organization. This get-together is a good chance for the juniors to meet with the firm's top management and listen to their views and vision about the future of the firm's business strategy. The individual's must be encouraged to provide their view and their views must be considered valuable, so that each individual can feel his/her importance in the decision making of the firm[10].
iv. The employer must try to keep the employees updated and should provide the employees with chances to refresh their knowledge about the emerging tools and technologies [19]. This can be done by arranging the courses at their own site, or by sending the employees to the specialized institutes for training.
v. The employer may introduce a loan scheme to help the needy individuals and the return may be in easy installments, without or at a minimal interest rate [20].
vi. It is necessary that the employer try maintaining the respect and honor of the employees, and it is never compromised in any situation. It is obvious that the respect just does not come by paying the employee more, but it comes by having the friendly and trust oriented relationship. The employees must not be in a position of continuous tease; horror and

torture, a work done under such circumstances can hardly be productive and badly affects the mental and sociological health of the employee. The governing force for the employees to work should not be the threat and anger but the affiliation and desire. Therefore, the polite handling of the staff must be the top priority of the management. A specialized human resource (HR) department may be established in the organization to keep track of all the employee related affairs: including the salary increments, hiring and firing, leave and holidays, productivity, expenses vs. productivity ratio (EPR) etc. The employees having the high EPR must be given the salary rise according to their contribution in the firm's profits. The employees having the normal EPR may or may not be give some benefits, while the employees having low EPR should be warned properly in advance, according to the condition of the contract, before their contracts are terminated [21].

vii. The employer may introduce a bonus scheme to make the employees a part of the profit that the firm gains. This would give a sense of ownership to the employee and the employee will try to deliver according to the best of his capabilities [21].

J. Backup Not Taken & Actual Document/Data Loss

i. Backup must be taken at multiple sites, so that in case of any physical or technical damage the backup itself remains intact, the smaller software development firms may opt not to take backups as they may consider this effort as wastage of time and resources. Actually, they oversee the risk by just being over optimistic about the fact that data neither can be lost nor be stolen.
ii. The management must try to introduce the paperless environment in the firm; this would help in maintaining the efficient, secure, and traceable working environment.
iii. The backup sites may be frequently updated and the updates should be inspected regularly to reduce the chances of any data not being updated on the server. The firm may hire the services of reputed individuals to provide help in this regard, as this is considered the one of the most critical risk factor to be managed.
iv. The team strictures should be implemented in the development environment, this not only improves the working environment but also helps in decreasing the dependency on the individuals as the team members remain active and keep knowledge of the trends and patterns that someone uses in its development. This will not only help in introducing the harmony in the team members but would also increase the efficiency in the working environment [22].

K. Fire, Flood And Building Loss

i. The firm must ensure that the working environment across the organization is not only conducive but also safe for the employees. Proper smoke detectors and fire alarms must be installed in the building to detect the fire and the emergency exit should be provided in case of any emergency.
ii. The organization must also ensure that the building codes have been followed and the structure is according to the prescribed standards. With the orientation of more earthquakes recently in the world, it is also important that the building structure is developed in a way that it can absolve the earthquake shocks of an adequate level.

L. Too Many Development Error

i. Although testing techniques can help in identifying errors yet it is more appropriate to try enforcing the clean room engineering approach [23]. The cost to identify the errors in a relatively large amount of code can be both expensive and difficult at the same time. The cost of rectification of these errors is also very high as the schedule of the development is disturbed and many changes are to be made in iteration in order to bring the software on right track. Clean room engineering, although requires the development of error free code yet it can only be adopted when ample time is available for software development.
ii. For this purpose not only the development team must try working accurately but also the continuous inspections of the work being done by the developer must be reviewed by some senior colleague, so that the guideline may be provided early and correction are made without serious harm [9].
iii. Along with the availability of the inspections, the developer must unit test the piece of software that he is developing and must ensure that the code is free of errors and that it is according to the prescribed requirements [24].
iv. The small software houses, consider testing as a sole responsibility of the developers, and do not have a specific testing department. Although individual components may work fine but the integrated application may still not work, because of the run-time and integration errors. These types of errors are generally beyond the scope of the developer and are to be addressed by the specific testing team in the organization. Absence of dedicated testing team may cause serious problems for the organization in delivering the correct software in-time.
v. The organization must adopt the team structure in the software development. Along with the unit testing, that generally, the developer will do on his

own, the team can help each other to test the code and to ensure that the test cases are correctly designed and are efficiently handled in order to save time and improve the productivity of the resting process [25].

vi. A sudden jump to the new tools and technology adds the risk of too many errors. It is suggested that the jump to a new technology should not be made without adequate thinking and must be supported by the discussion and should be a result of a decision governed by the logical thinking. It should also be noted that adequate training on the tools must be available and provided before the actual shift in the technology is made.

vii. Sometimes there are so many errors identified in a piece of code that correction may not only cause the wastage of time but also the resources. In such circumstance, the re-development of that component may be easier than correcting the existing one. The decision of re-development is a very critical decision and should be supported by the logical discussion among the management governing the project. Before any such decision, a mathematical calculation should be done to logically represent that the re-development is in the benefit of the organization. A re-development must logically be completed in much higher speed as compared to the initial development [26].

viii. It is also important that the testing process works fine, i.e. identification of too many errors can still be less harmful as compared to the ignoring errors or un-identified errors [27], because the identified errors can somehow be tackled and addressed for correction, but an un-identified error may cause harm after all the bugs have been fixed. The errors become more harmful when they exist even after the release of the software. An error identified in external environment costs the firm much more to rectify that error. A released software is like a thrown arrow, once becomes public can't be brought back [28].

M. Developer Run Away With Code

i. At the time of appointment, the Human Resource (HR) department must ensure that the person they are hiring, is adequately trustable and owes a good employment history. His credibility can be checked from the previous employer. The contact details provided by the employer must be verified before the employee is hired permanently.

ii. The organization may also opt to take the employees from the accredited universities and resource providers so that only, already verified, individuals can find a place in the organization.

iii. The organization may also decide to hire the employees based upon the references or recommendation of their existing employees or someone may provide the guarantee for the employee for the purpose of reliability and trust [29].

iv. Backup must be taken at multiple sites, so that in case of any physical or technical damage the backup itself remains intact. The backup sites may be frequently updated and the updates should be inspected.

N. Lack Of Intuition

i. It has been observed that the experienced individuals can help in estimating the cost, budget and manpower of any project by just using their intuition [11]. The guess provided by them is generally accurate, and thus causes a huge benefit for the organization. The organization must do adequate effort to retain such people and should continue befitting from their experience.

ii. Talented individual must be attached to work with the experienced individuals so that they can learn that how the estimations can be made by using the previous knowledge and intuition [30].

IV Conclusion

Software development process is complex and requires efficient handling of the available resources. Poor planning invites risk factors that are very difficult to deal with. The paper unleashes the possible strategies to avoid or overcome risk, once they have been identified in a software process. Although a complete list of software risk factors is impossible to produce, as the risk factors keep on growing with the new tools and technologies, yet a comprehensive list has been considered for providing knowledge about the handling and avoidance mechanism. In the last three decades ample stress has been given on the identification, management, avoidance and handling of risk factors. This paper after having identified the risk factors, proposes the avoidance and mitigation strategies for each risk factor based on the frequency of their occurrence. The software houses that are developing the small and medium software can especially benefit by following the avoidance strategy.

V Acknowledgments

We are thankful to the Research Center (RC) at College of Computer & Information Science of King Saud University, for providing partial support for completion of this work.

VI References

1) Rothfeder, "It's Late, Costly, and incomplete-But Try Firing a Computer System, " Business Week, November 7, 1988, pp. 164-65
2) Coper Jones, "patterns of software success and failure", 1996
3) Roger S. Pressman, "Software engineering: a practitioner's approach", 5th ed, McGraw-hill, pp 151-159

4) Barry W. Boehm, "software risk management: principles and practices", pp 13
5) Basit Shahzad, Tanvir Afzal, "Enhanced risk analysis and relative impact factorization", 1st international conference on information and communication technology, IBA Karachi, August 27-28, 2005 ,pp 290-295.
6) Basit Shahzad, Javed Iqbal, ""Software Risk Management – Prioritization of frequently occurring Risk in Software Development Phases. Using Relative Impact Risk Model", 2nd International Conference on Information and Communication Technology (ICICT2007), December 16-17, 2007, IBA Karchi.
7) Roger S. Pressman, "Software engineering: a practitioner's approach", 5th ed, McGraw-hill, pp 151-159
8) Borland, the open alm company, A Load Testing Strategy, white paper, April 2006,pp6
9) Jiantao Pan, Software Testing, Carnegie Mellon University, Dependable Embedded Systems, spring 1999, pp 1-14.
10) Duport, "how to control and manage the staff turnover "http://www.duport.co.uk/guides/staff%20issues/Controlling%20and%20managing%20staff%20turnover.htm, May 2006.
11) Magic intuition, "definition of intuition "http://www.magicintuition.com/intuition.html", 2009
12) T.E. Bell, T. A. Thayer, Software Requirements-Are they really a problem?, International Conference on Software Engineering Proceedings of the 2nd international conference on Software engineering San Francisco, California, United States, Pages: 61 – 68, 1976.
13) Gursimran Singh Walia , Jeffrey C. Carver, A systematic literature review to identify and classify software requirement errors, Information and Software Technology, v.51 n.7, p.1087-1109, July, 2009
14) Longstreet Consulting Inc., Software development estimation, http://www.softwaremetrics.com/Articles/estimating.htm
15) Baskeles, B.; Turhan, B.; Bener, A. Software effort estimation using machine learning method, 22nd international symposium on, Computer and information sciences, Volume , Issue , 7-9 Nov. 2007 Page(s):1 - 6 Digital Object Identifier 10.1109/ISCIS.2007.4456863
16) Matsumura, K. Yamashiro, A. Tanaka, T. Takahashi, I,Modeling of software reusable component approach and its case study, Proceedings of 4th International Conference on Computer Software and Applications, (COMPSAC1990), 10/31/1990 - 11/02/1990, Chicago, IL, USA, pp 307-313
17) Jhon McManus, Risk Management in Software Projects, Computer Weekly professional series by Elsevier, pp 4-18.
18) Kim Man Lui1 and Keith C.C. Chan, Test Driven Development and Software Process Improvement in China, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 219-222, Friday, May 14, 2004
19) Sarah Wilson, CCRP, How to reduce turnover and manage employees, http://www.ehow.com/how_4495943_reduce-turnover-motivate-employees.html
20) How to motivate employees, http://www.ehow.com/how_2094622_motivate-employees.html
21) Dovinea, How to Hire, Manage and Motivate employees effectively, http://www.ehow.com/how_2377369_hire-manage-motivate-employees-effectively.html
22) Moore, R.W.; JaJa, J.F.; Chadduck, R. Mitigating risk of data loss in preservation environments, Proceedings. 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies , 11-14 April 2005 Page(s): 39 – 48, Digital Object Identifier 10.1109/MSST.2005.20
23) S. wayne Shere, Ara Kouchakdjian, Paul G. Arnold, Experience Using Cleanroom Software Engineering, IEEE Software, Volume 13 , Issue 3 (May 1996), Pages: 69 – 76,Year of Publication: 1996, ISSN:0740-7459.
24) IEEE Standards Board, "IEEE Standard for Software Unit Testing: An American National Standard, ANSI/IEEE Std 1008-1987" in IEEE Standards: Software Engineering, Volume Two: Process Standards; 1999 Edition; published by The Institute of Electrical and Electronics Engineers, Inc. Software Engineering Technical Committee of the IEEE Computer Society.
25) Carnegie Mellon-Software Engineering Institute, Overview of team software process and personal software process, http://www.sei.cmu.edu/tsp/
26) Sachidanandam Sakthivel, A decision model to choose between software maintenance and software redevelopment, Dept. of Accounting & MIS, Bowling Green State University, Bowling Green, OH 43403, USA
27) Bill Curtis, Top five reasons for poor software quality, http://itmanagement.earthweb.com/entdev/article.php/3827841/Top-Five-Causes-of-Poor-Software-Quality.htm, July 1 2009.
28) Leo King, Businesses fear lost revenues after poor software testing, http://www.infoworld.com/d/developer-world/businesses-fear-lost-revenues-after-poor-software-testing-134, April 08, 2008.
29) Dr. Dobb's, Investigating software and source-code theft,

http://www.ddj.com/windows/184406134, July 19, 2005.

30) Naur, Intution in software development, Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT) on Formal Methods and Software, Vol.2: Colloquium on Software Engineering (CSE) , Berlin, Germany Pages: 60 – 79, Year of Publication: 1985 , ISBN:3-540-15199-0
31) Danny Lieberman,http://74.125.153.132/search?q=cache:4VSx1A9wqVUJ:www.software.co.il/downloads/EnterpriseSoftware_RiskReduction.pdf+reducing+operational+risk+by+improving+production+software+quality&cd=1&hl=en&ct=clnk&gl=pk&client=firefox-a.
32) 2005 Breach Analysis, April 2006 http://www.software.co.il/downloads/breachAnalysis2005.xls
33) Privacy Rights Clearinghouse, http://www.privacyrights.org
34) Developing Secure Software, Noopur Davis, http://www.softwaretechnews.com/stn8-2/noopur.htm
35) J.H. Persson, L.Mathiassen, T.S. Madsen, and F. Steinson, "Managing risks in distributed software projects: An integrative franework", IEEE Transactions on engineering management, Vol. 56, No. 3, August 2009.
36) B.J. Alge, C.Witheoff, and H.J. Klein, "When does the medium matters? Knowledge building experiences and opportunities in decision making teams", Organ.Behav, Hum, Dects, Process, vol. 91, no. 1, pp 26-37, 2003.
37) BRADNER Erin ,MARK Gloria,HERTEL Tammie D." Team size and technology fit: Participation, awareness, and rapport in distributed teams", EEE transactions on professional communication ISSN 0361-1434, 2005, vol. 48, no 1 (104 p.) , pp. 68-77
38) R. N. Burns, A. R. Dennis, "Selecting the appropriate application development methodology ",SIGMIS Database, Vol. 17, No. 1. (1985), pp. 19-23.
39) Robert N. Charette, Software engineering risk analysis and management, McGraw-Hill, Inc. New York, NY, USA,Pages: 325,1989,ISBN:0-07-010719-X.
40) Ronald P. Higuera, Yacov Y. Haimes, "Software Risk Management", Technical Report CMU/SEI-96-TR-012 ESC-TR-96-012, June 1996.