An Integrated Framework for Software Development Using Risk Mitigation & Conflict Resolution

Apoorva Mishra
Department of Computer Science & Engineering, CSIT, Durg., India

***Abstract*— The software development life cycle (SDLC) has made the task of developing a software easy by making it systematic and reducing the chance of getting unexpected output at the end of the project. The problem of software crisis, which was once dominant in the software industry has been reduced to much extent by the application of the correct SDLC models. SDLC has made it possible to develop the software in a smooth and controlled manner, but still many software projects- are not delivered within the time deadline, don't fulfil the quality standards or suffer from excessive budget overrun. Some of the major reasons for the delay in delivery of the software or degrading its quality or causing excessive budget overrun are: improper selection of the SDLC model, use of improper approach for dealing with risks and conflicts that occur during the software development. This paper proposes an integrated framework for software development. It focuses on effectively resolving the conflicts that arise during the phases of a software development life cycle using a 'win-win' approach and dealing with risks that may occur during the development of the software.**

***Keywords*— *Software Development Life Cycle (SDLC), risk mitigation, risk exposure, time deadline, conflict resolution, win-win model, collaboration*.**

I. INTRODUCTION

Software development life cycle (SDLC) is a concept used for developing the software product in an organized and smooth manner [1]. The aim of a SDLC is to deliver the software within the time deadline and maintaining the quality of the software product as per the standard, without incurring excessive budget overrun. The performance of the SDLC model is measured on the basis of following parameters:

- Time duration of the Project.
- Quality of the software project.
- Budget overrun (if any) and
- Improvement of the existing knowledge base of the company.

"A risk is a potential problem – it might happen and it might not"-Pressman [10]. A risk may cause serious problems, if not handled properly, like: huge loss of Man-power, funds, delay in project development and could even cause the failure of the project [11]. A risk can be broadly categorized as :

- Project risk
- Technical risk
- Business risk

A risk that can be predicted by some means like past experience is called as a 'predictable' or 'known risk' and a risk which cannot be predicted in any way is termed as 'unknown risk'.

A conflict is "to come into collision or disagreement or to be contradictory" [9]. Some of the general causes of conflicts are as follows:

- Poorly defined goals
- Lack of cooperation/trust
- Different approach to solve a problem
- Clash of the egos
- Work place politics
- Difference in priorities
- Different interpretation of a problem etc-.

If the conflict is not resolved as soon as it arises, then, it tends to escalate and may result in the loss of productive time as both the parties involved in the conflict may not be able to focus on the project due to the unresolved conflict [2]. A conflict that has not been resolved might jeopardize the project; hence it is very important to resolve a conflict as soon as it arises. In this paper a 'win-win' approach for dealing with conflicts is discussed in the methodology section.

The integrated framework for software development is a variant of the general SDLC model with some additional features of risk mitigation & conflict resolution.

II. PHASES OF SOFTWARE DEVELOPMENT

The phases of a general software development model are as follows:

- Planning
- Requirements gathering.
- Requirement analysis
- high level design

- low level design
- Implementation
- Testing
- Deployment & maintenance.

The common phases of a software development model can be represented by the following diagram [8]:

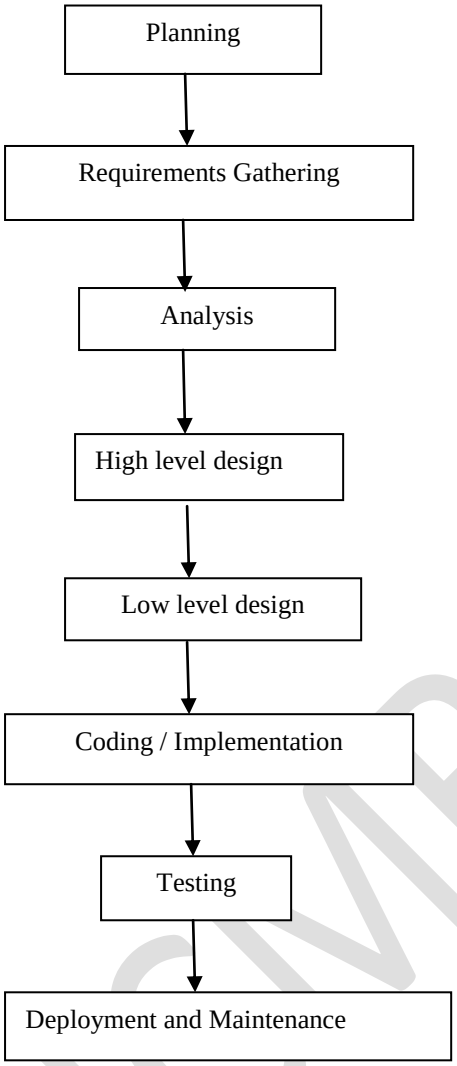

Fig. 1 Phases of a Software Development Model

Some of the famous traditional SDLC models are: Waterfall model, spiral model, win-win spiral model, incremental model, RAD, evolutionary model etc-. Each model has its own advantages and disadvantages and is suitable for a particular situation [4]. The common disadvantage of traditional models is that they are heavy weight in nature, means time requirement is too high for these models, which is practically not possible in many cases. The concept of light weight models came into existence to overcome the disadvantage of more time requirement in case of traditional models. Some of the famous agile models are: XP, scrum, DSDM etc-. The main disadvantage of agile models is that the control over the project is not that much strong as in case of traditional models. The selection of the most suitable model for a particular project is very crucial for the success of the project [6] and is one of the most challenging tasks.

III. PROBLEM IDENTIFICATION

The common problems that still exist in the field of software development are as follows:

- Selection of the appropriate model for a specific project is a matter of personal choice which depends on the cognizance of a person or a team, which might prove to be wrong and the project will suffer.
- Many models for handling risks and conflicts have been proposed, but none of them has been integrated properly as a part of the software development life cycle.
- A generalized framework for software development has not yet been proposed.

IV. METHODOLOGY

The integrated framework for software development is based on the basic concepts of the standard life cycle models. The model has embedded the concepts of:

- Risk Mitigation and
- Conflict Resolution.

As risk and conflict are two major reasons for the delay in the delivery of the project, the concept of handling the risks by mitigating their impact and handling the conflicts as soon as they occur, has been proposed. The standard phases have been modified by adding the concept of risk mitigation and conflict resolution after every phase. The two broad categories of dealing with risks are reactive strategies and proactive strategies [5]. Proactive strategies have a contingency plan, which is prepared during the initial planning phase of the project and keeps on updating after every phase. Whenever a risk occurs, its corresponding contingency plan has the solution of dealing with that risk. Reactive risk strategies deal with the risk as soon as it becomes a reality; there is no planning required in advance. In the proposed framework the proactive strategy will be used for handling and mitigating known risks as depicted in figure 2. (For unknown risks a reactive strategy will be used).

The risks are prioritized on the basis of the risk exposure [3]. The formula for calculating the risk exposure is given in equation1.

$$RE = P \times C \quad \text{——————} \quad 1$$

Where,

RE = Risk Exposure

P = Probability of occurrence of a risk
C = the cost to the project, if the risk actually occurs

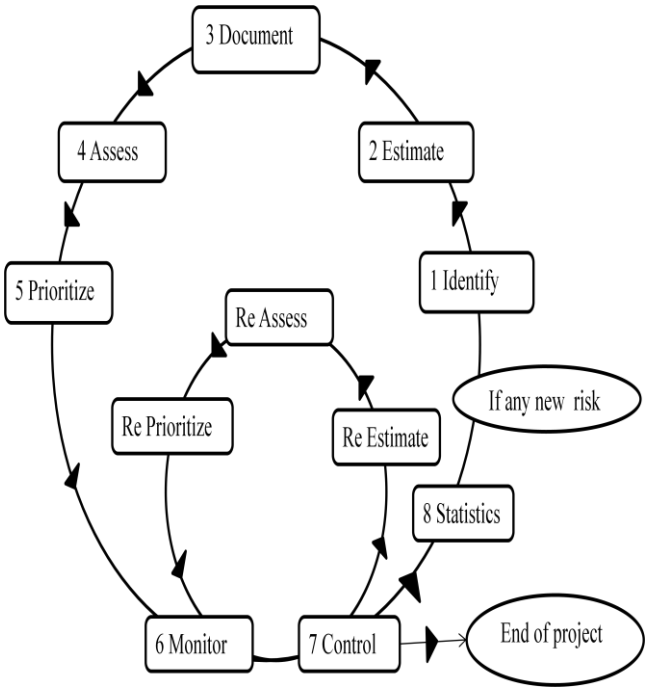

Fig. 2 Methodology for Handling & Mitigating Risks during Software Development

As depicted by figure 2, the steps for handling a risk are as follows:

A. Identification of the Risk

This step involves the identification of different types of risks involved with particular software. Some of the common types of risks are:

1) Software Requirement Risks:

Common risks that belong to this category are:

- Lack of analysis for change of requirements.
- Lack of report for requirements
- Poor definition of requirements etc-.

2) Software Cost Risks:

These types of risks are related to the budget of the software project. Some of the common cost risks are:

- Lack of good estimation in projects
- Unrealistic schedule
- The hardware does not work well
- Human errors

3) Software Scheduling Risks:

Software scheduling risks are those which can cause the delay in the schedule of the software. Risks that come under this category are:

- Inadequate budget
- Human errors
- Inadequate knowledge about tools and techniques etc-.

4) Software Quality Risks:

Software quality risks may degrade the quality of the software, if they become reality. Some of the common Software quality risks are:

- Inadequate documentation
- Lack of project standard
- Lack of design documentation etc-.

B. Risk Estimation

In this step the probability of occurrence of the risk is predicted by conducting a meticulous analysis of the past experience and the internal as well as external environment.

C. Documentation

As soon as the risks are identified and estimated, they are entered into the risk register, so that they can be prioritized and can be used further.

D. Assessment and Prioritization

Risks are assessed and prioritized on the basis of the risk exposure, which is calculated by the formula given in equation 1. The risks having the higher value of risk exposure are given the higher priority.

E. Monitoring and Controlling

The contingency plan is used for monitoring and controlling the risks. The risks are picked from the risk registers and are addressed by taking the corresponding action as mentioned in its contingency plan [7].

Conflicts can arise among the employees of an organization due to clash of egos, different approaches to solve a problem, or due to any other reason as mentioned in the introduction section. If the conflict will remain unresolved, then it may cause the serious problems to the project in terms of the loss of constructive time, as both the parties involved in the conflict will not be able to dedicate themselves completely

for the project until and unless the conflict is resolved. The models for resolving a conflict are of following types:

- Competition (win-lose situation)
- Avoidance (lose-lose situation)
- Compromise (lose-lose situation)
- Collaboration (win-win situation)

As, collaboration is the way of resolving the conflict in a 'win-win' manner, it is the best possible way of resolving any conflict [12], so that both the parties involved in the conflict can be satisfied. In the proposed framework for software development, collaboration is used (which is a win-win solution) to resolve conflicts, as shown in the figure 3 [9].

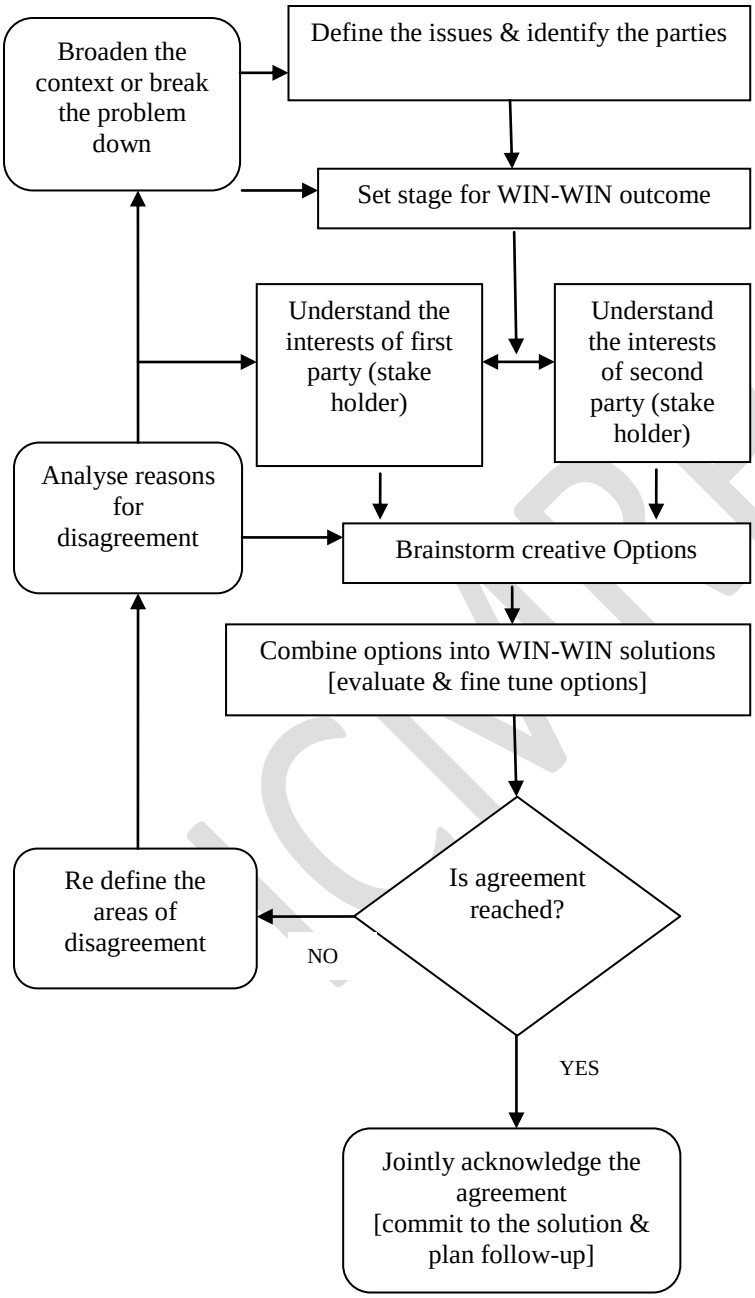

Fig 3. Win-Win Model for Resolving Conflicts Using Collaboration

The integrated framework for the development of the software includes risk mitigation and conflict resolution after every phase of the standard life cycle model as represented by figure 4.

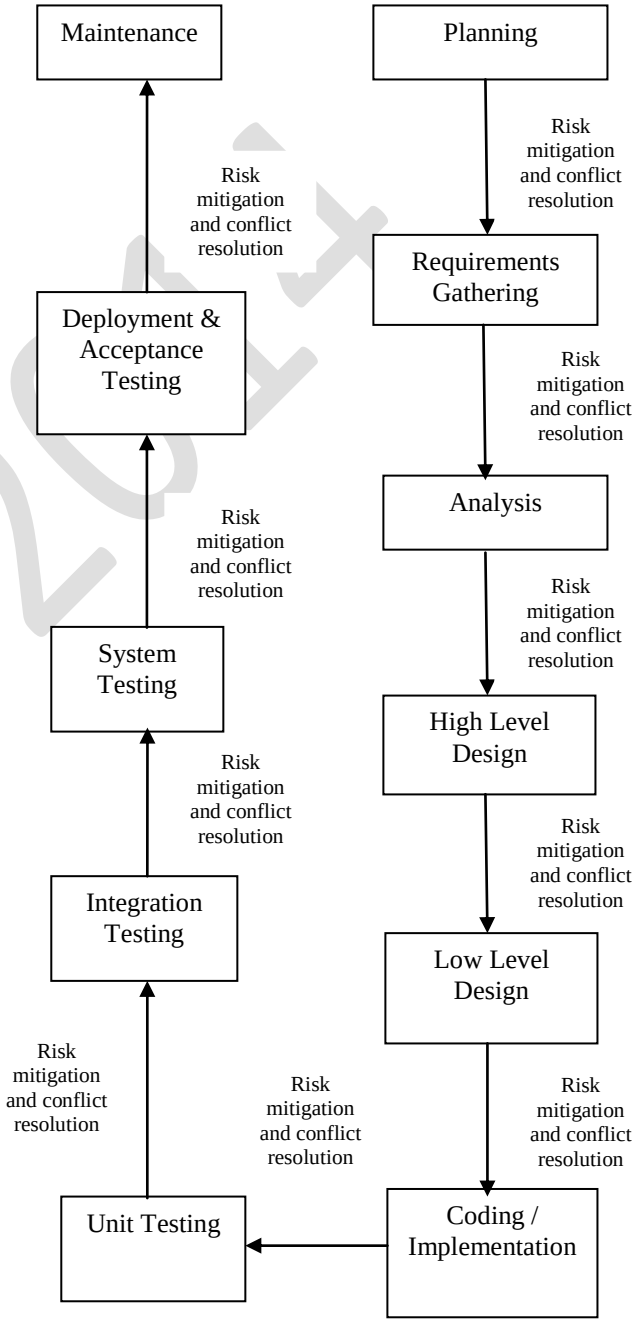

Fig 4. Integrated Framework for Software Development Using Risk Mitigation and Conflict Resolution

The integrated framework serves as a general purpose model for a software project and minimizes the chance of getting unexpected output at the end of the project. Using this framework risks and conflicts can be resolved as soon as they occur. In this model after every phase it is analyzed that if there are any risks related to the next phase and if the

conflicts have occurred in the previous phase. In both the cases the models for handling risks and conflicts is applied respectively.

V. CONCLUSION

"An Integrated Framework for Software Development Using Risk Mitigation & Conflict Resolution" applies the concept of risk mitigation and conflict resolution after every phase of a standard software development life cycle model. The risks are identified and then, prioritized on the basis of the values of their risk exposures as mentioned in equation 1. The conflicts are resolved using a 'win-win' approach through collaboration as described in figure 3. As, risks and conflicts can be handled properly using the framework at an early stage, it increases the chance of delivering the project within the time deadline, with proper quality and without excessive budget overrun.

Hence, the framework serves as a general purpose model for the software development and is very useful for software companies to achieve better customer satisfaction.

ACKNOWLEDGMENT

I take this opportunity with great pleasure to express heartfelt gratitude to my father Dr. Alok Mishra, my mother Mrs. Sangeeta Mishra and my wife Mrs. Sonam Mishra for their cooperation during my work.

I express my indebtedness and gratitude to Mr. Om Prakash Yadav, Head of the Department, Computer Science and Engineering, Chhatrapati Shivaji Institute of Technology, Durg (C.G) for his constant motivation and support.

I extend my thanks to the almighty and the management of the Chhatrapati Shivaji Institute of Technology, Durg (C.G), for inculcating an environment of research in the campus.

REFERENCES

[1] Fairley, R. (2008), "Software Engineering Concepts", Tata-Mcgraw-Hill, 30th reprint, pp 30-62.

[2] Faisal, H. & Mohammad, E. (2011) "Model for conflict resolution in aspects within Aspect Oriented Requirement engineering", Master Thesis in Software Engineering.

[3] Greene, J. & Stellman, A. (2009) "Head First PMP", O'Reilly Media, 2nd Edition. pp- 543-582.

[4] Jalote, P. (1997), "An Integrated Approach to Software Engineering", Springer 2nd Edition, pp 23-73.

[5] Janusz, G. & Jakub, M. (2001), "Implementing risk management in software projects", proc. of 3rd National Conference on Software Engineering, Otwock, Poland pp 1-10.

[6] Khurana, G. & Gupta, S. (2012), "Study & Comparison of Software Development Life Cycle Models", International Journal of Research in Engineering & Applied Sciences, Volume 2, Issue 2, ISSN: 2249-3905.

[7] Kusar, J., Rihar, L., Urban, Z. & Marko, S. (2013), "Extended risk-analysis model for activities of the project", SpringerPlus.

[8] Mishra, A. & Dubey, D. (2013), "A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios", International Journal of Advance Research in Computer Science and Management Studies, Vol 1, Issue 5, ISSN 2321-7782 (Online), pp 64-69.

[9] Mishra, A. & Dewangan B. (2014), "A Model for Software Development Involving Conflict Resolution", proc. of National Conference on Emerging Trends in Engineering & Technology, Shri USB College of Engineering and Management, Abu Road, Rajasthan, India pp 120-123.

[10] Pressman, R. (2010), "Software Engineering: A Practitioner's Approach", McGraw-Hill Higher Education, 7th Edition, pp 200-240.

[11] Shahzad, B., Azween, A. & Yousef, A. (2011), "Trivial model for mitigation of risks in software development life cycle", International Journal of the Physical Sciences Vol. 6, Issue 8, ISSN 1992 – 1950.

[12] Wieland, K., Langer, P., Seidl, M., Wimmer, M. & Kappel, G. (2012), "Turning Conflicts into Collaboration", Computer Supported Cooperative Work © Springer DOI 10.1007/s10606-012-9172-4.